webinos project deliverable

Phase II Security Framework

September 2012

This work is partially funded by webinos, an EU-funded project under the EU FP7 ICT Programme, No 257103.

This report is a public deliverable of the webinos project. The project members will review any feedback received; updates will be incorporated as applicable. The webinos project reserves the right to disregard your feedback without explanation. Later in the year, update to the report may be published on www.webinos.org as well as being made available as a live and community maintainable wiki.

If you want to comment or contribute on the content of the webinos project and its deliverables you shall agree to make available any Essential Claims related to the work of webinos under the conditions of section 5 of the W3C Patent Policy; the exact Royalty Free Terms can be found at: http://www.w3.org/Consortium/Patent-Policy-20040205/.

This report is for personal use only. Other individuals who are interested to receive a copy, need to register to http://webinos.org/downloads. For feedback or further questions, contact: **editors@webinos.org**

DISCLAIMER: webinos believes the statements contained in this publication to be based upon information that we consider reliable, but we do not represent that it is accurate or complete and it should not be relied upon as such. Opinions expressed are current opinions as of the date appearing on this publication only and the information, including the opinions contained herein, are subject to change without notice. Use of this publication by any third party for whatever purpose should not and does not, absolve such third party from using due diligence in verifying the publication's contents. webinos disclaims all implied warranties, including, with limitation, warranties of merchantability or fitness for a particular purpose. webinos, its partners, affiliates, and representatives, shall have no liability for any direct, incidental, special, or consequential damages or lost profits, if any, suffered by any third party as a result of decisions made, or not made, or actions taken, or not taken, based on this publication.

Abstract

The webinos project defines and delivers an open source web application runtime compatible with a wide range of smart devices, including smartphones, tablets, PCs, in-car systems and set-top boxes. A key aim of the project is building a platform which is both secure and protects user privacy. This document describes the security and privacy rational, threat model and architectural risk analysis used by the project. It is a companion document to the webinos system and API specifications and explains why certain security and privacy controls exist and what risks remain. It provides a set of recommendations and describes the outstanding weaknesses and issues of which webinos stakeholders may need to be aware.

Keyword list

Security, privacy, architecture, policy, threat model, architectural risk analysis

Content

1 Introduction

The webinos project delivers a cross-platform web application runtime environment; it provides standard JavaScript APIs as well as inter-device communication and interaction. The development of this runtime environment will help to provide a seamless end-user experience for web applications across multiple devices. The webinos consortium aims to make several innovations in the runtime environment and, as a research project, it aims to go beyond the current state of the art in web application technology.

One of the most important areas for improvement in existing web application technology is the provision of better security and privacy. webinos-enabled web applications will be able to support important and high value functionality such as electronic payment and may store confidential and valuable information belonging to companies or individuals. At the same time, vulnerabilities in web technology are being discovered regularly, with large projects such as OWASP (OWASP) dedicated to cataloguing and mitigating the most common and severe. Furthermore, user privacy is an increasing concern, and mobile applications frequently appear in the news for violating user expectations for how their data are collected and used (Leyden2011).

A key challenge facing the webinos project is that existing threats to security and privacy could potentially have a greater impact on webinos than on existing systems, due to the capability for cross-device interaction and standardised architecture. From the outset we have been aware that an insecure webinos platform could result in the creation of cross-device malware. This malware could capture sensitive private information or commercially valuable data or even create a large, cross-platform botnet capable of launching denial of service attacks against people and organisations. These threats are real, and must be solved in the webinos architecture. The webinos project has therefore been considering security and privacy issues from the beginning, and this document represents the rationale behind the webinos specifications with regards to security and privacy.

There is another compelling reason for the creation of a webinos security and privacy architecture: the standardisation of security and privacy controls and interfaces which will increase usability and reduce development effort. At present, each device manufacturer provides different interfaces and conceptual models for securing applications and protecting users. This makes the task of securing all personal devices challenging for users. By unifying the interface and allowing the management of security policies on all devices to be done on the most appropriate platform (on a device with a large screen and keyboard, for example) users will be able to make better decisions than they can at present.

In a departure from the approach taken in the previous year's report this document does not provide specification material relating to the webinos security and privacy framework. This can be found in related specifications (Webinos-D33) and (Webinos-D34) and aims to be a precise, implementable description of how webinos is designed and should work in practice. Instead, this document describes the rationale and background to the webinos specifications with regards to security and privacy. It identifies the risks and threats facing webinos, explains how these are mitigated and what the residual issues are. It contains detailed information about cloud security an privacy issues and how they affect the webinos architecture. It also lists specific security and privacy concerns with the APIs being developed by webinos and ways of mitigating them. This document is closely related to the specifications given above as well as the previous updates to webinos requirements (Webinos-D25) and the User Expectations of Security and Privacy deliverables (Webinos-D27,Webinos-D28).

1.1. Intended audience

This deliverable provides *informative* content for implementers of the webinos platform and webinos applications. This will help with any security and privacy analysis performed by individual stakeholders and organisations who may be considering deploying webinos.

Perhaps more importantly, this deliverable is useful for the webinos project itself in making sure that security and privacy threats are being mitigated by the architecture. It provides immediate recommendations and feedback to platform developers and API designers. This document represents a snapshot of the overall webinos development process, with some security and privacy issues in webinos addressed already and others still in progress. It is hoped that this work will be continued into the third year of the project.

1.2. Document structure

This document attempts to present the methodology and findings within the main sections, followed by references and an appendix containing source data. We begin by introducing the architectural risk analysis process, describing the methodology followed and the main results. These include an attack resistance analysis, ambiguity analysis, weakness analysis and additional work on securing data at rest. We then outline the threat model that has been adopted in the webinos architecture, defining the trusted and untrusted features of each component. Next, we address security and privacy issues directly related to APIs and their implementations, as well as how these have been addressed. Following this we present a substantial chapter on cloud security, relating primarily to webinos' use of cloud services. We then make a set of recommendations to webinos stakeholders, and in the final section we conclude.

Attached are a set of appendices containing:

- Background on related systems, academic papers and webinos deliverables
- The full list of API threats
- Architectural risk analysis data, including architectural patterns and attack patterns
- Additional draft attacks and attack patterns that are yet to be fully included in the analysis
- A complete set of actions used to inform the threat model

1.3. Security and Privacy Mission

The webinos project has security and privacy as a primary goal. In this section we attempt to define the broad, overall objective of security activities in webinos which help to define whether we have succeeded or not.

1.3.1. Encouraging the development and deployment of secure web applications

- Allow developer with good intentions to create secure applications without introducing new security overheads.
- Create sensible default rules which make it harder to develop insecure applications and easier for users to protect themselves.
- Introduce a secure personal network infrastructure which protects data in transit and properly authenticates endpoints, allowing for distributed applications which do not need to provide inter-application security features themselves.
- Allow for the creation and use of secure distributed services based on shared web APIs

1.3.2. Limiting the impact of insecure applications

- Mobile and web malware is a genuine threat: webinos must try to mitigate the impact of a malicious application on a user's devices.
- Security requirements from all relevant stakeholders (as defined by the consortium) should be taken into account, including - manufacturers, users, developers and network operators.
- Follow the principle of least privilege by making sure that applications define the privileges they are requesting and are given nothing more.
- Allow for revocation of permissions by users who may change their mind about an application or component.

1.3.3. Mitigating the most likely risks to user data and personal devices

- Identify the key risks and threats and have convincing mitigations for as many as possible.
- Make the remaining risks public by sharing the security analysis and source data.
- Make assumptions and expectations on other components clear.
- Make security and privacy part of the design process - influence the architecture such that key risks are balanced against functionality.

1.3.4. Making sure that webinos only enhances online privacy

- Limit the potential for webinos to be used to fingerprint end users and reveal personally identifiable information without consent.
- Create APIs which reveal only the data that developers require, and no more.
- Minimize the potential for surprise generated by new multi-device information sharing.
- Encourage the storage of personal data in areas owned by the user, both for their sake and for developers data handling obligations.

1.3.5. Ensuring that the security and privacy functionality within webinos does not conflict with usability

- Design based on a consistent set of personas, assets, threats and attackers.
- Balance the need for informed user consent with expectations: enable safe APIs where sensible and highlight those where additional consideration is required.
- Avoid the impact of 'click through' such that goal-orientated users are not expected to make all-or-nothing decisions about access control.

1.3.6. Providing security and privacy features in an open environment

- Allow for open source development without sacrificing the integrity of the security and privacy architecture
- Create security and privacy controls which are appropriate for the web and do not depend solely on a closed ecosystem such as an 'app store'

1.4. Key Changes from D3.5

This document replaces D3.5 as the security and privacy framework for the webinos platform. There are several changes between the documents, both in the structure of the framework and in the content.

1.4.1. Structure

In the first phase of specification, the security and privacy specifications were split between both the D3.1 and D3.5, with D3.1 containing details on policy management and authentication, and D3.5 containing everything else. In this set of documents, D3.6 contains purely *informative* sections whereas all *normative* parts of the specification are now in D3.3 and D3.4. This make it clear where implementers need to go for definitive information.

Furthermore, the focus of D3.6 has been narrowed to contain the rationale behind the platform in terms of how it achieves security goals and how it mitigates threats. It also includes specific analysis of key topics, such as cloud security.

The following table outlines where each section of the security framework and specifications can be found:

Section	In phase 1 deliverables	In phase 2 deliverables
Security policy architecture	D3.1 Security	D3.3 Policy
Privacy policy architecture	D3.5 Privacy policy architecture	D3.3 Policy

Authentication	D3.1 Authentication	D3.3 Authentication
Authorisation	D3.5 Authorisation	D3.3 Policy & Application Security
Privileged applications	D3.5 Privileged applications. D3.1 Security	D3.3 Policy (policy editor only)
Secure storage	D3.5 Secure storage	D3.3 PZP Deployment & D3.6 Threat model
Extension handling	D3.5 Extension Handling	
Personal zone security	D3.5 Personal Zone Security	D3.3 Personal Zone PKI & PZH Admin
Platform integrity protection	D3.5 Platform Integrity Protection	
Application certification	D3.5 Application certification	D3.3 Application Security
Device permissions	D3.5 Device Permissions & D3.1 Security	D3.3 Policy & D3.3 Widget runtime
Session security	D3.5 Session Security	D3.3 Personal Zone PKI & Application Security
Implementation guidelines	D3.5 Implementation guidelines	D3.3 Informative deployment specifications
Cloud security models	D3.5 Cloud security models	D3.6 Cloud security
Threat model	D2.7 & D2.8 & D3.5 Background	D3.6 Threat Model
Architectural risk analysis		D3.6 Architectural risk analysis
API security and privacy analysis	D3.1 Security	D3.6 API security and privacy analysis
DDoS considerations		D3.6 DDo Considerations

Platform integrity protection has been removed because integrity assurance was considered unlikely to be implemented within the time scale of webinos. Considering the number of supported platforms this was overly ambitious in phase 1. Further investigations as part of standards activities may re-open this possibility. Extension handling has been removed from webinos in general.

1.4.2. Framework changes

The main changes in the webinos specifications with relation to security and privacy are given below. In general, we have aimed to reduce the scope and complexity of the security and privacy framework and remove ambiguity.

1.4.2.1 *Expanded certificate and key hierarchy*

The webinos overlay network is secured using TLS connections between all endpoints. The full key hierarchy for this is now specified in Deliverable D3.3. It has also been extended to support client and server certificates in browsers and widget renderers. The kye hierarchy has also been designed to allow for multiple user identities such that the use of a specific key does not necessarily reveal linkable user information. More details can be found in D3.3.

1.4.2.2 *Improvements to the authentication system*

Authentication in webinos has now been clarified. Authenticating to the personal zone in general is implemented through a combination of OpenID as well as the device keys described in the personal zone key infrastructure. An important novelty of webinos is that we have avoided introducing any new usernames and passwords: users can re-use their existing credentials from other identity providers. Locally, webinos will provide user to device authentication through the authentication API. This can also be used internally. A full description of the authentication framework can be found in D3.3.

1.4.2.3 *Removal of the attestation API*

The attestation API was removed from webinos in phase 2 due to the perceived difficulties in implementing it across multiple platforms. We intend to continue investigating it in our research, but do not expect to specify its use in the rest of the project.

1.4.2.4 *Removal of webinos-provided secure storage*

The phase 1 specifications included secure storage. However, on further analysis (see the Data Security section in the risk analysis) we concluded that implementing encryption at the webinos level would, in most cases, be unnecessary, lower performance and be a time consuming task not directly related to the main focus of the project. This is motivated by the increase in dull disk encryption products available on Android, Windows and Linux. Instead, we have defined the key requirements for platforms implementing webinos.

1.4.2.5 *Changes to the policy architecture with respect to privacy and exceptions*

The policy framework now includes an 'exceptions' policy file on each device which is not editable or synchronised by the rest of the personal zone. This mitigates the impact of a personal zone device being malicious, as well as the potential for misuse should the personal zone hub be compromised.

We have also updated the specifications of privacy policies to align with work from PrimeLife and integrate better into the widget manifest. This replaces the reduced P3P proposals from Deliverable 3.5. More information is available in the "Policy" section of Deliverable D3.3.

1.5. Terminology

Throughout this document we will use terminology described in the Webinos-D33). In particular we reference:

- Security goals, such as confidentiality, integrity, availability and accountability.
- Access control terms, such as authorisation, authentication and identification.
- Privacy goals, such as anonymity, unlinkability, undetectability, unobservability and pseudonymity.

We would also like to draw the readers attention to the following definitions which are important for the understanding of this document:

1.5.1. Architectural Risk analysis terms

Term	Definition
Ambiguity Analysis	This activity involves eliciting undesirable behaviour resulting from ambiguity or inconsistency in the software architecture.
Architectural Pattern	These express a model of part of a software system, pre-defined sub-systems, responsibilities, and guidelines for organising the relationships between model elements.
Attack pattern	Attack patterns are descriptions of common methods for exploiting software that both provide an attacker's perspective, together with guidance towards mitigating them.
Attack Resistance Analysis	This identifies general flaws from the literature and knowledge basis of known attacks and, based on these, identifies potential risks and their viability.
CAIRIS	CAIRIS (Computer Aided Integration of Requirements and Information Security) is a Requirements Management tool for specifying secure and usable systems. This tool was developed at the University of Oxford and has been extended during the project to support the webinos design process.
Goal model	Goals describe the objectives that a system aims to achieve when it has been fully developed. A goal model is an artefact used in goal-oriented requirements engineering to link high-level goals to the sub-goals that the can be refined to. Meeting a high-level goal may require meeting one or more of the sub-goals. Goal trees provide a way to structure the requirements of a system and trace the

	rationale behind goals. They also allow for individual sub-goals to be assigned to responsible parties.
Obstacle model	Obstacles are undesirable behaviours which a system aims to avoid. An obstacle model is the logical inverse of a goal tree. It defines a set of behaviours which prevent a goal from being met.
Weakness analysis	This identifies weaknesses that might arise in a system due to the impact of the architecture's dependencies.

1.5.2. Trust

Term	Definition
Trusted	Alice believes that Bob is 'trusted' if she believes that Bob will behave in the way she expects, not violating her security or privacy requirements. Similarly, an application is 'trusted' if the user believes that it will behave in the expected manner and not violate their security or privacy requirements. Generally the notion of trusting an entity only occurs when the entity has the ability (but hopefully not the intention) to abuse that trust. We aim to limit the number of components in webinos that are trusted while carefully documenting how each component is trusted and what for.
Trustworthy	An entity is trustworthy if it *will* behave as expected. For example, an application which is granted access to user location data is trusted by that user to not misuse that data. The application is *trustworthy* if it uses the data in the way the user expects. This might mean not sharing it with a third party, or not storing it at all.
Trustable	An entity is trustable if sufficient information is available in order to work out whether it is trustworthy. For example, a website being served over HTTP is arguably not *trustable* because we cannot authenticate whether it is the website it says it is. It may be being spoofed by another party. In comparison, a website served over HTTPS with a certificate from a trusted authority is more *trustable* because they can be identified. Being trustable is a necessary condition for establishing that an entity is trustworthy, but not sufficient. The website served over HTTPS still may be untrustworthy. Another example is a downloaded application. It might be considered trustable if its integrity and authenticity is checked (e.g., it is signed and the signature is verified before execution) and if it is running on a platform that is trusted. If these conditions do not hold, then the same application might have been modified by an attacker or might be running on a malicious platform altering its behaviour.

2 Architectural Risk Analysis

2.1. Motivation

The design of webinos did not start with a blank page, but with a bricolage of different model elements. By re-using existing software components, we introduced design elements into our architecture. Assumptions about possible attacks might also influence architectural decision making. Even designers with a good understanding of both the problem and solution domains may not appreciate the implications of protocol selection, or the wording of requirements. Moreover, because the abstractions used by a designer don't always match those used by an attacker then flaws missed by the former may be found and exploited by the latter. Consequently, we need tools that help us assess the security consequences of bringing together different model elements.

The aim of an architectural risk analysis is to identify design-level flaws in a software architecture. This process was first described by McGraw (McGraw2006), and motivated by the claim that design flaws account for a significant number of security problems; such flaws cannot be identified by code-inspection alone. An architectural risk analysis shares many of the characteristics of classic risk analysis: it emphasises tangible assets of business value and, as a process, is knowledge intensive, requiring knowledge of both the problem domain and security expertise about potential flaws and attacks. In other respects, however, architectural risk analysis is more challenging. It relies on additional knowledge about solution-based models that form the basis of a software architecture, along with a sense of the requirements and constraints implicitly assumed when these are adopted.

In D2.7 and D2.8, we illustrated how the IRIS meta-model can deal with risks in the broader socio-technical environment within which a software system is situated. The IRIS (Integrating Requirements and Information Security) meta-model was developed at the University of Oxford to integrate concepts from Usability, Security, and Requirements Engineering (Faily2010b); this laid the foundations for subsequent work that evaluated the security and usability implications of mitigating risks (Faily2010b), and representations for archetypical attackers behind these risks (Atzeni2011). This work has also contributed to the development of the open-source Computer Aided Integration of Requirements and Information Security (CAIRIS) requirements management tool (CAIRIS), which has been validated using real-world case studies, e.g. (Faily2010c, Faily2011a).

Two further augmentations are needed for undertaking a more detailed analysis of software architectures. First, although the IRIS meta-model provides substantial support for modelling the problem domain, solution domain concepts are limited to the notion of assets. While modelling assets is necessary for architectural risk analysis, it is not sufficient as many features of an architecture warrant analysis in their own right; these include the architecture's *attack surface* -- the measure of its exposure to attack -- and the properties of connections between its elements. Second, model representations are needed for specifying the elements of software architecture and the attacks these need to resist. These representations need to match the thinking that designers might have about different perspectives of a system, and it should be possible for them to quickly evaluate the consequences that attack and defence elements might have on each other.

2.2. Approach

The approach prescribed by McGraw for carrying out an architectural risk analysis involves carrying out three steps. In the first step, an *attack resistance analysis* is carried out to identify general flaws from the literature and knowledge bases of known attacks and, based on these, identifying potential risks and their viability. In the second step an *ambiguity analysis* is carried out to discover new risks resulting from ambiguity and inconsistency in a design. In the final step, a *weakness analysis* identifies weaknesses that might arise due to the impact of the architecture's dependencies. Interested readers may wish to refer to (McGraw2006) for a more detailed presentation of this process, although Khan et al. (Khan2011) provide a more recent illustrative example based on an analysis of the Chromium browser.

To support a *model-driven* architectural risk analysis, we build two model-based constructs: architectural patterns and contextualised attack patterns. The following sections describe how these constructs are defined and used.

2.2.1. Architectural patterns

Architectural patterns were proposed by Buschmann et al. (Buschmann1996) to express a model of a software system, provide a set of pre-defined sub-systems, specify responsibilities, and include rules and guidelines for organising the relationships between model elements. To capture the elements of an architectural design pattern, we introduce new concepts to the IRIS meta-model, while making use of existing concepts and relationships. Collectively, the architectural patterns meta-model in Figure 1 provides three different views of an architectural pattern.

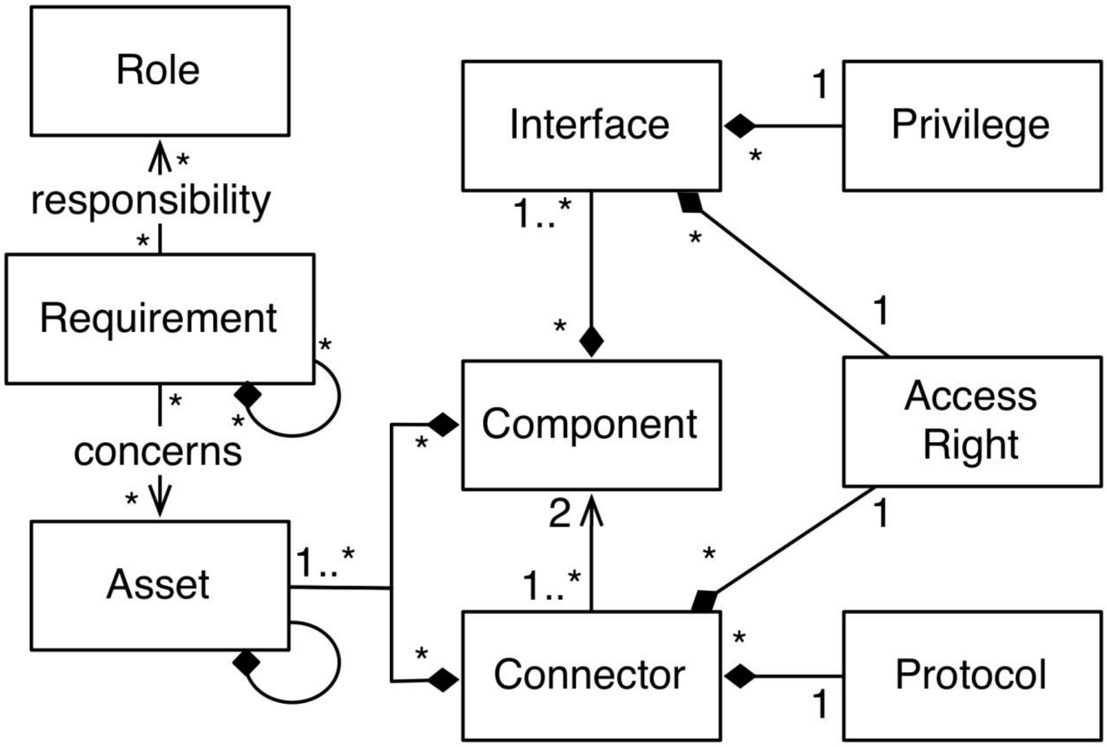


Figure 1: Architectural Pattern Meta-Model

The first of these is a component and connector view, which is expressed using UML component diagrams. This captures the runtime attributes of a system in terms of its computational elements (components) and the interaction pathways between them. Components are attached to connectors via interfaces; these are services or methods through which component interaction takes place. Interfaces are associated with an access right to indicate the level of authorisation needed to use the interface. Interfaces also have a particular privilege level. Connectors, like components, are characterised by their access rights and also by the protocol upon which the connector runs. These meta-model components are closely aligned with the model of software architecture described by Gennari & Garlan, which was recently adapted to capture the elements of a software architecture's attack surface (Gennari2012). This involves specifying the model elements associated with components and connectors, and assigning numeric privilege, access right, and protocol values to these elements. These values range between 0 and 10 and represent an element's exposure to attack; the higher the value, the greater the exposure. These values make it possible to formally evaluate the damage potential associated with interfaces, data transmitted through a connector, and untrusted data items with respect to restrictions placed on the data they contain. This model is described in more detail in the ambiguity analysis section.

The second is a goal view, and is characterised by the system requirements that motivate or constrain the architectural components; within the IRIS meta-model, these system requirements are expressed using KAOS (Knowledge Acquisition in automated Specification) goals and goal models. (VanLamsweerde2009). KAOS responsibility links describe the roles responsible for satisfying the behaviour associated with requirements, while concern links are used to describe instances where requirements reference or constrain assets.

The third view is based on a module view of assets. These assets are salient concepts of value that are specific to components or connectors; this view is expressed using UML class diagrams.

2.2.2. Contextualised attack patterns

Attack patterns are descriptions of common methods for exploiting software that both provide an attacker's perspective, together with guidance towards mitigating them (CAPEC). In recent years, work on attack patterns has been popularised by the development of open-source intelligence Common Attack Pattern Enumeration and Classification (CAPEC) (CAPEC) and Common Weakness Enumeration (CWE) (CWE) repositories. Although these repositories offer a wealth of useful attack data, the patterns are deliberately abstract in order that they can be applicable in as many contexts as possible. To provide this context and re-use as much existing model data as possible when applying these patterns, we have developed a meta-model for *contextualised attack patterns*. These model both the attack and design elements necessary to instantiate an attack within a specific context of use.

The model upon which contextualised attack patterns are based is the *Gang of Four* design pattern template (Gamma1995). However, the meta-model is based exclusively on existing concepts and associations in the IRIS meta-model. As such, when a contextualised attack pattern is introduced into CAIRIS, it introduces a new risk, together with the IRIS model elements that act as its rationale. The relationship between the contextualised attack pattern structure and the meta-model is illustrated in Figure 2 by the UML comment nodes denoting the name of the pattern elements.

The intent and consequences elements are used to describe the overall intent of the attack, and the external impact of the attack being successful. This impact is broader than the impact of a particular architectural pattern and is described using terminology that all system stakeholders can understand. The applicability element states the environment within which the attack pattern will be introduced. This is also the same environment within which architectural patterns will be situated to determine whether the design elements are resistant to this or other attack patterns within the environment.

The structure element describes the details of the attack itself. Attacks and Exploits are drawn from both CAPEC and CWE. The structure is closely complemented by the participant element, which models information about the attack; this includes the attacker's capabilities and motives for carrying out the attack. This information is derived from personas that have been created for possible attackers. Personas are specifications of archetypical user behaviour that are grounded in empirical data collected from representative users (Pruitt2006). These add a substantial amount of context to the analysis because not only do these add a human face to the attackers behind attack patterns, these are grounded in data sources about real attackers.

To describe how the attack defined by the structure might be implemented, leaf obstacles are associated with threats and vulnerabilities and a KAOS obstacle model is defined to describe how these might arise. Like the KAOS goal model in the architectural pattern, these obstacles might concern assets, and responsibility associations describe the roles responsible for satisfying obstacles.

As van Lamsweerde et al. have observed (VanLamsweerde1998), a KAOS obstacle model can be seen as a goal-driven form of a fault tree. However, unlike fault trees, our approach to obstacle modelling is closely tied to other artifacts such as previous knowledge about attacks and information about the attackers that might carry these out. Collectively, where useful statistical data about possible attacks exists, this information can help us predict the likelihood of particular obstacles being satisfied. When a probability value is specified for this likelihood then a rationale statement also needs to be provided to justify it. This is necessary because, when attack patterns are imported into a CAIRIS model, it may not be immediately obvious that the obstacle or the obstacle model arose from them. By proving this justification, we have some way of understanding the thinking that motivated this value. Based on these values, we can evaluate the probability of a particular cut of an obstacle tree based on the same equations used to evaluate the faults in a fault tree. For example, for an obstacle O_x with leaf goals O_1 and O_2, the probability of O_x (i.e. P(O_x)) where O_1 and O_2 are AND-refinements is O_1 x O_2; where O_1 and O_2 are OR-refinements then P(O_x) is O_1 + O_2.

Like classic design patterns, the collaboration concept describes the classes necessary to achieve the designer's intent. However, in the case of attack patterns, the classes are assets and the designers are attackers. As such, the collaborating assets are those which are targeted by threats or exploited by vulnerabilities. Closely aligned with this concept are motivating security properties of interest to an attacker realising this pattern.

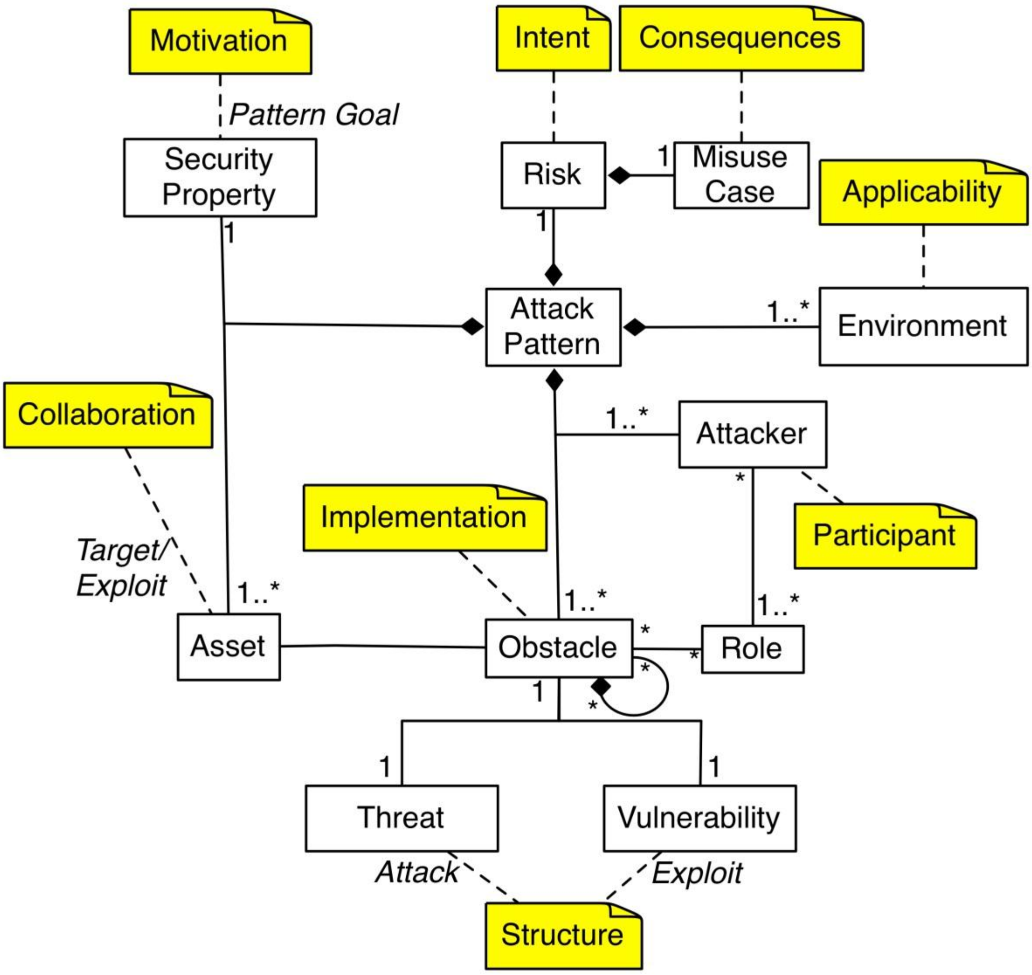


Figure 2: Attack Pattern Meta-Model

2.2.3. Architectural Risk Analysis Process

For D3.6, we have applied an architectural risk analysis process which is based on that proposed by McGraw. These are described in more detail in the following steps.

2.2.3.1 Attack resistance analysis

In this step, we begin to populate the contextualised attack pattern template based on potential security concerns that may be associated with the pattern. This includes searching the imported knowledge bases for the pattern structure elements, and identifying attacker personas with the ability to carry out the identified attack. If the existing attacker personas do not have either the capabilities or motives for carrying out the attack, then it may be necessary to create a new, more meaningful attacker persona. The process for doing this is beyond the scope of this paper, but is described in more detail by (Atzeni2011).

To illustrate how each attack pattern comes about, KAOS obstacle models are developed to illustrate the conditions that make the attack possible. Where appropriate, attack and exploit elements are associated with root or leaf obstacles in the obstacle model. As further obstacles are elicited, these are refined to identify other potential threats and vulnerabilities. As this model evolves then, where possible, probability values are assigned to obstacles, and potential goal and responsibility links are assigned to known system assets and roles referenced in the contextualised attack pattern.

Once the attack patterns were finalised, the leaf obstacles in each attack pattern obstacle model were noted for subsequent ambiguity analysis. These would either need to be satisfied by the software architecture, or their non-satisfaction would need to be motivated.

2.2.3.2 Ambiguity analysis

For a specific areas of architectural significance, architectural patterns are created to encompass the component and connector, requirement, and asset views associated with this area. As McGraw suggests (McGraw2006), this is the most intellectually demanding part of the process because the information necessary to populate the pattern needs to be elicited from various sources, including design documentation and source code. It is also necessary to involve other designers and domain experts to validate the architectural pattern as it is specified. For this reason, the pattern itself will invariably be revised throughout the architectural risk analysis process.

Once the architectural patterns have been finalised, the *Damage Effort Ratios* (DER) were calculated for each architectural pattern. The DERs are a ratio of the potential damage done by exploiting resources over the amount of effort an attacker expends to access it (Gennari2012). Building on previous work by Gennari and Garlan [ibid], it is possible to evaluate the damage potential across the interfaces (DER_m), channels (DER_c), and untrusted surfaces (DER_i) are calculated. These formulae for these ratios are defined below:

- DER_m: Privilege / Access right
- DER_c: Protocol / Access right
- DER_i: Surface type / Access right

These ratios not only provide a high-level quantative measure of the webinos attack surface, their automatic evaluation also provides a quick spot check for unexpected results that might suggest an incomplete analysis, or hitherto unaddressed areas of the architectural risk analysis. If there are unusual or unexpected values, the source architectural patterns can be reviewed and, where appropriate, revised before proceeding.

The next step in this analysis involves taking the leaf obstacles from the attack resistance analysis, and eliciting one or more requirements that mitigate them. Each mitigating requirement is categorised with the affected architectural pattern component/s, an indication of whether it is satisfied or not, and the rationale for how the mitigating requirement is treated. Where mitigating requirements are satisfied, the requirements are added to the architectural patterns and incorporated into the KAOS goal model associated with each component. A KAOS *resolution* link is also added to the attack pattern obstacle model to indicate that the leaf obstacle has been mitigated. Where mitigating requirements are not satisfied, a KAOS domain property object is created to capture the rationale for not addressing the requirement. These domain properties represent hypothesis or assertions about the environment that assume as part of our analysis, e.g. that a particular requirement is out of scope. Like mitigating requirements, the KAOS obstacle model associated with the respective attack pattern is updated to reflect that the obstacle is resolved [albeit imperfectly] by the domain property.

2.2.3.3 Weakness analysis

The final step in the architectural risk analysis considers whether any residual weaknesses remain in each architectural pattern. This analysis involves considering assets that are associated with both architectural and contextualised attack pattern, and -- for threats or vulnerabilities associated these assets -- whether the software architecture adequately addresses them. For each architectural pattern, the measures in place to address the affected threats or vulnerabilities are described.

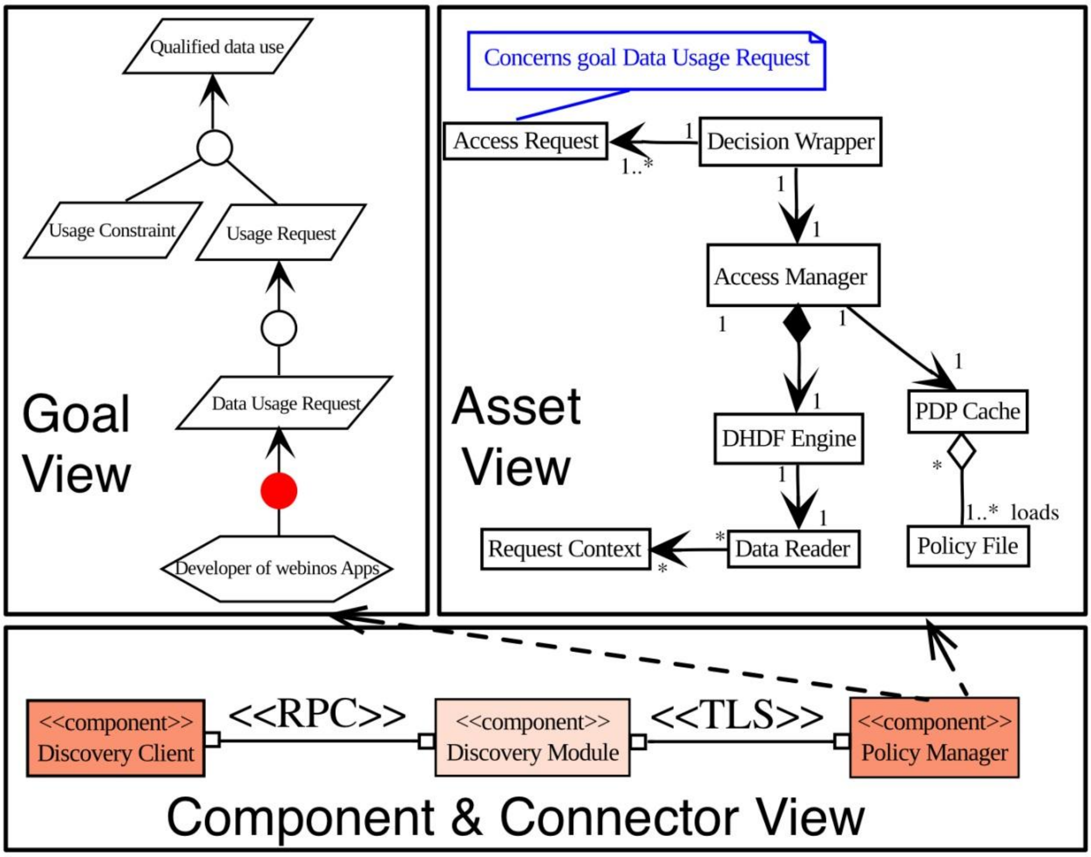


Figure 3: Asset, Goal and Component Views of Contextual Policy Management Architectural Pattern

2.2.3.4 Tool-support

To support it, we modified CAIRIS to support the aforementioned changes to the IRIS meta-model. We have also modelled architectural patterns and contextualised attack patterns as XML Data Type Descriptions; these facilitate the import of both types of artifact directly into CAIRIS. More information about CAIRIS's facilities for importing threat and vulnerability directories can be found in (Faily2011b).

2.3. Attack Resistance Analysis

The security concerns which formed the basis of the attack resistance analysis were initially drawn from the misuse cases in D2.8. However, these were also supplemented by several additional security concerns which had been identified by the project in the last year. These concerns were used to populate 16 contextualised patterns.

The definitions for the attack patterns and their supplemental obstacle models can be found in the appendix, but these patterns -- and the leaf obstacles arising from them -- are summarised in the table below.

Attack Pattern	Attack	Exploit	Leaf obstacles
Application DoS	Malicious Automated Software Update	Improper Verification of Cryptographic Signature	Application blacklisted after negative reviews, Application developer signing key compromised, Application signing key not checked, Bad default policy, Bad user-selected policy, JavaScript injection overwrites webinos.js, JavaScript injection triggers security violation, Webinos backdoor
Capture Hidden Analytics	Spyware	Lack of data provenance	Apps share usage data, findService API reveals permanent user identifier
Device availability loss	Denial of Service through Resource Depletion	Uncontrolled Resource Consumption ('Resource Exhaustion')	Installed app exploited, Installed app misbehaving, Malicious background application installed, Native malware running, Webinos widget processor bug, XSS attack on hosted app
Exploit network bandwidth	Repudiation Attack	Permissive convergence preferences	Post spoofed message, Spoof network settings message origin
Exploiting transitive permissions	Data Interception Attacks	Improper Preservation of Permissions	Installed App allows unrestricted postMessage, Installed App allows unrestricted XHR, Installed App given API permissions, Installed App uses unauthenticated webinos event

			messages, Installed App vulnerable to content injection, Malicious App misuses communication interface
Identity theft with webinos messaging	Mobile Phishing (aka MobPhishing)	Insufficiently Protected Credentials	Attacker obtains user password, Bad trust decisions, Malware installed, Permission prompt click-through, SMS intercepted and relayed
Linkability through findServices	Cross Site Identification	Information Exposure Through Persistent Cookies	Apps share usage data, findService API reveals permanent user identifier
Loss of personal zone administration access	Denial of Service	Unverified Password Change	Account deleted, Account lock-out, Forgotten credentials, OpenID provider offline, Password guessed and reset, Password recovery process attacked, PZH offline
Man In The webinos Browser	Man-in-the-browser attack	Inclusion of Functionality from Untrusted Control Sphere	Application data readable, Malicious plugin not detected, Widget renderer supports extensions
Overlay network facilitated relay attack	NFC replay attack	System data trust	Malicious code evaluated, Malicious NDEF tag, Overwrite valid authentication data, Overwrite valid PZP Configuration, Revoke device from valid personal zone, Run multiple personal zone proxies, Spoof message origin
Policy evasion through Browser APIs	Accessing Functionality Not Properly Constrained by ACLs	Missing Authorization	App running in browser, Browser API authorised, Policy misconfigured
PZH Pharming	Pharming	UI Misrepresentation of Critical Information	PZH Admin URL displayed without prominence, PZH Admin URL not well known, PZH Admin URL too complicated, User clicks on link within application
Request	Network	Missing XML	Eavesdrop Context Database,

footprinting	Eavesdropping	Validation	Eavesdrop Policy Manager, Eavesdrop request enforcement channel, Eavesdrop RPC Call Log
Steal In-Car Data	Session hijacking	Automatic login	Malicious code evaluated, Malicious NDEF tag, Overwrite valid authentication data, Overwrite valid PZP Configuration, Revoke device from valid personal zone, Run multiple personal zone proxies, Spoof message origin
Steal webinos Session	Cross-Site Request Forgery	Session Fixation	Malicious code evaluated, Malicious NDEF tag, Overwrite valid authentication data, Overwrite valid PZP Configuration, Revoke device from valid personal zone, Run multiple personal zone proxies, Spoof message origin
Test footprinting	Locate and Exploit Test APIs	Allocation of Resources without Limits or Throttling	Ambiguous request specification, Non-mandated request, Overrestricted resource, Test API enabled, Test configuration enabled, Unrestricted request specification, Unrestricted resource, Unspecified resource

2.4. Ambiguity Analysis

Based on our review of the webinos software architecture, we have defined 7 architectural patterns that characterise the webinos attack surface. For brevity, these are defined within the appendix, but are summarised in the table below:

Architectural pattern	Description	Components	No. of assets
Context Policy Management	Model illustrating how policy management mediates the	Context API, Context Database, Context Manager, Policy Manager, webinos API	15
NFC	Model illustrating the components associated with	Application Client, NFC Manager, NFC Manager B	14

	NFC		
PZH Authentication	Model illustrating the components associated PZH authentication.	Discovery Manager, OpenID Proxy, PZH Provider, PZH Session Handler, User Agent	15
PZP Enrolment	Components associated with enrolling and authenticating PZPs to PZHs	Certificate Manager, Discovery Manager, Keystore Manager, PZH Provider, PZH Session Handler, PZP Session Handler	16
TV Service Discovery	Model illustrating discovery and use of TV services	Application Client, Discovery Manager, Policy Manager, TV Manager	24
Widget Processing	Model illustrating the components associated with widget processing	Policy Manager, Widget Manager	21
Widget Rendering	Model illustrating the components associated with widget rendering	Discovery Manager, Widget Renderer	8

Based on both the pattern specifications and the damage effort ratio formula defined the methodology, the quantitative attack surface of webinos is summarised in the below table:

Architectural pattern	DER_m	DER_c	DER_i
Context Policy Management	1.2	6.1	28.6
NFC	1.4	4	39.5
PZH Authentication	9.6	6.1	29.7
PZP Enrolment	11.2	5	16.6
TV Service Discovery	3.7	4.2	29
Widget Processing	1.1	1	22.9
Widget Rendering	2.8	2	28.6

At a one-day workshop held in Berlin in August 2012, the leaf obstacles resulting from the attack resistance analysis were reviewed and, based on these, 71 mitigating requirements were elicited. Each requirement was analysed to determine its affected components and whether these were adequately addressed by the webinos software architecture. A summary of this analysis is documented in the table below:

Obstacle	Mitigating Requirement Name	Mitigating Requirement Definition	Affected Components	Satisfied (Y/N)	Rationale
Account deleted	Authorised account removal	The OpenID account shall be removed only by an authorised user.	N/A	N	Deletion of OpenID account out of scope
Account lock-out	OpenID lock-out recovery	Locked out OpenID credentials shall be recoverable.	OpenID Proxy	N	OpenID account recovery procedure is out of scope.
Ambiguous request specification	Canonical request specification	The request specification shall be validated before use.	Policy Manager	Y	Each request is enforced separately by definition. Hence, it's impossible to grant access to more resources than required.
Ambiguous resource spec	Canonical request specification	The request specification shall be validated before use.	Policy Manager	Y	Each request is enforced separately by definition. Hence, it's impossible to grant access to more resources than required.
App running in browser	App running in widget renderer	webinos applications shall run only in approved widget renderers.	Widget Renderer	N	There are no approved widget renderers.
Application blacklisted after negative reviews	Application blacklist checks	webinos supported app stores shall periodically check the	Widget Manager	N	webinos supported app stores are out of scope.

		validity of submitted reviews.			
Application data intercepted	Widget data authorisation	Application data from widgets shall be accessible only to authorised components.	Widget Manager	N	Although applications shall only access data permitted by the underlying system, circumventing cross-origin resource sharing restrictions via native application access to application data is out of scope.
Application data readable	Widget data authorisation	Application data from widgets shall be accessible only to authorised components.	Widget Manager	N	Although applications shall only access data permitted by the underlying system, circumventing cross-origin resource sharing restrictions via native application access to application data is out of scope.
Application developer signing key compromised	Application developer signing key storage	The application developer's signing key shall be safeguarded from unauthorised access.	Keystore Manager	N	Keys are currently saved in $HOME/.webinos. If we use the keyring, platform applications can access contained keys once keyring is unlocked
Application has user identifier without permission	User identifier permission	Application shall be authorised to access to user identifier.	Policy Manager	Y	User identifier is a controllable resource.
Application impossible to use	Application QoS	A webinos application shall satisfy its specified quality of service	Application Client	N	We have no way of asserting how an application's quality of service expectations might be expressed or satisfied.

		expectations.			
Application signing key not checked	Application signing key verified	Update procedure shall verify that the update signing key matches the original signing key.	Widget Manager	Y	widgetmanager.js performs the author matching (calling matching code in comparisonresult.js)
Apps share usage data	Usage data sharing restriction	The sharing of usage data shall be restricted between applications.	Context Manager, Widget runtime	N	Because there are several ways applications can share data, including server-side out of band methods, usage sharing restrictions are out of scope. However, privacy policies can explain what an application proposes to do with user supplied data, which may allow users to make informed choices at install time.
Attacker obtains user password	User password authorisation	Authorisation shall be necessary to obtain a user login password.	PZH Provider	N	While we can recommend safe OpenID providers and make recommendations, controlling how OpenID providers are run is out of scope.
Authentication failure	OpenID authorisation	OpenID authentication process shall authenticate users.	OpenID Proxy	N	webinos should use PAPE extension and set max_auth_age=0 in order to prevent authentication caching
Automate personal zone enrolment	Manual personal zone enrolment.	Devices shall be enrolled into a personal zone only through	PZH Provider, PZP Session Handler	N	Manual personal zone enrolment is circumvented by the latest specification which allows in-band authentication.

		the use of an out-of-bound challenge.			
Bad default policy	Permissive default policy	The default webinos policy shall be permissive enough to require no modification for the majority of webinos applications.	Policy Manager	Y	The API default options are reasonable given the majority of expected applications.
Bad trust decisions	Trust information.	Users shall be provided sufficient information to identify malware when making authorisation decisions.	Widget Manager	N	Recommending what might be sufficient information to identify prospective malware is out of scope.
Bad user-selected policy	Best-practice policy	The user shall select best-practice policy settings.		N	The default policy is only a static template rather than anything specific to a device or personal zone.
Badly configured policy	Best-practice policy	The user shall select best-practice policy settings.		N	The default policy is only a static template rather than anything specific to a device or personal zone.
Browser API authorised	Browser API unauthorised	The use of browser APIs shall be forbidden in webinos applications.	Widget Renderer	N	Forbidding access to browser-provided APIs is not possible without specifying a webinos specific browser.

Browser Geolocation API accessed	Browser API unauthorised	The use of browser APIs shall be forbidden in webinos applications.	Widget Renderer	N	Forbidding access to browser-provided APIs is not possible without specifying a webinos specific browser.
Credentials changed	Credentials change authorisation.	OpenID credentials shall be changed only by authorised users.	N/A	N	Changing OpenID credentials out of scope
Eavesdrop Context Database	Policy management secrecy	Policy management requests shall be visible only to authorised users.	Policy Manager	Y	Policy management calls are sent over TLS
Eavesdrop Context Manager	Policy management secrecy	Policy management requests shall be visible only to authorised users.	Policy Manager	Y	Policy management calls are sent over TLS
Eavesdrop Policy Manager	Policy management secrecy	Policy management requests shall be visible only to authorised users.	Policy Manager	Y	Policy management calls are sent over TLS
Eavesdrop request enforcement channel	Secret request enforcement channel	Request enforcement channels shall be visible only to authorised users.	PZP Session Handler, Policy Manager	Y	Access requests come either over the overlay network (which is only served using TLS) or between the widget renderer and PZP using a secure websocket. However, there is the potential for man-in-the-browser attacks.

Eavesdrop RPC Call Log	Secret RPC Call Log	RPC call logs shall be visible only to authorised users.	PZP Session Handler, Context Manager	N	Misusing the secrecy of the context database can be misused for this purpose. However, context logging is turned off by default and must be requested with permissions.
findService API reveals permanent user identifier	Pseudonymous API user identifier	The output of the findServices API shall be a temporary pseudonymous identifier.	Discovery Manager	N	Pseudonymous identifiers are currently not supported.
Forgotten credentials	Memorable credentials	webinos user credentials shall be memorable.	PZH Provider	N	The creation of memorable credentials for managing PZHs is not mandated.
Installed App allows unrestricted postMessage	Installed app postMessage non-repudiation	Installed applications shall disallow postMessages from unrecognised origins	PZP Session Handler	Y	Origins can be verified on Messaging.
Installed App allows unrestricted XHR	Installed app XHR non-repudiation	Installed applications shall disallow XHRs from unrecognised origins.	PZH Provider, PZP Session Handler	N	While use of W3C WARP or Mozilla's Content Security Policy can achieve this non-repudiation, this is not explicitly supported by webinos.
Installed app exploited	Application use authenticy	An installed application shall be controlled only by authorised users and applications.	Widget Manager	N	Verifying the authenticity of prospective malware is out of scope.

Installed App given API permissions	Personal data unauthorised to trusted apps	Installed apps shall be unable to access personal data.	N/A	N	Controlling whether or not webinos applications may be trusted with personal data is out of scope.
Installed app misbehaving	Installed application behaviour cannot be interfered with	The behaviour of installed applications shall not changed based on outside influence or attackers.	PZH Provider, PZP Session Handler, Policy Manager, Discovery Manager	N	Protecting the integrity of the application packages and isolation / freedom from influence by other entities is not currently supported.
Installed App uses unauthenticated webinos event messages	Installed app message non-repudiation	Installed applications shall verifiy the authenticity of event message origin.	PZP Session Handler	Y	We can rely on the Events API read-only "from" field that is written only by PZP.
Installed App vulnerable to content injection	Installed app content injection tests	Installed applications shall be resistant to content injection attacks	Widget Renderer	N	Control over browser based widget renderers is out of scope.
JavaScript injection overwrites webinos.js	Immutable webinos modules	webinos code modules shall be non-modifiable by webinos applications.	Application Client	N	There are no File API restrictions for accessing webinos code modules
JavaScript injection	Verified webinos.js	The integrity of webinos.js	Widget renderer	N	Verifying webinos.js is out of scope while webinos supports

overwrites webinos.js		is verified by the widget renderer			web browsers and web apps must include their own webinos.js file. Having a common include path for webinos.js may be an improvement, or replacing this approach in a browser model.
JavaScript injection triggers security violation	Prevent javascript from other domains	Prevent malicious entities from injecting javascript into web applications	Widget renderer	N	Preventing hosted applications from being vulnerable from JavaScript injection is out of scope. Various approaches can protect against attacks on the client side, such as CSP restrictions, but these are not implemented.
JavaScript injection triggers security violation	Verified injected javascript	Injected javascript running within webinos components is verified.	Widget renderer	N	Verification against code injection is not currently implemented for webinos.js and javascript from other domains.
Malicious App installed	Verified application installation	webinos applications shall be verified before installation.	Widget Manager	Y	widgetprocessor.js performs the signature validation (the called validation code is in widgetvalidator.js)
Malicious background application installed	Verified application installation	webinos applications shall be verified before installation.	Widget Manager	Y	widgetprocessor.js performs the signature validation (the called validation code is in widgetvalidator.js)
Malicious App misuses	Installed app communication verification	Installed applications shall verify	PZP Session Handler	Y	Application could read the "from" field of the received event and attest it.

communication interface		communication to webinos applications.			
Malicious background application running	Verified background application	The behaviour of background applications shall be verified.	PZP, Widget manager	N	While we can recommend the use of app stores with verified applications and suggest the setting of sensible default policies, satisfying this goal is out of scope.
Malicious code evaluated	Malicious code unevaluated	Event handlers shall process only non-executable JSON content.	Application Client, PZP Session Handler	N	There are no restrictions within event handlers for executing JSON.
Malicious NDEF tag	Trusted NDEF message content	Processed NDEF messages shall contain trusted code.	NDEF Manager	N	There are no restrictions planned on how NDEF messages shall be processed within webinos.
Malicious plugin not detected	Plugin verification	Plugins shall be verified before user installation.	Widget Renderer	N	Control over browser based widget renderers is out of scope.
Malware installed	Verification before installation	Software shall be verified before it is installed by end users.	Widget Manager	N	While we can recommend the use of app stores with verified applications, satisfying this goal is out of scope.
Missing Access Request validation	Access request validation	Access requests shall contain a validating DTD or schema	Policy Manager	Y	Currently being implemented for the second phase.
Missing Policy file	Policy file validation	Policy files shall contain a validating	Policy Manager	Y	Currently being implemented for the second phase.

validation		DTD or schema			
Native malware running	Native malware not running	webinos software shall be isolated from native malware running on a device.	N/A	N	webinos makes no assurances about a potentially compromised platform.
Non-mandated request	Mandated request	Access requests shall be non-repudiable.	Policy Manager	Y	All access requests are served between PZPs over TLS
OpenID provider offline	OpenID provider online	PZH administrative access shall be possible only if the OpenID provider is online.	OpenID Proxy	Y	If the OpenID provider is offline it is not possible to carry out the OpenID authentication
Overrestricted resource	Restricted resource	Resource restrictions shall be commensurate with their specifications.	Policy Manager	Y	Each request is enforced separately by definition. Hence, it's impossible to grant access to more resources than required.
Unrestricted resource	Restricted resource	Resource restrictions shall be commensurate with their specifications.	Policy Manager	Y	Each request is enforced separately by definition. Hence, it's impossible to grant access to more resources than required.
Overwrite valid authentication data	Authenticate authentication data changes	Re-authentication shall be necessary to update authenticatio	PZH Provider	Y	Need to authenticate to the hub to change synchronised policies

		n data.			
Overwrite valid PZP Configuration	Authenticate PZP changes	Re-authentication shall be necessary to update PZP configuration data	PZP Session Handler	Y	Every time PZP has update configuration data, it will be treated as a new PZP to authenticate
Password guessed and reset	Password brute force resistance	OpenID credentials used for personal zone authentication shall be resistant to brute force attacks.	OpenID Proxy	N	OpenID credentials' details, which are a secret shared between OpenID providers and users, are out of scope.
Password recovery process attacked	Password recovery resistance	The reset of personal zone credentials shall be isolated from the OpenID account recovery procedure.	OpenID Proxy	N	OpenID account recovery procedure is out of scope.
Permission prompt click-through	Click-through controls	Permission prompts shall prevent dialog click-through.	PZH Provider	Y	Based on a proposed mock-up, click-throughs will not result in a default policy.
Policy misconfigured	Policy validation	Policy settings shall be user validated before use.	Policy Manager	N	Supporting the user validation of policy files is out of scope.
Post spoofed message	Message non-repudiation	Event messages shall be non-	PZP Session Handler	Y	Message 'from' fields are filled in by the PZP, which is able to authenticate the

		repudiable.			source of each message.
Spoof message origin	Message non-repudiation	Event messages shall be non-repudiable.	PZP Session Handler	Y	Message 'from' fields are filled in by the PZP, which is able to authenticate the source of each message.
Spoof network settings message origin	Message non-repudiation	Event messages shall be non-repudiable.	PZP Session Handler	Y	Message 'from' fields are filled in by the PZP, which is able to authenticate the source of each message.
Post spoofed message	PZP message authenticity	PZP shall authenticate the origin of messages it sends.	PZP Session Handler	Y	PZPs add the 'from' field to messages based on the sender who has been authenticated through the webinos PKI
Spoof message origin	PZP message authenticity	PZP shall authenticate the origin of messages it sends.	PZP Session Handler	Y	PZPs add the 'from' field to messages based on the sender who has been authenticated through the webinos PKI
Spoof network settings message origin	PZP message authenticity	PZP shall authenticate the origin of messages it sends.	PZP Session Handler	Y	PZPs add the 'from' field to messages based on the sender who has been authenticated through the webinos PKI
PZH Admin URL displayed without prominence	PZH admin URL displayed prominently	The PZH admin URL bar is visible on supported webinos device platforms.	PZH Provider	N	While a customised browser could present additional GUIs which authenticate the PZH to the user in a more visible way, control over browser displays is out of scope.
PZH Admin URL not well known	PZH Admin URL well known	The PZH Admin URL shall be recognisable to users.	PZH Provider	N	While recommendations can be made for how admin URLs should be formatted, we cannot control the format of URLs, which users are bad at

					parsing.
PZH Admin URL too complicated	Human-readable PZH admin URL	PZH admin URLs shall be human readable.	PZH Provider	N	We cannot control the format of URLs but the specifications should recommend their format.
Revoke device from valid personal zone	Device revocation authentication	Re-authentication shall be necessary to revoke devices from personal zone.	PZH Provider	Y	Authentication carried out via the PZH admin interface.
Run multiple personal zone proxies	Single personal zone proxy	Running more than a single personal zone proxy on a device shall be disallowed.	PZP Session Handler	N	There are currently no explicit checks for instances of multiple PZP configuration data on a single device.
SMS intercepted and relayed	Messaging authentication	Messaging API users shall be authenticated before API use.	Policy Manager	Y	This can be configured using the policy framework, but this is not the default.
Test API enabled	Test API disabled	Test APIs shall be disabled in installation environments.	Widget Manager	N	There are no supported means for distinguishing test APIs from those which are officially supported. We do, however, intend to impose a naming scheme that will enable the disabling of test API's in installation environment, either when packaging/building webinos or on install time.
Test configuratio	Test configuration	Test and sample	Widget	N	There are no supported means for distinguishing test

n enabled	disabled	configuration data shall be disabled in installation environments.	Manager		and sample configuration data from valid platform or application data. But we intend to impose a naming scheme that will enable the disabling of test and sample configuration in installation environment, either when packaging/building webinos or on install time.
Unrestricted request specification	Restricted request specifications	User agent requests to network resources shall be restricted.	Policy Manager	Y	Requests to access network resources are represented by the feature http://webinos.org/action/network-access
Unspecified resource	Specified resource	Access requests shall specify the resources requiring authorisation.	Policy Manager	Y	Policy manager permits or denies access to resources only after matching the requested feature list against policies. Therefore all required resources must be specified in the requests. However, resources that don't have associated resources cannot be directly controlled by the policy manager
User clicks on link within application	Unspoofable PZH admin URLs	PZH admin page URLs shall be non-spoofable	PZH Provider	N	Control over URLs visited by browsers is out of scope.
Webinos backdoor	Webinos backdoor tests	webinos platform shall test for the presence of backdoor procedures.	N/A	N	Testing for the presence of backdoors is not currently within scope. Gatekeeping processes are currently being considered for change requests which will consider the potential for such backdoors being introduced.

Webinos widget processor bug	Widget processor verification	Webinos widget processors shall be verified before release.	Widget Manager	N	Widget processors are not part of the webinos system specification.
Widget renderer supports extensions	Authorised widget renderer extensions	Widget renderers shall support only authorised extensions	Widget Renderer	N	Security of the widget renderers is out of scope.
XSS attack on hosted app	Hosted app XSS resistance	Hosted applications shall be resistant to known XSS attacks.	Widget Renderer	N	Security of hosted applications is out of scope.

2.5. Weakness Analysis

To validate the results of the attack and ambiguity analysis, the below table summarises the mitigations in place within the webinos architecture for addressing related threats or vulnerabilities.

Architectural pattern	Threats	Vulnerabilities	Mitigation Summary
Context Policy Management	Locate and Exploit Test APIs, Man-in-the-browser attack, Mobile Phishing (aka MobPhishing), Network Eavesdropping	Allocation of Resources without Limits or Throttling, Insufficiently Protected Credentials, Missing XML Validation	TLS connections between components mitigates network eavesdropping. Message authentication and default policy design mitigates mobile phishing. Plans to implement a naming scheme re: test data to help mitigate man-in-the-browser attacks.
NFC	NFC replay attack	Improper Preservation of Permissions, Information Exposure Through	Attack mitigating by re-authenticating before changing personal zone

		Persistent Cookies, System data trust	information.
PZH Authentication	Denial of Service, Pharming	Inclusion of Functionality from Untrusted Control Sphere, UI Misrepresentation of Critical Information, Unverified Password Change	Application signing key verification and the checking of application signing keys helps mitigate DoS.
PZP Enrolment	None	None	None
TV Service Discovery	Locate and Exploit Test APIs, Man-in-the-browser attack, Mobile Phishing (aka MobPhishing), Network Eavesdropping, NFC replay attack	Allocation of Resources without Limits or Throttling, Improper Preservation of Permissions, Information Exposure Through Persistent Cookies, Insufficiently Protected Credentials, Missing XML Validation, System data trust	TLS connections between components mitigates network eavesdropping. Message authentication and default policy design mitigates mobile phishing. Plans to implement a naming scheme re: test data to help mitigate man-in-the-browser attacks.
Widget Processing	Denial of Service through Resource Depletion, Locate and Exploit Test APIs, Man-in-the-browser attack, Mobile Phishing (aka MobPhishing), Network Eavesdropping	Allocation of Resources without Limits or Throttling, Improper Verification of Cryptographic Signature, Insufficiently Protected Credentials, Missing XML Validation, Uncontrolled Resource Consumption ('Resource Exhaustion')	TLS connections between components mitigates network eavesdropping. Message authentication and default policy design mitigates mobile phishing. Application signing key verification and the checking of application signing keys helps mitigate DoS. Plans to implement a naming scheme re: test data to help mitigate man-in-the-browser attacks.
Widget Rendering	Malicious Automated Software Update	Inclusion of Functionality from Untrusted Control Sphere	None

2.6. Data Security

The webinos platform uses and stores data which may have security and privacy requirements. For example, many of our personas may use webinos applications to monitor health data, to read personal emails, or to store valuable work files. As such, it is important to address threats and mitigations to vulnerabilities affecting data at rest.

Some of this analysis has been included in the main Architectural Risk Analysis process, but some still remains separate.

2.6.1. Where and what data are stored in webinos?

2.6.1.1 Applications (The File API)

Applications may store data locally on each device, as well as using data (such as media files) exposed by each device. To support this, webinos provides the File API. The File API will expose to each application an application-specific, isolated storage area. In addition, the File API can also expose arbitrary data storage. However, access to arbitrary data storage will be mediated by policies and require a different permission. Isolated storage from one application is never exposed to another through webinos.

2.6.1.2 Local devices and PZPs

Each device with a PZP will store some or more of the following:

1. Application data
2. Data in policies, certificates, and preferences. This may include the names of applications the user has installed, the devices they use and their friends identities. It is therefore considered private.
3. Browser histories and system logs
4. Context data (a temporary log file), if enabled.
5. Downloaded widget data containing potentially valuable intellectual property

2.6.1.3 Cloud-based components (PZH, online services)

Cloud-based components may store:

1. Context data
2. Data in policies, certificates, and preferences. This may include the names of applications the user has installed, the devices they use and their friends identities. It is therefore considered private.
3. Application data (outside of webinos control)

This is discussed in more detail in the cloud security analysis section.

2.6.2. What are the threats?

Two obstacles in the existing architectural analysis are relevant:

- "Application data intercepted" - Application data from webinos widgets is intercepted in the widget renderer.
- "Application data readable" - Application data from webinos widgets is readable outside of the widget renderer.

In addition, the following threats exist:

Threat	Description	Attackers	Attacker Motivation
Native malware	Malware is installed on webinos-enabled device and is used to access and transmit application data to a third party. This may be targeted to particular apps or users	Irwin	Corporate espionage or discrediting an existing application. Monetary gain.
Device theft	A webinos-enabled device is stolen and data is downloaded by the thief.	David	Most likely selling the device, but this could be a targeted attack on an individual or corporation
webinos malware	A malicious webinos application accesses user data	Ethan, Frankie	Stealing personal data for personal or monetary gain, may be looking for credentials or credit card details.
Online data leak	A PZH provider exposes their entire file system by mistake. This compromises the certificates, keys and settings of each user	Frankie, Gary	May be discrediting former employee, or may be looking for recognition from the user community
App content theft	A webinos widget data is stolen by another developer	Jimmy	Monetary gain - copying valuable IPR
Data removal blackmail	User data is encrypted and the key held by an attacker. The user is extorted to get back their personal data	Ethan	Monetary gain

2.6.3. Mitigations

2.6.3.1 Webinos-provided mitigations

The webinos platform mitigates some of these threats in the following ways:

Threat	Mitigation
Native malware	No mitigation
Device theft	No mitigation
webinos malware	Provide isolated storage for each application, require additional permissions to access shared areas
Online data leak	Provide recommendations for PZH design to minimize risk, recommend key storage approaches, minimize data stored by the PZH
App content theft	No mitigation
Data removal blackmail	No mitigation

2.6.3.2 Device-provided mitigations

We suggest the following mitigations should be provided by webinos device platforms.

Threat	Mitigation
Native malware	Provide anti-malware tools and allocate each native application in its own private storage area
Device theft	Provide full disk encryption
webinos malware	N/A
Online data leak	PZH providers should provide disk encryption and should follow best practice guidelines to avoid vulnerabilites. Keys should be stored privately using either a trusted hardware device or a separate file system.
App content theft	No mitigation
Data removal	Offer backup and recovery tools.

blackmail		

As this table demonstrates, the majority of threats we suggest mitigating at the device level, not the webinos level.

2.6.3.3 Current situation

At present, the following webinos platforms provide some of the aforementioned mitigations

2.6.3.3.1 Windows

Threat	Mitigation	Provided
Native malware	Anti-malware tools, isolated application storage	Anti-malware tools exist. No isolated storage is available.
Device theft	Disk encryption	Disk encryption tools exist.
Data removal blackmail	Backup	Backup solutions exist

2.6.3.3.2 Linux

Threat	Mitigation	Provided
Native malware	Anti-malware tools, isolated application storage	Few anti-malware tools. Isolated storage can be implemented through Linux Security Modules such as SELinux.
Device theft	Disk encryption	Disk encryption tools exist in the main kernel.
Data removal blackmail	Backup	Backup solutions exist

2.6.3.3.3 Android

Threat	Mitigation	Provided
Native malware	Anti-malware tools, isolated application storage	Anti-malware tools exist. Isolated storage provided by default.
Device theft	Disk encryption	Disk encryption tools exist on Android ICS and above.
Data removal blackmail	Backup	Backup solutions exist

2.7. Summary

We have analysed the webinos architecture based on 14 attack patterns motivated by the findings of Deliverable 2.7 and 2.8, as well as ongoing development and emerging security and privacy issues. This provides a substantial range of attacks and obstacles to consider in the architecture. Indeed, all of the leaf obstacles obtained during this process have been included in the analysis and shown to either be satisfied by a goal supported by the architecture or shown to be lacking a solution at present. Our process is sufficiently rigorous that we are now able to draw several conclusions about the design of the architecture. However, more attack patterns and architectural patterns should be developed in the coming months in order to provide a more complete analysis and to prioritise future development of security functionality. It is also important to note that this analysis refers primarily to the *specification* of the system rather than the implementation. We have not analysed the level in which the specifications are implemented, nor have we considered the strength of the implementation against known code-level vulnerabilities.

Based on the findings of the attack-resistance, weakness and ambiguity analysis, we have made the following conclusions.

2.7.1. Application rendering

At present, our architecture does not fully address attacks on content rendered by widget renderers and browsers. This leaves webinos and webinos applications vulnerable to attacks such as content injection (cross site scripting, for example). This is primarily due to the re-use of existing web browsers as opposed to customising a browser with a webinos-specific extension. Customisation would allow additional rules and restrictions to prevent some of the more prevalent and likely attacks.

2.7.2. Comparative exposure of personal zone hubs and proxies

Based on the quantitative analysis of the attack surface in the ambiguity analysis, the webinos attack surface associated with setting up personal zone hubs is greater than the surface exposed when enrolling devices to personal zones; this is largely due to the attack surfaces of the assets associated with each architectural pattern rather than the damage potential associated with component interfaces or connectors. While a lot of emphasis has been placed on the device-side webinos architecture, the webinos hub-side architecture is less well understood with undefined surface types for the administration console and the potential for rendering engine weaknesses compromising traffic to/from a personal zone hub.

2.7.2.1 Reliance on OpenID authentication

The webinos platform is novel in its approach to authentication: we introduce no new passwords or usernames into the system and delegate most authentication tasks either to the underlying operating system or to an OpenID provider. However, this puts many mitigations out of scope, and the system is at risk when a weak or insecure OpenID provider is used. We believe this to be a reasonable approach, but it also means that webinos personal zone hub providers ought to limit the number of OpenID providers they support for security reasons, despite the negative impact on interoperability.

2.7.2.2 Policy management remains crucial

The policy-based access control system remains an important aspect of webinos and is responsible for satisfying at least 15 goals from the ambiguity analysis. This means that additional effort should be spent making sure that policy features are implemented in the platform, and that appropriate tools and user interfaces are provided. It is also true that default policy settings will be important to mitigate several obstacles, and these may need to be improved based on experience and the findings from Deliverable 2.8.

2.7.2.3 Privacy and service discovery

A privacy issue that has been discovered throughout the analysis is the use of the `findService` call, which returns persistent device identifiers and could potentially be used to re-identify the user, breaking unlinkability requirements. This problem has been reported to the rest of the project, and motivates further improvements and investigations of the Web Intents alternatives, as discussed in Deliverable D3.3.

2.7.2.4 Protecting webinos source code

The current webinos development process must protect users against the potential for a malicious contributor from adding back-door vulnerabilities to the platform. While this is solved (to some extent) through the gate-keeping implemented as part of the GitHub development process, further improvements are needed. In particular, creating a larger distinction between tests and platform code and making it easy to remove testing code from deployments of the platform.

A related issue is that of code quality and input sanitisation. Any communication with the PZP should be thoroughly checked and validated against a provided schema to prevent malicious code injection exploiting the runtime. This should be part of the requirements for accepting new code into the webinos source code repository.

2.7.2.5 Denial of service issues

Several of the attack patterns we considered in this analysis relate to denial of service issues. We believe that the architecture mitigates denial of service attacks in the following way:

1. Because the PZP does not rely on having a PZH available for local functionality, loss of the PZH doesn't prevent use of the PZPs. The PZH being made unavailable is, therefore, less important in many situations than a PZP being made unavailable. Indeed, the PZH is not required for local, peer-to-peer interaction between devices.

2. Support for downloadable widgets means that application can be used offline, and can be designed to be resistant to loss of network connectivity. Installing applications does not require an internet connection.
3. Process isolation between the web browser or widget renderer and the PZP should make the PZP harder to attack directly.
4. Restrictions on widget content should make content-injection DoS attacks harder to perform.
5. The use of long-term certificates means that credential expiration is not an availability issue.

However, we remain aware of the following issues with respect to availability:

1. The policy system is vulnerable to misconfiguration, which could make PZPs unable to communicate.
2. The availability of the OpenID provider is important, but is out of the control of webinos.
3. Attacks on platform integrity from malware is not something we can mitigate.
4. API-specific resource exhaustion attacks may be a problem. These are discussed in the API Security and Privacy Analysis section.

2.7.2.6 Data storage

The security of data used within webinos is largely dependent on the devices running webinos, and therefore is largely outside of the scope of the webinos architecture. The webinos implementation has avoided providing additional data encryption tools on grounds that most platforms already provide these options. In most cases, users have the tools available to protect themselves. The outstanding threats include:

- Protecting valuable IPR stored within downloaded widgets. A solution to this problem would require Digital Rights Management. This is hard to implement across multiple platforms and is no longer within the scope of webinos.
- Isolated data storage on windows and Linux. Native malware may be able to access webinos data as it is not protected on these systems by default.
- PZH implementations may have varying levels of security from online attacks. We refer to the cloud security section to provide more details on potential solutions.

3 Threat Model

3.1. Sources of Threat Data

We have used the following sources of data to identify threats, vulnerabilities and attacks that may relate to webinos:

3.1.1. CAPEC

The Common Attack Pattern Enumeration and Classification (CAPEC) is a database of methods for attacking systems based on the style of *Design Patterns*. Attack Patterns try to be sufficiently abstract that they can be translated between different systems. For webinos, the CAPEC database has been imported into the CAIRIS tool and used as a source of *threats*. The webinos attack patterns are defined with the additional details of the webinos architecture but are often inspired and closesly related to CAPEC attack patterns. (CAPEC)

3.1.2. CWE

The Common Weakness Enumeration (CWE) provides a "unified, measurable set of software weaknesses". The CWE database of vulnerabilities was added to the CAIRIS tool and aligned with attack patterns. (CWE)

3.1.3. OWASP

The Open Web Application Security Project (OWASP) provides a wealth of data on attacks and threats to web applications. Dozens of threats and attacks have been added to the CAIRIS tool based on the OWASP project and then aligned with attack patterns. (OWASP)

3.2. Threat Model and Trusted Components

In this section we describe the activities and actions that each of the core webinos components are trusted or explicitly *not* trusted to perform. The value of this threat model is that it makes explicit the assumptions behind which components we are relying on and, therefore, how each may be exploited to attack the overall system. The threat model helped to inform the attack patterns and obstacles described in the risk analysis.

This trust model is taken from the perspective of a webinos personal zone user.

The following components are covered in this model:

- Personal Zone Proxy (PZP)
- Personal Zone Hub (PZH)
- Identity Provider
- Name resolution service

- Platform: device + OS + other applications
- Widget Renderer
- Browser
- Widget Processor
- Applications

A complete list of potential actions and activities is given in the appendix. Some actions are generic and therefore a *modifier* may exist to explain what this action refers to in this context. For example, "Storing data - confidentiality" may have the modifier "policies" referring to the fact that this component is trusted to store policies confidentially. This is represented as a separate column.

3.2.1. Personal Zone Proxy (PZP)

Users need to trust the PZP to have almost complete control of the local device, and to access all *local* APIs it provides. However, as designers should be wary of trusting every PZP absolutely. The inability of a PZP to protect itself against malware makes the possibility of a zone being hijacked by a rogue PZP extremely likely. It should also not be overly trusted for authenticating the end user: different devices will have different capabilities with regards to authenticating users.

There is also a significant threat from a device being joined into a malicious personal zone without the user's knowledge. This is why it is trusted to maintain the integrity of certificates and policies.

3.2.1.1.1 Trusted for

Action	Modifier	Description and rationale
Access control via policy enforcement		The PZP is the main area for policy enforcement
Storing data - confidentiality	policies, app data	The PZP must be able to store policies without outside modification
Editing XACML policies (local)		The PZP will need to make changes to XACML policies based on user input
Handling device keys		Passing keys to the operating system to store
Storing data - integrity	policies, app data, PZP and PZH certificates	PZPs could be attacked by changing their certificate hierarchy
Authenticating devices against device certificates		PZPs may authenticate other PZPs and PZHs via their public key certificates
Displaying information to the local user		PZPs may present information to the user for policy enforcement or installation reasons

Taking user input	Policy decisions, installation	PZPs need to take access control decisions from users
Routing	To the PZH	PZPs route data from applications to the PZH and peer-connected PZPs
Creating sessions	To the PZH, other local PZPs	PZPs establish TLS connections to other PZHs
Authenticating users to local device		PZPs call operating system methods to authenticate users to the local device, thus implementing the Authentication API
Untrusted Input validation	incoming API requests and messages	PZPs must validate all input from web runtimes and browsers
Authenticating the widget runtime and web browser		PZPs must be able to authenticate their input to protect against unauthorised local software
Name resolution	Applications, the PZH	PZPs need to resolve the names of applications and the PZH, which will be hosted on a web server

3.2.1.1.2 Sometimes trusted for

Action	Modifier	Description and rationale
Authenticating user's online identities		The PZP may be configured as a "blessed" device, where no online authentication is needed.

3.2.1.1.3 Explicitly not trusted for

Action	Modifier	Description and rationale
Adding personal zone devices		PZPs should not be able to add new PZPs to a personal zone as they do not hold zone-wide CA keys
Maintaining platform integrity		Delegated to the underlying platform

3.2.2. Personal Zone Hub (PZH)

The Personal Zone Hub is trusted with some of the most important aspects of the personal zone. It is not given access to individual device private keys (this allows any device to implement its own storage and generation mechanism) or allowed to edit local policies on individual devices, but almost everything else is permitted.

3.2.2.1.1 Trusted for

Action	Modifier	Description and rationale
Access control via policy enforcement		
Adding personal zone devices		
Authenticating devices against device certificates		
Authenticating identity providers		
Authenticating user's online identities		
Certificate and signature processing		
Content isolation		
Creating sessions		
Displaying information to the local user		
Editing XACML policies (zone-wide)		
Identifying applications		
Protect itself against runtime attack		
Revocation of devices		
Routing		
Storing data – integrity	settings and certificates	
Storing data – confidentiality	settings and certificates	
Storing keys		
Taking user input		
Untrusted Input validation		

3.2.3. Sometimes trusted for

Action	Modifier	Description and rationale
Storing data –	Yes: settings and certificates. No:	The PZH is not used to store

integrity	application & context data	application data or context data
Storing data – confidentiality	Yes: settings and certificates. No: application & context data	The PZH is not used to store application data or context data

3.2.3.1.1 Explicitly not trusted for

Action	Modifier	Description and rationale
Editing XACML policies (local)		There are local device policies the PZH is not trusted to edit
Handling device keys		The PZH never receives the private keys created by PZPs

3.2.4. Identity Provider (OpenID Provider)

We *require* that the user has a trustworthy identity provider. However, the webinos framework has no way of enforcing this, and therefore must trust that users select an appropriate option. In addition, we have no mitigation for attacks resulting from flaws in OpenID

3.2.4.1.1 Trusted for

Action	Modifier	Description and rationale
Authenticating user's online identities		
Authenticating web domains		
Protect itself against runtime attacks		

3.2.4.1.2 Explicitly not trusted for

Action	Modifier	Description and rationale
Maintaining sessions	User sessions to the OpenID provider	
Authenticating users to the local device		

3.2.5. Name resolution service (WebFinger / User lookup service)

The name resolution service converts user identities into personal zone hub URLs. We assume that the name resolution service is either provided by a well known authority (such as webinos.org) or the identity provider themselves.

3.2.5.1.1 Trusted for

Action	Modifier	Description and rationale
Name resolution	User email addresses to PZH URLs	This is the primary function of this entity
Authenticating user's online identities		Updating routing information requires authorisation from the correct user
Revocation of devices	PZH, not PZPs	The name resolution service is used to revoke a PZH by linking to a different location

3.2.6. Platform: device + OS + other applications

The webinos PZP software will run on an existing platform, including an operating system and applications. This is the source of many potential attacks, particularly related to malware. Device operating systems and applications are responsible for protecting themselves against malware. However, this is often shown not to be reasonable as new malware and vulnerabilities are published frequently. As such, a key threat is the presence of malware on a device and therefore on webinos.

3.2.6.1.1 Trusted for

Action	Modifier	Description and rationale
Process isolation	Isolation of the PZP from other applications, Isolation of two PZPs on the same platform	
Storing keys		webinos uses the key storage mechanisms provided by the platform
Authenticating users to local device		webinos calls local authentication mechanisms to implement the Authentication API
Storing data – confidentiality	Application data, webinos platform data	
Protecting itself against runtime attacks		Protecting against native malware

3.2.6.1.2 Explicitly not trusted for

Action	Modifier	Description and rationale

Authenticating user's online identities		The platform has no way of authenticating the user online

3.2.7. Widget Renderer

The widget render displays widgets to the end user. They may be vulnerable to a wide range of attacks from malicious users or applications, including:

- Shoulder-surfing attacks
- Widgets could be designed to exploit the renderer code, either by modifying what is rendered or by launching attacks on the underlying system
- XSS, CSRF, session hijacking

3.2.7.1.1 Trusted for

Action	Modifier	Description and rationale
Rendering applications		This is the Widget Renderer's primary task
Communicating with the local PZP		Establishing a secure connection with the PZP
Taking user input		
Displaying information to the local user		
Authenticating web domains		Checking SSL domain certificates for hosted applications
Enforcing cross-origin restrictions	Enforcing WARP	
Certificate and signature processing		Authenticating the PZP
Identifying applications		Being able to identify the application being rendered is essential for maintaining a trusted path
Sandboxing applications		The widget renderer does not expose other APIs to widgets except for those defined in webinos

3.2.7.1.2 Explicitly not trusted for

Action	Modifier	Description and rationale
Access control via policy enforcement		Delegated to the PZP

3.2.8. Web Browser

Applications in webinos may be displayed from within a web browser. These are vulnerable to a wide range of attacks, similar to a widget renderer.

3.2.8.1.1 Trusted for

Action	Modifier	Description and rationale
Rendering applications		This is the Widget Renderer's primary task
Communicating with the local PZP		Establishing a secure connection with the PZP
Taking user input		
Displaying information to the local user		
Authenticating web domains		Checking SSL domain certificates for hosted applications
Enforcing cross-origin restrictions	Enforcing same-origin policies and CSP restrictions	
Certificate and signature processing		Authenticating the PZP
Identifying applications		Being able to identify the application being rendered is essential for maintaining a trusted path
Sandboxing applications		The browser should not expose other APIs to widgets except for those defined in webinos

3.2.8.1.2 Explicitly not trusted for

Action	Modifier	Description and rationale
Access control via policy enforcement		Delegated to the PZP

3.2.9. Widget Processor

Widget processors unpack widget packages and install them onto the personal zone. They must be able to validate signatures and maintain records of trusted applications. A widget processor may be part of a widget runtime environment (WRT).

3.2.9.1.1 Trusted for

Action	Modifier	Description and rationale
Validating application content		Checking that application signatures are valid
Authenticating users to local device		Authentication may be needed for installing applications
Certificate and signature processing		Processing signatures for applications
Displaying information to the local user		Presenting install screens
Installing web applications		This is the primary function of a widget processor
Untrusted Input validation		Protecting against malformed application packages
Editing XACML policies (zone-wide)		Changing XACML policies based on application installation

3.2.10. Applications

Applications are generally not trusted with user data or APIs unless granted explicit permission. They can be a significant attack vector, as well as being attacked themselves.

Applications can be attacked through:

- Content injection attacks such as XSS and CSRF
- Code could be stolen by competing developers
- Content could be stolen
- Any tokens or secrets contained within the application may be stolen

Applications can be attack vectors:

- Stealing user data
- Privacy violations - monitoring, tracking
- Acting as a botnet client
- Denial of service attacks on the end user
- Exploiting vulnerabilities in the renderers

We have not tried to account for attacks on content within the application.

3.2.10.1.1 Trusted for

Action	Modifier	Description and rationale

Untrusted Input validation		Applications may receive input from many untrusted sources
Storing data – confidentiality		Applications may store confidential data on remote servers
Storing data – integrity		Applications may store high-integrity data on remote servers
Self-imposed least privilege		Applications must only request permissions they use to avoid being misused
Maintaining sessions		Any sessions being established between the client and the hosted page must be maintained
Fulfilling data handling policies		Applications may store private data on remote servers

3.2.10.1.2 Sometimes trusted for

Action	Modifier	Description and rationale
Installing web applications		Some applications are allowed to install child applications
Correct API use (after permission has been granted)		Some APIs may be vulnerable to misuse by applications, but most will impose rate limits
Protecting itself against runtime attacks		The hosted portion of an application must resist attacks, the local is not expected to

3.2.10.1.3 Explicitly not trusted for

Action	Modifier	Description and rationale
Access control via policy enforcement		Applications must gain permission through the user granting access
Authenticating users to local device		Applications have no way of directly authenticating the user to webinos, they may trust that webinos does it for them.
Authenticating user's online identities		Applications are not (by default) granted access to user identity information
Maintaining platform integrity		Applications may be actively trying to modify the underlying system
Editing XACML policies		

(local)	
Editing XACML policies (zone-wide)	
Validating application content	Applications may be corrupted

4 API Security and Privacy Analysis

4.1. API Analysis Methodology

The API Security and Privacy analysis was carried out using the following methodology and process.

Several partners were invited to review APIs, either due to their security expertise or their involvement with the development of the API and therefore insights into potential threats. These partners included Telecom Italia, Polito, BMW F&T, Sony, and The University of Oxford. Most analysis data was reviewed by another partner before it was considered finished. Participants were invited to fill in the template given below while studying the API specification carefully. The template provided suggestions as to the personas, data sources and attack vectors to consider. Creativity was encouraged as part of this exercise.

The following two data sources were recommended:

- The Mozilla WebAPI security analysis (MozillaWebApi)
- The related W3C / WAC / Other "security and privacy" sections of specifications

Once the process had concluded, the findings were conveyed as a series of recommendations and reported to the API developers and rest of the specification and implementation teams. At the time of writing, several recommendations are still under consideration.

4.1.1. Analysis Template

4.1.1.1 Overview of API

Link to API - http://dev.webinos.org/specifications/new/example.html

Brief description of the API

4.1.1.2 Threats

4.1.1.2.1 API-Specific threats and misuse cases

1. These are the obvious threats that misuse of this API could cause for users.
2. I have elicited these by considering the assets that the API gives access to, as well as what happens if the API is excessively used.
3. I have taken into consideration several personas.
4. I have looked at the Mozilla approach for this/a similar API - (MozillaWebApi).
5. I have taken into consideration any "security and privacy considerations" section of related specifications.

4.1.1.2.2 Threats based on remote invocation of this API from another device

1. When this API is called from another device, what are the additional security concerns?
2. When this API is called from another user on another device, what are the additional security concerns?

4.1.1.2.3 Implementation threats and possible attacks

1. Based on my experience in implementation in phase 1, the following attacks might be possible...

4.1.1.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. How might a developer be caused harm by relying on this API?
2. How might an app be damaged or attacked through this API?

4.1.1.2.5 Threats to device manufacturers, operators, other stakeholders

1. How might other stakeholders be affected - for example, could it cause excessive traffic?

4.1.1.3 Mitigations

1. How should the API be designed, implemented or constrained such that the threats are mitigated?
2. Should the API be turned off by default?
3. Should the API require authorisation always?
4. Should the API result in some kind of notification or logging event?

4.1.1.4 Recommended default policy rules for applications

The following 'types' of application are supported in webinos:

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

For each application type, one of the below API default policies is recommended. This applies to an application which explicitly requests access to this API.

B-	W-	W-	Policy	Explanation (Widgets)	Explanation (Browser Apps)

A	R	U			
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

Several iterations of this table may exist if the API provides distinctly different methods or permissions.

4.2. API Security and Privacy Analysis Summary

The API security and privacy analysis makes the following suggestions which are not yet incorporated into the API specifications or webinos platform, including:

4.2.1. Investigate a more privacy-friendly way to implement the Discovery API

An issue also highlighted as part of the architectural risk analysis, the Discovery API is privacy-invasive because due to its use of persistant identifiers for webinos services; this facilitates user fingerprinting. For applications needing only relatively impersonal services, this API provides unnecessary information. We propose that web intents would be a potential alternative.

4.2.2. Provide an alternative interface to the Messaging API

For applications which require only visibility to certain incoming patterns and the need to send occasional messages, an alternative API should be available. The current API is more invasive than may be strictly necessary. A new API might include greater restrictions - such as only being able to send a message if it has first been viewed by the end user - but could also be allowed by default to applications.

4.2.3. Provide an alternative interface to the Calendar API

The Calendar API, like the Messaging API, exposes a great deal of information to requesting applications. This is unnecessarily dangerous for an application which requires only the free/busy status of an application rather than precise entry details.

4.2.4. Create a 'system level' and 'web app level' distinction

A final consideration for webinos is that there are two quite separate use cases considered by the APIs.

- Smaller applications requiring only slightly more functionality than that provided by the web browser. These are currently given more resources than they need by the Messaging, Calendar and Discovery API.
- System-level applications which are more trusted and need to have greater access to device features. More needs to do more to ensure their webinos runtime environment is secure and immune to attacks on any hosted components.

'web app level' apps should be allowed to run in the browser, and use a more privacy-friendly discovery system. At the same time, these apps should restrict access to some APIs completely.

'system level' apps should run exclusively in a widget renderer, but have the potential to access more APIs. We recommend these only be installed from app stores and must be pre-vetted.

5 Cloud Security and Privacy

5.1. Introduction

Cloud computing has emerged as a distributed system of choice for many use cases. As a utility, cloud platforms enable a pay-per-use basis, while their dynamic scalability attributes enable automated increase or decrease of resources for a particular service in response to increasing or falling demand, respectively. However, security and privacy remains one of the biggest challenges (Kandukuri-cloud-sec-09); as a result many organisations are reluctant in adopting cloud computing, especially when it comes to critical or sensitive applications.

Despite these challenges, significant strides have been made in the adoption of cloud computing as can be seen by an increase in cloud service offerings in the area of storage (Dropbox), communication and social networking. It is therefore impossible for webinos to completely ignore cloud computing; several applications running on webinos-enabled devices will use cloud services, others applications may be hosted on the cloud or indeed some components, such as the Personal Zone Hub (PZH), may be deployed in the cloud. For example, see the possible PZH deployment profile in the D3.3 informative specifications (Webinos-D33). However, in order for webinos to successfully take advantage of cloud computing or indeed work with services on the cloud, it is important that security and privacy implications are understood.

This section aims to meet this goal by investigating how the use of cloud computing could affect the security and privacy of webinos. More specifically, the section investigates the possible assets that could exist in a cloud-based PZH and how these are affected by common threats and vulnerabilities associated with cloud computing.

The investigations are based on the following scenario. *Alice* is a technology enthusiast who always wants to take advantage of new technology to make her life easier. She has been encouraged by cloud computing's promise of on-demand provision of resources, global accessibility and pay-per-use model. She recently signed up to Amazon Web Services and has now started using these services; she has so far managed to deploy a gaming application and streaming service. *Alice* has realised that there are some free compute cycles on her account and she has decided to use those for webinos. Depending on how this goes, she may decide to offer this as a service to her friends. However, before she can do that she needs to understand what the impact of this move on the security of her data and her privacy would be.

To help *Alice* decide on whether or not it would be ok to use cloud computing, the section begins with some background on general cloud computing issues including its definition and some of the top threats to its adoption. Following this, a security and privacy analysis is performed on a cloud-based Personal Zone Hub. This analysis identifies the assets that may exist in a cloud environment and then investigates how these assets are affected by some of the common threats identified in the background. The section then describes how these threats might be realised.

5.2. Background

Cloud computing has emerged as one of the hot topics in distributed computing. However, there is still confusion on how cloud computing is defined. Some suggest that it is a new architecture and approach, while others argue that it is simply a modification of the business models rather than a new computation paradigm. In order for the analysis to be useful, this section needs to make clear the view of cloud computing adopted; with the hope that this view will be shared with *Alice*. For this reason, the next section covers the definition and taxonomy of cloud computing before discussing the security challenges of using it.

5.2.1. Cloud definition, taxonomy and examples

Despite significant attempts at defining cloud computing, there is still no consensus on its definition. However, one commonly accepted definition is that from NIST (NistCloudDefinition2011):

> Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Central to this definition is the idea of on-demand provision of computational resources (CloudSecMather2009, CloudCompRittinghouse2010) so that users can start with a minimal set of resources, which can be dynamically increased or decreased depending on demand. This enables a pay-per use model where users only get to pay for the amount of resources used rather than for the ownership of those resources. To support this, cloud computing is designed with a number of characteristics.

5.2.1.1 Service Models

Cloud computing services can be provided using three main models:

- “Software as a service” (SaaS) - the customer does not purchase software, but rather rents it for use on a subscription or pay-per-use model
- “Platform as a service” (PaaS) - the vendor offers a development environment to application developers, who develop applications and offer those services through the provider’s platform
- “Infrastructure as a service” (IaaS) - the vendor provides the infrastructure to run the applications, but the cloud computing approach makes it possible to offer a pay-per- use model and to scale the service depending on demand.

Other less common used models include:

- “Communication as a service” - a subtype of Software-as-a-Service model, where providers are responsible for the management of hardware and software, e.g. VoIP, instant messaging, and
- “Security as a service” - a vendor offers security functionalities like e-mail and web content filtering, vulnerability management, etc.

5.2.1.2 Essential Characteristics

NIST identifies five key characteristics of cloud computing including:

1. On-demand self-service - users can request more services any time and the system used can provide these services automatically without requiring human interaction.
2. Broad network access - the services can be globally accessed using standard network mechanisms, enabling users to use different devices such as mobile phones, tablets, laptops, and workstations.
3. Shared resource - a multi-tenant model enables resources to be shared among a number of users while giving an illusion that each user owns the resource.
4. Rapid elasticity - the capabilities and/or capacity of resources can be increased or decreased automatically without affecting the availability of the services.
5. Measurable services - resources used can be transparently monitored and measured for the purpose of billing and monitoring of service-levels

5.2.2. Cloud security models

Security concerns about cloud computing include a mix of old and new threats, targeting software bugs, exploiting social engineering, and involving network security and access controls. These exist at different levels including: network, host and application. Sensitive data must be adequately secured at each level to avoid confidentiality and integrity breaches. Furthermore, proper access control, accountability identity and access management must be designed in order to achieve efficient procedures, mitigate complexity, improve user experience, and reduce errors and insecure short-cuts.

5.2.2.1 Cloud security - network

Like all network-based systems, cloud systems expose significant risk at network level. Data in transit to and from the cloud infrastructure may suffer from confidentiality and integrity problems resulting from activities by malicious agents along the path. Unauthorized access can also be problematic when cloud providers do not adopt effective IP address reallocation mechanisms (DHCPDynamicIPAddressing) that ensure that an IP address that was originally assigned to a resource that is no longer needed cannot be used to access that same resource. As a result of this, customers cannot assume that network access to their resources is terminated upon release of its IP address, opening the path to unwanted access.

Cloud system must also guarantee high availability. For example, BGP hijacking (BGPAttacks04) can affect the availability of cloud-based resources, while the use of external DNS querying exposes clouds to DoS attacks.

The use of cloud systems also introduces new and different security boundaries. The established model of network zones and domains is replaced with less precise and firm “security groups”, “security domains” or “virtual data centers”. Conceptually, this allows separation between tiers, but these need to be properly understood if security breaches and improper use is to be avoided.

Network security hardening is necessary to avoid common network attacks. For example, confidentiality and integrity can be assured by appropriate encryption and digital signature, network filters (e.g., firewall) should be shaped and managed by cloud providers. For the sake of accountability and forensics, provider-managed aggregation of security event logs is desirable, and to timely address ongoing problems, network-based intrusion detection system/intrusion prevention system is useful.

5.2.2.2 Cloud security - host level

Although there are few new host level security threats and vulnerabilities, some are inherited from related environments, e.g. some virtualization security threats carry into the public cloud computing environment. Even if the threats are not conceptually new, cloud computing harnesses the power of thousands of compute nodes, combined with the homogeneity of the operating system employed by hosts. This means threats can propagate quickly and easily.

Different cloud models imply slightly different security requirements.

In SaaS and PaaS, host security is opaque to customers. The responsibility of securing the hosts is relegated to the CSP (Cloud Service Provider), who institutes the necessary security controls. These include restricting physical and logical access to hypervisor and other forms of employed virtualization layers.

In the IaaS cloud model, the CSP has to secure the virtualization layer. A vulnerable hypervisor could expose all user domains to malicious insiders. The customer guest OS (or virtual server) is also a point of interest. It has an operating system provisioned on top of the virtualization layer that customers have full access to. Since the virtual server may be accessible to anyone on the Internet, access mitigation steps should be taken to restrict access to virtual instances. It is therefore necessary to adopt a *secure-by-default* configuration; this includes tracking the inventory of VM images and OS versions that are prepared for cloud hosting.

5.2.2.3 Cloud security - application level

In addition to well known application vulnerabilities, and well known attack types, cloud-based systems can experience other attacks. For example, an application-level DoS attack could manifest itself as high-volume web page reloads, XML web services requests, or protocol-specific requests supported by a cloud service. These kinds of attacks can be particularly dangerous because it is difficult to selectively filter the malicious traffic without impacting the service as a whole (AppLayerDDOSXie). This is because attackers use legitimate HTTP requests to the victim, which are indistinguishable from requests coming from genuine users (AppLayerDDOSXie, AppLayerDDOSMonitor09). DoS attacks on pay-as-you-go cloud applications could result in an increased cloud utility bill due to increased use of network bandwidth, CPU, and storage consumption. Therefore, sufficient protections need to be put in place to prevent resources used by a hijacked or exploited cloud accounts from being enrolled into botnets.

End users should be conscious about security and take appropriate steps to protect their web browsers from attacks. They must install patches and updates in a timely basis to mitigate threats related to browser vulnerabilities. However, as cloud-based services become widespread, relying on users alone will not be sufficient, so providers need to give some kind of assurance of adequate security. This assurance might be in the form of legal responsibility; for example, SaaS providers are largely responsible for securing the applications and components they offer to customers, while customers are usually responsible for operational security functions, including user and access management as supported by the provider.

In a PaaS context, developers need to become familiar with specific APIs for deploying and managing software modules to enforce security controls as part of their product life cycle. In theory, developers should expect CSPs to offer a set of security features, including user authentication, single sign-on (SSO), authorization, and SSL or TLS support, made available via the API. In the IaaS model , however, matters are predominantly left in the hands of end users. Customers should not expect any application security assistance from CSPs beyond basic guidance on firewall policies that may affect the application's communications with other applications, users, or services within or outside the cloud. customers are responsible for keeping their applications and runtime platform patched to protect the system from malware and hackers scanning for vulnerabilities to gain unauthorized access to their data in the cloud, and highly recommended to design and implement applications with a "least-privileged" runtime model.

5.2.3. Cloud security alliance Top Threats

Several organisations and communities have invested significant effort in identifying and characterising cloud computing security issues. An example of such an effort is that by the Cloud Security Alliance (CSA) who have identified top threats to cloud computing (CSA-TopThreats), discussed below.

5.2.3.1 Abuse and nefarious use of cloud computing

The registration process offered by some devices accept anyone with a valid credit card or even worse offer limited trial periods, which may allow spammers, worm author and criminals in general to conduct unauthorized activities with relative anonymity and impunity. This particularly affects IaaS and PaaS models. This threat can be mitigated by employing stricter registration and validation processes, e.g. monitoring of fresh credit card frauds, monitoring of user network traffic (possibly in aggregate form, to respect privacy issue) and monitoring procedures.

5.2.3.2 Insecure interfaces and APIs

Providers should be very careful in exposing software interfaces to interact with cloud services, and design these interfaces adopting proper encryption, authentication, access controls and monitoring to avoid policy circumvention, like anonymous access and or reusable authentication tokens, monitoring and logging capabilities unable to identify key events, unknown service and API dependencies. This kind of problem is quite generic and involve Cloud models of IaaS, PaaS and SaaS types.

To mitigate such a problem, knowledge of dependency chain associated with the API, as well as mandatory strong authentication in concert with encrypted transmission is needed.

5.2.3.3 Malicious insiders

IaaS, PaaS and SaaS models suffer from the convergence of services and customers under a single management domain, often combined with lack of transparency of providers internal processes. In this way, opportunities for harvesting confidential data or to gain unauthorized control over the cloud service can be achieved with little risk of detection.

This is remediable by a proper organizational (e.g. a strict supply chain management with associated supplier assessment) and legal (specification of resource requirements as part of legal contracts) framework. Clear, transparent and well-established information security management, compliance reporting as well as security breach notification processes also contribute towards mitigation of this issue.

5.2.3.4 Shared technology issues

Shared technology must implement strong isolation properties for a multi-tenant architecture. Access of resources should be mediated by the virtualization hypervisor, which manages guest operating systems and prevents improper resource access or control on the underlying system. However, even hypervisors can be flawed, and cause unwanted influence of the underlying platform by the guest operating system.

To prevent this critical issue, which is especially sensitive in IaaS model, security best practice must be adopted when installing and configuring the system. Monitoring system must adequately identify unauthorised changes and suspect activities. Strong authentication and fine-grained access controls should enforce proper administrative access and operations. Vulnerability scanning and configuration audit should be performed periodically, and a proper threat management process should ensure prompt patching and vulnerability remediation.

5.2.3.5 Data loss or leakage

Since data in the cloud can possibly be physically unbound to local users, cloud-based data management must avoid alteration of records with no backup, as well as loss of encoding keys (which may result in effective data destruction) and unauthorized access to sensitive data. This is a general problem which any of IaaS, PaaS and SaaS model have to prevent.

This kind of problem can be mitigated by some security best practice, like detailed API access control, encryption and integrity protection of data in transit (driven by a careful analysis of data protection scheme both at design and run time), implementation of secure key generation and life cycle management (e.g. storage and destruction procedures). An appropriate legal framework is useful as well (e.g. contractual obligations for providers to wipe persistent media before releases of sensitive data, contractual specification of backup and data retention strategies)

5.2.3.6 Account or service hijacking

Being a network based model, clouds are vulnerable to phishing, fraud and software vulnerability exploitation. Given that credentials and password are often reused, the impact of such attacks is amplified. This is because account owners are often unaware that their accounts have been exploited to form the basis of further malicious actions. All IaaS, PaaS and SaaS can suffer of these issues.

A mandatory policy on credentials (e.g. prohibit sharing of passwords or credentials between users and services), and leveraging multi-factor authentication techniques (whenever possible) as well as proactive monitoring to promptly identify unauthorized entities can alleviate impact of this threat.

5.2.3.7 Unknown risk profile

A detailed knowledge of security posture is important for threat prevention. Understanding the software, software updates, security practices, intrusion attempts, and employed security controls are important factors in understanding current risk level and planning adequate modifications. The temptation of employing *security by obscurity* should be avoided as opaque designs make in-depth analysis difficult. This is a general concern that is valid for all cloud models.

Mitigating practices include providing information about who is sharing the infrastructure, disclosure of network intrusion logs, redirection attempts (and success), and partial/full disclosure of infrastructure details (e.g. patch level, firewalls, etc). Even if not intuitive, these processes foster a more vulnerability-free and secure system.

5.3. webinos in the cloud

Although webinos is designed to run on devices such as in-car systems and mobile devices, it relies on inter-device connectivity for several aspects of its operation. In particular, keeping a list of preferences synchronized across a user's devices or sharing media content requires the devices to connect to each other or to a mediating component. These devices could be geographically distributed across the globe. As a result, webinos relies on global scale infrastructure such as the Internet to provide connectivity. However, such infrastructures are faced with a multitude of security and privacy problems. webinos' use of these infrastructures imply that it will also be exposed to these problems, and its cross device nature would make it an even more attractive target for attackers. For this reason it is important to understand the impact of webinos' reliance on these infrastructures on the security and privacy of both webinos devices and the data they handle.

5.3.1. Why in the cloud?

The webinos architecture proposes to make use of cloud infrastructure to host personal zone hubs (PZHs). The primary requirement of a PZH is that it is constantly available via the internet and is at a known address. High availability is one of the key selling points of a cloud, so this implementation choice makes sense. Alternative scenarios such as a PZH being on a home router are also possible, but unlikely to be as available for users such as our personas Georg and Clara (since these users are quite mobile and may require global access to their PZH).

A cloud-based PZH would need to be provided by a cloud service provider. This would be a service which inevitably cost money, perhaps as part of a mobile contract or with home broadband connection providers. The service could be charged on a per usage basis or as a one-off fee. Cloud hosting would allow for many additional bundled services, such as cloud storage and backup or enhanced security and assurance (see later).

Deploying services on a cloud infrastructure has some widely recognized barriers, which seems acceptable in the webinos case for the following reasons:

1. **Availability of service**: since the use is limited to a single person, the assurance level provided by cloud providers is acceptable for a personal zone. The synchronisation and data caching features provided by personal zones may be seen as a way of mitigating availability issues. However, in the case of a personal zone hub used by an entire organisation (and 'enterprise zone hub'), it is likely this assumption should be discarded. In that case, one possible risk mitigation strategy would be to have this enterprise hub hosted on multiple cloud providers. In this case, some functionality to synchronising between the two personal zone hubs (hosted by different cloud providers) would be desirable.
2. **Data lock-in**: occurs when a cloud provider stores or manages a user's data in a way that is only compatible with their system, so that it becomes difficult for the user to move away from the provider. To avoid this problem, the user can export the PZH data and to import it in another cloud. This can be achieved by either directing transferring data between providers or exporting it in some format and re-importing at the other providers.

One of the main expected use of cloud systems in webinos is the hosting of PZHs. For this reason, the following sections provide more comprehensive analysis of the threats and attacks related to running a PZH in the cloud.

5.3.2. Cloud-based personal zone hub and hub provider

A personal zone hub (PZH) is a key component in webinos. It provides a consistent means of communication among devices; through its API and services such as synchronization. In this section, we investigate the security and privacy impact of running a cloud-based PZH --- where cloud-based implies that the PZH has been deployed on a cloud infrastructure.

5.3.2.1 Security Analysis

Security analysis is based on previous work described in D2.7 and D2.8. More specifically, the assets, threats and vulnerabilities identified in the two tasks are used as input into the analysis. These are analysed to identify how they are affected by the adoption of a cloud environment.

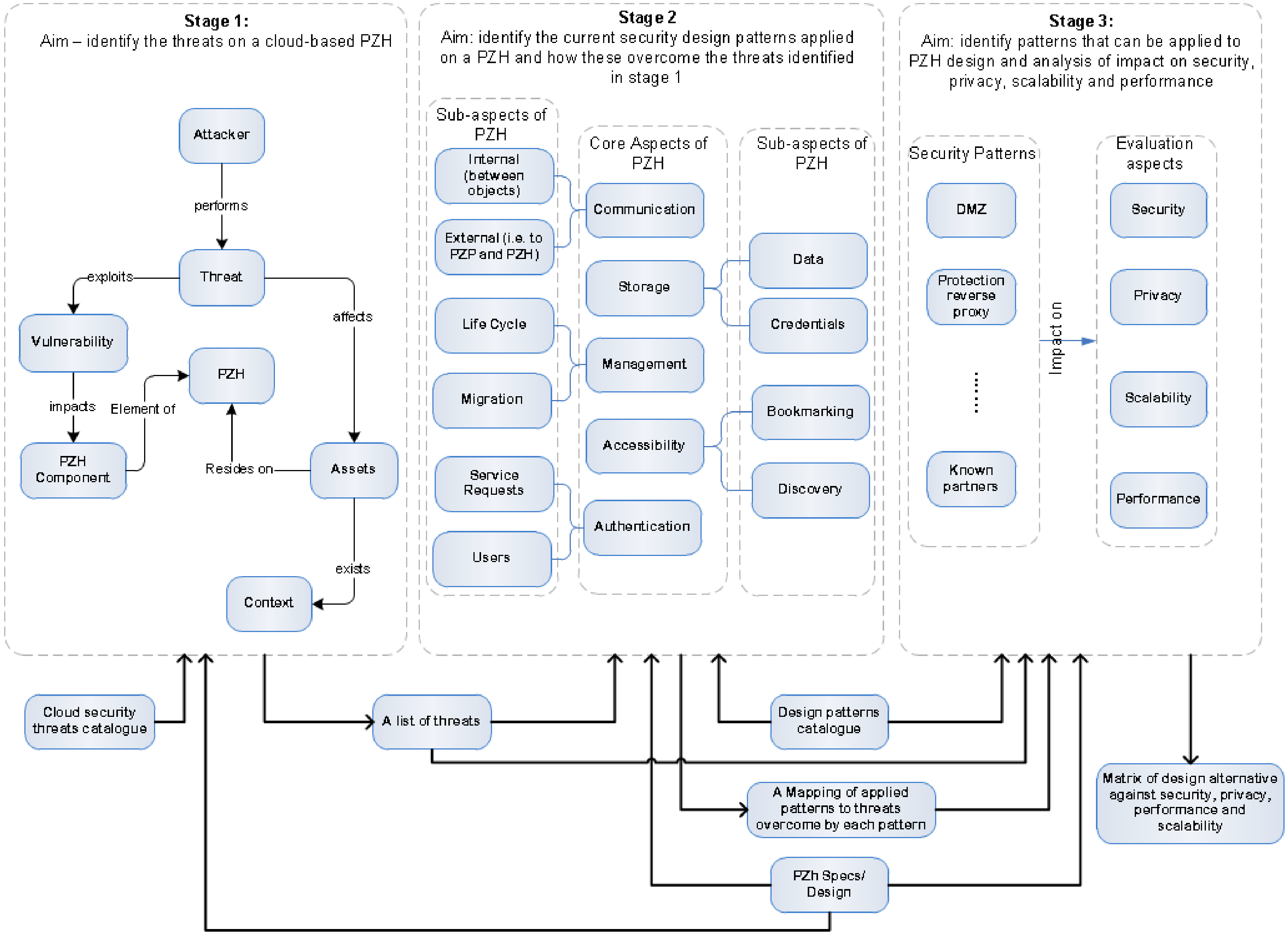
5.3.2.2 Analysis approach

In order to understand the implications of running a PZH in the cloud, a three-staged analysis approach was taken.

The first stage identified the threats faced by a cloud-based PZH design and the assets that may reside on or be transmitted through a PZH. Using the literature on cloud computing security as it relates to web applications, common vulnerabilities that may exist on a cloud-based PZH were identified, together with the possible attacks that might result from compromising these vulnerabilities. The output of this stage is a list of threats that webinos could be exposed to as a result of running a cloud-based PZH.

Stage 2 involved analysing the current design of the PZH to (i) mine architectural patterns currently in use in the PZH design, and (ii) identify which or to what extend these patterns addressed the threats identified in stage 1.

Stage 3 used the catalogue of architectural patterns to identify the patterns that could be applied to the PZH design. This stage also included an analysis of how these patterns and the ones identified in stage 2 impacted security, privacy, performance and scalability. This analysis produced a matrix specifying the relation of each pattern to its security, privacy, performance and scalability impact on webinos. The matrix was used to identify deployment profiles for PZHs. The figure below illustrates the input and output of each stage as well as the aspects considered in each stage. However, stages 2 and 3 are outside the scope of this document, and will instead be documented on an on-going basis.


Stage 1:
Aim – identify the threats on a cloud-based PZH
Attacker
performs
Threat
exploits
Vulnerability
impacts
PZH Component
Element of
PZH
affects
Assets
Resides on
exists
Context
Stage 2
Aim: identify the current security design patterns applied on a PZH and how these overcome the threats identified in stage 1
Sub-aspects of PZH
Internal (between objects)
External (i.e. to PZP and PZH)
Life Cycle
Migration
Service Requests
Users
Core Aspects of PZH
Communication
Storage
Management
Accessibility
Authentication
Sub-aspects of PZH
Data
Credentials
Bookmarking
Discovery
Stage 3:
Aim: identify patterns that can be applied to PZH design and analysis of impact on security, privacy, scalability and performance
Security Patterns
DMZ
Protection reverse proxy
....
Known partners
Impact on
Evaluation aspects
Security
Privacy
Scalability
Performance
Cloud security threats catalogue
A list of threats
Design patterns catalogue
A Mapping of applied patterns to threats overcome by each pattern
PZh Specs/ Design
Matrix of design alternative against security, privacy, performance and scalability

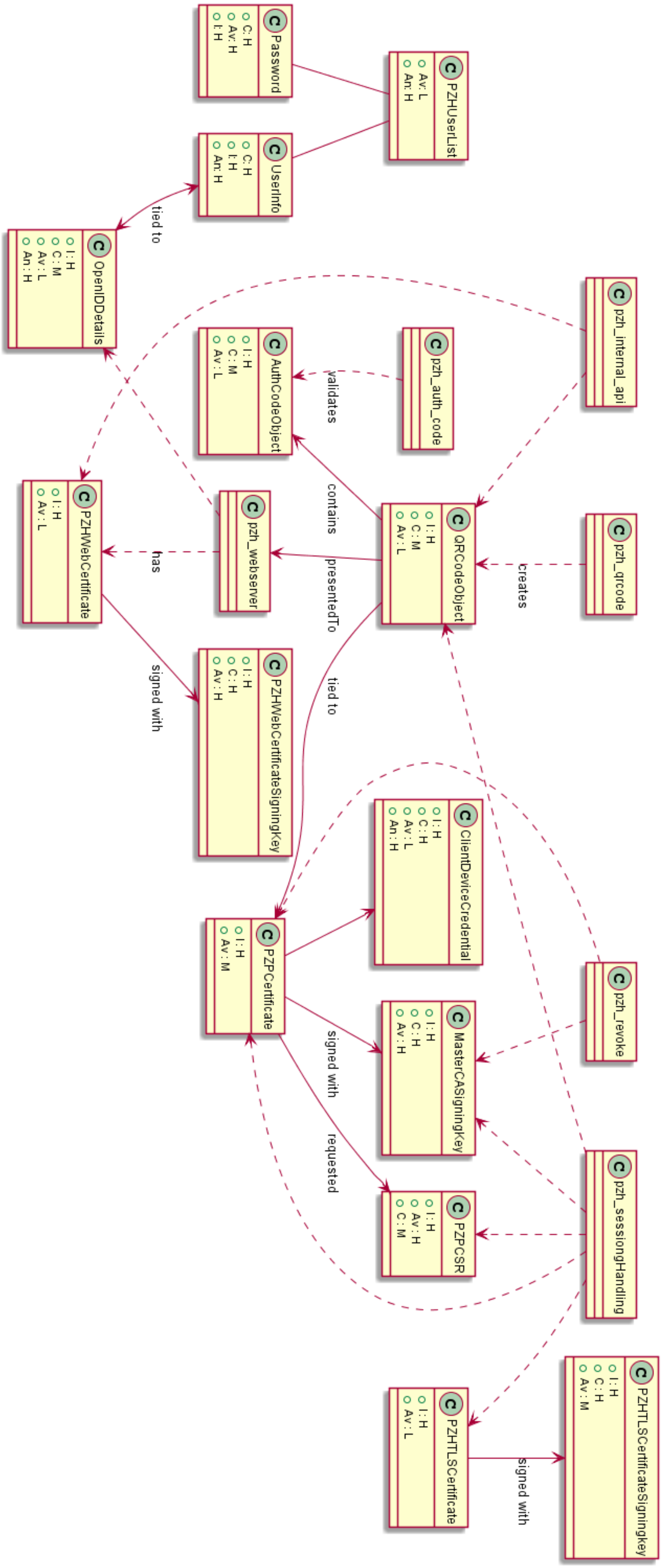
5.3.2.3 Asset Model

D2.7 and D2.8 identified a number of assets and assigned security and privacy values associated with each assets in a particular environment. These assets where defined at higher level of granularity, i.e. system level. This section is focused on understanding the impact of running a PZH in the cloud. The PZH is a component of webinos and therefore, to understand the threats associated with it, we would need to firstly map the assets to a granularity suitable for a component.

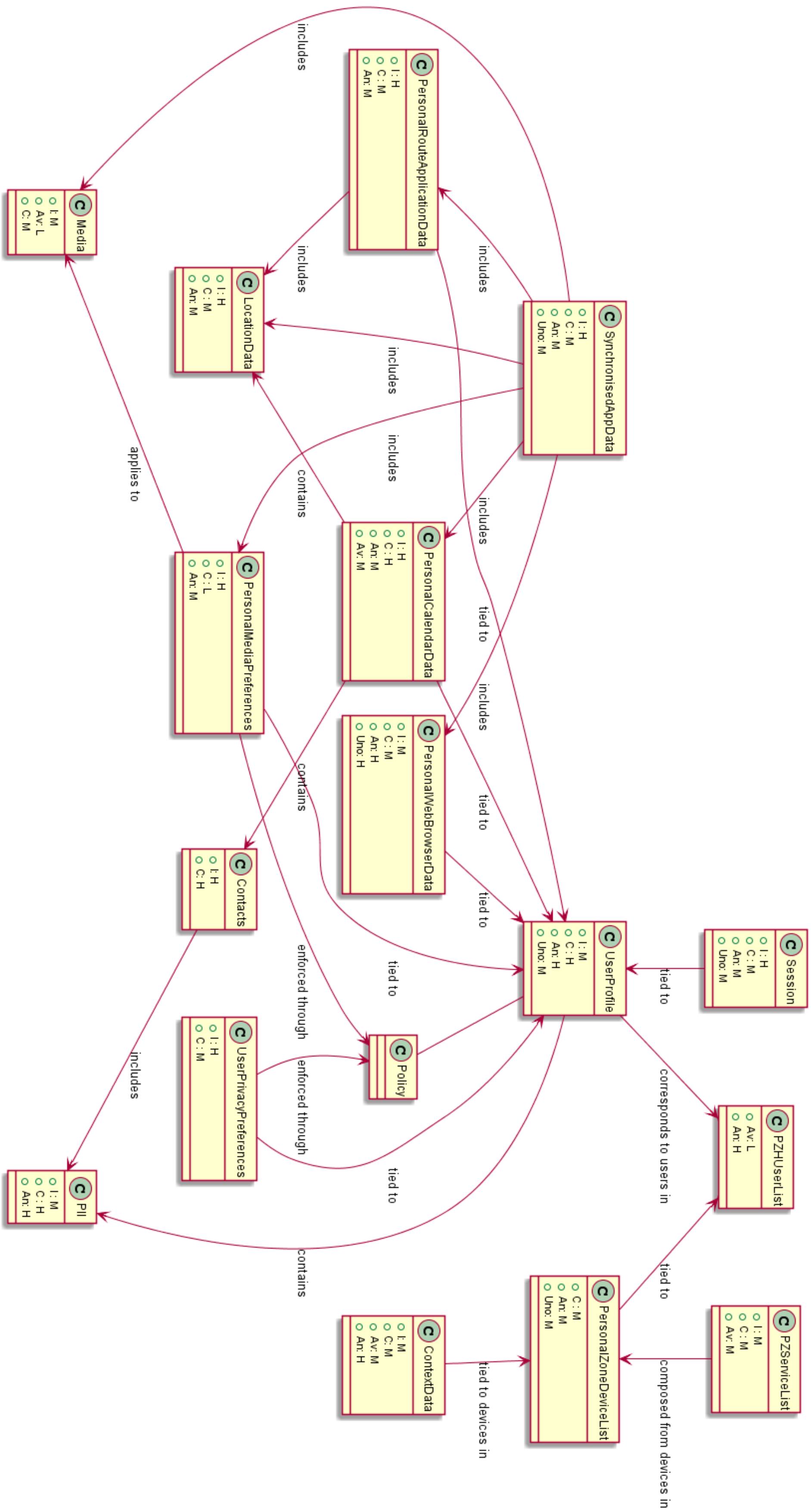
The approach adopted for specifying the granularity is *subclassing*, where each asset is broken down into subtypes that are as specific as possible. For instance, an asset such as *SynchronisedAppData* could be broken down to subtypes such as *Media*, *UserProfile* and *LocationData*. Subclassing allows the association of assets to components to be as specific as possible, thus enabling better analysis.

There are several assets used, generated or stored on a PZH. For simplicity, we categorise the assets into two categories, namely: credentials and functionality-related.

Credentials exist in many forms including certificates, passwords and cryptographic keys. The diagram below illustrates the assets that fall into this category.


Password
C: H
Av: H
I: H
PZHUserList
Av: L
An: H
UserInfo
C: H
I: H
An: H
tied to
OpenIDDetails
I: H
C: M
Av: L
An: H
pzh_internal_api
AuthCodeObject
I: H
C: M
Av: L
validates
pzh_auth_code
contains
QRCodeObject
I: H
C: M
Av: L
creates
pzh_qrcode
PZHWebCertificate
I: H
Av: L
has
pzh_webserver
presentedTo
signed with
PZHWebCertificateSigningKey
I: H
C: H
Av: H
tied to
ClientDeviceCredential
I: H
C: H
Av: L
An: H
PZPCertificate
I: H
Av: M
MasterCASigningKey
I: H
C: H
Av: H
pzh_revoke
signed with
requested
PZPCSR
I: H
Av: H
C: M
pzh_sessionHandling
PZHTLSCertificate
I: H
Av: L
signed with
PZHTLSCertificateSigningKey
I: H
C: H
Av: M

The PZH is intended to provide certain key functionality (though the functionality can be extended). As a result of the services it offers, the PZH comes into contact with a number of assets. The diagram below shows the key assets associated with the functionality of the PZH.


Media
I: M
Av: L
C: M
PersonalRouteApplicationData
I: H
C: M
An: M
LocationData
I: H
C: M
An: M
SynchronisedAppData
I: H
C: M
An: M
Uno: M
PersonalMediaPreferences
I: H
C: L
An: M
PersonalCalendarData
I: H
C: H
An: M
Av: M
PersonalWebBrowserData
I: M
C: M
An: H
Uno: H
Contacts
I: H
C: H
UserProfile
I: M
C: H
An: H
Uno: M
Session
I: H
C: M
An: M
Uno: M
Policy
UserPrivacyPreferences
I: H
C: M
PZHUserList
Av: L
An: H
PII
I: M
C: H
An: H
ContextData
I: M
C: M
Av: M
An: H
PersonalZoneDeviceList
C: M
An: M
Uno: M
PZServiceList
I: M
C: M
Av: M
includes
includes
includes
includes
includes
includes
includes
applies to
contains
contains
tied to
tied to
tied to
tied to
tied to
enforced through
enforced through
includes
corresponds to users in
contains
tied to
tied to devices in
composed from devices in

5.3.2.4 Security and privacy requirements for the PZH

Before conducing the analysis, however, we need to understand what the requirements are for the PZH towards the assets identified above. Below, we discuss some of the crucial ones.

- Credentials must be held securely, including
    - Certificate authority keys
    - Any passwords or tokens used to access other services and devices - e.g. OAuth tokens
    - Lists of trusted certificates
- Personal data stored on the PZH must be protected. While only the minimum has been specified, webinos may store a wide variety of data such as:
    - Contacts
    - Photos, videos and media
    - Location data
    - The context database
    - Lists of available applications and services
    - Personal settings and information, such as date of birth, home address, identities on other social networks
- The PZH will provide access control through policies and lists of trusted users. These lists are both private and have integrity requirements: they must not be modifiable by another party
- The PZH to some extent represents user identity.
    - Unauthorised access to this identity would allow impersonation and potential identity fraud.
    - Disclosure of this identity may enable a user to be tracked by applications or other devices.

5.3.2.5 Threats to a cloud-based PZH

Significant effort by organisations such as Mitre Corporation (Mitre), NIST (NIST), cloud security alliance (CSA) and other communities, has been directed towards identifying and classifying common threats that impact computer systems. Some of the most significant threats that exist in a cloud environment were discussed earlier. These threats affect different parts of a system including input, output and in memory. The PZH is not immune to these attacks. Moreover, some of the design choices for the PZH, such as running it in the cloud, could make the risks of these threats greater. For this reason, it is important to understand the threats that affect the PZH, especially in relation to its cloud adoption.

In this section, we analyse the common threats to identify which of these affect the PZH and in which instances they do so.

Insider credential misuse

In the asset model discussed above, several credentials were identified. For example, the private key used to sign digital certificates issued to devices connecting to the PZH was identified to both be highly confidential and requiring high integrity. These credentials may be required to be stored in the PZH to enable it to provide its services. The CSA includes malicious insiders as one of the top threats to cloud computing. A PZH deployed in the cloud could suffer from this threat and allow malicious insiders to steal the credentials. Alternative, non-malicious users, eg. administrators, may unwittingly expose these credentials, eg. through backups. These leaked credentials can be used to impersonate the PZH instance (eg. when its private key is compromised) or issue certificates on behalf of the PZH.

Unauthorised PZH duplication

One of the main advantages of cloud computing is the ease of service provisioning, enabled by mechanisms such as migration and virtual machine cloning. This feature enables PZHs to be easily instantiated in response to client requests for PZH instances. Additionally, new PZH instances can also be created in response to increased demand for services from a particular PZH. In this case provisioning of a PZH may be achieved by either cloning an existing PZH or using a standard PZH template. In the former, data stored within the PZH may be exposed as a result. Alternatively sensitive PZH configurations may be leaked. In the latter, PZH may end up sharing configurations, which if if insecure and not modified, may result in break-once-run-everywhere kind of attacks.

Loss of data such as synchronisation data

A PZH once deployed provides a number of services such as synchronisation and routing. These services handle various data such as media, personal media preferences or contacts. This data does not have high availability requirements, but if stored on the PZH alone, could have significant effects on usability if lost. The issue with cloud computing, as discussed in the data loss threat from CSA, is that it provides a number of interfaces through which data may flow and thus increasing the attack surface.

Misuse or compromise of interfaces

Cloud infrastructures are complex in nature. To manage this complexity, cloud infrastructure provide several abstraction layers which hide the underlying complexity by limiting accessibility of the services to specially designed interfaces. These interfaces may, however, be poorly designed or have vulnerabilities which may lead an attacker to either misuse them (eg. sending well crafted exploit code) or access services they are not authorised to access. Deployment of a PZH in the cloud magnifies this problem. This is because the PZH itself, provides a number of interfaces through which its services may be accessed. These interfaces interact with with interfaces provided by the underlying cloud infrastructure. Therefore any weakness or flaws in either the interfaces of the cloud infrastructure or the PZH itself could lead to compromise of the PZH or the assets stored on it.

PZH integrity compromise resulting from insecure isolation

A PZH deployed on the cloud would have to share the resources with possibly other PZHs or other applications. While cloud computing relies on virtualization to provide isolation, several attacks have demonstrated that virtualisation may not always provide complete isolation between services hosted on the same physical resource. CSA identifies this as a threat resulting from sharing technology.

In the asset model, several PZH components were identified to be assets as well, due to their relationships with other assets. For example, the session Manager handles authentication and also comes into contact with other assets such as private keys. A compromised session manager could easily expose sensitive data or compromise its integrity. For this reason, the integrity of some of the PZH components is considered to be critical for the well functioning of the PZH. The threat of shared technology could affect the integrity of the PZH components. In other words PZH instances running in the cloud may not be securely isolated from other services sharing the same resources, resulting in the compromise of the PZH integrity.

Privacy breach as a result of identifiable PZH

When hosted in the cloud, the PZH will need to be linked to the user for the purpose of billing. For this same purpose, PZH providers may need to monitor activities on a PZH instance, e.g. to enable pay-per-use of particular added-on services. As a result of this, PZH cloud providers may be able to deduce sensitive information about the activities of the PZH owner such as the services they accessed, leading to a privacy breach. Furthermore, as indicated in the account or service hijacking threat, attackers may perform certain activities which would be traced back to the owner of the PZH.

Sensitive residue migration data

The use of cloud computing for the deployment of a PZH makes it easy to switch from one provider to another; this is achieved through migration services provided by most cloud offerings. This means that a PZH instance can be migrated from one provider to another or from one platform to another, i.e. as part of load balancing. Migrating a PZH instance may involve moving the entire execution environment, such as virtual machine, from one system to another or copying data that is specific to a particular PZH instance into another environment capable of running a PZH. In both cases residue data, which could potentially include sensitive data such as keys, database backs or PZH configurations, may be left on the original host. This data could be used to mount attacks on the new PZH instance or leak sensitive information.

Data leakage by VM replication

Similar to the threat of residue migration data, sensitive data could also be leaked by cloning a PZH instance. Suppose *Alice* requires high availability for her PZH, then as a solution, a cloud provider could create several instances of the PZH, each with a copy of all of *Alice*'s data, so that in an event that the main instance goes down, eg. for maintenance, any of the cloned instances could be provisioned to serve any incoming requests. The main threat here is that of an increase in the attack surface. So that if any of the cloned instances have some misconfiguration, then an attacker could take advantage of that and exploit the PZH.

Compliance and Unauthorised Data Disclosure

One of the challenges faced in cloud computing is that of data location (Kandukuri-cloud-sec-09, Binning-top-cloud-issues-09). Whereas traditional dataware mechanisms provide visibility of the location of data, the use of a cloud environment makes this completely transparent. This means that data could be stored in a data centre anywhere in the world. This means that information may cross the border, raising concerns about forced data disclosure.

In *Alice*'s case, the PZH could be running on a server where the cloud provider might be forced, under laws such as the Patriot Act (USPatrioticAct-UnderFire-04), to release the data to government agencies. This has the potential to disclose all the sensitive information such as *Alice*'s activities and behavioural patterns and media content.

5.3.3. Cloud specific vulnerabilities

The previous section presented some possible threats to a PZH running in the cloud. However, in order for an attacker to realise these into attacks, there has to be some exploitable vulnerabilities in the system. One of the challenges in analysing the security and privacy impact of cloud computing is understanding if at all there are specific vulnerabilities that exist solely because of the use of a cloud environment. Grobauer et al. (Understanding-cloud-vulns) specifies the characteristics of cloud specific vulnerabilities to include:

- is intrinsic to or prevalent in a core cloud computing technology,
- has its root cause in one of NIST’s essential cloud characteristics,
- is caused when cloud innovations make tried-and-tested security controls difficult or impossible to implement, or
- is prevalent in established state-of-the-art cloud offerings.

Using this characterisation, they identify some vulnerabilities specific to cloud environment as well as how some common vulnerabilities equally apply to cloud. These vulnerabilities are summarised in the table below, which also indicates relationship to CWE where possible.

category	Vulnerabilities	Description	CWE Equivalency
Intrinsic to technology			
	Virtual machine escape	a virtual machine is meant to operate without influencing other VMs on the same host, however, incorrect	

		configuration of the hypervisor could lead to a VM escaping from its containment and affect other VMs	
	Session riding/hijacking	sessions maintained by web applications could be exploited by attackers	CWE-613
	insecure cryptography	cryptographic algorithms or protocols may be insecurely designed or implemented and advancements in cryptanalysis may render them insecure	CWE-261
Cloud characteristic			
	Unauthorized access to management interface	management interfaces provide means of controlling the life-cycle of resident VMs. Unauthorised access to the management interface could lead to unexpected behaviour of the VM as their state can be changed	CWE-284, CWE-522
	Internet protocol vulnerabilities	vulnerabilities in the underlying Internet protocols could allow man-in-the-middle attacks	CWE-300
	Data recovery vulnerability	resources assigned to a PZH instance might be re-allocated to a different user at a later time making it possible to recover data written by a previous user	
Use of existing controls			
	Insufficient network virtualization	limited administrative access to IaaS network infrastructure imply that standard network controls such as IP-based network zoning can’t be applied	
	poor key management procedures	lack of a fixed infrastructure makes it more difficult to apply standard controls—such as hardware security module	close to CWE-321, CWE-324

Prevalent in state-of-the-art			
	weak authentication mechanisms or implementations	use of authentication mechanisms such as usernames and passwords have inherent weaknesses which may allow credential interception and replay	CWE-287, CWE-290, CWE-303
	vulnerable VM templates (Kandukuri-cloud-sec-09)	the VM templates could contain vulnerable programs or data that the creator did not intende to expose	
	Insufficient logging and monitoring possibilities	log files may not be linked to a particular tenant or contain sufficient information making harder to imply security controls that rely on logging and monitoring	CWE-778

5.3.4. Extending the PZH functionality

While the PZH is designed to provide services necessary to support cross device connectivity, it is not hard to imagine instances where such a design may be insufficient (i.e. in terms of features). Furthermore, it is hoped that webinos will inspire some creativity, some of which will be applied to the PZH functionality. For example, *Alice* may come up with value-added services and offer her PZH to other people, for a fee. These services may be implemented as applications which are strongly tied to a particular PZH instance, PZH farm or PZH session. In order to understand the security and privacy implications of a cloud-based PZH, it is necessary to expand the scope of a PZH to include applications that may be integrated into it. These applications may be vulnerable to certain kinds of attacks, and thus may expose webinos to even more attacks.

Alice could decide to integrate one of the applications that she uses to manage her documents with webinos, so that all the preferences and documents could be synchronized on her devices. This application may need to know about the nature of *Alice* devices, for example to create versions of documents compatible with each device. To achieve this, the application would need to communicate with the PZH, enrol into the personal zone and discover information about other services in the personal zone.

webinos supports web applications that are downloadable widget packages as well as fully web-based hosted applications. Hosted web applications are often cloud based, relying on cloud storage, databases, or even hosted on a cloud infrastructure. The design and structure of these applications is largely out of scope. However, the project will be considering the security and privacy of the concept applications, as well as how webinos infrastructure can assist cloud-hosted applications to make them more trustworthy.

Hosting such an application could be beneficial for *Alice*, as she could use it on-demand and only pay according to how much she uses. However, the question of the implication on the security and privacy of assets still remains. Firstly, the application could be considered a *virtual PZP*. In this case, the application might have access to all the data in the personal zone, including contacts, personal preferences, location data and API usage data. Most likely, such an application will be designed to store this information in a database. Therefore, even though the PZH could be secure, flaws in such an application or the underlying database system might lead to compromise of asserts in the personal zone.

The application might also incorporate other services from unknown or untrusted providers. These services may have access to the data handled by the application and therefore to the asserts. Storage mechanisms used by the application could also be another source of concern. Furthermore, the authenticity of the application and the trustworthiness of the developers deserve some attention. And finally, the level of permissions given to such *virtual PZPs* could lead to them accessing data they should otherwise not be expected to access or perform actions that they should not.

5.3.5. Mitigations and opportunities

This section has identified a number of threats and vulnerabilities that could be used to compromise the assets on a PZH. The question that this raises is, how well placed is webinos to counteract these threats? webinos already employees some mechanisms to counteract these threats. However, more needs to be done in order to adequately cater for each of these issues. Some of the important ones are discussed below:

- **webinos credential storage** - credentials have been identified as one of the main assets in a PZH, which if compromised could have significant impact on the security and privacy or other aspects of webinos. These credentials could be accessed due to insufficient protection mechanisms on them. For example malicious insiders could access the credentials and use them to impersonate the PZH. For this reason, better protection mechanisms should be put in place to prevent these attacks. A common principle that might help with this problem is isolating the credentials from the data and or services to which they are applied.

- **Better credential management** - in addition to secure storage, the use of the credentials also needs to be properly controlled. Any key that is used must be tied to a well specified usage policy and must explicitly be specified with an expiry period after which the key should be considered compromised. Any backup procedures used for the keys must also be implemented to avoid leakage of these keys. Furthermore, when a PZH instance is migrated, the previous keys must be carefully destructed and new ones created.

- **PZH instance life-cycle management** - a PZH will run on VM instances that may be incorrectly configured or have outdated software. In order to avoid the attacks resulting from exploiting the underlying environment in which a PZH runs, it is important the that life-cycle of each VM used to host webinos is properly managed. This involves putting in place effective patch and configuration management processes as well as transparency in the creation process of these VMs.

- **Potential for cloud-based attestation of devices** - the devices used in a personal zone could also be attested to ensure that they have some minimum configurations that ensure protection of personal zone asserts. The Attestation API, already implemented in webinos, could be used more often to identify devices and their configurations. This could also serve as a basis for identifying the possible behaviour of *virtual PZPs* before admitting them to the personal zone.

- **Use of cloud based intrusion detection and traffic analysis** - the issue of inadequate logging and monitoring facilities in the cloud has significant impact on the security and privacy of the PZH. For this reason, it is essential that logging and tracking mechanisms are employed to track events on a cloud host and to detect and find the cause of breaches. Dedicated log servers and intrusion detection systems capable of collecting security relevant events must be used all the time.

- **Reclaiming cloud-based data** - one advantage that cloud-based web applications may gain from webinos is the ability to store data reliably on the user's own platform. At present, web applications such as Google Docs store data centrally, to provide access from anywhere, backup, and synchronisation. As webinos will provide these through the PZH and its own infrastructure, private data can be reliably stored using browser local storage. This has advantages for privacy: while this data is still accessible to the application, if the user decided to delete it, the data is no longer available. Furthermore, the user can enforce their own policies for passing data to a third party. This may be an advantage for application providers, as they will be able to avoid data protection requirements by not holding personal data themselves.

- **Better and verifiable VM provenance** - the VMs used to host the PZH might introduce vulnerabilities due to improper configuration or use of vulnerable components. For this reason, the provenance of the VMs must be securely collected in a manner that enables verification of key properties (NamilukoVMProvenance2012). This provenance should include the origin and integrity of the components used, the configurations applied and operations performed to prepare the VM in readiness for use in hosting a PZH.

- **PZH architecture considerations** - the PZH could be designed using several approaches. For example, the current implementation is such that the PZH operates similar to a monolithic kernel - storing all the data and providing all the services from a single component. Other alternative designs should be careful considered and evaluated to determine how they fair against cloud related threats. This work will be performed as part of stages 2 and 3 in the analysis, as discussed above.

5.4. Summary and conclusion

The promise of unlimited resources, on-demand service and pay-per-use model of cloud computing is significantly attractive to webinos. However, the use of cloud computing has the potential to expose webinos to significant security and privacy risks. While several attempts have been made to identifying and classifying cloud-related security and privacy issues, these need to be put into context in order to understand how the use of a cloud could affect webinos. This understanding enables users to make informed decisions about the choice of a cloud environment for webinos. To a user thinking about deploying a PZH in the cloud, understanding the kinds of assets such as credentials and synchronisation data that could exist in the cloud and the threats that these could be exposed to is vital in deciding whether or not the risks of running a PZH in the cloud are worth taking. To a provider, or indeed some one thinking about offering PZH services to other users, this understanding is vital in deciding the level of risk that their customers would be exposed to as well as in deciding how much of the responsibility they would be willing to carry. This section provides information necessary to develop such an understanding. By analysing the functionality of a PZH, several critical asserts were identified. These include media content, credentials such as private keys used in authentication, PZH users list and calendar data such as contacts and appointment schedules.

5.4.1. Key Findings

The key findings of the investigations in this section can be summarised as follows:

- webinos can take advantage of flexible, on-demand and pay-per-use services enabled by the adoption of a cloud-based system,
- security and privacy issues must, however, be identified, characterised and addressed before webinos can take full advantage of cloud computing,
- the analysis demonstrates that significant number of assets may be exposed to common security and cloud-specific threats and therefore require careful consideration of the risks involved,
- vulnerabilities in the underlying infrastructure or environment used to host a PZH or resulting from inherent characteristics of cloud computing or the technology it employs could be exploited to compromise the security and privacy of the assets on a PZH,
- a number of mitigations could be employed to reduce or eliminate some of the threats, such as better architecture for the PZH that employs principles such as separation of concern and security in-depth. However, certain kinds of threats such as compliance and jurisdiction may be outside the reach of webinos, instead, webinos would have to hope for a quicker maturity in the cloud offerings.

5.4.2. Recommendations

This section has presented the threats and vulnerabilities that a cloud-based PZH could be exposed and which *Alice* would have to aware about. But what does this mean to *Alice*, or anyone thinking about using webinos in the cloud? Based on the analyses performed, the following recommendations are provided:

- **Minimal PZH services** - the architecture of the PZH must be designed with a minimal set of services following the principle of separation of concern. The impact of the architecture of the PZH on security and privacy will be investigated as part of stages 2 and 3
- **Secure design principles** - well known security principles such as security in-depth and separation of concern must be employed in the design of the PZH, while other principles such as security by obscurity must be avoided.
- **Transparency** - cloud offerings must be more transparent to allow the users to see the configurations of their instances, staff hiring procedures and management processes involving their data and credentials.
- **More research in personal clouds** - webinos bridges the gap between a user's devices, allowing them to share data and services. The use of the cloud makes this more interesting, inspires creativity and opens up a number of challenges in terms of how this personal space, which was initially restricted to physical devices, can be securely managed. This calls for further research. Perhaps more collaboration with projects such as TClouds could have mutual benefits to webinos and other projects.
- **To *Alice* on hosting a cloud-based PZH** - *Alice* can go ahead and set-up a personal zone with minimal devices and keep her VM up to date. Furthermore, she must take advantage of current security mechanisms such as secure cloud storage and take advantage of the latest advancements in cloud security.

5.4.3. How will this analysis be useful in the future?

One of the efforts that have already began is putting the attacks within the context of the webinos architecture. For example the Architectural Risk Analysis involved creating attack patterns for various aspects of webinos and linking them to architectural patterns for the purpose of understanding how the architecture deals with these threats. The work presented in this section could be used as a first step towards understanding webinos architecture's readiness for the cloud.

6 Recommendations

This section gives advice and recommendations to stakeholders responsible for certain aspects of the webinos environment in order to minimize security and privacy concerns. These are in addition to general best practice guidelines on security and privacy. A list of webinos stakeholders can be found in the (Glossary) accompanying the main D3.3 specification.

6.1. For Identity Providers

Identity providers are highly trusted with the webinos architecture as their ability to authenticate users is relied upon to protect the personal zone. Compromise of an OpenID identity used to manage a personal zone would result in a potential loss of data, personal privacy and could be used to perform a number of other attacks.

We therefore recommend that identity providers do the following:

1. Implement OpenID PAPE extensions. This allows webinos to dictate when a user should re-authenticate, rather than relying on the OpenID provider to correctly implement authentication caching.
2. Provide and encourage two-factor authentication. The threat of identity theft and compromise of OpenID credentials is significant and hard for webinos to mitigate. A solution, however, might involve integrating the webinos PKI infrastructure with OpenID in order to provide a second factor of user identity.
3. Provide recovery mechanisms based on secure out-of-band communication methods. This will help avoid attacks based on a malicious user recovering the credentials of another person, an attack we cannot mitigate in webinos.
4. Follow best-practice guides (OpenIDBest) to avoid insecure implementations of OpenID protocols.

6.2. For PZH Providers

Personal zone hub providers are trusted in the webinos architecture with the ability to control a user's personal zone. In the T3.3 informative sections on PZH deployment we describe several potential architectures which can mitigate some of the trust placed in webinos personal zone hubs.

For a provider attempting to offer services and preserve user privacy and security, we suggest the following:

1. Store all user data on encrypted disks. Do not allow any third parties to access this data directly.
2. Monitor incoming connection traffic to detect potential attacks on web interfaces and PZH TLS servers.
3. Only support OpenID providers who follow the recommendations specified above

4. Support trusted hardware modules in order to secure keys and credentials

6.3. For Device Manufacturers

1. Provide user-to-device authentication methods which will mitigate threats from devices places in shared environments. For example, providing PINs or locks that are easy to use yet provide some real resistance from casual attackers. This is important in webinos as access to the device is sufficient, in some cases, to access part of the entire personal zone.
2. Provide disk encryption by default. Webinos makes no attempt to encrypt application data, recognising that this is something that can be provided more effectively by devices. Enable the disk encryption facilities of the device before shipping it with webinos.
3. Use an operating system that provides process isolation. The webinos platform assumes that processes are isolated from each other and is more secure if each application has its own area of protected storage. This can be found in operating systems such as Android and iOS, and can be configured in Linux using Linux Security Modules such as SELinux and AppArmor. Native malware is a threat that webinos cannot protect against, and therefore operating systems must provide isolation between processes and data used by each process.
4. Provide data backup solutions.

6.4. For Application Developers

There are numerous guidelines and recommendations for the secure and privacy-preserving development of web applications. Secure software development is a topic covered by hundreds of books and articles which we do not repeat here. However, we suggest that developers are aware of the following existing literature:

- The OWASP development guide (OWASP-Guide)
- W3C "Web Application Privacy Best Practices" - (W3CAppPriv)

We also encourage application developers to request as few webinos feature permissions as possible, and to register their applications with well-known app stores.

Finally, for hosted applications we recommend the WebSand project (WebSand) as a source of useful security guidance.

6.5. For Users

Users of webinos should be aware that webinos is only as secure as the various components and entities it relies upon. As such, users should be careful in choosing:

1. The devices they use webinos with. If webinos is to be used securely, devices should provide disk encryption. Devices should also either be used only by the zone owner, or offer user accounts which separate users from each other. For shared devices, such as TVs, users should be careful to avoid granting them too many permissions

2. The OpenID provider they use. We recommend the use of an OpenID provider who offers two-factor authentication
3. The PZH Host they choose. PZHs are highly trusted within the architecture, and users should not choose a provider they do not believe to be reliable
4. The applications they grant privileges to. Applications are a major source of threats, and we recommend that applications are not granted privileges without reason.

However, most of these responsibilities should not be shouldered by the user as (in many cases) they are not in a good position to make informed decisions. As such, these recommendations should primarily be followed by device manufacturers, service providers and app stores.

7 Conclusion

This deliverable presents the results of a large analysis of the webinos architecture with respect to security and privacy. We have incorporated data from related systems, academia, other webinos deliverables, the OWASP, CAPEC and CWE projects in order to assess the resistance of webinos to attacks and how it can be improved in the future. The document departs from the format of D3.5, in that all specifications for functionality can now be found in the D3.3 specification documentation.

As part of this work we have performed an architectural risk analysis, developed a threat model, analysed the majority of webinos APIs for security and privacy issues, performed an in-depth analysis of how webinos is affected by the use of cloud components, and presented a set of recommendations for stakeholders. In this final section we present our overall conclusions.

7.1. Summary of results

The webinos architecture, as specified in Deliverables 3.3 and 3.4 is resistant to several of the attacks described by attack patterns in the architectural risk analysis. While some unmitigated obstacles were found as part of the architectural risk analysis, many of these can be delegated to other entities such as the identity providers, operating systems and browser developers. Denial of service issues do not appear to be a great threat, and primarily affect components outside of webinos. Cloud security and privacy issues have been considered, including threats from hosting providers, errors in service migration and misuse of complex interfaces. However, assuming that best practices in cloud security are followed and that the scope of a webinos personal zone hub does not increase too dramatically this seems a reasonable risk considering the potential benefits. Furthermore, there is an opportunity for further research into mitigating some of the risks to cloud-based personal zone services. Finally, a significant number of API-related issues were identified and several mitigations have been designed and proposed for developers and specifiers.

Our analysis should not be considered final or complete. We have considered 14 attack patterns and 24 API specifications, but more exist and will need to be specified in the coming months. Our models of the webinos architecture can also be expanded to include more areas. However, the data we have presented is accurate, grounded in the specifications and results of previous deliverables, and covers a wide variety of functional areas in webinos.

The following changes to webinos may be considered as a result of this deliverable. Firstly, support for unmodified web browsers in webinos might be dropped in favour of customised webinos extensions. This will help satisfy many of the goals in the ambiguity analysis. The introduction of "Sensor Widgets" as described in (Gibraltar) may be considered in the final phase of the project to counter many API-specific misuse cases. Some APIs could be improved for less trusted applications, such as the messaging and discovery APIs. With regards to the overall system architecture, the webinos personal zone hub could benefit from restructuring to minimise the burden placed on any individual cloud provider.

7.2. Making use of these findings in webinos

These findings can be used by webinos in the following ways:

1. The threat model can be used to check design decisions and to justify the placement of new features
2. Obstacle models and the leaf obstacles can be used to motivate and encourage the development of the platform
3. The API analysis will be used by implementers and specifiers to avoid the threats identified.
4. This process can be integrated with the requirements update process to make sure that new features, proposed internally or externally, do not violate any of the security assumptions or existing mitigations in the architecture
5. The recommendations from the 'summary' page of the architectural risk analysis can inform the roadmap of webinos deliverables

7.3. Making use of these findings outside of webinos

Because the process followed in the architectural risk analysis is novel and based on publically-available data on threats and weaknesses, we intend to publish our findings and data on a public GitHub repository. This will allow other projects to identify how we have performed our analysis and compare with their own. It will also allow implementers, application developers, platform integrators and others to assess the risks involved with using webinos and whether it matches their environment. We also believe it will be useful as a source of material for academic data.

We have also proposed a set of recommendations for stakeholders within webinos, with particular emphasis on identity providers, pzh providers, device manufacturers and application developers. We have also outlined some key mitigations to security issues associated with the APIs being standardised by webinos which will be useful for any developers wishing to extend the platform.

7.4. Future security and privacy activities in webinos

The work described in this deliverable provides a consistent framework for changing and updating the security and privacy features in webinos. Security and privacy activities in webinos continue after August 2012, and during the remaining time we intend to do the following:

1. Integrate the data security and API analysis with the formal CAIRIS-based process
2. Create more attack patterns and integrate the draft patterns using CAIRIS.
3. Search for more input from outside sources of threats and weaknesses, such as the Cloud Security Alliance and privacy literature.
4. Analyse the remaining APIs not covered in this deliverable.
5. Begin the process of addressing the weaknesses and ambiguities identified in the system specifications.
6. Update the attacker personas; these were found useful during the creation of attack patterns - to represent more relevant attackers. In particular, attackers of cloud systems.

7. Make use of the premortems approach described in (Faily2012)

8 References

8.1. Akhawe2012

Devdatta Akhawe, Prateek Saxena, and Dawn Song
Privilege Separation in HTML5 Applications
USENIX Security Symposium, 2012
https://www.usenix.org/conference/usenixsecurity12/privilege-separation-html5-applications

8.2. Anderson2008

Ross Anderson,
Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd edition,
John Wiley & Sons, August 2008.

8.3. AndroidManifestPermission

Android Developers Reference: Manifest.permission API
Fetched June 2011
http://developer.android.com/reference/android/Manifest.permission.html

8.4. AndroidOverview

Sunitha Medayil Vijayamma,
A Security Overview in Google's Open Source Android Phone
2009
http://www.scribd.com/doc/25036401/A-Security-Overview-in-Google-s-Android-Phone

8.5. AndroidSecurity

Android DeveloperGuide: Security and Permissions
Fetched June 2011
http://developer.android.com/guide/topics/security/security.html

8.6. AndroidSecurityOverview

Android Security Overview
Google, 2012
http://source.android.com/tech/security/index.html

8.7. AndroidSurvey

Kamran Habib Khan and Mir Nauman Tahir,
Android Security, A survey. So far so good.
July 2010
http://imsciences.edu.pk/serg/2010/07/android-security-a-survey-so-far-so-good/

8.8. AppleIosSec

iOS Security
Apple
May 2012
http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf

8.9. AppLayerDDOSXie

Yi Xie; Shun-Zheng Yu;
A Novel Model for Detecting Application Layer DDoS Attacks
Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on , vol.2, no., pp.56-63, 20-24 June 2006
doi: 10.1109/IMSCCS.2006.159
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4673677&isnumber=4673661

8.10. AppLayerDDOSMonitor09

Yi Xie and Shun-Zheng Yu.
Monitoring the application-layer DDoS attacks for popular websites.
IEEE/ACM Trans. Netw. 17, 1 (February 2009), 15-25. DOI=10.1109/TNET.2008.925628
http://dx.doi.org/10.1109/TNET.2008.925628

8.11. Atzeni2011

Andrea Atzeni and Cesare Cameroni and Shamal Faily and John Lyle and Ivan Flechais
Here's Johnny: a Methodology for Developing Attacker Personas
Proceedings of the 6th International Conference on Availability, Reliability and Security
IEEE Computer Society, 2011
pp. 722--727

8.12. B2GAppSec

Boot 2 Gecko App Security
Mozilla

August 2012
https://wiki.mozilla.org/Apps/Security

8.13. B2GAppSecDis

Boot 2 Gecko App Security Model Discussion
Mozilla
May 2012
https://wiki.mozilla.org/Apps/Security/Discussion

8.14. B2GRuntimeSec

Boot 2 Gecko Runtime Security
Mozilla
June 2012
https://wiki.mozilla.org/B2G/Architecture/Runtime_Security_Model

8.15. BGPAttacks04

Ola Nordstr\&\#246;m and Constantinos Dovrolis
Beware of BGP attacks.
SIGCOMM Comput. Commun. Rev. 34, 2 (April 2004), 1-8. DOI=10.1145/997150.997152
http://doi.acm.org/10.1145/997150.997152

8.16. Binning-top-cloud-issues-09

Bin09 David Binning, Top Five Cloud Computing Security Issues, Computer Weekly, April 24, 2009, <URL: http://www.computerweekly.com/Articles/2010/01/12/235782/Topfive-cloud-computing-security-issues.htm>.

8.17. BONDI

BONDI Architecture and Security Requirements
July 2009
http://bondi.omtp.org/1.01/security/BONDI_Architecture_and_Security_v1_01.pdf

8.18. BONDIv1.1

BONDI Architecture and Security Requirements v1.1
January 2010
http://bondi.omtp.org/1.11/security/BONDI_Architecture_and_Security_v1.1.pdf

8.19. Buschmann1996

Frank Buschmann and Regine Meunier and Hans Rohnert and Peter Sommerlad and Michael Stal
Pattern-oriented software architecture: a system of patterns
Wiley, 1996

8.20. CAIRIS

Computer Aided Integration of Requirements and Information Security
August 2012
http://github.com/failys/CAIRIS

8.21. ChromeNpapiExtensions

Google Chrome Extensions: NPAPI Plugins
Fetched June 2011
http://code.google.com/chrome/extensions/npapi.html

8.22. CAPEC

The MITRE Corporation
Common Attack Pattern Enumeration and Classification (CAPEC) web site
July 2012
http://capec.mitre.org

8.23. Carlini2012

An Evaluation of the Google Chrome Extension Security Architecture
Nicholas Carlini, Adrienne Porter Felt, and David Wagner
USENIX Security Symposium 2012
https://www.usenix.org/conference/usenixsecurity12/evaluation-google-chrome-extension-security-architecture

8.24. ChromeHosted

Hosted Applications Documentation
Google
June, 2012
https://developers.google.com/chrome/apps/docs/developers_guide

8.25. ChromeUserId

Chrome User Identification
Google, 2012
Originally: http://developer.chrome.com/trunk/extensions/apps/app_identity.html
Now: https://developers.google.com/chrome/web-store/docs/identify_user
(On the 30th August Google changed and broke many of their links, causing some of these pages to move or be deleted)

8.26. ChromePackaged

Packaged Apps - Disabled Web Features
Google, 2012
Originally: http://developer.chrome.com/trunk/extensions/apps/app_deprecated.html
(On the 30th August Google changed and broke many of their links, causing some of these pages to move or be deleted)

8.27. ChromeManifest

Chrome Manifests
Google, 2012
Originally: http://developer.chrome.com/trunk/extensions/apps/manifest.html
Now: http://developer.chrome.com/extensions/manifest.html
(On the 30th August Google changed and broke many of their links, causing some of these pages to move or be deleted)

8.28. ChromeCSP

Chrome CSP for Packaged Apps
Google, 2012
Originally: http://developer.chrome.com/trunk/extensions/apps/app_csp.html
(On the 30th August Google changed and broke many of their links, causing some of these pages to move or be deleted)

8.29. ChromePermission

Chrome API Permission Warnings
Google, 2012
Originally: http://developer.chrome.com/extensions/permission_warnings.html
(On the 30th August Google changed and broke many of their links, causing some of these pages to move or be deleted)

8.30. ChromiumSec

Chromium OS Security Overview
Google, 2012
http://www.chromium.org/chromium-os/chromiumos-design-docs/security-overview

8.31. Cloudberry

Taivalsaari, Antero; Systä, Kari.
Cloudberry: An HTML5 Cloud Phone Platform for Mobile Devices
In IEEE Software, Volume 29, Issue 4, August 2012
http://dx.doi.org/10.1109/MS.2012.51

8.32. CloudCompRittinghouse2010

John W. Rittinghouse and James F. Ransome
Cloud Computing. Implementation, Management, and Security
CRC Press, 2010

8.33. CloudSecMather2009

Tim Mather, Subra Kumaraswamy, and Shahed Latif
Cloud Security and Privacy, an Enterprise Perspective on Risks and Compliance
O'Reilly, September 2009

8.34. Cox2006

Cox, Richard S. and Gribble, Steven D. and Levy, Henry M. and Hansen, Jacob Gorm
A Safety-Oriented Platform for Web Applications
In Proceedings of the 2006 IEEE Symposium on Security and Privacy
Pages 350 - 364
IEEE Computer Society Washington, DC, USA
http://dx.doi.org/10.1109/SP.2006.4

8.35. CSA

Cloud Security Alliance
https://cloudsecurityalliance.org/Cloud Security Alliance (CSA)

8.36. CSA-TopThreats

Cloud Security Alliance (CSA),
Top Threats to Cloud Computing V1.0 Prepared by the Cloud Security Alliance, March 2010
https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf

8.37. CSP-MOZILLA

Using Content Security Policy
June 2011
https://developer.mozilla.org/en/Security/CSP/Using_Content_Security_Policy

8.38. CWE

The MITRE Corporation
Common Weakness Enumeration (CWE) web site
July 2012
http://cwe.mitre.org

8.39. CVE-2011-2107

Security update available for Adobe Flash Player: APSB11-13 / CVE-2011-2107
June 2011
http://www.adobe.com/support/security/bulletins/apsb11-13.html

8.40. Dakin2011

Dakin, S.
Privacy Policies, What Good Are They Anyway?
ApplicationPrivacy.org, June 2011
http://www.applicationprivacy.org/?p=764

8.41. DeGrouf2012

Willem De Groef, Dominique Devriese, Nick Nikiforakis and Frank Piessens.
FlowFox: a Web Browser with Flexible and Precise Information Flow Control.
To Appear at the 19th ACM Conference on Computer and Communications Security (ACM CCS)
October 2012
http://www.securitee.org/files/flowfox_ccs2012.pdf

8.42. DHCP The Internet Engineering Task Force (IETF) Dynamic Host Configuration Protocol http://tools.ietf.org/html/rfc2131

8.43. DoNotTrack

J. Mayer, A. Narayanan and S. Stamm
Do Not Track: A Universal Third-Party Web Tracking Opt Out
IETF Network Working Group Internet-Draft
March 7, 2011
http://tools.ietf.org/html/draft-mayer-do-not-track-00

8.44. Draft-iab-priv

A. Cooper et al.
Privacy Considerations for Internet Protocols
draft-iab-privacy-considerations-03.txt
IETF IAB Network Working Group
July 2012
http://www.ietf.org/id/draft-iab-privacy-considerations-03.txt

8.45. dropbox

Dropbox website
https://www.dropbox.com/

8.46. ENISA2011

A Security Analysis of Next Generation Web Standards
ENISA
July 2011
http://www.enisa.europa.eu/act/application-security/web-security/a-security-analysis-of-next-generation-web-standards/at_download/fullReport

8.47. FaceNiff

FaceNiff Website
June 2011
http://faceniff.ponury.net/

8.48. Faily2010a

Shamal Faily and Ivan Flechais
A Meta-Model for Usable Secure Requirements Engineering
Proceedings of the 6th International Workshop on Software Engineering for Secure Systems
IEEE Computer Society, 2010
pp.126--135

8.49. Faily2010b

Shamal Faily and Ivan Flechais
Analysing and Visualising Security and Usability in IRIS
Proceedings of the 5th International Conference on Availability, Reliability and Security
IEEE Computer Society, 2010
pp. 543--548

8.50. Faily2010c

Shamal Faily and Ivan Flechais
Barry is not the weakest link: eliciting secure system requirements with personas
Proceedings of the 24th BCS Interaction Specialist Group Conference
British Computer Society, 2010
pp. 124--132

8.51. Faily2011a

Shamal Faily and Ivan Flechais
User-Centered Information Security Policy Development in a Post-Stuxnet World
Proceedings of the 6th International Conference on Availability, Reliability and Security
IEEE Computer Society, 2011
pp. 716--721

8.52. Faily2011b

Shamal Faily
A framework for usable and secure system design
DPhil thesis University of Oxford, 2011

8.53. Faily2012

Shamal Faily, John Lyle and Simon Parkin
"Secure System? Challenge Accepted: Finding and Resolving Security Failures Using Security Premortems"

To appear in proceedings of the BCS HCI workshop on Designing Interactive Secure Systems. 2012.
http://www.cs.ox.ac.uk/files/4880/premortem.pdf

8.54. Farrell2011

Farrell, N.
More security woes hit Apple's iOS
TechEYE.net, May 2011
http://www.techeye.net/security/more-security-woes-hit-apples-ios

8.55. Firesheep

Butler, E.
Firesheep Website
October 2010
http://codebutler.com/firesheep

8.56. Gamma1995

Erich Gamma and Richard Helm and Ralph Johnson and John Vlissides
Design patterns: elements of reusable object-oriented software
Addison-Wesley, 1995

8.57. Garfinkel1996

Garfinkel, S.
Public key cryptography
Computer, June 1996, 29, 101 -104
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=507642&tag=1

8.58. Garfinkel2005

Garfinkel, S. L.
Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable
PhD Thesis, Massachusetts Institute of Technology, May 2005
http://simson.net/thesis/

8.59. Gennari2012

Jeffrey Gennari and David Garlan
Measuring Attack Surface in Software Architecture
Carnegie Mellon University Technical Report CMU-ISR-11-121, 2012

8.60. Gibraltar

Kaisen Lin, UC San Diego; David Chu, James Mickens, Li Zhuang, Feng Zhao and Jian Qiu
Gibraltar: Exposing Hardware Devices to Web Pages Using AJAX
In the proceedings of the Third USENIX Conference on Web Application Development (WebApps'12), June 2012.
http://research.microsoft.com/apps/pubs/default.aspx?id=161386

8.61. GlobalPlatform2010

TEE Client API Specification
GlobalPlatform Device Technology, Document Reference: GPD_SPE_007
Version 1.0
July 2010
http://www.globalplatform.org/specificationdownload.asp?id=7339

8.62. Gollmann2010

Dieter Gollmann,
Computer Security, 3rd edition,
John Wiley & Sons, 2010.

8.63. Goodin2011

Goodin, D.
Android app brings cookie stealing to unwashed masses
The Register, June 2011
http://www.theregister.co.uk/2011/06/03/android_cookie_stealing_app/

8.64. Goodin2011a

Goodin, D.
Google Web Store quietly purged of nosy apps
The Register, May 2011
http://www.theregister.co.uk/2011/05/26/google_web_store_privacy_threats/

8.65. GoogleNativeClient

Google Native Client (NaCl): Technology Overview
Fetched June 2011
http://code.google.com/intl/de-DE/games/technology-nacl.html

8.66. GuiffySureMerge

Ritcher, B.
Guiffy SureMerge - A Trustworthy 3-Way Merge
September 2004 http://www.guiffy.com/SureMergeWP.html

8.67. ICO-Privacy

Information Commissioner's Office (ICO),
Privacy Impact Assessment Handbook, version 2.0,
http://www.ico.gov.uk/for_organisations/data_protection/topic_guides/privacy_impact_assessment.aspx.

8.68. IETF-TLSWG

IETF Transport Layer Security Working Group,
Fetched June 2011
http://datatracker.ietf.org/wg/tls/charter/

8.69. IntelTXT

Technology Overview: Intel® Trusted Execution Technology
Fetched June 2011
http://www.intel.com/technology/security/downloads/TrustedExec_Overview.pdf

8.70. iOS-TechOverview

iOS Developer Library: iOS Technology Overview Introduction
November 2010
http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

8.71. iPhoneOS-swcuc3m

Escribano, R. M. & Almena, J. P.
iPhone OS Tutorial
Fetched June 2011
https://sites.google.com/site/swcuc3m/home/iphone/iphoneos_en

8.72. Kandukuri-cloud-sec-09

Kandukuri, B.R.; Paturi, V.R.; Rakshit, A.; , "Cloud Security Issues," Services Computing, 2009. SCC '09. IEEE International Conference on , vol., no., pp.517-520, 21-25 Sept. 2009

doi: 10.1109/SCC.2009.84
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5283911&isnumber=5283890

8.73. Khan2011

Mati Ullah Khan and Mansoor Munib and Umar Manzoor and Samia Nefti
Analyzing Risks At Architectural Level
2011 International Conference on Information Society
IEEE Computer Society, 2011
pp. 231--236

8.74. Lederer04

Lederer, S.; Hong, I.; Dey, K. & Landay, A.
Personal privacy through understanding and action: five pitfalls for designers
Personal Ubiquitous Comput., Springer-Verlag, November 2004, 8, 440-454
http://dx.doi.org/10.1007/s00779-004-0304-9

8.75. Leyden2011

Leyden, J.
Wave of Trojans breaks over Android
The Register, June 2011
http://www.theregister.co.uk/2011/06/01/android_trojan_rash/

8.76. Lindholm2001

Lindholm, T.
A 3-way Merging Algorithm for Synchronizing Ordered Trees -- the 3DM merging and differencing tool for XML
Helsinki University of Technology, September 2001
http://www.cs.hut.fi/~ctl/3dm/thesis.pdf

8.77. Lyle2010

Lyle, J.
Trustable Services Through Attestation
DPhil Thesis, Department of Computer Science, University of Oxford, June 2011.
http://www.cs.ox.ac.uk/people/John.Lyle/thesis-final-25-06-11.pdf

8.78. LyleBlog2012

Lyle, J.
Do Garfinkel’s design patterns apply to the web?
(Web page) Department Of Computer Science, University of Oxford
March 2012
http://www.cs.ox.ac.uk/blogs/sss/2012/03/23/garfinkel-design-patterns-for-the-web/

8.79. MacOSX-SecurityArchitecture

Mac OS X Developer Library: Security Architecture
July 2010
http://developer.apple.com/library/mac/#documentation/Security/Conceptual/Security_Overview/Architecture/Architecture.html#//apple_ref/doc/uid/TP30000976-CH202-TPXREF101

8.80. MacOSX-SecurityServices

Mac OS X Developer Library: Security Services
July 2010
http://developer.apple.com/library/mac/#documentation/Security/Conceptual/Security_Overview/Security_Services/Security_Services.html#//apple_ref/doc/uid/TP30000976-CH204-CHDDJIDG

8.81. MAP

TNC IF-MAP Binding for SOAP Specification
Trusted Computing Group Website, Version 2.0, revision 36
http://www.trustedcomputinggroup.org/resources/tnc_ifmap_binding_for_soap_specification

8.82. McGraw2006

Gary McGraw.
Software Security: Building Security In,
Addison-Wesley Longman, Amsterdam, The Netherlands, 2006.

8.83. MeegoSec

The Meego Security Architecture
June 2011
http://wiki.meego.com/Security/Architecture

8.84. MicrosoftPrivacyGuide

Microsoft Corp., Privacy Guidelines for Developing Software Products and Services, v3.1, http://www.microsoft.com/downloads/en/details.aspx?FamilyID=c48cf80f-6e87-48f5-83ec-a18d1ad2fc1f&displaylang=en.

8.85. Mitre

Mitre Corporation
http://www.mitre.org

8.86. MozillaAppManifest

Application Manifest format for Open Web Applications
Mozilla
October 2010
https://wiki.mozilla.org/Labs/Apps/Manifest

8.87. MozillaPluginDirectory

Mozilla Wiki Plugins: Plugin Directory
March 2010
https://wiki.mozilla.org/Plugins:PluginDirectory#Goals

8.88. MozillaPrivacyRoadmap

Mozilla Privacy Roadmap 2011
May 2011
https://wiki.mozilla.org/Privacy/Roadmap_2011

8.89. MozillaProcessIsolation

MozillaWiki: Security Process Isolation
April 2009
https://wiki.mozilla.org/Security/ProcessIsolation

8.90. MozillaWebApi

Web API
Mozilla
August 2012
https://wiki.mozilla.org/WebAPI

8.91. MozillaWebAppSec

MozillaWiki,
WebAppSec/Secure Coding Guidelines,
https://wiki.mozilla.org/WebAppSec/Secure_Coding_Guidelines.

8.92. Munawar2006

Munawar Hafiz,
A collection of privacy design patterns,
Proceedings of the ACM 2006 conference on Pattern languages of programs, October 2006.

8.93. NamilukoVmProvenance2012

**Provenance-Based Model for Verifying Trust-Properties
Cornelius Namiluko and Andrew Martin
TRUST, 2012
http://www.springerlink.com/content/4q296203515854x0/**

8.94. NIST

**National Instituent of Standards and Technology (NIST)
http://www.nist.gov**

8.95. NistCloudDefinition2011

Peter Mell and Timothy Grance
The NIST Definition of Cloud Computing
NIST, 2011
http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

8.96. NoScript

NoScript: JavaScript/Java/Flash blocker for a safer Firefox experience
Fetched June 2011
http://noscript.net/

8.97. O'Brien2011

O'Brien, T.
FaceNiff makes Facebook hacking a portable, one-tap affair
Engadget, June 2011
http://www.engadget.com/2011/06/02/faceniff-makes-facebook-hacking-a-portable-one-tap-affair-vide/

8.98. OMTP

OMTP App Security Framework
Version 2.2
June 2008
http://www.omtp.org/OMTP_Application_Security_Framework_v2_2.pdf

8.99. OpenIDBest

OpenID Security Best Practices
PauAmma, 2010
http://wiki.openid.net/w/page/12995200/OpenID%20Security%20Best%20Practices

8.100.OWASP

OWASP: The Open Web Application Security Project Website
Fetched June 2011
https://www.owasp.org/

8.101.OWASP-ASVS

Open Web Application Security Project (OWASP),
OWASP Application Security Verification Standard 2009, June 2009,
https://www.owasp.org/images/4/4e/OWASP_ASVS_2009_Web_App_Std_Release.pdf.

8.102.OWASP-CRG

Open Web Application Security Project (OWASP),
OWASP Code Review Guide V1.1, February 2009,
https://www.owasp.org/index.php/Category:OWASP_Code_Review_Project.

8.103.OWASP-ESAPI

OWASP ESAPI: The OWASP Enterprise Security API
Fetched June 2011
https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

8.104.OWASP-Guide

Open Web Application Security Project (OWASP).
A Guide to Building Secure Web Applications and Web Services, 27 July 2005,
https://www.owasp.org/index.php/OWASP_Guide_Project#tab=Downloads.

8.105.OWASP-Top10

Open Web Application Security Project (OWASP),
OWASP Top 10 – 2010 – The Ten Most Critical Web Application Security Risks, 2010,
http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf.

8.106.PalmWebOS-swcuc3m

Macias, S. G. & Gutiérrez, B. J.
Palm webOS Tutorial
Fetched June 2011
https://sites.google.com/site/swcuc3m/home/webos/english-version

8.107.PermissionsDAP

Byers, P.; Hirsch, F. & Hazaël-Massieux, D.
Permissions for Device API Access: W3C Working Draft
October 2010
http://www.w3.org/TR/api-perms/

8.108.Pearson2010

Siani Pearson, Yun Shen,
Context-Aware Privacy Design Pattern Selection,
LNCS, Volume 6264, 2010, Springer-Verlag.

8.109.Perry2011

Perry, M.
Improving Private Browsing Modes: "Do-Not-Track" vs Real Privacy by Design
Tor Project Blog, June 2011
https://blog.torproject.org/blog/improving-private-browsing-modes-do-not-track-vs-real-privacy-design

8.110.PorterFelt2011

Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner
A survey of mobile malware in the wild
Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices
October 2011
http://doi.acm.org/10.1145/2046614.2046618

8.111.PorterFelt2012

Adrienne Porter Felt, Serge Egelman, Matthew Finifter, Devdatta Akhawe, and David Wagner
How To Ask For Permission
USENIX Workshop on Hot Topics in Security (HotSec) 2012
http://www.eecs.berkeley.edu/~afelt/howtoaskforpermission.pdf

8.112.PorterFelt2012b

The Effectiveness of Application Permissions
Adrienne Porter Felt, Kate Greenwood, and David Wagner
USENIX Conference on Web Application Development (WebApps) 2011
http://www.cs.berkeley.edu/~afelt/felt-permissions-webapps11.pdf

8.113.PorterFelt2012c

Android Permissions: User Attention, Comprehension, and Behavior
Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin and David Wagner
Symposium on Usable Privacy and Security (SOUPS) 2012
http://www.cs.berkeley.edu/~afelt/felt-androidpermissions-soups.pdf

8.114.PrimeLife

PrimeLife Project Website
Fetched June 2011
http://www.primelife.eu/

8.115.PRiMMA

The Privacy Rights Management for Mobile Applications (PRiMMA) Project Website
Fetched June 2011
http://www.open.ac.uk/blogs/primma/

8.116.Pruitt2006

John Pruitt and Tamara Adlin
The persona lifecycle: keeping people in mind throughout product design
Elsevier, 2006

8.117.rfc2560

Myers, M.; Ankney, R.; Malpani, A.; Galperin, S. & Adams, C.
X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP

The Internet Society, June 1999
http://www.ietf.org/rfc/rfc2560.txt

8.118.rfc5246

Dierks, T. and Rescorla, E.
The Transport Layer Security (TLS) Protocol
IETF Website, version 1.2, August 2008
http://tools.ietf.org/html/rfc5246

8.119.rfc6462

A. Cooper
RFC 6462: Report from the Internet Privacy Workshop
Internet Architecture Board (IAB)
ISSN: 2070-1721
http://tools.ietf.org/html/rfc6462

8.120.Rooney2011

Rooney, B.
More Android Malware Uncovered
The Wall Street Journal Website, TechEurope, June 2011
http://blogs.wsj.com/tech-europe/2011/06/06/more-android-malware-uncovered/

8.121.Sailer2004

Sailer, R.; Zhang, X.; Jaeger, T. & van Doorn, L.
Design and Implementation of a TCG-based Integrity Measurement Architecture
Proceedings of the 13th USENIX Security Symposium, USENIX, 2004, pages 223-238
http://www.usenix.org/publications/library/proceedings/sec04/tech/sailer.html

8.122.Saltzer75

Saltzer, J. & Schroeder, M.
The protection of information in computer systems
Proceedings of the IEEE, September 1975, 63, 1278 - 1308

8.123.SEAndroid

Security Enhanced (SE) Android
August 2012
http://selinuxproject.org/page/SEAndroid

8.124.Shields2011

Shields, T.
Mobile Apps Invading Your Privacy
Veracode: ZeroDa Labs Blog, April 2011
http://www.veracode.com/blog/2011/04/mobile-apps-invading-your-privacy/

8.125.Singh2010

Singh, Kapil; Moshchuk, Alexander; Wang, Helen J.; Lee, Wenke.
On the Incoherencies in Web Browser Access Control Policies
Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP), vol., no., pp.463-478, 16-19 May 2010

8.126.TCG2007

TCG Architecture Overview, Version 1.4
August 2007
http://www.trustedcomputinggroup.org/resources/tcg_architecture_overview_version_14

8.127.TCGMobile

TCG Mobile Trusted Module Specification
Version 1.0, Revision 7.02
29 April 2010
http://www.trustedcomputinggroup.org/resources/mobile_phone_work_group_mobile_trusted_module_specification

8.128.TClouds

TClouds "Trustworthy Clouds" Project
Fetched June 2011
http://www.tclouds-project.eu/

8.129.TizenManifest

Application installation and Manifest
Tizen, August 2012
https://wiki.tizen.org/wiki/Security/Application_installation_and_Manifest

8.130.TizenSec

Tizen Security Framework Overview
Tizen, May 2012
http://download.tizen.org/misc/media/conference2012/tuesday/ballroom-c/2012-05-08-1600-1640-tizen_security_framework_overview.pdf

8.131.TrustZone

ARM TrustZone Webpage and API
Fetched June 2011
http://www.arm.com/products/processors/technologies/trustzone.php

8.132.Understanding-cloud-vulns

Grobauer, B.; Walloschek, T.; Stocker, E.; , "Understanding Cloud Computing Vulnerabilities," Security & Privacy, IEEE , vol.9, no.2, pp.50-57, March-April 2011
doi: 10.1109/MSP.2010.115
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5487489&isnumber=5739630

8.133.USPatrioticAct-UnderFire-04

USA Patriot Act Comes under Fire in B.C. Report, CBC News, October 30, 2004,
URL: http://www.cbc.ca/canada/story/2004/10/29/patriotact_bc041029.html

8.134.VanLamsweerde1998

Axel van Lamsweerde and Emmanuel Letier
Integrating obstacles in goal-driven requirements engineering
Proceedings of the 1998 International Conference on Software Engineering
IEEE Computer Society, 1998
pp. 53--62

8.135.VanLamsweerde2009

Axel van Lamsweerde
Requirements Engineering: from system goals to UML models to software specifications
John Wiley & Sons, 2009

8.136.W3CApiPriv

Proceedings of the W3C Workshop on Privacy for Advanced Web APIs
12/13 July 2010, London
http://www.w3.org/2010/api-privacy-ws/papers.html

8.137.W3CApiSec

http://www.w3.org/2010/api-privacy-ws/papers.html
Proceedings of the W3C Workshop on Security for Access to Device APIs from the Web
December 2008
http://www.w3.org/2008/security-ws/papers/

8.138.W3CAppPriv

Frederick Hirsch
Web Application Privacy Best Practices
W3C Working Group Note
03 July 2012
http://www.w3.org/TR/2012/NOTE-app-privacy-bp-20120703/

8.139.W3CDAP-Perms

Byers, P; Hirsch, F & Hazaël-Massieux, D.
Permissions for Device API Access, W3C Working Draft
W3C Website, October 2010
http://www.w3.org/TR/api-perms/

8.140.W3CDataMinimization

W3C Technical Architecture Group (TAG),
Data Minimization in Web APIs,
http://www.w3.org/2001/tag/doc/APIMinimization.html.

8.141.W3CFeaturePermissions

Gregg, J. & Gombos, L.
Feature Permissions, W3C Editor's Draft
W3C Website, May 2011
http://dev.w3.org/2009/dap/perms/FeaturePermissions.html

8.142.W3CMobileWebApp

W3C Recommendation,
Mobile Web Applications Best Practices,
http://www.w3.org/TR/mwabp/.

8.143.W3CPatternsPriv

Robin Berjon and Daniel Appelquist
Patterns for Privacy by Design in Javascript APIs
W3C Draft TAG Finding
06 June 2012
http://www.w3.org/2001/tag/doc/privacy-by-design-in-apis

8.144.W3CWARP

W3C Widget Access Request Policy Candidate Recommendation
W3C Website, April 2010
http://www.w3.org/TR/widgets-access/

8.145.WAC

WAC 2.0 Core Specification: Widget Security and Privacy
January 2011
http://www.wacapps.net/web/portal/wac-2.0-spec

8.146.Wang2009

Wang, Helen J. and Grier, Chris and Moshchuk, Alexander and King, Samuel T. and Choudhury, Piali and Venter, Herman
The multi-principal OS construction of the gazelle web browser
Proceeding of the 18th USENIX security symposium (SSYM'09), 2009
Pages 417-432

8.147.Webinos-D21

Webinos Deliverable: D2.1 Use Cases and Scenarios
January 2011
http://webinos.org/archives/744

8.148.Webinos-D22

Webinos Deliverable: D2.2 Requirements & developer experience analysis
March 2011
http://webinos.org/archives/704

8.149.Webinos-D25

Webinos Deliverable: D2.5 Updates on Requirements and Available Solutions
February 2012

8.150.Webinos-D27

Webinos Deliverable: D2.7 User Expectations of Security and Privacy
March 2011
http://webinos.org/blog/2011/03/16/d2-3-industry-landscape-ipr-licensing-governance/

8.151.Webinos-D28

Webinos Deliverable: D2.8 User Expectations of Security and Privacy (update)
June 2011
http://webinos.org/blog/2011/11/01/webinos-repot-user-expectations-of-security-and-privacy-phase-2/

8.152.Webinos-D31

Webinos Deliverable: D3.1 System Specifications
August 2011
http://webinos.org/blog/2011/11/01/webinos-report-phase-i-architecture-and-components/

8.153.Webinos-D32

Webinos Deliverable: D3.2 API Specifications
August 2011
http://webinos.org/blog/2011/11/01/webinos-report-phase-i-device-network-and-server-side-api-specifications/

8.154.Webinos-D33

Webinos Deliverable: D3.3 System Specifications
Working Draft
August 2012

8.155.Webinos-D34

Webinos Deliverable: D3.4 API Specifications
Working Draft
August 2012

8.156.Webinos-D35

Webinos Deliverable: D3.5 Security and Privacy Framework
August 2011
http://webinos.org/blog/2011/08/04/webinos-report-phase-1-security-framework/

8.157.WebOSIntro

Mobile Application Security : WebOS Security - Introduction to the Platform
February 2011
http://programming4.us/mobile/3158.aspx

8.158.WebSand

WebSand Project Website
August 2012
https://www.websand.eu/

8.159.WidgetSignatures

Cáceres, M.; Byers, P.; Knightley, S.; Hirsch, F. & Priestley, M.
XML Digital Signatures for Widgets: W3C Working Draft
June 2011
http://www.w3.org/TR/widgets-digsig/

8.160.WidgetUpdates

Cáceres, M.; Tibbett, R. & Berjon, R.
Widget Updates: W3C Working Draft
September 2010
http://www.w3.org/TR/widgets-updates/

8.161.XACML

Moses, T.
eXtensible Access Control Markup Language (XACML) Version 2.0

February 2005
http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

8.162.Zhou2012

Yajin Zhou, Xuxian Jiang
Dissecting Android Malware: Characterization and Evolution
pp.95-109, 2012 IEEE Symposium on Security and Privacy, 2012
http://www.computer.org/portal/web/csdl/doi/10.1109/SP.2012.16

9 Appendices

10 Background

To provide context for the security and privacy analysis in webinos we give the following background summary of related projects, academic papers and webinos deliverables.

10.1.1. Related Security and Privacy Architectures in Industry

This page summarises the main related work in industry on mobile web application security and privacy, as well as some summaries of how this has affected webinos design.

10.1.1.1 Web Application Systems

10.1.1.1.1 WAC and BONDI Security architectures

- WAC Core Specification: Security and Privacy (WAC)
- OMTP App Security Framework (OMTP)

From WAC we borrow much of the widget handling specifications and installation behaviour with regards to signatures and identities. The webinos policy architecture is also based on the WAC approach, but with modifications for cross-platform flows and privacy.

10.1.1.1.2 Google Chrome

- Hosted Applications Documentation (ChromeHosted)
- Chrome User Identification (ChromeUserId)
- Packaged Apps - Disabled Web Features (ChromePackaged)
- Chrome Manifests (ChromeManifest)
- Chrome CSP for Packaged Apps (ChromeCSP)
- Chrome API Permission Warnings (ChromePermission)

From these references we intend to follow recommendations on limitations to javascript within web applications, such that inline javascript and 'eval' statements are not permitted.

10.1.1.1.3 Boot2Gecko / FirefoxOS / OpenWebDevice and Mozilla Web APIs

- Boot 2 Gecko Runtime Security (B2GRuntimeSec)
- Boot 2 Gecko App Security (B2GAppSec)
- Boot 2 Gecko App Security Model Discussion (B2GAppSecDis)
- Application Manifest format for Open Web Applications (MozillaAppManifest)
- Mozilla Web APIs (MozillaWebApi)

10.1.1.1.4 Recommendations and specifications on web privacy from the W3C and IETF

- Patterns for Privacy by Design in Javascript (W3CPatternsPriv)

- Report from the Internet Privacy Workshop (rfc6462)
- Privacy Considerations for Internet Protocols (Draft-iab-priv)
- Do Not Track: A Universal Third-Party Web Tracking Opt Out (DoNotTrack)

Based on these guidelines, we intend to try and avoid introducing APIs which allow for fingerprinting without any form of user consent. I.e., web applications should not be able to gain additional information about a user, or any form of user identity, unless the policy architecture allows it. We also intend to make information flows more visible to users by encouraging developers to explain their data handling policies when requesting data.

10.1.1.1.5 Security analysis by ENISA

We focus on one of the recommendations given by ENISA in their report (ENISA2011) - End User Policing. From the report:

> Several specifications depend on the end user to ensure security, typically by means of a permission system. With the rapid development of new APIs offering new functionality, this approach is reaching its limits. Typical web users are unable to assess the ramifications of granting certain permissions to certain origins. Additionally, involving the user too often might quickly lead to blindly approving permissions.
>
> A first recommendation is to require an awareness indicator, so the user knows when a site is using a granted permission (e.g. when a site is locating the user). Currently, this is only included as a suggestion in the Geo-location API and is missing from other APIs, such as the System Information API.
>
> A second recommendation is to offer the users a way to select predefined security profiles or to create a security profile (e.g. by means of a wizard, which could explain the ramifications of sharing your location and then provide the option to allow this or not.). Once a security profile is chosen, it is used to determine the appropriate actions when a site requests permission for an action.

10.1.1.1.6 Web intents

An analysis of web intents security issues can be found in the D3.4 deliverable.

10.1.1.2 Mobile application security and privacy frameworks

While the following projects are less directly applicable, they provide an important context for webinos applications. In particular, we borrow concepts from the Android permission system.

10.1.1.2.1 Android

- Android Security Overview (AndroidSecurityOverview)
- Security Enhanced Android (SEAndroid)

10.1.1.2.2 iOS

- iOS Security (AppleIosSec)

10.1.1.2.3 Chromium OS

- Chromium OS Security Overview (ChromiumSec)

10.1.1.2.4 Meego security architecture

- The Meego Security Architecture (MeegoSec)

10.1.1.2.5 Tizen security architecture

- Security Overview Presentation (TizenSec)
- Application installation and Manifest (TizenManifest)

10.1.1.3 Summary

The webinos security framework is informed by all of the sources given above. In particular, we make use of recommendations from WAC, Chrome, W3C and ENISA to try and avoid threats that they have identified.

10.2. Related Academic Work

This section covers the key related academic material and concludes with an explanation for how these have been taken into account in the webinos security framework.

10.2.1. Papers

10.2.1.1 Exposing additional capabilities to web browsers

The Gibraltar system (Gibraltar) takes a similar approach to webinos in exposing device-level capabilities to web browsers. This work takes a capability approach (issuing tokens to trusted sites) in contrast to the policy-based system implemented in webinos.

10.2.1.2 Cross-device synchronisation

The Cloudberry system (Cloudberry) is related to webinos but is more cloud-driven.

10.2.1.3 Access control, permissions and API security

- "On the Incoherencies in Web Browser Access Control Policies," (Singh2010)
- "How to ask for Permission" (PorterFelt2012)
- "Android Permissions: User Attention, Comprehension, and Behavior" (PorterFelt2012b)
- "The Effectiveness of Application Permissions" (PorterFelt2012c)

- "An Evaluation of the Google Chrome Extension Security Architecture" (Carlini2012)
- "Privilege Separation in HTML5 Applications" (Akhawe2012)

10.2.1.4 Web Browsers

- The Gazelle Web Browser (Wang2009)
- The Tahoma Web Browser (Cox2006)
- FlowFox (DeGrouf2012)

10.2.1.5 Mobile Malware

- An analysis of Android malware is presented in (Zhou2012)
- A wider survey of mobile malware can be found in (PorterFelt2011)

10.2.1.6 Principles and Design patterns

- Chapter 10 of (Garfinkel2005), and an analysis of their application to the web by The University of Oxford (LyleBlog2012).
- We aim to avoid the Privacy Pitfalls described by Lederer et al. (Lederer04)
- We also aim to follow the principles outlined in the classic Saltzer and Schroeder paper (Saltzer75).

More details of our guiding principles are available in the D3.5 deliverable.

10.2.2. Workshops

W3C API Security and Privacy Workshops from 2010 and 2008

- (W3CApiSec)
- (W3CApiPriv)

10.2.3. Recommendations for webinos

- The principle of least privilege should be followed through the policy architecture.
- Motivations and attacks based on existing mobile malware should be fed into risk analysis.
- Recommendations for web application based on the Google Chrome Extension Security Architecture should be considered as part of the security architecture.
- Garfinkel's design patterns - install before execute - are relevant for webinos.
- We recommend the use of status icons ("Sensor Widgets") based on the Gibraltar paper for consideration as a future webinos security control.

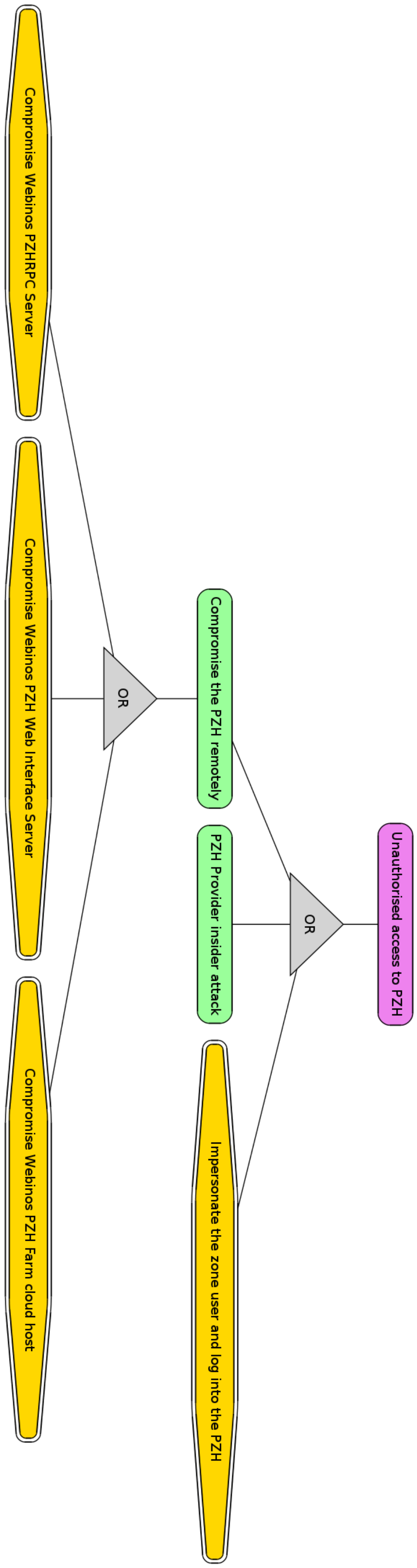
10.3. Related Webinos Deliverables

Before compiling this document we used input from Deliverable 2.7 and 2.8: User Expectations of Security and Privacy

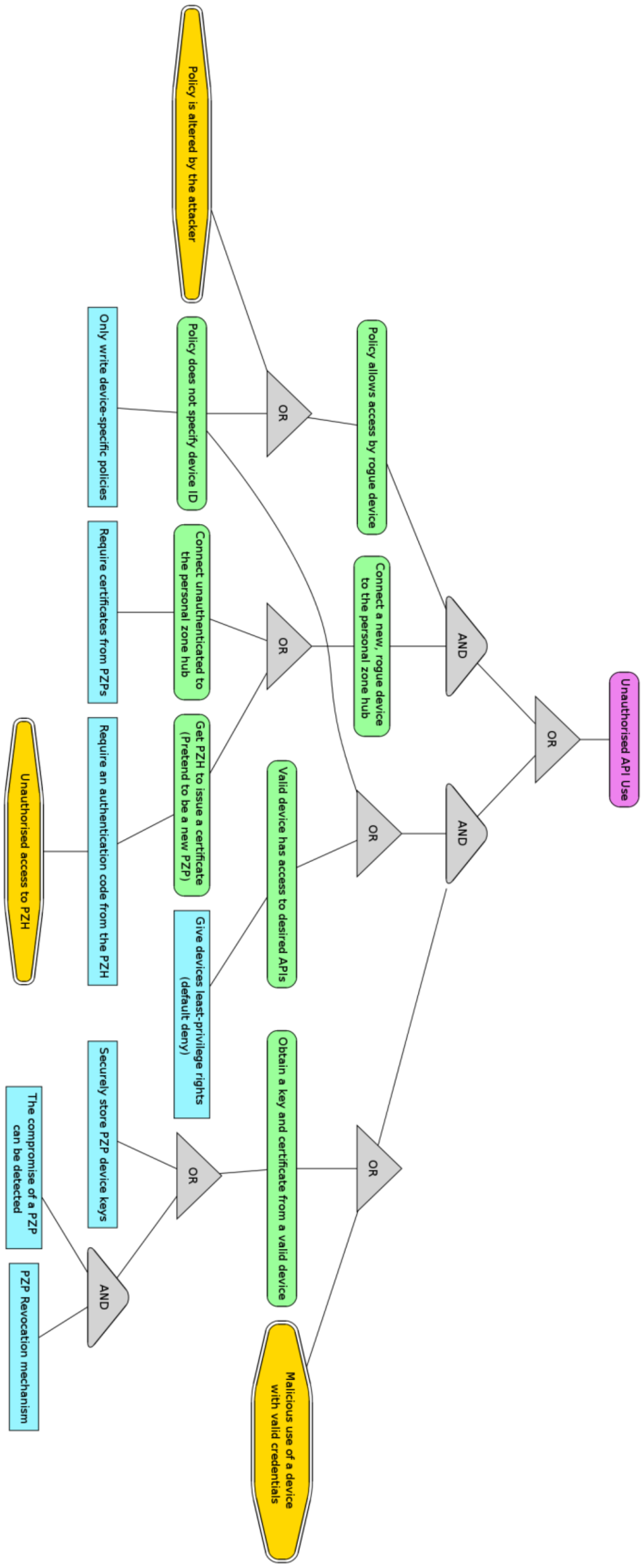
- *Personas* to contextualise attacks and risks
- *Attacker personas* to provide motivation and potential capabilities of attackers
- *Misuse cases* and *Tasks* to provide inspiration for attack patterns
- *Environment models* and *Asset models* to inform the architectural patterns
- *Attack trees* to generate attack patterns

As part of this we also developed several more attack trees before beginning our main architectural risk analysis. They are included here as an addendum.

10.3.1. Unauthorised access to APIs


Compromise Webinos PZHRPC Server
Compromise Webinos PZH Web Interface Server
Compromise Webinos PZH Farm cloud host
OR
Compromise the PZH remotely
PZH Provider insider attack
Impersonate the zone user and log into the PZH
OR
Unauthorised access to PZH

10.3.2. Unauthorised access to the PZH


Unauthorised API Use
OR
AND
AND
Policy allows access by rogue device
Connect a new, rogue device to the personal zone hub
OR
OR
OR
Policy is altered by the attacker
Policy does not specify device ID
Only write device-specific policies
Connect unauthenticated to the personal zone hub
Require certificates from PZPs
Get PZH to issue a certificate (Pretend to be a new PZP)
Require an authentication code from the PZH
Unauthorised access to PZH
Valid device has access to desired APIs
Give devices least-privilege rights (default deny)
OR
Obtain a key and certificate from a valid device
Malicious use of a device with valid credentials
OR
Securely store PZP device keys
AND
The compromise of a PZP can be detected
PZP Revocation mechanism

11 API Security Analysis

This section describes an individual analysis of each API and the security and privacy issues they have.

11.1. Authentication API

11.1.1. Overview of API

http://dev.webinos.org/specifications/new/authentication.html

Provides information to applications about the current authentication status of users, based on their authentication to the local device. This API may also be used within webinos to ask the runtime to authenticate the user.

11.1.2. Threats

11.1.2.1 API-Specific threats and misuse cases

1. This API could be used to monitor the current state of the user. e.g., requesting an authentication might show that the end user is present.
2. Authentication method and Last Auth Time could be used for fingerprinting the user.
3. If this is mapped to device authentication, users could be inundated with authentication requests. It also might confuse them into clicking-through or automatically entering passwords when prompted. This would be a particularly big problem if this password/code is reused on other systems.
4. The authentication prompt could be spoofed by an application.

11.1.2.2 Threats based on remote invocation of this API from another device

1. Remote invocation could be misused to monitor the user's activity - e.g., to work out whether the user was active on the device.
2. Misuse of authentication - if all authentication requests are delegated to a device which does not support (or only weakly supports) the Authentication API, this might be unexpectedly weak or strong.

11.1.2.3 Implementation threats and possible attacks

1. The current implementation means that authentication applies to all apps at once - a user authenticating in one App will automatically be authenticated for others. This may be confusing if the API is misused as authorisation.

11.1.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. The use of this API could create false expectations for developers. It is meant to be used to re-assert the users identity as the owner of the PZP in question. However, it is not supposed to be a strong authentication statement, as the method will be device-specific. Developers must be aware that it is weak, and may not correlate with the current user precisely. It should not be used to protect high-value or high-risk events.
2. Similarly, authentication is not necessarily authentication for the current action - a user may have authenticated for another reason. It should not be confused with authorisation.
3. Authentication time-outs and delays are platform-specific, and developers need to be aware that they should check the 'lastAuthTime' parameter to see how fresh it is.

11.1.2.5 Threats to device manufacturers, operators, other stakeholders

11.1.3. Mitigations

1. Provide examples for the correct scenarios in which the API should be used. Make sure that authentication is not confused with authorisation. Consider adding authorisation methods to this API.
2. Allow plausible deny-ability - callbacks should trigger after timeouts of random periods (within certain limits) to avoid user monitoring.
3. Secure UI - make sure the webinos prompt is not spoofable.
4. Do not make use of the authentication credentials in any way.
5. Make sure that authentication API implementations do not allow for excessive querying of the user. Authentication caching is required.
6. Make explicit which application is asking for authentication when authentication is requested.

11.1.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
	x		Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
x		x	Default allow (install	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will	Webpages will prompt for consent (Yes / No / Always) at runtime, this can be just a

	time)	automatically be filled in).	warning rather than a prompt
	Default ask at runtime (one-time)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Webpages will prompt for consent (Yes / No / Always) at runtime, but the PZP will remember the setting.
	Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Webpages will prompt for consent (Yes / No / Always) at runtime
	Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Webpages will display a short notification at first-use saying that access was denied, with a button to change settings
	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.2. Context API

11.2.1. Overview of API

http://dev.webinos.org/specifications/new/context.html

Allows access to a user’s context data through either explicit queries or a subscription model. Context events include all API calls.

11.2.2. Threats

11.2.2.1 API-Specific threats and misuse cases

1. Privacy loss - any application can monitor what the user is doing and has done, and can react to particular events. This might be used for targeted adverts, physical or cyber stalking, targeted theft or burglary, identity theft.
2. Confidentiality loss - any application potentially see which files have been opened, where the user has been, contact information, etc. Depending on implementation, this could include the content of files and more. An application might use this to gain access to APIs it does not have permission for.
3. Non-repudiation. Users may want to go unmonitored and need to turn off context collection at times. If this is hard to do or unclear, it might result in embarrassment, loss of reputation, etc.

11.2.2.2 Threats based on remote invocation of this API from another device

Because this API primarily uses a central database, remoting has no obvious implications.

11.2.2.3 Implementation threats and possible attacks

1. Availability - subscribing to common context events might slow down the platform considerably, rendering it unusable.
2. Availability - too many queries to the context API might over-use bandwidth and cause either a loss of battery power of expensive mobile phone bills

11.2.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. Context data could be inconsistent or misused to provide a false impression for developers. For example, if a user turns on and off their context data, it may make them appear to have different behaviour to reality.

11.2.2.5 Threats to device manufacturers, operators, other stakeholders

1. Availability - too many queries to the context API might over-use bandwidth and reduce battery life.

11.2.3. Mitigations

1. Turn off context collection by default
2. Provide controls for turning on/off collection and clearing the database
3. Optionally - provide more granular controls for users who are aware of aspects of their context they are happy to log to the database and make available. Allow for selective deletion.
4. Provide feedback for when applications query context data
5. Clean context data so that only a bare minimum of information about API calls is collected. This could be implemented through data tagging.
6. Integrate context querying permissions with permission to access the underlying data. Investigate how the data in the Context Database can be classified based on its sources.

11.2.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-	W-	W-	Policy	Explanation (Widgets)	Explanation (Browser Apps)

A	R	U			
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
x	x		Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
		x	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.3. AppLauncher API

11.3.1. Overview of API

http://dev.webinos.org/specifications/new/launcher.html

Provides the capability to check if a webinos app is present on the device and to start it.

11.3.2. Threats

11.3.2.1 API-Specific threats and misuse cases

1. Start applications on the device until the system crashes;
2. Start a malicious application and send it to background; when it prompts for a permission, the user may allow it thinking it is requested by the main application;
3. Checking which applications are installed on the device can reveal personal information;

4. Using the AppLauncher API to invoke a known-vulnerable application and then stealing data or misusing its access to credentials.

11.3.2.2 Threats based on remote invocation of this API from another device

Remotely launching applications may cause excessive battery usage from the target device, or may be used as part of any of the above threats, but with the added difficulty that the user may not see what is happening.

11.3.2.3 Implementation threats and possible attacks

11.3.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

11.3.2.5 Threats to device manufacturers, operators, other stakeholders

1. Starting applications that overload cpu, memory, network access can cause device dysfunctions; this can result in bad publicity for the manufacturer. This may also cause additional cost in fixing devices with malicious applications installed.

11.3.3. Mitigations

1. the user must be able to select which applications can be executed by a local or remote entity;
2. this can be achieved with a ui showing the list of applications; the user can enable each app to be locally or remotely executed (and select which entity can execute them);
3. make sure that the policy prompt clearly states which is the app requesting it;
4. Make sure that an explicit consent step is required before an application is launched. This consent request should include the calling application's name as well as the target application.

11.3.4. Recommended default policy rules for applications

Only applications selected by the user should be checkable or runnable (localy or remotely). This means that the policy must have a parameter identifying the invoked application.

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

Policy for general applications:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
X	X		Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
		X	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

Policy for user selected applications:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
X	X		Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime	Widgets will require one-off user consent at runtime. This fact will	Web pages will prompt for consent (Yes / No / Always) at

	(one-shot)	be visible & modifiable at install time.	runtime, this preference will be saved.
X	Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
	Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.4. Payment API

11.4.1. Overview of API

http://dev.webinos.org/specifications/new/payment.html

"This API provides generic shopping basket functionality to provide in-app payment. It is not linked to a specific payment service provider and is designed to be sufficiently generic to be mapable to various payment services like GSMA OneAPI, BlueVia, Android Payment API or PayPal."

11.4.2. Threats

11.4.2.1 API-Specific threats and misuse cases

- Misuse case from 2.8
- Misuse case from 2.8
- An application could add items to the shopping basket with user consent. This could result in the user being charged more money or purchasing something they didn't mean to.
- An application could remove items from the shopping basket without user consent. This would be irritating, and might put them off using this application or payment provider.
- An application could modify the item price, making it cost more for the end user. Alternatively, it could mean that the user thinks something is cheap. Or it could defraud the merchant.
- An application could fail to update the basket after a change, resulting in an incorrect total amount being displayed.
- An application could accidentally double-add items to the basket
- An application could use an inappropriate out-of-band authenticator with the payment provider, resulting in unauthorised payment

- The payment provider might use SMS-based authentication, which webinos could intercept and allow an attacker who has stolen one device to buy lots of things.
- The invocation of the API may be logged by the context DB. This might mean that a item purchased privately or semi-anonymously is then recorded in the user's personal history. This could cause embarrassment.
- An application might create and load a shopping basket without user consent.
- The process of authenticating might reveal user contact details or address information.
- The API could accidentally link payments through different mechanisms (e.g. PayPal and Visa) which would otherwise be anonymous.
- A child might misuse an app with access to the payment API - e.g. downloading pay-per-view content.

11.4.2.2 Threats based on remote invocation of this API from another device

- The API could be invoked remotely on another device owned by the user, resulting in the payment provider's authentication mechanism appearing on the wrong device. This could either be misused to trick the user into purchasing the wrong thing, or make it impossible for the user to properly authenticate.
- The API could be invoked on someone else's device, taking advantage of any credentials they have for authenticating to the payment provider.
- If the webinos web/widget runtime allowed unauthorised cross-origin communication, then the App might be able to eavesdrop or intercept key presses / input from the user during payment authentication.

11.4.2.3 Implementation threats and possible attacks

11.4.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

- A failure of the payment API could result in the app developer missing out on revenue. E.g., the payment API does not work, or results in difficult to use behaviour, resulting in the user cancelling the payment.
- Misuse of the payment API might result in the developer appearing fraudulent.

11.4.2.5 Threats to device manufacturers, operators, other stakeholders

- Misuse of the payment API would cause bad publicity for any operator or manufacturer using/shipping webinos on their handsets.

11.4.2.6 Further considerations

- The payment provider's authentication mechanism may be fundamentally flawed, or make assumptions that do not hold in webinos.
- Payment doesn't fundamentally introduce anything new in webinos compared to the browser.

11.4.3. Mitigations

- Remove from webinos? Doesn't add any innovation, does add complexity and an attack vector?
- Do not allow remote invocation
- Require user consent and authentication before invoking the checkout operation. However, this ought to be provided by the service rather than necessarily by the policy framework.
- Recommend that a custom analysis on each payment provider is performed, to check that webinos is not introducing new attack vectors.

11.4.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
X	X	X	Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not	

be asked at install time. This can only be modified using a policy editor.

Rationale: the payment API does not introduce anything new compared to regular browser payment.

11.5. Discovery API

11.5.1. Overview of API

http://dev.webinos.org/specifications/new/servicediscovery.html

"The Webinos Discovery API provide web applications with an API to discover services without any previous knowledge of the service. The Discovery API is not limited to discovery of local services but also enables discovery of remote services."

11.5.2. Threats

11.5.2.1 API-Specific threats and misuse cases

1. Privacy loss – any application can read out which type of services that are available in the personal zone or on local networks. The service description contains information such as service type, name and a description. This information could contain the model name of your home TV, which type of personal phone you have in your zone. This information can be used by applications for fingerprinting. Potential candidates are targeted advertising, targeted theft or burglary. If an application is authorized access to a service availability could be used to monitor availability of different services. This can be used to estimate what an end user is doing and potential whether the end user is at home or not.
2. Denial of Service – any application that is accidently authorized access to a service in the personal, could use this for DoS attacks and service Hijacking.
3. Masquerade service access – when finding a services the application can fake which services that are found and give the end user the impression that service x is selected whilst service y is choosen by the application.

11.5.2.2 Threats based on remote invocation of this API from another device

The discovery API is an enabler for doing remote invocation but the API as such should not be possible to invoke remotely.

11.5.2.3 Implementation threats and possible attacks

1. Currently there are no limitations in number of discovery operations that can be invoked in parallel. This could be misused, resulting in denial of service or exploiting a bug in an underlying service platform.

11.5.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. Unreliable application behaviour. If the discovery API produces unreliable results then the application will appear unreliable and may be unpopular with users.
2. Discovery of privacy-invasive or insecure services. Application developers might be blamed for their applications making use of services which are invasive of their privacy or security. The discovery API has no way of presenting this information to the application.
3. Inappropriate use of an application-defined service. If Jimmy develops a service designed to take, for example, public data about sports results with very few security implications he may get in trouble if it is misused by users for private data.

11.5.2.5 Threats to device manufacturers, operators, other stakeholders

1. The API defines that service availability shall be reported as events. This could generate a lot of traffic of there a lot of services running in parallel and could potentially be devastating for the power consumption and load the network with traffic.

11.5.2.6 Further considerations

11.5.3. Mitigations

1. The API shall only be available for authorized applications or applications trusted by the end user.
2. Provide feedback when services are discovered and which services that are discovered.
3. There shall be means for end user to easily revoke access to the discovery API

11.5.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
	X		Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.

X	X	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
		Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
		Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
		Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.6. Generic Sensor API

11.6.1. Overview of API

Link to API - http://dev.webinos.org/specifications/new/sensors.html

The Webinos Generic Sensor API provides web applications with an API to access data from sensors in the device, connected to the device or in another device.

The API consists of two interfaces:

1. A sensor interface that provides attributes for the sensors and a method to configure a selected sensor.
2. A DOM level 3 event that provides sensor data.

11.6.2. Threats

11.6.2.1 API-Specific threats and misuse cases

1. Sensors may be receiving data from a wide range of sources or inputs. These might potentially be privacy-invasive, e.g.,
    1. A power-usage sensor might indicate the user's current activity
    2. A sleep monitor (motion sensor) has some obvious privacy issues
    3. Sensors might provide clues to the location of the end user - e.g. a temperature sensor at a known location during winter would expect to be warm when people are

nearby and cold otherwise. This might be used by thieves to determine whether a house is occupied before a burglary.

   4. Sound sensors might be able to overhear conversations

2. Sensors might produce a large amount of data and therefore overuse the bandwidth or storage capacity of the device, resulting in slow network/device, high costs or denial of service altogether.
3. Sensors may be more or less accurate than expected. If used for security-related tasks (such as where a door is open or closed, or in establishing user presence) this could cause security violations.

There are likely to be more threats for each type of sensor - this threat analysis is vague by definition.

11.6.2.2 Threats based on remote invocation of this API from another device

1. Access to sensors on remote devices may be unexpected by the end user. They may not expect an application on a device in a different location to have access to this information
2. Remote eavesdropping or spying on someone else may be enabled by this device, even accidentally. For example, in a shared house it might inform one occupant of the activities of the other by accident. This could cause embarrassment or expose sensitive information. The assumption that a sensor is only accessible for the place where the sensor is has been violated

11.6.2.3 Implementation threats and possible attacks

This API has not been implemented.

Suspected implementation issues:

- The underlying sensor is going to be controlled by a device driver. This driver may be running in kernel mode. It may have implementation flaws which are exploitable from API invocation, such as buffer overruns or double free vulnerabilities. This could cause remote code execution from webinos APIs.

11.6.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. No assurance is provided to developers of the trustworthiness of the sensor. This should not be relied upon by developers.
2. Apps could be fed a large stream of fake or unhelpful data by this API.

11.6.2.5 Threats to device manufacturers, operators, other stakeholders

1. Cross-zone API use could result in high network traffic, particularly if combined with the context API

2. If the sensor relates to the device itself - e.g. a sensor recording engine temperature - it could cause end users to believe their devices are faulty when they are not, or vice versa. This would be a problem for a manufacturer.

11.6.3. Mitigations

1. When adding a sensor to webinos, make it clear that this sensor is accessible from any device in the personal zone and provide users with the opportunity to disable this feature.
2. If the sensor relates to part of a product and is known to be unreliable, the manufacturer may want to allow access to it only from applications with the manufacturer's signature.
3. When roaming or using a mobile network, the amount of data being sent by this API might be limited
4. By default, user consent should be required when accessing this API both locally and remotely.
5. Use of this API by an application should be logged
6. Recommend to implementers of sensors that any free-form untyped text input (sensorType and sensorId in the initSensorEvent method) should be validated properly.
7. Recommend to implementers that rate limits are set where appropriate on sensors.
8. Display visual warnings on the device with the sensor when the sensor is in use, to avoid eavesdropping / privacy invasion by a remote or local app.

11.6.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

For each application type, select one of these as the default policy for this API. This applies to an application which explicitly requests access to this API.

Local access:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be

				automatically be filled in).	saved.
x	x	x	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

Remote access:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
x	x	x	Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny	Widgets will require user consent	Web pages will display a short

(install time)	at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	notification at first-use saying that access was denied, with a button to change settings
Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.7. Generic Actuator API

11.7.1. Overview of API

Link to API - http://dev.webinos.org/specifications/new/actuators.html

11.7.2. Threats

11.7.2.1 API-Specific threats and misuse cases

1. Misuse of an actuator could damage the underlying device or local environment. E.g., a motor could be used so frequently it gets too hot.
2. Misuse of a battery-powered actuator could cause a denial of service through exhausting the battery.
3. An actuator could be capable of harming people or the environment. E.g., actuators for home automation could cause temperatures to drop or rise too high
4. Actuators may through success or error callbacks reveal information about the underlying device which could, in turn, be privacy sensitive. E.g. an actuator that has recently been used might be unusable for several minutes. This would therefore be a covert channel.
5. An expensive to use actuator (e.g. one with a support service or one which uses a limited supply of raw materials) could be expensive for the end user if abused.
6. An application or webinos flaw could allow an attacker to misuse **all** the actuator APIs at once. This would be spectacular!

There are likely to be more threats for each type of actuator - this threat analysis is vague by definition.

11.7.2.2 Threats based on remote invocation of this API from another device

1. Remote invocation of an actuator could cause alarm or surprise by a user who is not expecting it.
2. Remote invocation could cause battery life issues or denial of service if the user did not realise they were remotely invoking it.
3. Remote invocation greatly increases the likelihood and success of many of the denial of service threats.

11.7.2.3 Implementation threats and possible attacks

This API has not been implemented.

Suspected implementation issues:

- The underlying actuator is going to be controlled by a device driver. This driver may be running in kernel mode. It may have implementation flaws which are exploitable from API invocation, such as buffer overruns or double free vulnerabilities. This could cause remote code execution from webinos APIs.

11.7.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. The developer might design for the wrong kind of actuator and therefore accidentally damage the actuator that the API uses. They could be sued for this by angry users

11.7.2.5 Threats to device manufacturers, operators, other stakeholders

1. Actuators related to a device might be damaged through unauthorised or inappropriate use. This could cause expense or product recalls.

11.7.3. Mitigations

1. When the actuator is used it ought to notify the user on the device he or she is using it from as well as the device it is connected to.
2. Recommend to implementers of actuators that any free-form untyped text input (actuatorType and actuatorId in the initActuatorEvent method) should be validated properly.
3. Recommend to implementers that rate limits are set where appropriate on actuators.
4. User consent should be required *at least* the first time the actuator is used
5. When adding a new actuator, users should be alerted to warnings and, by default, policies should disallow access to this API remotely.
6. Use of this API by an application should be logged
7. If the actuator is expensive to use, or could be damaged through misuse, the manufacturer may want to allow access to it only from applications with the manufacturer's signature. A default policy might be bundled with the actuator driver

11.7.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

Local access:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
x	x	x	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

Remote access:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask	Widgets will require one-off user	Web pages will prompt for

			at runtime (one-shot)	consent at runtime. This fact will be visible & modifiable at install time.	consent (Yes / No / Always) at runtime, this preference will be saved.
x	x	x	Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.8. Messaging API

11.8.1. Overview of API

http://dev.webinos.org/specifications/new/messaging.html

Provides the capability to read, receive and send sms, mms, email and instant messages.

11.8.2. Threats

11.8.2.1 API-Specific threats and misuse cases

1. an application (or remote user) can read user sms/mms/emails/im; they may contain sensitive data (for example important information about their personal life or work activities).
2. an application (or remote user) can send sms/mms/emails; this may be used, for example, to subscribe the user to a paid service.
3. an application can modify/replace the message or the recipient list before sending it, causing embarassment or other problems to the user.

11.8.2.2 Threats based on remote invocation of this API from another device

1. Remote invocation of this API could have unexpected consequences as it might break the natural accountability that messaging would be expected to have. For example, an application might remotely invoke SMS functionality on a smartphone and the user, who is on their PC at the time, would only find out when they check their mobile device.

2. Remote invocation could be used to impersonate the user for second-factor out-of-band authentication. For example, letting the application read an SMS created by a payment provider, or reading an email password reset message.

11.8.2.3 Implementation threats and possible attacks

11.8.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

11.8.2.5 Threats to device manufacturers, operators, other stakeholders

1. sending many sms/mms may consume user credit; this may cause disappointment with the operator or platform/application developer.

11.8.3. Mitigations

1. access to send functionalities by default should be prompted every time (and denied to untrusted apps);
2. should the prompt allow the user to check the message to be sure it's not modified by the application?

11.8.4. Recommended default policy rules for applications

The following features are defined for this api:

http://webinos.org/api/messaging (access to all functionalities)
http://webinos.org/api/messaging.send (access to send message)
http://webinos.org/api/messaging.find (access to find message)
http://webinos.org/api/messaging.subscribe (access to message subscription)
http://webinos.org/api/messaging.attach (access to message attachment)

The control of message sending should be more restrictive than the others.

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

Message send:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently	Applications will be granted access to this API without user consent	

			allow	being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
x	x		Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
		x	Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

Other messaging methods:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
x	x		Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
		x	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask	Widgets will require user consent	Web pages will prompt for

	at runtime (every time)	at runtime, every time. This fact will be visible & modifiable at install time.	consent (Yes / No / Always) at runtime
	Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.9. NFC API

11.9.1. Overview of API

http://dev.webinos.org/redmine/projects/t3-4/wiki/NFC_API

Allows applications to read and write to NFC tags, to push messages to other NFC devices, and to establish a temporary bidirectional asynchronous message channel between the host device and another. NFC uses short range inductive radio frequency communication. The range is typically of the order of 4 centimetres. NFC tags lack batteries and are powered through inductive coupling. The webinos phase 2 API doesn't (yet) support more advanced techniques such as APDU messages for contact and contact-less smart cards, as used for electronic payments.

11.9.2. Threats

11.9.2.1 API-Specific threats and misuse cases

1. Privacy and confidentiality loss - An attacker could place a hidden antenna near enough to listen into the communication protocol between the NFC devices as NFC communications is typically unencrypted. In practice with a hand held NFC device like a mobile phone, and a handheld NFC tag (e.g. a smart card), the risk is low.
2. Loss of an NFC Tag - if data is held on an NFC Tag in an unencrypted form, there is a risk of the Tag falling into the wrong hands. This is analogous to losing a metro travel payment card (e.g. London's Oyster card), a credit card or a security access card used for access to restricted premises. A related scenario involves the loss of your NFC enabled smart phone.

11.9.2.2 Threats based on remote invocation of this API from another device

For this to work, the user would have to be persuaded to cooperate with physically moving the NFC device close to the other device, e.g. a phone or tag.

11.9.2.3 Implementation threats and possible attacks

NFC is sometimes used to exchange credentials for securing other protocols, e.g. WiFi networks or Bluetooth connections. NFC has also been suggested as part of the process for inducting new devices into the user's personal zone. Unencrypted communication between NFC devices could be used to compromise security, however, see above for the practical issues in doing so.

11.9.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

NFC can be used as part of payment solutions, and web developers may feel unwarranted confidence in the security of NFC devices. We need to explain the limitations of basic NDEF NFC devices and point people are more secure solutions where appropriate, e.g. the use of NFC-SEC and APDU.

11.9.2.5 Threats to device manufacturers, operators, other stakeholders

1. Availability - NFC is low power and not a big factor in battery life.

11.9.3. Mitigations

The wireless connection can be safeguarded through the use of NFC-SEC as defined by ECMA-385 and ECMA 386, see:

- ECMA-385
- ECMA-386
- ECMA TC-47 NFC-SEC White paper

Note: ECMA 385, 385 have been republished as ISO/IEC 13157-1 and -2, respectively.

NFC-SEC provides for a Diffie-Helman secure key exchange, allowing NFC devices to set up a secure communication path. In principle, this could be subject to a man-in-the-middle attack, but that would be hard to achieve in practice due to the nature of the very short range radio frequency communication paths involved.

Further work is needed to explore how to integrate support for NFC-SEC in webinos, as this is dependent on the availability of support by the underlying libraries, e.g. libnfc and the Android SDK.

11.9.4. Recommended default policy rules for applications

The need for users to explicitly bring NFC devices close together acts as barrier to silent abuse of the NFC APIs, as a result user consent shouldn't be requested at run-time, except when the system needs to re-format an existing Tag. It may still be a good idea to ask users for consent to at install time as a reminder of the capability of the application in question.

The policy language could be designed to limit the APIs applications can use according to the need, e.g. if an application only needs to share personal contact data (an electronic business card) then it could be restricted to reading and writing such data to NFC devices/tags. The policy language could further limit the capabilities to applications that are authenticated by the host platform.

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
x	x	x	Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.10.TV Control API

11.10.1. Overview of API

http://dev.webinos.org/specifications/new/tv.html

Interface for TV control and managment.

11.10.2. Threats

11.10.2.1 API-Specific threats and misuse cases

1. This API could be used to monitor the current state of the broadcast service. Depending on time and channel information users could potentially be fingerprinted.
2. With knowledge of channel list the location of a user could be derived (e.g. local channels appear in list if terrestrial service is used)
3. Generation of user profiles, preferred channels (sports, adult shows, etc)
4. Annoyance - an application could change channel on the end user unexpectedly. This could also cause embarrassment - e.g., selecting a not-safe-for-work channel at the wrong time.

11.10.2.2 Threats based on remote invocation of this API from another device

1. In case one specific TV service is accessed from two or more apps then the user might experience a denial of service, e.g. user could be prevented from watching a desired channel by an other app changing programmatically to other channels. This is due to API implementation controlling one specific service, in other words: lack of exclusive usage of service.
2. Remote invocation might be used to monitor the user's presence, e.g. one could monitor if a user is changing channels?
3. Remote (mobile) access to streams could increase costs for internet access

11.10.2.3 Implementation threats and possible attacks

1. The current implementation means that TV API applies to all apps at once.
2. Currently, one implementation of TV API uses VLC as backend, which might introduce its own security issues (foreign code execution, due to application flaws)

11.10.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. As mentioned above ("lack of exclusive usage of service") an app developer could experience unexpected behavior in his apps

11.10.2.5 Threats to device manufacturers, operators, other stakeholders

1. Not a security threat, but some broadcasters could have concerns, as apps might filter them out, censor or hiding ads or overlaying content.

11.10.3. Mitigations

1. Provide examples for the correct scenarios in which the API should be used.
2. Provide a black-list option for some channels, making them unable to be selected by the API

11.10.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
	x		Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
x		x	Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.11.Vehicle API

11.11.1. Overview of API

Link to API - http://dev.webinos.org/specifications/new/vehicle.html

The API provides read access to vehicle specific data.

11.11.2. Threats

11.11.2.1 API-Specific threats and misuse cases

1. The API could be used to generate driving profiles of a user based on the trip computer data. this might be wanted by targeted advertisers trying to profile the user to send them product advertisements.
2. The API could be used to monitor the state of the vehicle (e.g. enginge-oil status and trip computer).
3. Inferring the location of the end user, based on vehicle data

11.11.2.2 Threats based on remote invocation of this API from another device

1. Vehicle properties with high update frequencies can slow down the communciation within the personal zone.

11.11.2.3 Implementation threats and possible attacks

1. One implementation of the Vehilce API has access to the MOST-Bus. Depending of the vehicle property the update frequency is short (a few ms). Registering listeners to more high frequency properties can crash the PZP.
2. Updates on a high frequency properties can freeze the GUI

11.11.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. Binding to too many vehicle properties can lead to unresponsiveness of the app, as the communication channel between PZP and browser is flooded with RPC messages of the API . The updates can also freeze the GUI of the application for a few seconds.

11.11.2.5 Threats to device manufacturers, operators, other stakeholders

1. bad publicity for device manufactures, especially as the car is still perceived as one system. Currently customers still do not distinguish between a problem with the engine or with a third application running on in-car headunit. It is perceived as a problem with the car.

11.11.3. Mitigations

1. Provide examples for the correct scenarios in which the API should be used.

2. Use thresholds for updates on high frequency vehicle properties
3. Limit the amount of vehicle properties an application can register listeners to.
4. Suggest that manufacturers consider only allowing signed apps from known authors or distributors to access this API

11.11.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

For each application type, select one of these as the default policy for this API. This applies to an application which explicitly requests access to this API.

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
X			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
	X		Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
		X	Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy	

editor.

11.12.Webinos core interface

11.12.1. Overview of API

Link to API - http://dev.webinos.org/specifications/new/example.html

The webinos core interface is a way of accessing all the other interfaces.

11.12.2. Threats

11.12.2.1 API-Specific threats and misuse cases

1. Replacing this core interface with another, malicious core interface. This could access a different personal zone proxy, perhaps one that is not on the current device.

11.12.2.2 Implementation threats and possible attacks

1. Are there any threats due to the way the webinos.js shim is implemented?

11.12.2.3 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. A developer has little assurance that the webinos object is communicating with the right endpoint.

11.12.2.4 Threats to device manufacturers, operators, other stakeholders

No obvious threats.

11.12.3. Mitigations

1. Ensure that the browser can only talk to the right PZP through authentication of the IPC / WebSocket.

11.13.Widget API

11.13.1. Overview of API

http://dev.webinos.org/specifications/draft/widget.html

Provides informations about the widget, the capability to store preferences as well as methods to monitor and control widget status (start, stop, background, foreground). It also provides a method to install a child widget on any user's device. All data refers to the widget itself, so a widget cannot control or access data belonging to another widget.

11.13.2. Threats

11.13.2.1 API-Specific threats and misuse cases

1. This api can be used to track or change user preferences (only for the current widget);
2. It is not clear what happens if this api is called from a browser context. Can the deployChild be used to install a child widget from a web page?

11.13.2.2 Threats based on remote invocation of this API from another device

1. The method deployChild can be remotely called on the device where the widget should be installed; it can be used to install (maliciuos) widgets on all the devices discoverable by the user/device.
2. Other attributes/methods mainly provide informations about the current widget, it makes no sense to call it on a remote device.

11.13.2.3 Implementation threats and possible attacks

No particular threat identified.

11.13.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

No particular threat identified.

11.13.2.5 Threats to device manufacturers, operators, other stakeholders

No particular threat identified.

11.13.3. Mitigations

1. Disable this api from the browser context;
2. Disable remote invocation of this api;
3. Require user consent before installing a child widget;

11.13.4. Recommended default policy rules for applications

At the moment this api does not define a feature for policy control.
Access to widget information or status is not a problem. Installation of child widgets should require user approval (on both devices?).

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate

W-U	Widget, unrecognised

Policy for access to widget info or status:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
	X	X	Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

Policy for child widget installation:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
	X		Default allow (install	Widgets will need user consent at install time, but users will expect	Web pages will prompt for consent (Yes / No / Always) at

	time)	to allow it (the tick-box will automatically be filled in).	runtime, this preference will be saved.
	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
X	Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
	Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.14.The W3C calendar module

11.14.1. Overview of API

http://www.w3.org/TR/2011/WD-calendar-api-20110419/

The Calendar API defines a high-level interface to access Calendar information such as events, reminders, alarms and other calendar information. The API itself is designed to be agnostic of any underlying calendaring service sources.

Security and Privacy considerations are already described in this part of the specification .

11.14.2. Threats

11.14.2.1 API-Specific threats and misuse cases

1. Distribution of user calendar details - privacy issues abound. Sharing with unauthorised parties may result in embarrassment, loss of income (meeting information may be valuable to businesses) or impact on the safety of the end user.
2. Targeted advertising based on calendar event information
3. Stalking possible if the wrong person is given access to calendar entries

4. Misuse case - Calendar details shared over unprotected communication medium and accessed by unauthorised parties, allowing them to rob the victim when they know he will be at an appointment.
5. Not all calendars are created equally - some will be used for business, others personal, others are common events. Granting access to one should not automatically grant access to them all.
6. The details of contacts who have been added to meetings or appointments might be revealed through use of the calendar API in an unexpected way.

11.14.2.2 Extended threats based on editing calendars

1. Spam through an unauthorised application adding calendar entries
2. Missing appointments or loss of information due to an application deleting entries or adding too many unwanted entries

11.14.2.3 Threats based on remote invocation of this API from another device

1. Peter might allow his application access to the Calendar on his PC, knowing that there is nothing important in it. However, it then has access to his mobile phone calendar, which is much more personal and sensitive.
2. There could be an unexpected ripple-effect when deleting a calendar entry - deleting it on one calendar could impact all calendars on all devices unexpectedly.

11.14.2.4 Implementation threats and possible attacks

11.14.2.5 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

- The Calendar API might return enormous amounts of data, causing any Web Application to crash or fail.
- Calendar data from public calendars might be used to perform XSS or XSRF attacks.

11.14.2.6 Threats to device manufacturers, operators, other stakeholders

11.14.3. Mitigations

These apply: http://www.w3.org/TR/2011/WD-calendar-api-20110419/#security-and-privacy-considerations

1. User consent required for accessing each calendar
2. Separation of calendars is important - access to one calendar should not imply access to others
3. Could the API support "busy/free" invocation methods rather than precise information about each event? (this would involve a modification to the API)
4. Granularity is important
    1. Applications should state how much calendar information they will request and why.

    2. permission might be granted just for an application to see whether a particular time is free or busy, but not any details
    3. permission might be granted to view calendar items but not edit (editing is not technically in the specification)
5. Revocation of consent is necessary.
6. Suggest a three-phase policy per calendar: "allow all", "allow view busy/free", "deny"

11.14.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
X	X		Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
		X	Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

Rationale: accessing a calendar ought to require explicit consent, and most pages ought to deny it by default. For those pages and widgets which are from known sources, it is reasonable to expect that consent is likely to be granted after one question.

11.15.The W3C contacts module

11.15.1. Overview of API

http://dev.webinos.org/specifications/new/contacts.html

Provides the capability to read, write, update and delete user contacts.

11.15.2. Threats

11.15.2.1 API-Specific threats and misuse cases

1. an application can read all my contact details and send them to an attacker; this way the attacker can use it for spam or to collect sensitive information about contacts (not only phone number or email, but also address, birthday, photo, ...).
2. an application can modify or delete my contacts; for example change the phone number to a premium rate service.
3. giving an application/user access to a contact can give access also to other contacts; for example I share my work contacts, but accidentially I also give access to my family and friends contacts.
4. giving an application/user access to a contact phone number can reveal other personal information (address, birthday, photo...); for example I share the phone number of my sister, but also accidentally it gives access to her address and photo.
5. a malicious application (or remore user) can fill the address book with dummy data (this way the user cannot add other contacts);

11.15.2.2 Threats based on remote invocation of this API from another device

No particular threats identified.

11.15.2.3 Implementation threats and possible attacks

No particular threats identified.

11.15.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. an application loading all contacts details (including photos) may risk memory problems;
2. data protection laws and issues might arise for remotely-hosted applications loading contact details.

11.15.2.5 Threats to device manufacturers, operators, other stakeholders

1. The privacy issues (ie the problems of private information being disclosed without control) convince potential users not to use the platform.

11.15.3. Mitigations

1. The policy must support parameters to select contacts (by name, by detail type, by organization...); this way only selected contacts can be accessed;
2. The policy must allow the user to share only some details of his contacts (for example share details with type="work"); this way only selected details can be accessed;
3. The user should be able to verify which data are available to applications/users;
4. Untrusted applications should not be allowed write access to contacts;
5. It may help to have an interface to show which contacts can be accessed by a particular application or user.

11.15.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

Access to contacts.read feature:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
	X		Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
X			Default ask at runtime	Widgets will require user consent at runtime, every time. This fact	Web pages will prompt for consent (Yes / No / Always) at

			(every time)	will be visible & modifiable at install time.	runtime
		X	Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

Access to contacts.write feature:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
X	X		Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
		X	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.16.The WAC devicestatus module

11.16.1. Overview of API

http://specs.wacapps.net/devicestatus/index.html

Provides information about various "aspects" of a device.

11.16.2. Threats

11.16.2.1 API-Specific threats and misuse cases

1. This API could be used, accessing IMEI code or other device properties, to fingerprint the user of a personal device.
2. Operating System version and Web Runtime version can be used to exploit version-specific vulnerabilities.

11.16.2.2 Threats based on remote invocation of this API from another device

1. Remote invocation could be used to access personal information. E.g. language (`OperatingSystem` aspect, `language` property), country (`CellularNetwork` aspect, `mcc` property), mobile operator (`CellularNetwork` aspect, `operatorName` property)
2. It could be used to fine-tune remote DoS attack (e.g., the bettery level is a nice information to shape how much aggressive a DoS have to be). #

11.16.2.3 Implementation threats and possible attacks

11.16.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. Wrong result of this API can influence the behavior of an application. The result can vary from an odd feeling from the user to severe application faults.
2. If the API can provide unverified results (so, especially in case of untrusted browser or widget) the information with the user can be partially compromised. E.g., an App which want to display a contract and a disclaimer. If the devicestatus does not provide the correct size of the screen, the disclaimer could be not showed to the user since "visualized" in a zone out of the screen.
3. If not wisely used, may lead to excessive power consumption (e.g. through watchPropertyChange() method)

11.16.2.5 Threats to device manufacturers, operators, other stakeholders

1. If the API provides wrong results from the sensors/components, components manufacturers could have bad reputation due to the suspects they components do not work well.

11.16.3. Mitigations

1. Main problem with this API, seems information leakage. Thus, remote access should be provided only to trusted entities, to avoid information leakage and attack finetuning
2. Set adequate limit to watch the properties change

11.16.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

Local Invocations:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
	x		Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Webpages will prompt for consent (Yes / No / Always) at runtime, this can be just a warning rather than a prompt
x		x	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Webpages will prompt for consent (Yes / No / Always) at runtime, but the PZP will remember the setting.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Webpages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Webpages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy	

				editor.	

Remote invocations:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Webpages will prompt for consent (Yes / No / Always) at runtime, this can be just a warning rather than a prompt
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Webpages will prompt for consent (Yes / No / Always) at runtime, but the PZP will remember the setting.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Webpages will prompt for consent (Yes / No / Always) at runtime
x	x		Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Webpages will display a short notification at first-use saying that access was denied, with a button to change settings
		x	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.17.The WAC deviceinteraction module

11.17.1. Overview of API

Link to API - http://specs.wacapps.net/deviceinteraction/index.html

This module provides a mechanism to interact with the end-user through features such as: device vibrator, device notifier, screen backlight, device Wallpaper.

11.17.2. Threats

11.17.2.1 API-Specific threats and misuse cases

1. Excessive use of device notification features (vibrate, backlight) would drain battery and be annoying. This could render the device unusable, or make the user (such as Peter) turn off these features altogether at the OS level, affecting other applications.
2. Using notifications could embarrass the user or alert others to their presence, impacting their privacy. E.g., making the phone vibrate / notify in a public setting.
3. Changing wallpaper could cause embarrassment or alarm. E.g., setting it to something work-inappropriate, or setting Georg's phone wallpaper to have a picture of a woman who is not his wife...

11.17.2.2 Threats based on remote invocation of this API from another device

1. It could cause some panic or confusion if invoked from the wrong device
2. Sending a notification to the wrong device would also render the notification useless - this might result in a loss of integrity/availability of application functionality. It could result in someone missing an appointment or meeting.
3. Remote battery drain is quite possible, as the easiest mitigation for misuse of this API would be the fact that it is obvious if too much vibrating / screen light is used. If the device is remote, however, it will not be obvious

11.17.2.3 Implementation threats and possible attacks

TBC.

11.17.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

Nothing obvious beyond loss of application functionality.

11.17.2.5 Threats to device manufacturers, operators, other stakeholders

Nothing obvious beyond loss of application functionality. Excessive use of certain features might damage devices?

11.17.3. Mitigations

Not a high risk API

1. Remote use of this API seems relatively unlikely. Suggest that it is denied by default for remote invocation.
2. Upon remote invocation of the setWallpaper API, have a mandatory confirm dialogue appear on remote devices.
3. Vibration API Mozilla analysis suggests limits on how often this make be invoked - e.g., don't allow hundreds of API invocations.

4. Limits should be placed on the light on / off command.
5. Require user consent before allowing the "setWallpaper" command.

11.17.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

For the notify, vibrate and light LOCAL invocations:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
X	X		Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
		X	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

For the notify, vibrate and light REMOTE invocations:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
X	X	X	Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

For setWallpaper LOCAL invocations:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
	X	X	Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime	Widgets will require one-off user consent at runtime. This fact will	Web pages will prompt for consent (Yes / No / Always) at

			(one-shot)	be visible & modifiable at install time.	runtime, this preference will be saved.
X			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

For setWallpaper REMOTE invocations:

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
	X	X	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
X			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings

	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.

11.18. The W3C DeviceOrientation Event specification

11.18.1. Overview of API

http://www.w3.org/TR/2011/WD-orientation-event-20111201/
http://www.w3.org/TR/orientation-event/

Provides information about the physical orientation and motion of a hosting device.

11.18.2. Threats

11.18.2.1 API-Specific threats and misuse cases

1. This API could be used to infer keystrokes on the touch screen.

11.18.2.2 Threats based on remote invocation of this API from another device

1. It could also be used to identify someone's general activities and whether they are moving. This might help narrow-down location, and give information about user habits, particularly if augmented by other information (e.g. what's the current user location plus some location specific information).

11.18.2.3 Implementation threats and possible attacks

11.18.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

11.18.2.5 Threats to device manufacturers, operators, other stakeholders

11.18.3. Mitigations

1. Put the device on a hard surface when typing.
2. Don't use the default on screen keyboard.
3. Disable the orientation sensors during touch-screen keyboard operation, when possible.

11.18.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
	X		Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
X		X	Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.18.5. Other refs

http://static.usenix.org/events/hotsec11/tech/final_files/Cai.pdf

11.19.The W3C File API

11.19.1. Overview of API

http://www.w3.org/TR/2011/WD-FileAPI-20111020/
http://www.w3.org/TR/FileAPI/

Provides read-only file objects representation.

11.19.2. Threats

11.19.2.1 API-Specific threats and misuse cases

1. This API could be used to access local files (read only) containing sensitive data.

This could be exploited, for example, by Harold, who develops a free and funny application which can provide the user a nice quote (from an external DB of quotes) chosen on the information present on the file system (e.g., retrieve the user name from the file system, and then look for quotes which contains the same name). To do so, the application access read the file system with the file API, whose behavior if not restricted allow access even to sensitive system sections (e.g. .webinos, where the private key is stored). Those private date can be sent outside during the exchange with the external quote database.

11.19.2.2 Threats based on remote invocation of this API from another device

1. This API could be used to access remote files (read only) containing sensitive data.
2. It may induce a false feeling in the user, which may assume that installing the application on one device limits what it can access to just that device, enabling access of sensitive information on other devices of the zone.
3. It is also a case of minimising the surprise to the end user - they may assume that installing the application on one device limits what it can access to just that device.

11.19.2.3 Implementation threats and possible attacks

1. Arguments are not validated, the Jimmy and Jessica have to provide by itself the proper validation.

11.19.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. Certain UAs, i.e., Browsers. In order to manage origin-private filesystems within webinos, additional information is required
2. Encoding Subtleties: The W3C File API specification(http://www.w3.org/TR/FileAPI/) requires the IANA: Character Sets to be supported (http://www.iana.org/assignments/character-sets). Internally, JavaScript uses UTF-16; the implementation uses UTF-8 and currently ignores any given encoding. This direct manipulation have to pay attention to not introduce encoding mismatch.
3. unproper data read by applications could contain malicious code, exploit bugs and enable bad application behaviour

11.19.2.5 Threats to device manufacturers, operators, other stakeholders

1. it could be a first step (information leakage, device fingerprinting) for another threat which influence device reputation
2. reading malicious code that enable bugs could provide root access or system crash

11.19.3. Mitigations

1. For developers threats, provide examples which clarify how the API works for each ambiguous case
2. Each application could have a filesystem section devoted to its own files. File access outside this area should be denied. (This can be done via File API - Directories and System)
3. Files (or application's file system area) should be encrypted to avoid data disclosure to unauthorized readers.
4. Files' content should be parsed to avoid malicious code execution.

11.19.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

Private Application file system

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
x	x		Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Webpages will prompt for consent (Yes / No / Always) at runtime, this can be just a warning rather than a prompt
		x	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Webpages will prompt for consent (Yes / No / Always) at runtime, but the PZP will remember the setting.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Webpages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-	Webpages will display a short notification at first-use saying that access was denied, with a

				box will automatically be empty).	button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

Non private application file system

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Webpages will prompt for consent (Yes / No / Always) at runtime, this can be just a warning rather than a prompt
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Webpages will prompt for consent (Yes / No / Always) at runtime, but the PZP will remember the setting.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Webpages will prompt for consent (Yes / No / Always) at runtime
	x		Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Webpages will display a short notification at first-use saying that access was denied, with a button to change settings
x		x	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.20.The W3C File API - Writer

11.20.1. Overview of API

http://www.w3.org/TR/2012/WD-file-writer-api-20120417/
http://www.w3.org/TR/file-writer-api/

Provides write-only file objects representation.

11.20.2. Threats

11.20.2.1 API-Specific threats and misuse cases

1. This API could be used to corrupt or forge local and remote files' content.
2. It can be used write data to fill the local or remote file system

This open the way to multiple threats, for example, by David to change system parameters or fill the available space of an in-car system, thus leading a not skilled user to frequently come back to the car repairer and spend lot of money.

11.20.2.2 Threats based on remote invocation of this API from another device

1. It is possible for a client to be impersonated, if not trusted identity is verified (and if many devices are regularly off).
2. Ethan, the spammer, could write exploit code in remote files (or in file which will be accessed by remote device), making possible creation of cross-device botnets

11.20.2.3 Implementation threats and possible attacks

11.20.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. Certain UAs, i.e., Browsers. In order to manage origin-private filesystems within webinos, additional information is required
2. Encoding Subtleties: The W3C File API specification(http://www.w3.org/TR/FileAPI/) requires the IANA: Character Sets to be supported (http://www.iana.org/assignments/character-sets). Internally, JavaScript uses UTF-16; the implementation uses UTF-8 and currently ignores any given encoding. This direct manipulation have to pay attention to not introduce encoding mismatch.

11.20.2.5 Threats to device manufacturers, operators, other stakeholders

1. In specific device (e.g. In-car system) should be restricted to avoid integrity loss of important system data.

11.20.3. Mitigations

1. Each application could have a filesystem section devoted to its own files. File access outside this area should be denied. (This can be done via File API - Directories and System).
2. Files encryption can avoid data forgery but can't avoid data corruption.
3. cross-device IDS to identify cross-device exploits.

11.20.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

Private Application file system

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
X	X		Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
		X	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

Non private application file system

B-	W-	W-	Policy	Explanation (Widgets)	Explanation (Browser Apps)

A	R	U			
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
	X		Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
X		X	Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.21.The W3C File API - Directories and System

11.21.1. Overview of API

http://www.w3.org/TR/2012/WD-file-system-api-20120417/
http://www.w3.org/TR/file-system-api/

Provides means to navigate file system hierarchies.
Provides means to expose sandboxed sections of a user's local filesystem.

11.21.2. Threats

11.21.2.1 API-Specific threats and misuse cases

1. This API could be used to navigate file system directories and other local applications directories.
2. It can take file system resources inappropriately using storage request API.

3. It can prevent legitimate data access moving local or remote files
4. it can delete data removing local or remote files

For example, if access to critical file, like credential ones (e.g. password or private key) is incorrectly allowed, Frankie (the alienated script kiddie) may be able to delete them, in fact "disconnecting" the device from the zone (since no longer able to authenticate versus the other devices), just to make "funny" annoyance to "friends"

11.21.2.2 Threats based on remote invocation of this API from another device

11.21.2.3 Implementation threats and possible attacks

1. Some operations invalidate the entry/blob used, e.g., remove. The implementation does not keep track of the entries/blobs returned, creating copies if the same entry/blob is requested multiple times, thus not propagating invalidation to all affected objects. The validation process is left to the underlying system. This could create integrity weaknesses (entries supposed to be deleted but yet there), and allow for persistent presence of entries, which could lead to information leakage if sensitive information are not properly deleted.

11.21.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

Developers could experience some discrepancies between actual implementation and expected behavior

1. Certain UAs, i.e., Browsers. In order to manage origin-private filesystems within webinos, additional information is required
2. Encoding Subtleties: The W3C File API specification(http://www.w3.org/TR/FileAPI/) requires the IANA: Character Sets to be supported (http://www.iana.org/assignments/character-sets). Internally, JavaScript uses UTF-16; the implementation uses UTF-8 and currently ignores any given encoding. This direct manipulation have to pay attention to not introduce encoding mismatch.
3. Some operations, e.g., removeRecursively, could continue after the occurrence of an error or exception. This implementation works differently, aborting any operation that encounters an error or exception. Thus, an operation may partially succeed, e.g., removeRecursively may remove some (up to the occurrence of an error or exception) but not all children of a directory. The app developer have to understand and check the success of those commands to avoid damage and leave the system in inconsistent state.
   Presumably an inconsistent delete operation might result in private/valuable data not being deleted, which might cause embarrassment due to unexpected data disclosure. E.g., when selling an old device, or even affect valuable intellectual property.

11.21.2.5 Threats to device manufacturers, operators, other stakeholders

No obvious threats. Not much significantly, an attacker could navigate the OS file system and move or delete OS files, causing OS instability and bad user experience.

11.21.3. Mitigations

1. For developers threats, provide examples which clarify how the API works for each ambiguous case
2. Each application could have a filesystem section devoted to its own files. File system navigation, movement, copy and deletion outside this area should be denied, as well as "anonymous" file system access

11.21.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
X	X		Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
		X	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.22.The W3C Gallery API

11.22.1. Overview of API

http://dev.w3.org/2009/dap/gallery/

Provides access to the media gallery.

NOTE: at the moment there is not implementation (deferred to phase 2) and no final decision about the specification (the W3C Gallery is a little outdated). Thus, only quite vague threats are mentioned.

11.22.2. Threats

11.22.2.1 API-Specific threats and misuse cases

1. This API could be used to access sensitive multimedia data stored in a local or remote gallery.
2. Adding objects to the gallery an attacker could fill the file system.

An attack perpetrated by an evil person could discredit or embarrass a specific individual, accessing sensible photos and even allowing for ransom in the worst cases.
The integrity of photos and gallery must be assured as well, to avoid possible cases of vilification (e.g. by changing a photo with a modified one).

11.22.2.2 Threats based on remote invocation of this API from another device

11.22.2.3 Implementation threats and possible attacks

11.22.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

11.22.2.5 Threats to device manufacturers, operators, other stakeholders

11.22.3. Mitigations

1. The Gallery implementation should impose restrictions to the gallery size,

11.22.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)

	X		Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
X			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
		X	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.23.The W3C Geolocation API

11.23.1. Overview of API

Link to API - http://www.w3.org/TR/geolocation-API/

"This specification defines an API that provides scripted access to geographical location information associated with the hosting device."

11.23.2. Threats

11.23.2.1 API-Specific threats and misuse cases

1. Leaking of location to unauthorised parties (e.g. through advertising networks or through spoofing of app identity)
2. Misuse of location data by authorised parties (e.g. an app starts using location to track users rather than provide contextual functionality)
3. Unintentional authorisation of malicious parties (e.g., users select 'OK' to an app they didn't mean to. Or an app gains access through another app)

4. Unexpected consequences of geolocation data disclosure (e.g., the user reveals their location but then realises it was a mistake in retrospect)

More details:

1. This API gives applications access to the location of the end user, which could be privacy sensitive.
2. It may result in advertising that the user finds annoying, it might be given away to third party advertisers, or be used for customer/user analytics at a later date (profiling). This data could end up on unauthorised websites.
3. The application might publish this information in a way that the user has no control over, and therefore might be seen by unexpected people. E.g., someone's friends or family might see their location without the user's knowledge. This would be extremely bad if they were visiting a potentially sensitive or embarassing place, such as a hospital or police station. It may also be embarassing if someone learns about a relationship or interest the end user has but wishes to keep a secret.
4. Multiple location details could be correlated over time to build up a picture of user activities. This might be used for predicting future behaviour and be inaccurate or misleading. It might also reveal habits that the user is uncomfortable with knowing.
5. Thieves might use location data to find out whether someone is away from their house in order to burgle them. It could also tell a mugger where to wait for someone with an expensive phone, car or tablet device.
6. Stalkers might use disclosed location data to track their target and intimidate or threaten them.
7. In dangerous situations (e.g. in unstable countries with terrorists or unrest) this API could accidentally give away location data which would expose the end user.
8. Inaccurate information could cause accidents or mistakes if users rely too heavily on the output of this API.
9. Geolocation data could connect otherwise unlinkable user actions. E.g., a tweeting application and a online shopping application could correlate user identity by spotting that the user was in the same place at the same time.
10. Users might allow location data for one action, but not want to allow it forever. If the API made it easy for applications to gain and then reuse permission, this would be a problem.
11. Users might realise at a later date that their location information, as attached to some record somewhere online, was either inaccurate or privacy-invasive for some reason. E.g., it might have revealed their presence at a movie theatre when they had claimed to be unwell and off work.
12. This API might be used for a "find my phone" type application for lost or stolen devices. If it loses availability or is inaccurate, such a feature would be useless.

From Mozilla - https://wiki.mozilla.org/WebAPI/Security/Geolocation

11.23.2.2 Threats based on remote invocation of this API from another device

1. Remote invocation can result in unexpected revelation of data. E.g., an application might access location information about the users' car, when they were expecting it to be about their TV. Different devices will be in privacy sensitive or insensitive locations, so accidentally authorising the wrong one is a problem.
2. Remote geolocation would allow users to pretend to be in a location they are not. This might be a problem for applications relying on this information.

11.23.2.3 Implementation threats and possible attacks

11.23.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. A developer might base their app on geolocation in order to implement a security or privacy feature. E.g., only allowing access to certain parts of the app if they are at work, or in a certain place. This would be easy to circumvent.

11.23.2.5 Threats to device manufacturers, operators, other stakeholders

1. Geolocation drains battery and (if it is assisted GPS) might use bandwidth
2. Geolocation (if unconstrained) could slow down the phone

11.23.3. Mitigations

1. Users must be prompted before geolocation is first requested or used.
2. Revocation of permission to access geolocation must be possible and easy to access.
3. Insist on plausible deniability. E.g., it should always be plausible for the API to return a "no data" result.
4. Implement a cap on the number of one-off 'getCurrentPosition' requests allowed
5. Make it visible to the end user that their location is being watched or accessed by a particular app. Make it easy to turn off/on geolocation data
6. Make the geolocation service selection easy, and do not provide access to remote geolocation by default. Device-agnostic policies should either not be possible, or not be easy to end up with.

11.23.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-	W-	W-	Policy	Explanation (Widgets)	Explanation (Browser Apps)

A	R	U			
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
X	X	X	Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
			Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

11.24.The W3C Media Capture and Streams API

11.24.1. Overview of API

http://dev.w3.org/2011/webrtc/editor/getusermedia.html

This specification defines an API that provides access to the audio, image and video capture capabilities of the device. Development on the previously referred *W3C Media Capture API* (http://www.w3.org/TR/media-capture-api/) has stopped, and further work is taking place as part of *Media Capture and Streams* (http://dev.w3.org/2011/webrtc/editor/getusermedia.html).

11.24.2. Threats

11.24.2.1 API-Specific threats and misuse cases

1. This API could be used to compromise the privacy of the user, as the accessed media may contain private information, either directly in audio or video form or could be derived from

the shown surrounding, e.g. location of the user, contact/dialogues with other persons or in the simplest case the presence of the user.

2. Accessed audio or video could be used to build user profiles, e.g. music/TV in background captured and delivered with the stream.
3. Users will expect the same security in Web based real-time communication as they are used to from traditional telephony service, e.g. the identification of callee and caller is up the application. A user may be communicating with someone else than he/she is expecting.

11.24.2.2 Threats based on remote invocation of this API from another device

1. Any mechanism that bypasses the user consent to allow the media capture could be misused by potentially malicious apps. As this API introduces programmatically privacy threats and was originally designed for the Web browser execution environment an user consent prompt is an inherent part of its rationale. Any remote invocation should therefore incorporate user consent as well. Implications of a remote prompt needs to be examined.
2. API usage could generate high costs through generation of high volume data

11.24.2.3 Implementation threats and possible attacks

In the first phase of the project this API was not implemented. Only theoretical assertions can be made currently.

1. The concurrent access to media capture devices could be made possible but could have consequences of the end user. For example, once a user has granted access within one application and switches to use a different application then the first one may still have access to the media capture devices. This leads to a requirement of exclusive usage of a specific device and revocation of granted access rights.

11.24.2.4 Threats to apps and developers using this API (E.g. Jimmy and Jessica)

1. In both cases exclusive and shared access to media capture devices unexpected behavior. Exclusivity may lead to denial of service and shared operation mode to non-transparency for the end user.

11.24.2.5 Threats to device manufacturers, operators, other stakeholders

1. Enabling audio and video communication in Web applications may have impact on the traditional telephony via mobile/fixed operated networks as well as VoIP networks
2. Device manufacturers may be forced to improve their products to support media processing for Web based communication (hardware media en/decoding) to lower the thread of high cpu usage and unresponsive devices

11.24.3. Mitigations

- Require explicit user consent before a service may be accessed

- Provide a notification when a device's media capture services are in use. Also make clear from the calling device where a media capture service is hosted (e.g., which device is providing the API).

11.24.4. Recommended default policy rules for applications

Code	Type
B-A	Browser-based, authenticated via TLS certificates
W-R	Widget, authenticated using a recognised certificate
W-U	Widget, unrecognised

B-A	W-R	W-U	Policy	Explanation (Widgets)	Explanation (Browser Apps)
			Silently allow	Applications will be granted access to this API without user consent being required. This can only be modified using a policy editor.	
			Default allow (install time)	Widgets will need user consent at install time, but users will expect to allow it (the tick-box will automatically be filled in).	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
X	X		Default ask at runtime (one-shot)	Widgets will require one-off user consent at runtime. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime, this preference will be saved.
		X	Default ask at runtime (every time)	Widgets will require user consent at runtime, every time. This fact will be visible & modifiable at install time.	Web pages will prompt for consent (Yes / No / Always) at runtime
			Default deny (install time)	Widgets will require user consent at install time, but users will expect NOT to allow it (the tick-box will automatically be empty).	Web pages will display a short notification at first-use saying that access was denied, with a button to change settings
			Silently deny	Applications will not be granted access to this API, and users will not be asked at install time. This can only be modified using a policy editor.	

12 Components

12.1. Application Client

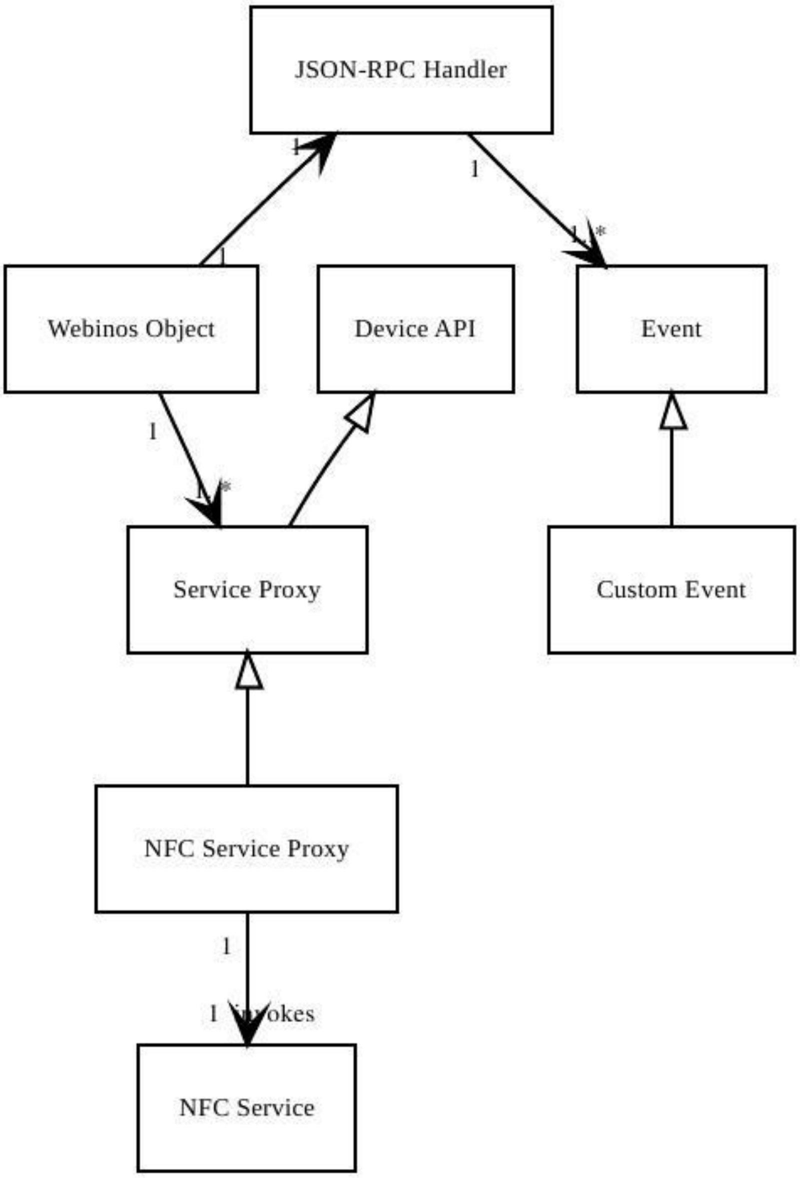
12.1.1. Description

Application using webinos APIs.

12.1.2. Interfaces

Interface	Type	Access Right	Privilege
shareTag	provided	authenticated	normal
findServices	provided	authenticated	normal
getTVSources	provided	authenticated	normal

12.1.3. Structure



Name	Type	Description	Surface	Access Rights
Custom Event	Software	webinos event	JSON	anonymous
Device API	Software	API for accessing device-specific functionality	Undefined	Undefined
Event	Information	An object incorporating messaging attributes and additional webinos specific payload data and meta-data	Undefined	Undefined

JSON-RPC Handler	Software	JSON-RPC Handler	Privileged application	trusted
NFC Service	Software	NFC Service Implementation	Privileged application	authenticated
NFC Service Proxy	Software	NFC Service Proxy	Client application	authenticated
Service Proxy	Software	webinos service interface	Client application	authenticated
Webinos Object	Software	webinos singleton	Client application	authenticated

12.1.4. Component Requirements

None

12.2. Certificate Manager

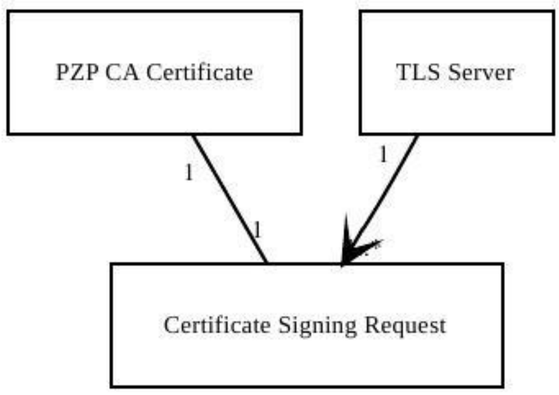
12.2.1. Description

Generates certificates for PZPs before device enrolement.

12.2.2. Interfaces

Interface	Type	Access Right	Privilege
createCertificateRequest	required	trusted	privileged

12.2.3. Structure



Name	Type	Description	Surface	Access Rights
Certificate Signing Request	Information	A message sent from an applicant to a certificate authority in order to apply for a digital identity certificate.	Structured Text	trusted
PZP CA Certificate	Information	PZP CA Certificate	Structured Text	authenticated
TLS Server	Software	TLS Server	Privileged application	trusted

12.2.4. Component Requirements

None

12.3. Context API

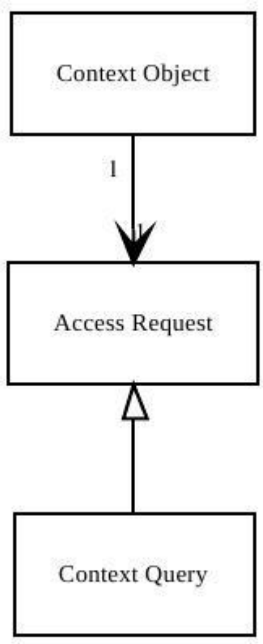
12.3.1. Description

Context API client

12.3.2. Interfaces

Interface	Type	Access Right	Privilege
queryContext	provided	authenticated	normal

12.3.3. Structure



Name	Type	Description	Surface	Access Rights
Access Request	Information	Request for a resources	JSON	authenticated
Context Object	Information	Structure contextual information	JSON	authenticated
Context Query	Information	Query from context database	JSON	authenticated

12.3.4. Component Requirements

None

12.4. Context Database

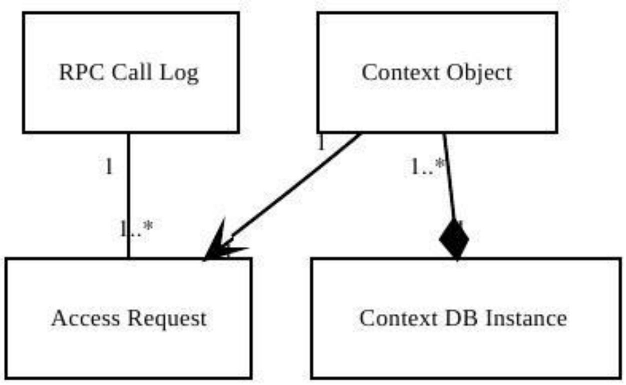
12.4.1. Description

Database of context objects

12.4.2. Interfaces

Interface	Type	Access Right	Privilege
handleContextData	required	authenticated	normal

12.4.3. Structure



Name	Type	Description	Surface	Access Rights
Access Request	Information	Request for a resources	JSON	authenticated
Context DB Instance	Software	SQLite context database instance	Privileged application	trusted
Context Object	Information	Structure contextual information	JSON	authenticated
RPC Call Log	Information	RPC call log	JSON	trusted

12.4.4. Component Requirements

None

12.5. Context Manager

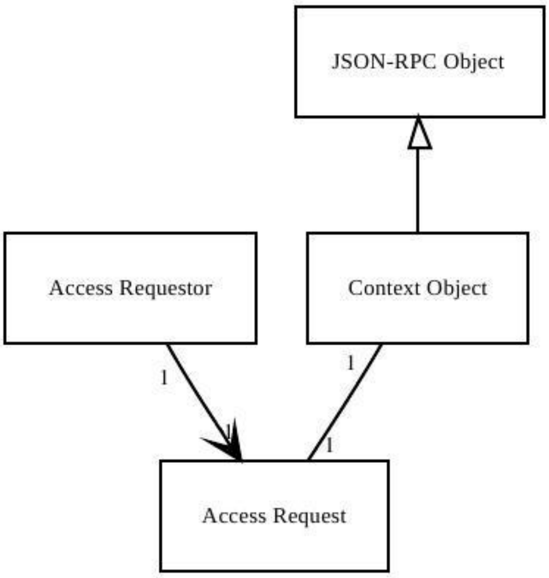
12.5.1. Description

Context Manager implementation

12.5.2. Interfaces

Interface	Type	Access Right	Privilege
enforceRequest	provided	trusted	normal
logContext	required	trusted	normal
handleContextData	provided	authenticated	normal
queryContext	required	authenticated	normal

12.5.3. Structure



Name	Type	Description	Surface	Access Rights
Access Request	Information	Request for a resources	JSON	authenticated
Access Requestor	Software	Requests access to resources	Client application	authenticated
Context Object	Information	Structure contextual information	JSON	authenticated
JSON-RPC Object	Information	JSON-RPC Object	JSON	authenticated

12.5.4. Component Requirements

None

12.6. Discovery Manager

12.6.1. Description

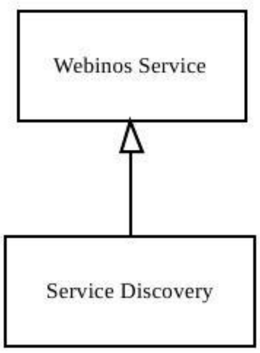
Discovery Manager

12.6.2. Interfaces

Interface	Type	Access Right	Privilege
findServices	required	authenticated	privileged
enforceRequest	provided	authenticated	normal

getAllServices	required	authenticated	normal
getRegisteredServices	required	authenticated	normal

12.6.3. Structure



Name	Type	Description	Surface	Access Rights
Service Discovery	Software	Service Discovery module	Privileged application	authenticated
Webinos Service	Software	Webinos Service Implementation	Privileged application	authenticated

12.6.4. Component Requirements

None

12.7. Keystore Manager

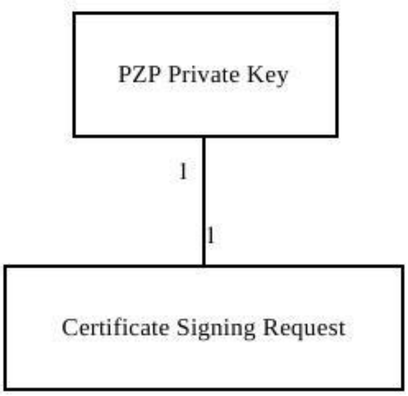
12.7.1. Description

Keystore Manager

12.7.2. Interfaces

Interface	Type	Access Right	Privilege
get	required	trusted	normal

12.7.3. Structure



Name	Type	Description	Surface	Access Rights
Certificate Signing Request	Information	A message sent from an applicant to a certificate authority in order to apply for a digital identity certificate.	Structured Text	trusted
PZP Private Key	Information	Device-held private key	JSON	trusted

12.7.4. Component Requirements

None

12.8. NFC Manager

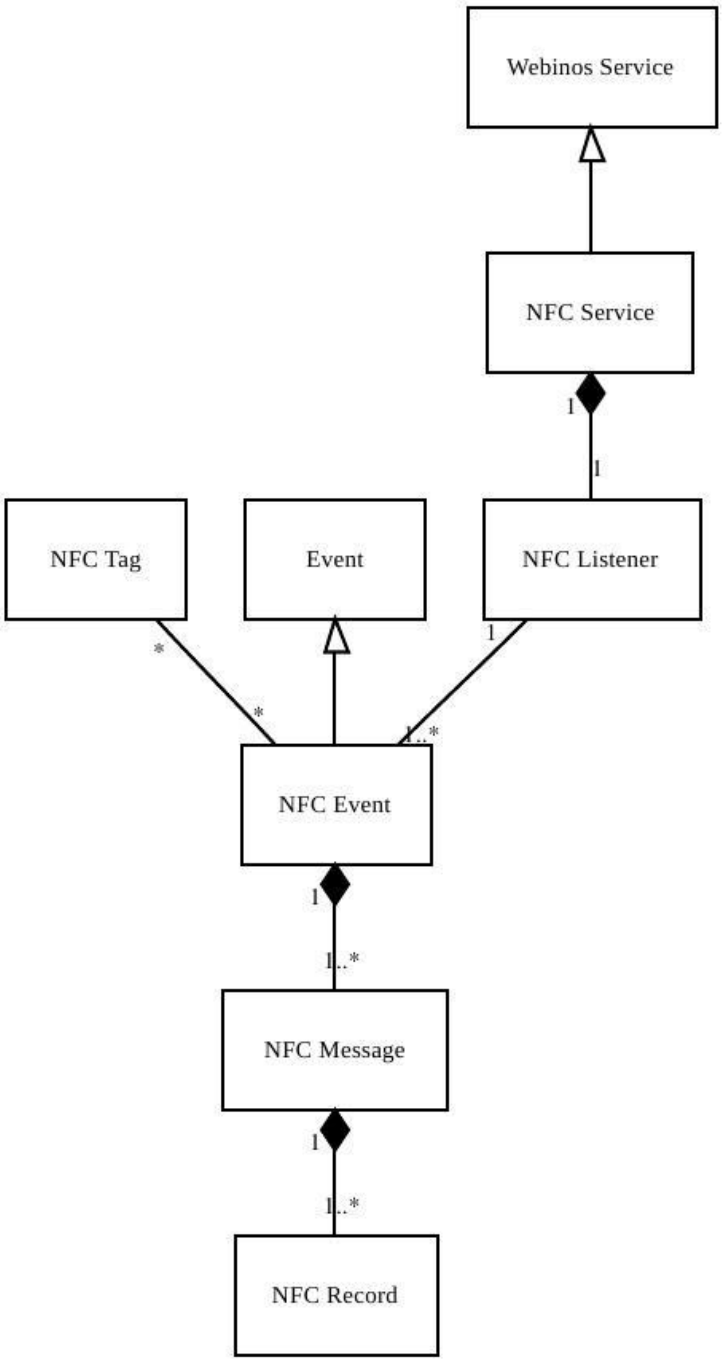
12.8.1. Description

NFC API implementation

12.8.2. Interfaces

Interface	Type	Access Right	Privilege
shareTag	required	authenticated	normal
peer	provided	authenticated	normal

12.8.3. Structure



Name	Type	Description	Surface	Access Rights
Event	Information	An object incorporating messaging attributes and additional webinos specific payload data and meta-data	Undefined	Undefined
NFC Event	Information	NFC Event	JSON	authenticated

NFC Listener	Software	NFC	Privileged application	authenticated
NFC Message	Information	NFC Message	NDEF	authenticated
NFC Record	Information	NFC Record	NDEF	authenticated
NFC Service	Software	NFC Service Implementation	Privileged application	authenticated
NFC Tag	Information	NFC Tag	NDEF	anonymous
Webinos Service	Software	Webinos Service Implementation	Privileged application	authenticated

12.8.4. Component Requirements

None

12.9. NFC Manager B

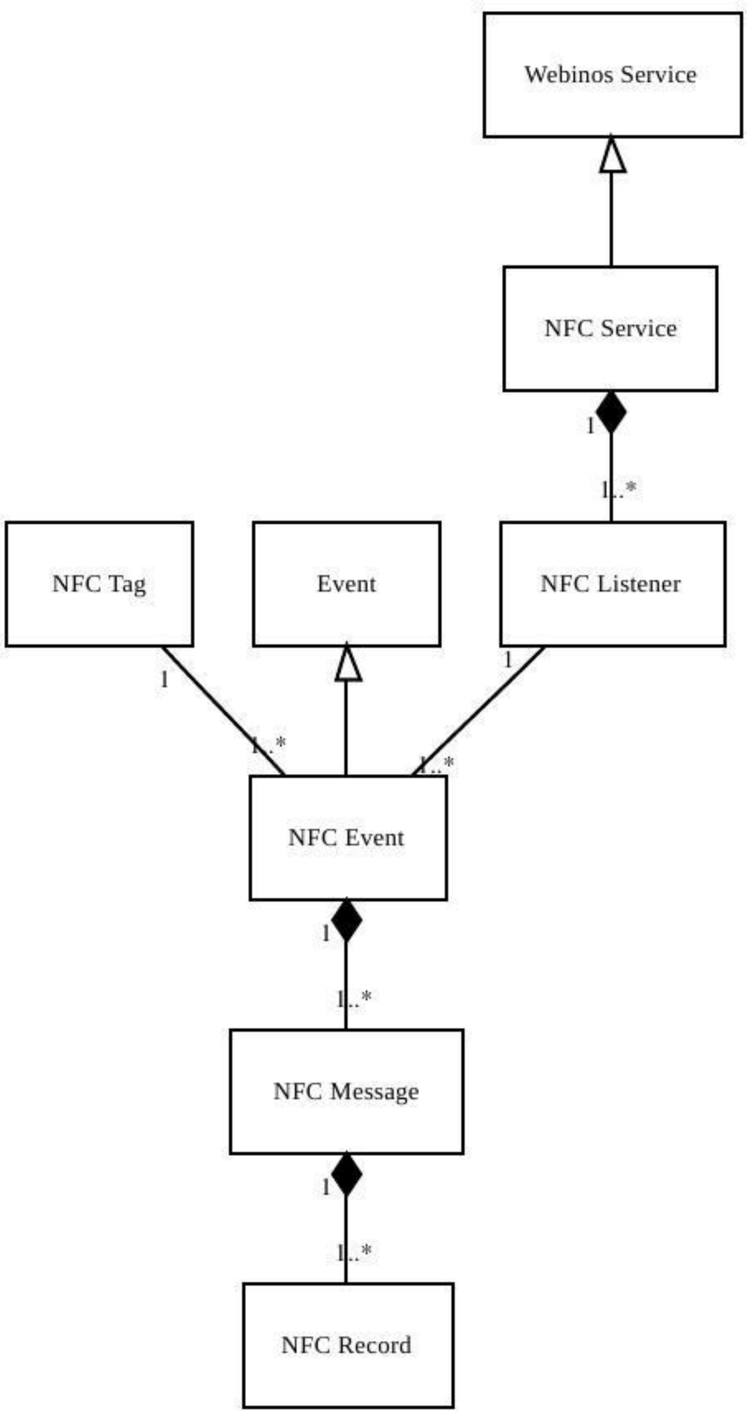
12.9.1. Description

NFC API implementation

12.9.2. Interfaces

Interface	Type	Access Right	Privilege
shareTag	required	authenticated	normal
peer	required	authenticated	normal

12.9.3. Structure



Name	Type	Description	Surface	Access Rights
Event	Information	An object incorporating messaging attributes and additional webinos specific payload data and meta-data	Undefined	Undefined

NFC Event	Information	NFC Event	JSON	authenticated
NFC Listener	Software	NFC	Privileged application	authenticated
NFC Message	Information	NFC Message	NDEF	authenticated
NFC Record	Information	NFC Record	NDEF	authenticated
NFC Service	Software	NFC Service Implementation	Privileged application	authenticated
NFC Tag	Information	NFC Tag	NDEF	anonymous
Webinos Service	Software	Webinos Service Implementation	Privileged application	authenticated

12.9.4. Component Requirements

None

12.10.OpenID Proxy

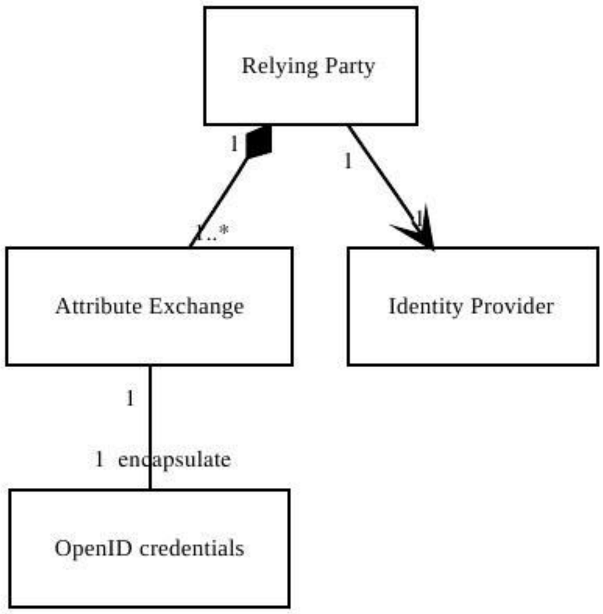
12.10.1. Description

OpenID Proxy

12.10.2. Interfaces

Interface	Type	Access Right	Privilege
authenticate	required	authenticated	normal

12.10.3. Structure



Name	Type	Description	Surface	Access Rights
Attribute Exchange	Information	Attribute Exchange	JSON	authenticated
Identity Provider	Systems	Interface through which an Identity Provider registers and provides authentication credentials.	Undefined	Undefined
OpenID credentials	Information	OpenID credentials	Plaintext	authenticated
Relying Party	Information	Relying Party	JSON	authenticated

12.10.4. Component Requirements

Name	Definition	Concerns	Responsibility
OpenID provider online	PZH administrative access shall be possible only if the OpenID provider is online.	Identity Provider	None

12.11. Policy Manager

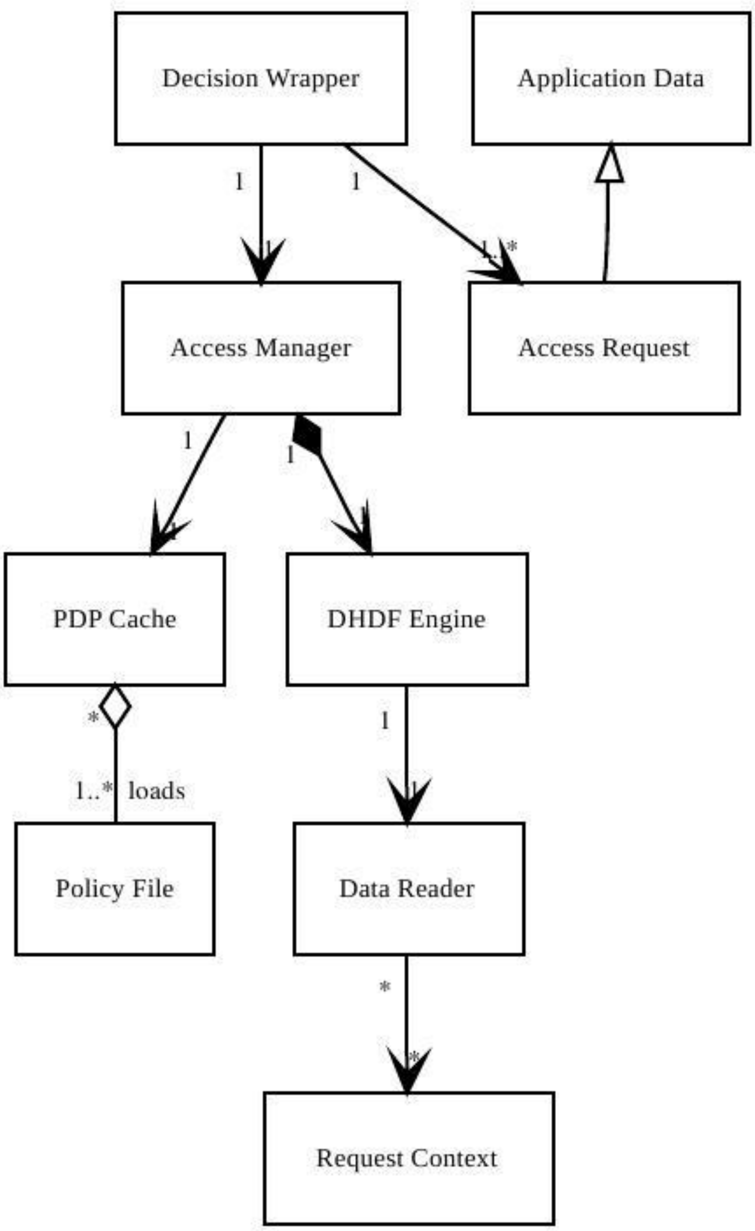
12.11.1. Description

XACML based policy manager component.

12.11.2. Interfaces

Interface	Type	Access Right	Privilege
enforceRequest	required	trusted	normal

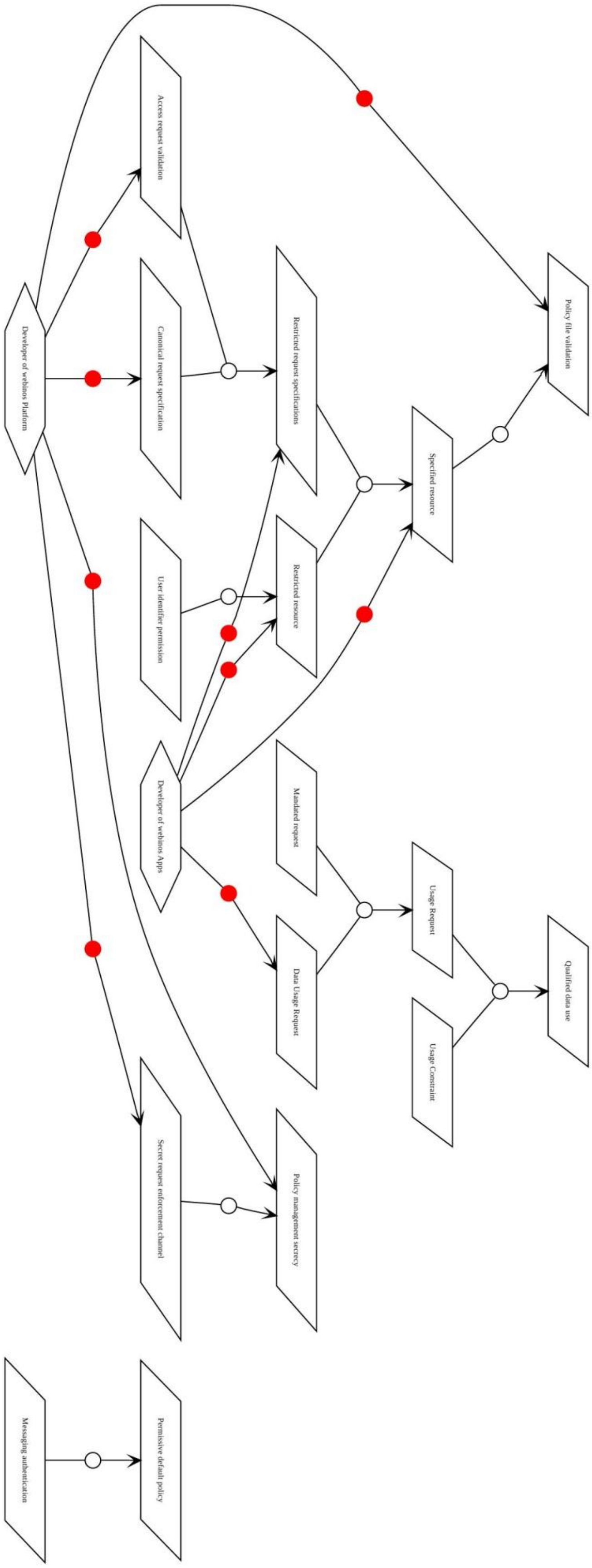
12.11.3. Structure



Name	Type	Description	Surface	Access Rights
Access Manager	Software	Makes policy decisions	Privileged application	trusted

Access Request	Information	Request for a resources	JSON	authenticated
Application Data	Information	Application Data	JSON	authenticated
Data Reader	Software	Data Reader	Privileged application	trusted
Decision Wrapper	Software	Manages the dialogue between clients and the policy manager	Privileged application	trusted
DHDF Engine	Software	Provides privacy and data handling functionality	Privileged application	trusted
PDP Cache	Software	PDP Cache	Privileged application	trusted
Policy File	Information	XACML Policy File	Unconstrained XML	trusted
Request Context	Information	Encapsulates the data and credentials released by a user in a given session.	Unconstrained XML	trusted

12.11.4. Component Requirements


Access request validation
Developer of webinos Platform
Canonical request specification
Restricted request specifications
Policy file validation
Specified resource
User identifier permission
Restricted resource
Developer of webinos Apps
Mandated request
Usage Request
Data Usage Request
Qualified data use
Usage Constraint
Secret request enforcement channel
Policy management secrecy
Messaging authentication
Permissive default policy

Name	Definition	Concerns	Responsibility
Access request validation	Access requests shall contain a validating DTD or schema.	Access Request	Developer of webinos Platform
API control	Users and other stakeholders shall be able to control access by web applications to JavaScript APIs. These APIs may allow access to local and remote resources.	Access Requestor	None
Canonical request specification	The request specification shall be verified before use.	Access Request	Developer of webinos Platform
Context-sensitive access control	The platform shall allow context-sensitive access control decisions: e.g. these may change depending on the environment.	Request Context	None
Data Usage Request	Users shall specify how they will use data they are requesting.	Access Request	Developer of webinos Apps
Device policy	Users shall be able to create both device-specific and device-agnostic policies.	Access Requestor	None
Mandated request	Access requests shall be non-repudiable.	None	None
Messaging authentication	Messaging API users shall be authenticated before API use.	None	None
Permissive default policy	The default webinos policy shall be permissive enough to require no modification for the majority of webinos applications.	Policy File	None
Policy file validation	Policy files shall contain a validating DTD or schema.	Policy File	Developer of webinos Platform
Policy management secrecy	Policy management requests shall be visible only to authorised users.	Access Request	Developer of webinos Platform
Policy synchronisation	The platform shall provide synchronisation for access control policies, so that policies can be desceibed on one platform and enforced on all.	PDP Cache	None

Qualified data use	The platform shall protect user privacy: access requestors shall be able to qualify how they will use the data they are requesting, and users shall be able to express constraints about data disclosure.	Request Context	None
Restricted request specifications	User agent requests to network resources shall be restricted.	Access Request	Developer of webinos Apps
Restricted resource	Resource restrictions shall be commensurate with their specifications.	Access Request	Developer of webinos Apps
Secret request enforcement channel	Request enforcement channels shall be visible only to authorised users.	None	Developer of webinos Platform
Specified resource	Access requests shall specify the resources required authorisation.	Access Request	Developer of webinos Apps
Usage Constraint	Users shall specify constraints about data disclosure.	None	None
Usage Request	Access requestors shall specify how they will use the data they are requesting.	None	None
User identifier permission	Application shall be authorised to access to user interface.	None	None

12.12.PZH Provider

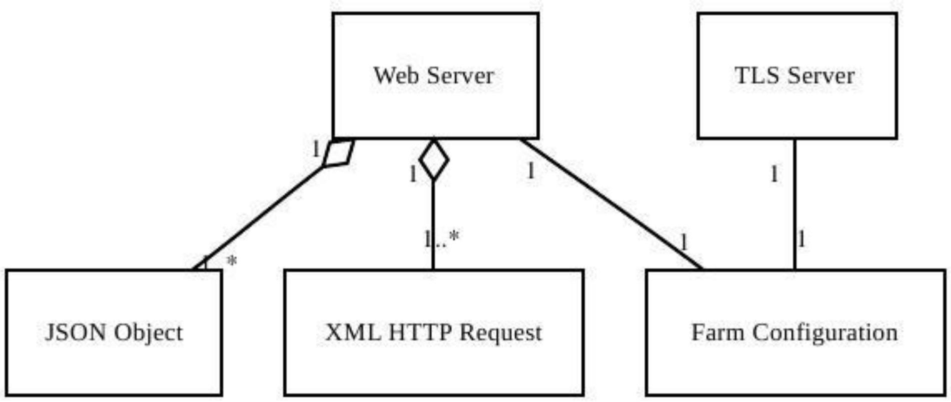
12.12.1. Description

PZH Provider

12.12.2. Interfaces

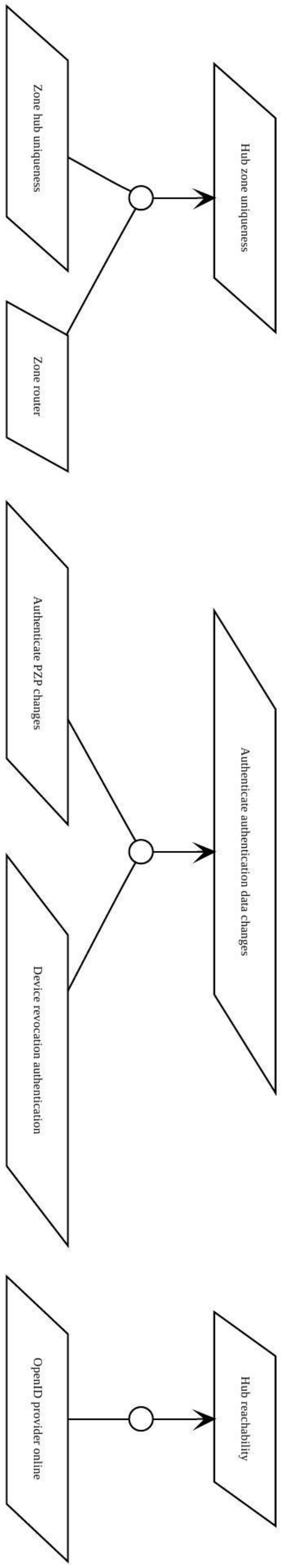
Interface	Type	Access Right	Privilege
on	required	trusted	privileged
addPzh	provided	authenticated	privileged
authenticate	provided	authenticated	normal
connect	required	trusted	normal

12.12.3. Structure



Name	Type	Description	Surface	Access Rights
Farm Configuration	Information	Configuration	JSON	authenticated
JSON Object	Information	JSON Object	JSON	authenticated
TLS Server	Software	TLS Server	Privileged application	trusted
Web Server	Software	Web Server	Privileged application	trusted
XML HTTP Request	Information	XML HTTP Request	JSON	trusted

12.12.4. Component Requirements



Name	Definition	Concerns	Responsibility
Authenticate authentication data changes	Re-authentication shall be necessary to update authentication data.	None	None
Authenticate PZP changes	Re-authentication shall be necessary to update PZP configuration data.	None	None
Click-through controls	Permission prompts shall prevent dialog click-through.	None	None
Device revocation authentication	Re-authentication shall be necessary to revoke devices from a personal zone.	None	None
Hub reachability	When online, personal zone hubs shall be reachable from the public internet.	TLS Server, Web Server	None
Hub zone uniqueness	A personal zone hub shall be associated with no more than one personal zone.	None	None
Zone hub uniqueness	Each personal zone shall be associated with a single personal zone hub.	None	None
Zone router	If required, the personal zone hub shall route communication between devices in a personal zone.	None	None

12.13.PZH Session Handler

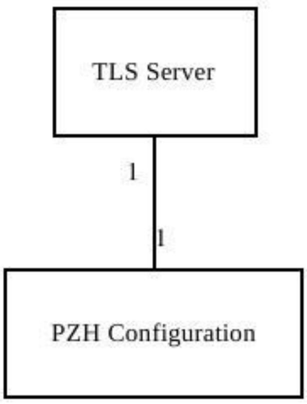
12.13.1. Description

PZH

12.13.2. Interfaces

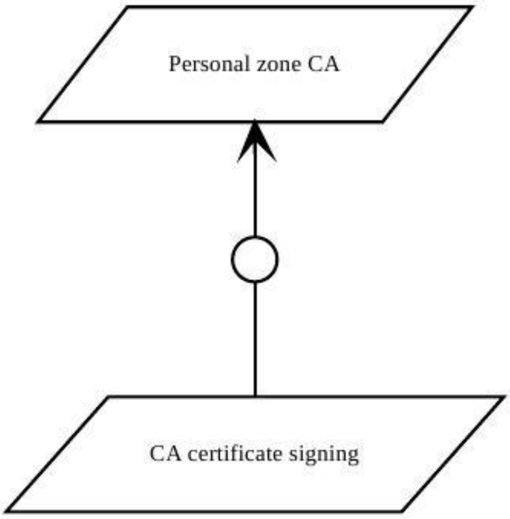
Interface	Type	Access Right	Privilege
addPzh	required	authenticated	privileged
getAllServices	provided	authenticated	normal
registeredServices	provided	authenticated	normal
addNewPZPCert	required	trusted	normal

12.13.3. Structure



Name	Type	Description	Surface	Access Rights
PZH Configuration	Information	PZH Configuration	JSON	authenticated
TLS Server	Software	TLS Server	Privileged application	trusted

12.13.4. Component Requirements



Name	Definition	Concerns	Responsibility
CA certificate signing	The personal zone's CA certificate shall be signed by the service provider who owns the infrastructure the PZH is running on.	None	None
Personal zone CA	The personal zone hub shall be the certificate authority for a personal zone.	None	None

12.14.PZP Session Handler

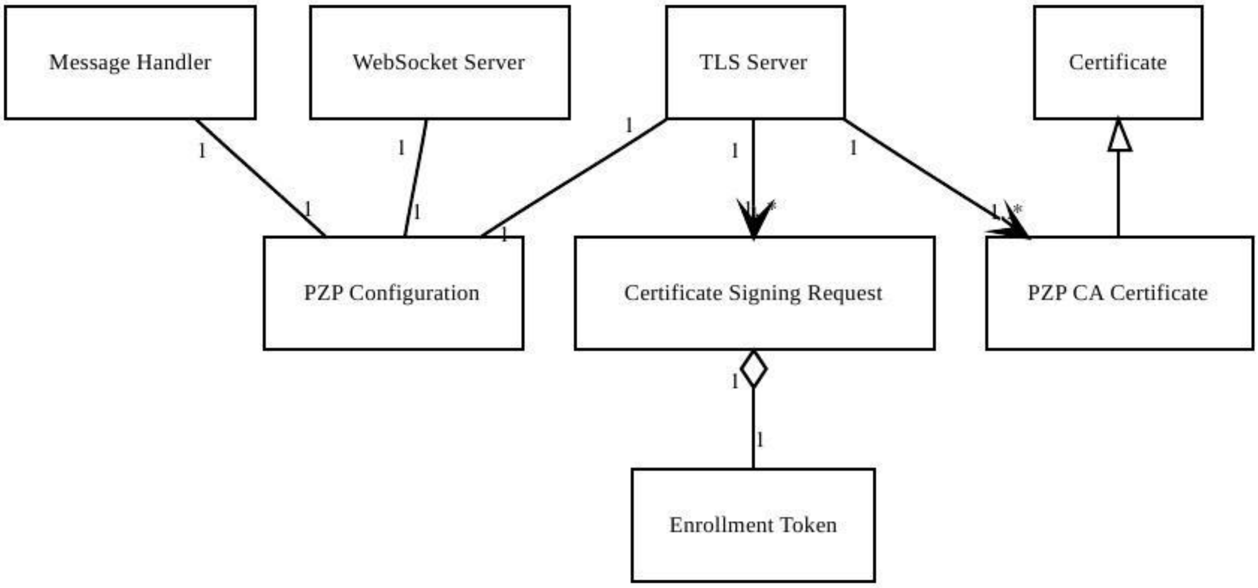
12.14.1. Description

Responsible for the creation of sessions, loading sessions during reconnections, and loading user preferences

12.14.2. Interfaces

Interface	Type	Access Right	Privilege
createCertificateRequest	provided	trusted	privileged
connect	provided	trusted	normal
addNewPZPCert	provided	trusted	normal
getKey	provided	trusted	normal
getRegisteredService	provided	authenticated	normal
registerServices	required	authenticated	normal

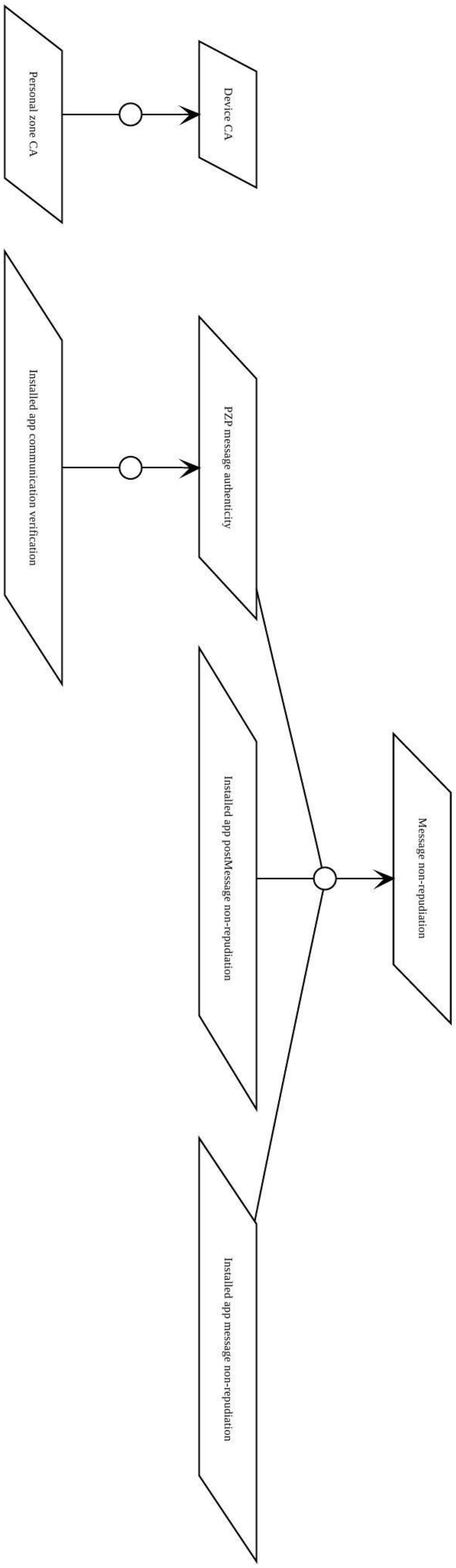
12.14.3. Structure



Name	Type	Description	Surface	Access Rights
Certificate	Information	X.509 Certificate	Structured Text	authenticated
Certificate	Information	A message sent from an applicant	Structured	trusted

Signing Request		to a certificate authority in order to apply for a digital identity certificate.	Text	
Enrollment Token	Information	Token generated by PZH in order that a PZP can be enrolled into a personal zone.	Structured Text	authenticated
Message Handler	Software	Message Handler	Privileged application	trusted
PZP CA Certificate	Information	PZP CA Certificate	Structured Text	authenticated
PZP Configuration	Information	PZP Configuration	JSON	authenticated
TLS Server	Software	TLS Server	Privileged application	trusted
WebSocket Server	Software	WebSocket Server	Privileged application	trusted

12.14.4. Component Requirements


Personal zone CA
Device CA
Installed app communication verification
PZP message authenticity
Installed app postMessage non-repudiation
Message non-repudiation
Installed app message non-repudiation

Name	Definition	Concerns	Responsibility
Device CA	The personal zone proxy shall be the certificate authority for a device enrolled in a personal zone.	None	None
Installed app communication verification	Installed applications shall verify communication to webinos applications.	None	None
Installed app message non-repudiation	Installed applications shall verify the authenticity of event message origin.	None	None
Installed app postMessage non-repudiation	Installed applications shall disallow postMessages from unrecognised origins.	None	None
Message non-repudiation	Event messages shall be non-repudiable.	None	None
PZP message authenticity	PZPs shall authenticate the origin of messages they send.	None	None
PZP token	A token generated by the personal zone's hub shall be presented by the PZP when enrolling a device to the personal zone.	None	None

12.15.TV Manager

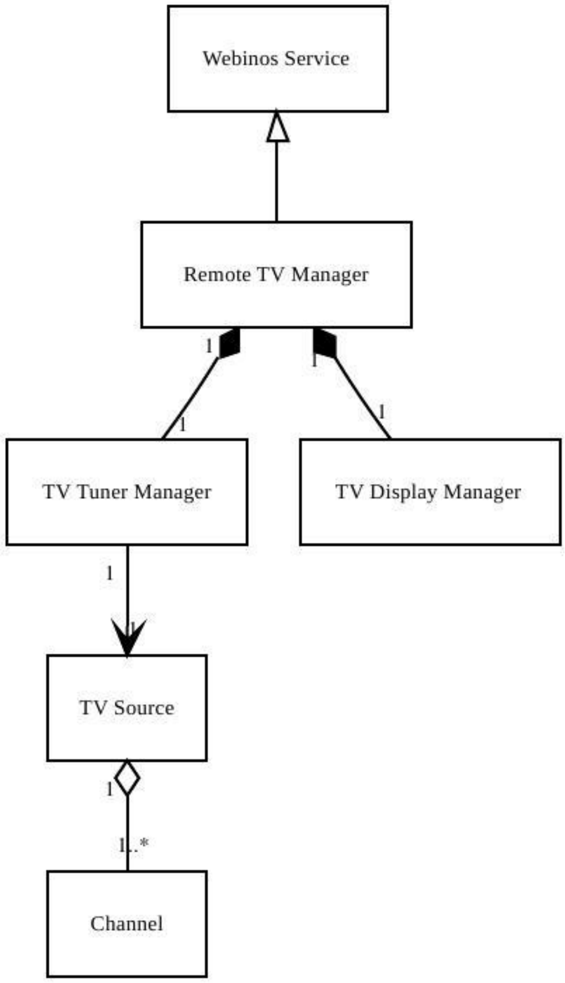
12.15.1. Description

TV API implementation implementation

12.15.2. Interfaces

Interface	Type	Access Right	Privilege
enforceRequest	provided	authenticated	normal
getTVSources	required	authenticated	normal

12.15.3. Structure



Name	Type	Description	Surface	Access Rights
Channel	Information	Channel	JSON	authenticated
Remote TV Manager	Software	Remote TV Manager	Privileged application	authenticated
TV Display Manager	Software	TV Display Manager	Privileged application	authenticated
TV Source	Information	List of channels with a name	JSON	authenticated
TV Tuner Manager	Software	TV Tuner Manager	Privileged application	authenticated
Webinos Service	Software	Webinos Service Implementation	Privileged application	authenticated

12.15.4. Component Requirements

None

12.16.User Agent

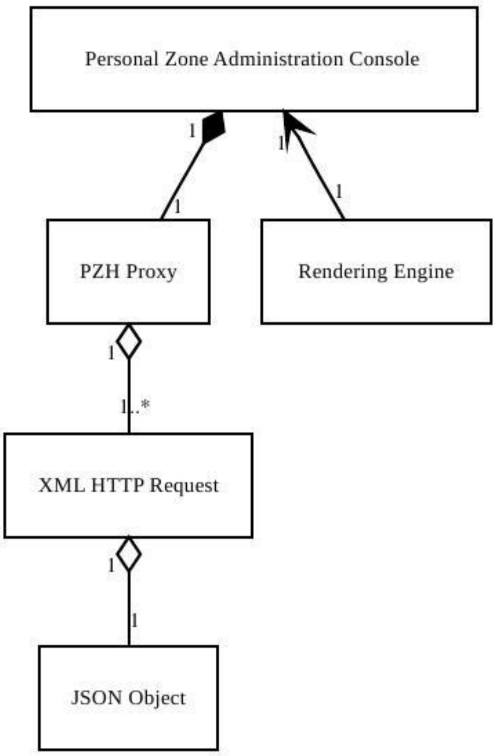
12.16.1. Description

Browser user agent

12.16.2. Interfaces

Interface	Type	Access Right	Privilege
post	required	trusted	privileged

12.16.3. Structure



Name	Type	Description	Surface	Access Rights
JSON Object	Information	JSON Object	JSON	authenticated
Personal Zone Administration Console	Systems	The web interface used to control the personal zone.	Undefined	Undefined
PZH Proxy	Information	PZH Proxy	JSON	authenticated

Rendering Engine	Software	Rendering Engine	Client application	anonymous
XML HTTP Request	Information	XML HTTP Request	JSON	trusted

12.16.4. Component Requirements

None

12.17.webinos API

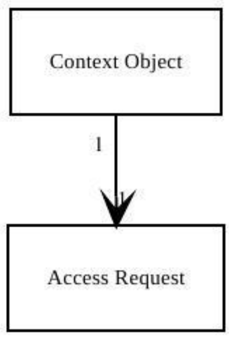
12.17.1. Description

General webinos API

12.17.2. Interfaces

Interface	Type	Access Right	Privilege
logContext	provided	trusted	normal

12.17.3. Structure



Name	Type	Description	Surface	Access Rights
Access Request	Information	Request for a resources	JSON	authenticated
Context Object	Information	Structure contextual information	JSON	authenticated

12.17.4. Component Requirements

None

12.18.Widget Manager

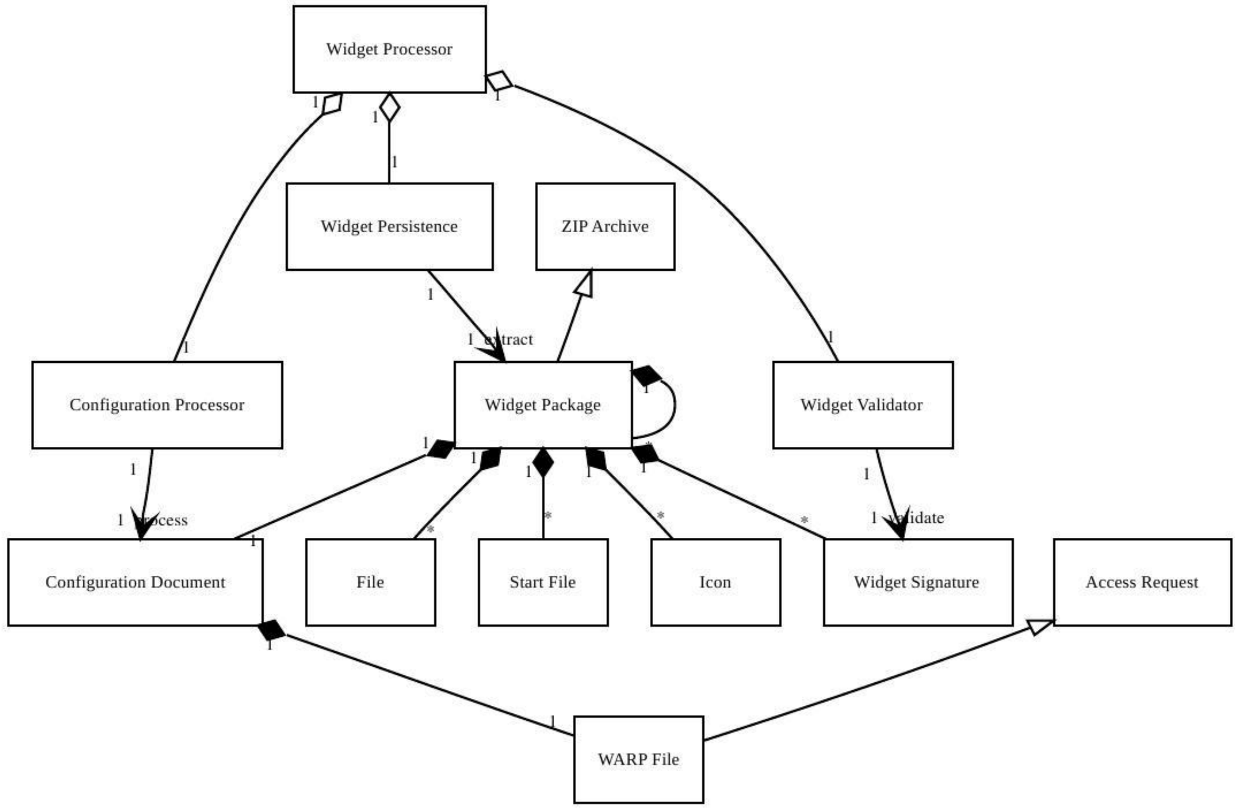
12.18.1. Description

W3C based Widget Manager

12.18.2. Interfaces

Interface	Type	Access Right	Privilege
validatePolicy	provided	trusted	privileged

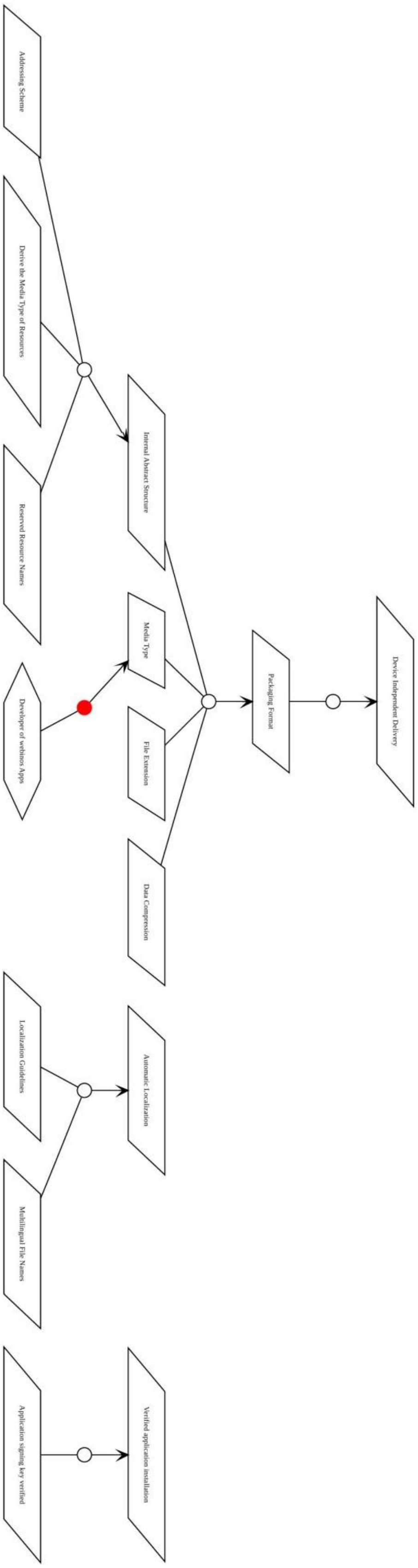
12.18.3. Structure



Name	Type	Description	Surface	Access Rights
Access Request	Information	Request for a resources	JSON	authenticated
Configuration Document	Information	The XML vocabulary that declares metadata and configuration parameters for a widget.	Unconstrained XML	authenticated
Configuration Processor	Software	Configuration Processor	Privileged application	trusted

File	Information	A decompressed physical representation of a file entry.	Unconstrained Image	anonymous
Icon	Information	File used to represent the widget in various application contexts; this helps users of visual browsers recognise the widget at a glance.	Unconstrained Image	authenticated
Start File	Information	A file from the widget package to be loaded by the user agent when it instantiates the widget.	HTML	authenticated
WARP File	Information	Extension to the configuration document controlling network access from within a widget.	Unconstrained XML	trusted
Widget Package	Software	Client side application packaged for distribution.	Client application	authenticated
Widget Persistence	Software	Widget Persistence implementation	Privileged application	trusted
Widget Processor	Software	Widget Processor	Privileged application	trusted
Widget Signature	Information	Widget Digital Signature	Constrained XML	trusted
Widget Validator	Software	Widget Validator	Privileged application	trusted
ZIP Archive	Information	Archive conforming to PKWare ZIP specification.	Client application	anonymous

12.18.4. Component Requirements


Addressing Scheme
Derive the Media Type of Resources
Reserved Resource Names
Developer of webinos Apps
Internal Abstract Structure
Media Type
File Extension
Data Compression
Packaging Format
Device Independent Delivery
Localization Guidelines
Multilingual File Names
Automatic Localization
Application signing key verified
Verified application installation

Name	Definition	Concerns	Responsibility
Addressing Scheme	A conforming specification MUST recommend a hierarchical addressing scheme that can be used to address the individual resources within a widget package from within a configuration document. The hierarchical addressing scheme MUST be capable of expressing both absolute and relative relationships between a resource and the widget package. In addition, the hierarchical addressing scheme MUST be interoperable with resources that might also need to address other resources within the widget package (e.g., HTML documents, CSS documents, JavaScript documents, etc.). The hierarchical addressing scheme SHOULD be one that Web authors would feel comfortable using or to which they are already accustomed.	Configuration Document	None
Application signing key verified	Update procedure shall verify the update signing key matches the original signing key.	None	None
Automatic Localization	A conforming specification SHOULD specify a processing model that automatically localizes content when authors follow the localization guidelines.	Configuration Processor	None
Data Compression	A conforming specification MUST recommend a packaging format that supports both uncompressed data and OPTIONAL data compression. A conforming specification SHOULD also recommend at least one royalty-free default compression/decompression algorithm that is compatible with market-leading widget user agents and implementations of the packaging format on mobile devices.	ZIP Archive	None
Derive the Media Type of Resources	In the case that the packaging format does not support labeling resources with a media type, a conforming specification MUST either specify or recommend a means of deriving the media type of resources for the purposes of rendering. A conforming specification MAY define a means to	Configuration Document	None

	override how a widget user agent derives the media type of a resource (e.g., treat resources with the file extension .php as text/html), but MUST NOT force a widget user agent to process resources of one media type as that of another type (e.g. treating a jpeg image at text/html).		
Device Independent Delivery	A conforming specification MUST recommend a packaging format that is suitable for delivery on many devices, particularly Web-enabled mobile devices.	ZIP Archive	None
File Extension	A conforming specification MUST specify a file extension that authors MAY assign to widget packages in contexts that rely on file extensions to indicate the content type, such is the case on many popular file systems.	ZIP Archive	None
Internal Abstract Structure	A conforming specification MUST recommend a packaging format that supports structuring resources into collections such as files and directories (understood in this document in a broader sense than in some popular file systems, namely as forms of generic logical containers). In addition, the packaging format SHOULD allow authors to add and remove resources of a widget package without needing to recreate the widget package.	Configuration Processor	None
Localization Guidelines	A conforming specification MUST provide guidelines that explain to authors how collections of resources need to be structured for the purpose of internationalization.	Configuration Document	None
Media Type	A conforming specification MUST recommend that a conforming widget package be sent over HTTP with a formally registered media type that is specific to the Widgets 1.0 Family of specifications. The Working Group MUST formally register the media type with the Internet Assigned Numbers Authority (IANA). A conforming specification MUST specify how a widget user agent will process a widget package that was served with an unsupported media	Widget Package	Developer of webinos Apps

	type or when the media type is unspecified.		
Multilingual File Names	A conforming specification MUST recommend a packaging format that allows for non-ASCII characters in file and directory names, allowing authors to create widgets suitable for various cultures and languages, as well as multilingual contexts. The packaging format MUST either provide some means to declare the character encoding or specify what the character encoding is. The UTF-8 character encoding SHOULD be either the default (if multiple encodings are allowed) or sole encoding used.	Widget Package	None
Packaging Format	A conforming specification must recommend a packaging format that is royalty free, open, and widely implemented across market-leading widget user agents and compatible with mobile devices. In addition, a conforming specification must specify exactly which aspects of the packaging format are to be supported by conforming widget user agents.	Widget Package	None
Reserved Resource Names	A conforming specification MUST indicate if any resources (files or directories or similar logical containers) are mandatory or reserved and what specific function they serve in the widget package. A conforming specification SHOULD specify graceful error handling/recovery procedures if those resources are used erroneously or missing.	Configuration Document	None
Verified application installation	webinos applications shall be verified before installation.	Widget Processor, Widget Validator	None

12.19.Widget Renderer

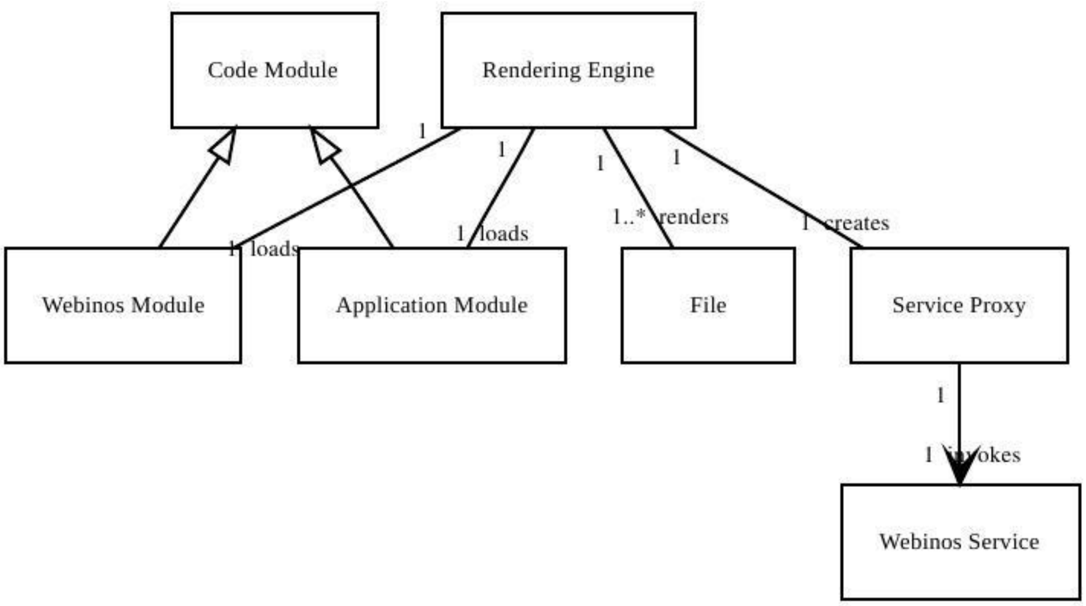
12.19.1. Description

W3C based Widget Manager

12.19.2. Interfaces

Interface	Type	Access Right	Privilege
findServices	required	authenticated	normal

12.19.3. Structure



Name	Type	Description	Surface	Access Rights
Application Module	Information	Application source module	Structured Text	anonymous
Code Module	Information	Source code module	Structured Text	anonymous
File	Information	A decompressed physical representation of a file entry.	Unconstrained Image	anonymous
Rendering Engine	Software	Rendering Engine	Client application	anonymous
Service Proxy	Software	webinos service interface	Client application	authenticated
Webinos Module	Information	webinos source module	Structured Text	anonymous
Webinos Service	Software	Webinos Service Implementation	Privileged application	authenticated

12.19.4. Component Requirements

None

13 Attack Surface metrics

13.1. Access Rights

These metrics define the access rights necessary for a subject to make use of a protocol, privilege level, or asset.

Name	Description	Value
anonymous	Subject can access resource anonymously.	1
authenticated	Subject needs to be explicitly authenticated to access resource.	5
trusted	Subject needs to be trusted to access resource.	10
Undefined	Access rights undefined.	10

13.2. Protocols

These metrics define the protocol used in component connections.

Name	Description	Value
TLS	Transport Layer Security (TLS)	1
PC	Synchronous procedural call	5
JSON-RPC	Unencrypted JSON-RPC	10
LLCP	NFC Logical Link Control Protocol	10
RPC	Generic remote procedure call	10
Undefined	Protocol undefined	10

13.3. Privileges

These metrics define the level of privilege that a component interface operates at.

Name	Description	Value
normal	Subject operates at a non-privileged level of operation.	1
privileged	Subject operates at a privileged level of operation.	10

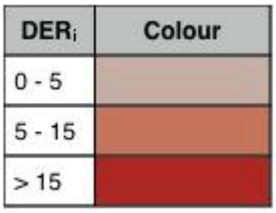
Undefined	Subject operates at an unknown level of privilege.	10

13.4. Surface Types

These metrics define the type of surface used by component assets.

Name	Description	Value
Constrained XML	Well-formed, non-externally validated XML document.	1
HTML	HTML document	1
Privileged application	Trusted webinos application or software component, usually running at an elevated privilege level.	1
Unconstrained Image	Image files	1
NDEF	NFC Data Exchange Format	5
Plaintext	ASCII plaintext	5
Structured Text	Structured Text	5
Unconstrained XML	Well-formed, non-externally validated XML document.	5
Client application	Untrusted client application or software component.	10
JSON	JavaScript Object Notation (JSON) representing data structures and algorithms.	10
Undefined	Access rights undefined.	10

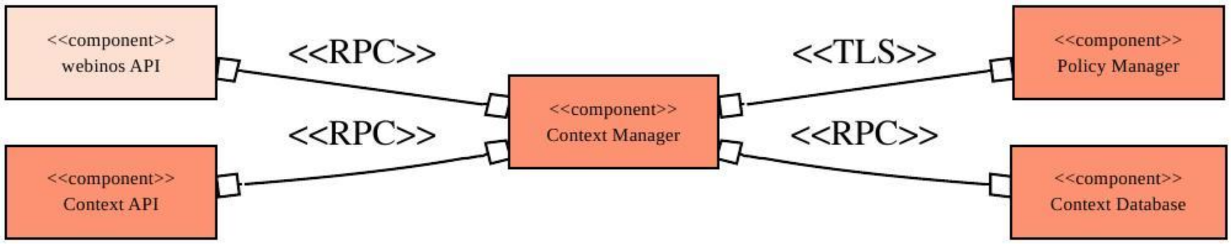
13.5. Damage potential for untrusted asset surface: Colour codes



DER$_i$	Colour
0 - 5	🟫
5 - 15	🟧
> 15	🟥

14 Architectural Patterns

14.1. Context Policy Management



14.1.1. Synopsis

Model illustrating how policy management mediates the

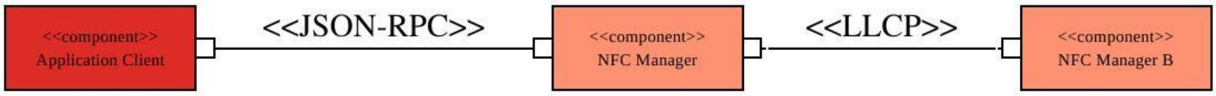
14.1.2. Components

- Context API
- Context Database
- Context Manager
- Policy Manager
- webinos API

14.1.3. Connectors

From	Role (Interface)	To	Role (Interface)	Protocol	Access Right
Context Manager	request-insertContext (handleContextData)	Context Database	provide-insertContext (handleContextData)	RPC	authenticated
webinos API	request-logcontext (logContext)	Context Manager	provide-logcontext (logContext)	RPC	authenticated
Context Manager	request-permission (enforceRequest)	Policy Manager	provide-permission (enforceRequest)	TLS	trusted
Context API	request-queryContext (queryContext)	Context Manager	provide-queryContext (queryContext)	RPC	authenticated

14.2. NFC



14.2.1. Synopsis

Model illustrating the components associated with NFC

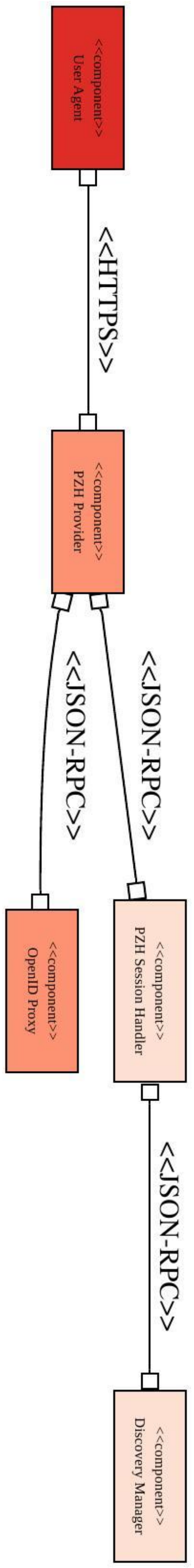
14.2.2. Components

- Application Client
- NFC Manager
- NFC Manager B

14.2.3. Connectors

From	Role (Interface)	To	Role (Interface)	Protocol	Access Right
Application Client	request-nfcapi (shareTag)	NFC Manager	provide-nfcapi (shareTag)	JSON-RPC	authenticated
NFC Manager	request-peerservices (peer)	NFC Manager B	provide-peerservices (peer)	LLCP	authenticated

14.3. PZH Authentication



14.3.1. Synopsis

Model illustrating the components associated PZH authentication.

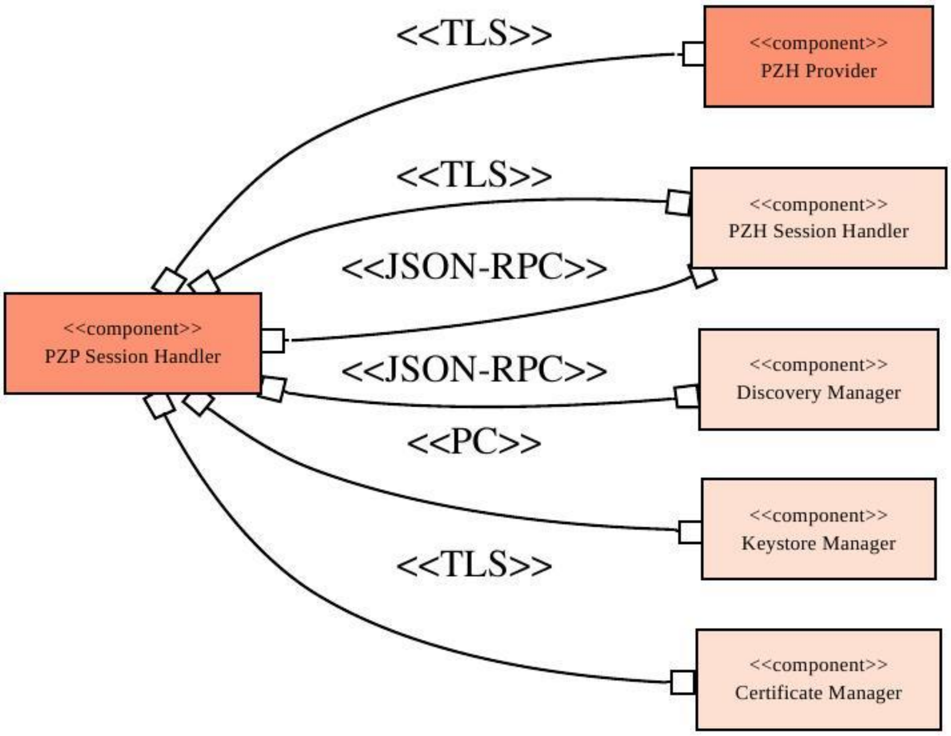
14.3.2. Components

- Discovery Manager
- OpenID Proxy
- PZH Provider
- PZH Session Handler
- User Agent

14.3.3. Connectors

From	Role (Interface)	To	Role (Interface)	Protocol	Access Right
PZH Provider	request-addpzh (addPzh)	PZH Session Handler	provide-addpzh (addPzh)	JSON-RPC	authenticated
PZH Session Handler	request-registeredservices (getAllServices)	Discovery Manager	provide-registeredservices (getAllServices)	JSON-RPC	authenticated
PZH Provider	request-authenticate (authenticate)	OpenID Proxy	provide-authenticate (authenticate)	JSON-RPC	authenticated
User Agent	post-http (post)	PZH Provider	receive-https (on)	TLS	trusted

14.4. PZP Enrolment



14.4.1. Synopsis

Components associated with enrolling and authenticating PZPs to PZHs

14.4.2. Components

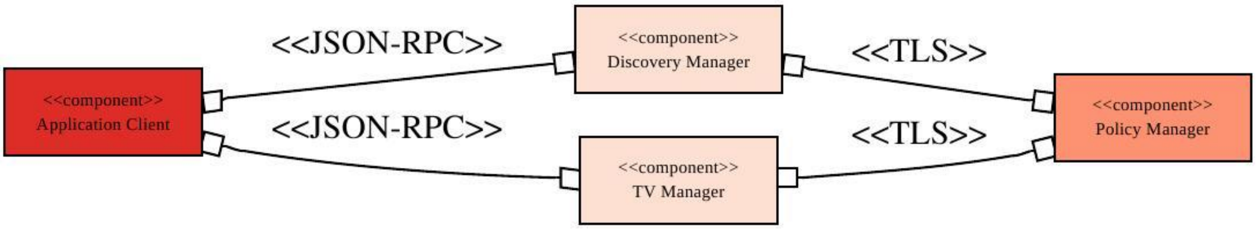
- Certificate Manager
- Discovery Manager
- Keystore Manager
- PZH Provider
- PZH Session Handler
- PZP Session Handler

14.4.3. Connectors

From	Role (Interface)	To	Role (Interface)	Protocol	Access Right
PZP Session Handler	request-addPZPCert (addNewPZPCert)	PZH Session Handler	provide-addPZPCert (addNewPZPCert)	TLS	authenticated
PZP Session Handler	request-createCSR (createCertificateRequest)	Certificate Manager	provide-createCSR (createCertificateRequest)	TLS	authenticated

PZP Session Handler	request-keyservices (getKey)	Keystore Manager	provide-keyservices (get)	PC	trusted
PZP Session Handler	request-pzpconnect (connect)	PZH Provider	provide-pzpconnect (connect)	TLS	trusted
PZP Session Handler	request-allservices (getRegisteredServices)	Discovery Manager	provide-registeredservices (getRegisteredServices)	JSON-RPC	authenticated
PZP Session Handler	send-remoteservices (registerServices)	PZH Session Handler	recv-remoteservices (registeredServices)	JSON-RPC	authenticated

14.5. TV Service Discovery



14.5.1. Synopsis

Model illustrating discovery and use of TV services

14.5.2. Components

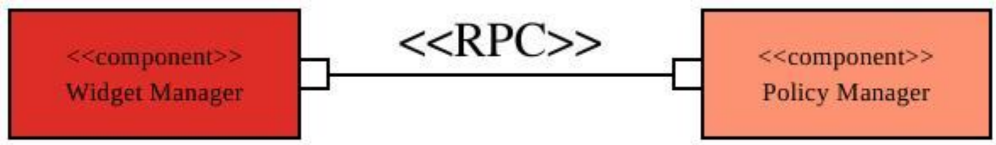
- Application Client
- Discovery Manager
- Policy Manager
- TV Manager

14.5.3. Connectors

From	Role (Interface)	To	Role (Interface)	Protocol	Access Right
Discovery Manager	request-permission (enforceRequest)	Policy Manager	provide-permission (enforceRequest)	TLS	trusted
Application Client	request-service (findServices)	Discovery Manager	provide-service (findServices)	JSON-RPC	authenticated

Application Client	request-tvapi (getTVSources)	TV Manager	provide-tvapi (getTVSources)	JSON-RPC	authenticated
TV Manager	request-permission (enforceRequest)	Policy Manager	provide-permission (enforceRequest)	TLS	trusted

14.6. Widget Processing



14.6.1. Synopsis

Model illustrating the components associated with widget processing

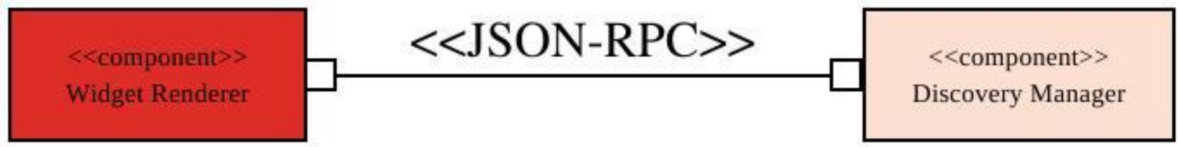
14.6.2. Components

- Policy Manager
- Widget Manager

14.6.3. Connectors

From	Role (Interface)	To	Role (Interface)	Protocol	Access Right
Widget Manager	request-validatePolicy (validatePolicy)	Policy Manager	provide-validatePolicy (enforceRequest)	RPC	trusted

14.7. Widget Rendering



14.7.1. Synopsis

Model illustrating the components associated with widget rendering

14.7.2. Components

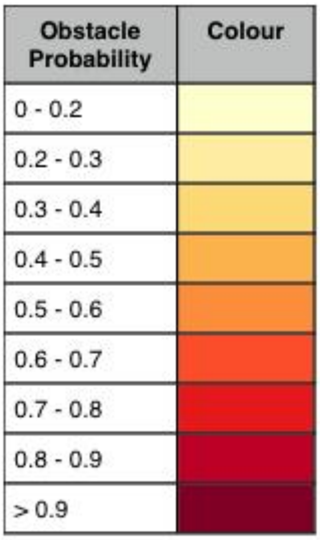
- Discovery Manager
- Widget Renderer

14.7.3. Connectors

From	Role (Interface)	To	Role (Interface)	Protocol	Access Right
Widget Renderer	request-findservices (findServices)	Discovery Manager	provide-findservices (findServices)	JSON-RPC	authenticated

15 Contextualised Attack Patterns

15.1. Obstacle probability: colour codes



Obstacle Probability	Colour
0 - 0.2	🟨
0.2 - 0.3	🟨
0.3 - 0.4	🟨
0.4 - 0.5	🟧
0.5 - 0.6	🟧
0.6 - 0.7	🟥
0.7 - 0.8	🟥
0.8 - 0.9	🟥
> 0.9	🟫

15.2. Application DoS

15.2.1. Intent

Harold has developed a webinos application he is very proud of. Unfortunately it is not receiving much attention because another app (which he believes to be inferior) is extremely popular. Harold decides to get rid of the competition, which he feels will be best for everyone: he gets the recognition he deserves and users get a better experience.

15.2.2. Motivation

Security Goal	Value	Description
Availability	High	Users become unable to reliably use a certain app

15.2.3. Structure

Attack: Malicious Automated Software Update	Origin: CAPEC	Obstacle: The Application is auto-updated by an attacker
Exploit: Improper Verification of Cryptographic Signature	Origin: CWE	Obstacle: Application signing key not checked

15.2.4. Participants

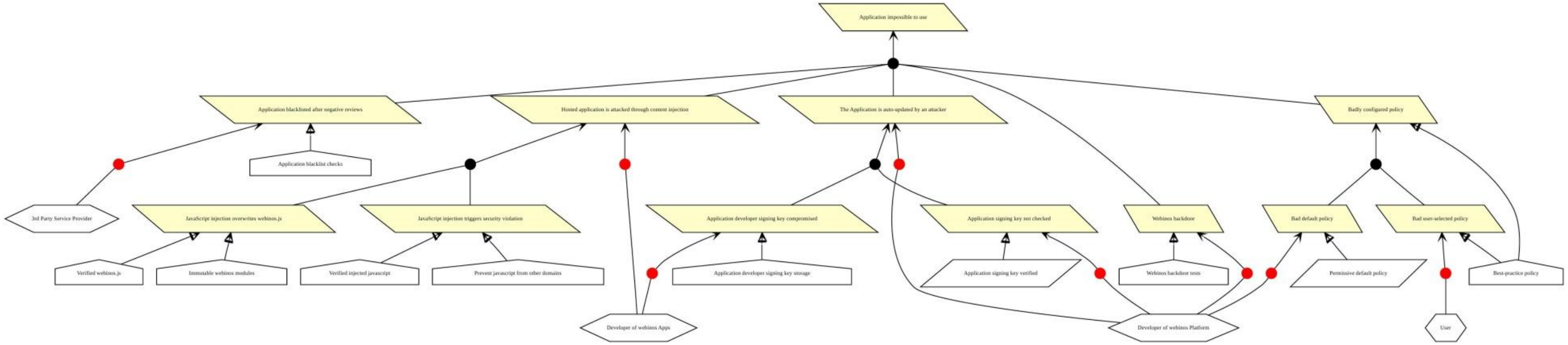
Attacker	Motives	Capabilities (Value)

Harold	Money	Knowledge/Methods(High), Software(Medium)

15.2.5. Collaboration

Target	Application Module
Exploit	Widget Processor

15.2.6. Implementation


Application impossible to use
Application blacklisted after negative reviews
Hosted application is attacked through content injection
The Application is auto-updated by an attacker
Badly configured policy
Application blacklist checks
3rd Party Service Provider
JavaScript injection overwrites webinos.js
JavaScript injection triggers security violation
Application developer signing key compromised
Application signing key not checked
Webinos backdoor
Bad default policy
Bad user-selected policy
Verified webinos.js
Immutable webinos modules
Verified injected javascript
Prevent javascript from other domains
Application developer signing key storage
Application signing key verified
Webinos backdoor tests
Permissive default policy
Best-practice policy
Developer of webinos Apps
Developer of webinos Platform
User

Harold performs a denial of service attack on an application, rendering it unusable by end users.

15.2.7. Obstacles

Obstacle	Category	Definition
Application blacklisted after negative reviews	Vulnerability	The application is given several negative reviews and warnings on app stores and social networks, putting it on a blacklist
Application developer signing key compromised	Vulnerability	The application developer's signing key is leaked to the attacker, perhaps due to it being embedded in a source file or another means.
Application impossible to use	Vulnerability	A webinos application is no longer usable
Application signing key not checked	Vulnerability	Update procedure does not check that the update signing key matches the original signing key
Bad default policy	Vulnerability	The default webinos policy settings are too restrictive
Bad user-selected policy	Vulnerability	The user has selected bad policy settings
Badly configured policy	Vulnerability	The webinos policy settings are poorly chosen and the application is denied access to too many APIs
Hosted application is attacked through content injection	Vulnerability	The application's hosted components are attacked via a content injection attack, possibly XSS.
JavaScript injection overwrites webinos.js	Vulnerability	Injected javascript prevents the application from accessing PZP functionality by overwriting webinos.js
JavaScript injection triggers security violation	Vulnerability	Injected javascript causes the application to trigger security warnings and get disconnected by the PZP or webinos runtime
The Application is auto-updated by an attacker	Vulnerability	The application downloads an update automatically and installs it. However, the update was issued by an attacker rather than the application author
Webinos backdoor	Vulnerability	The webinos platform is updated to specifically target this application and render it unusable

15.3. Capture Hidden Analytics

15.3.1. Intent

Subversely capture analytics about application usage.

15.3.2. Motivation

Security Goal	Value	Description
Anonymity	High	Jimmy relies on anonymity to capture context data without users realising it.

15.3.3. Structure

Attack: Spyware	Origin: OWASP	Obstacle: None
Exploit: Lack of data provenance	Origin: D2.8	Obstacle: Apps share usage data

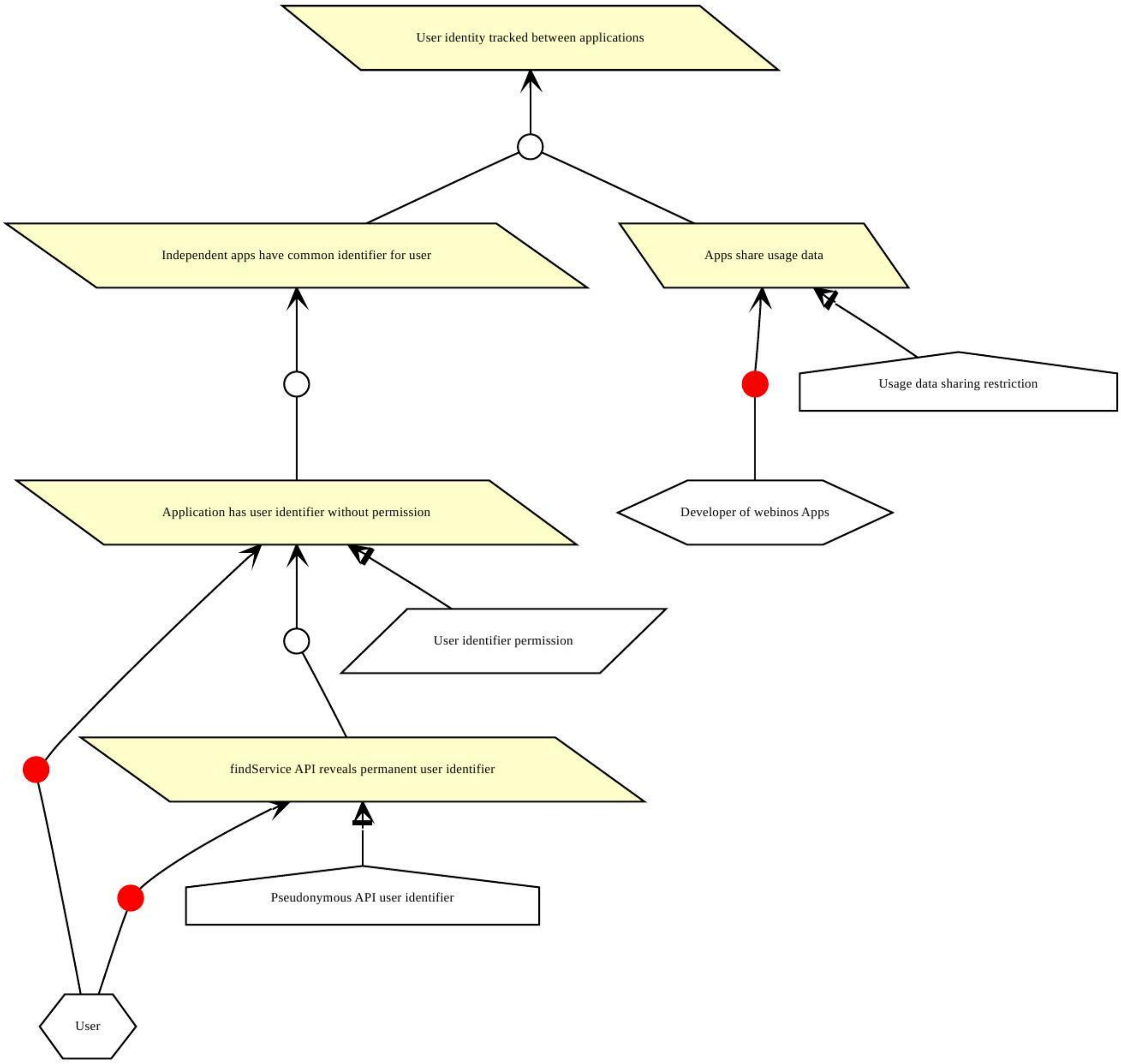
15.3.4. Participants

Attacker	Motives	Capabilities (Value)
Jimmy	Money	Knowledge/Methods(Medium), Software(Medium)

15.3.5. Collaboration

Target	Analytics Data
Exploit	Analytics Data

15.3.6. Implementation


User identity tracked between applications
Independent apps have common identifier for user
Apps share usage data
Usage data sharing restriction
Application has user identifier without permission
Developer of webinos Apps
User identifier permission
findService API reveals permanent user identifier
Pseudonymous API user identifier
User

Peter is about to head to Berlin for a long weekend and decides to download the Berlin Gallery Guide from the Android app store. This application gives Peter a quick overview of art galleries in Berlin and allows him to take snaps and submit reviews about places visited. Before downloading the app, Peter checks what the application has access to. Additionally, before running the app for the first time, he also scans the application's terms and conditions, briefly noting that the application captures information about time. Several days later, Peter is walking out of the Neue Nationalgalerie and submits some of his thoughts about the gallery as a review.

One month earlier, Jimmy was in the midst of a dilemma. His Berlin cafe guide was more successful than he thought, but he really wished that he had captured more analytics information; his new Berlin Gallery Guide might be just the opportunity he needs to get this data. While Peter really wants to capture information about location rather than time of review, this would mean re-drafting the privacy policy associated with the app to better reflect his intentions; this is both time consuming and, to his mind, unnecessary. "I mean" Jimmy thought, "I'm the only person using the data, and I'm not going to abuse it so who really cares if I capture location data rather than time in my app. The PDP associated with the running application won't notice the difference and, perhaps, maybe that means capturing this data is ok." Eventually, Jimmy convinces himself to reuse his previous privacy policy and configuration file entries related to permission, but change the Javascript code to store location data as context data rather than time.

Several weeks later, Jimmy feels vindicated by this decision as useful, fine-grained location data starts to arrive via his gallery guide app, including unauthorised location data arising about Peter's visit to Neue Nationalgalerie.

15.3.7. Obstacles

Obstacle	Category	Definition
Application has user identifier without permission	Vulnerability	One application is installed and is able to obtain a user identifier without requesting access to an API for this.
Apps share usage data	Vulnerability	The two application share usage data, either directly or through an intermediary analytics service
findService API reveals permanent user identifier	Vulnerability	The output of the findServices API is an effective user identifier.
Independent apps have common identifier for user	Vulnerability	Two or more independent applications have been installed in the personal zone and have the same user identifier.
User identity tracked between applications	Vulnerability	An application is able to successfully re-identify a user of a different application despite them restricting access to personally-identifying APIs.

15.4. Device availability loss

15.4.1. Intent

Part of a larger attack, Ethan tries to drum-up interest in his own "battery life extender" application by rendering his victim's devices unusuable due to battery life problems that he has caused.

15.4.2. Motivation

Security Goal	Value	Description
Availability	Medium	The device rapidly becomes unavailable as battery life is spent too quickly

15.4.3. Structure

Attack: Denial of Service through Resource	Origin:	Obstacle: Device effectively

Depletion	CAPEC	unavailable
Exploit: Uncontrolled Resource Consumption ('Resource Exhaustion')	Origin: CWE	Obstacle: Processor always busy

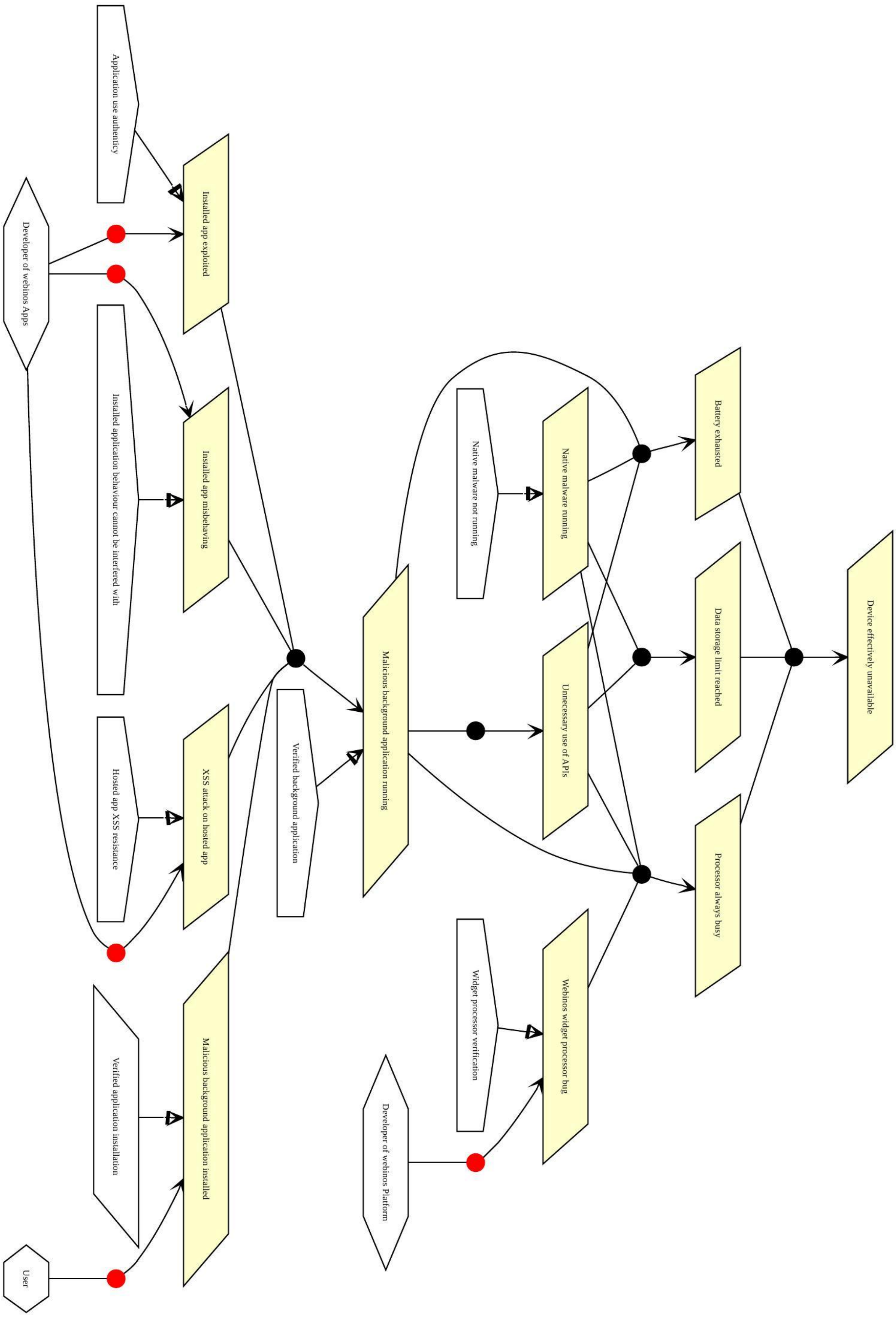
15.4.4. Participants

Attacker	Motives	Capabilities (Value)
Ethan	System resource theft	Knowledge/Methods(Medium), Software(Medium), Technology(Medium)

15.4.5. Collaboration

Target	Widget Processor
Exploit	Widget Processor

15.4.6. Implementation


Application use authenticity
Developer of webinos Apps
Installed app exploited
Installed application behaviour cannot be interfered with
Installed app misbehaving
Hosted app XSS resistance
XSS attack on hosted app
Verified application installation
Malicious background application installed
User
Verified background application
Malicious background application running
Native malware not running
Native malware running
Unnecessary use of APIs
Widget processor verification
Webinos widget processor bug
Developer of webinos Platform
Battery exhausted
Data storage limit reached
Processor always busy
Device effectively unavailable

Ethan exploits commonly used web applications to drain the battery life of users and encourage them to use his software.

15.4.7. Obstacles

Obstacle	Category	Definition
Battery exhausted	Vulnerability	The device's battery is out of capacity
Data storage limit reached	Vulnerability	The local storage is full, preventing normal operation
Device effectively unavailable	Vulnerability	The mobile device becomes effectively unusable for any function
Installed app exploited	Vulnerability	An installed application has been compromised and is now under the control of an attacker
Installed app misbehaving	Vulnerability	An installed and trusted application is behaving in unexpected ways with bad consequences
Malicious background application installed	Vulnerability	A malicious webinos application has been installed
Malicious background application running	Vulnerability	A background application is running and is behaving maliciously
Native malware running	Vulnerability	There is native malware installed on the device
Processor always busy	Vulnerability	The device's processor is constantly active, perhaps due to infinite loops or too many processes running
Unnecessary use of APIs	Vulnerability	APIs are being called repeatedly, just for the purpose of exhausting the device
Webinos widget processor bug	Vulnerability	The webinos platform contains a bug resulting in it acting unpredictably or looping infinitely
XSS attack on hosted app	Vulnerability	An installed app has been attacked through a XSS vulnerability, allowing code injection

15.5. Loss of personal zone administration access

15.5.1. Intent

Part of a larger attack to misuse user credentials or perform identity theft, an attacker deletes or changes the password on the user's OpenID account.

15.5.2. Motivation

Security Goal	Value	Description
Availability	Medium	The attacker makes the personal zone administration interface unavailable to the user, resulting in an inability to control policy or authentication in the zone
Confidentiality	High	The attacker can view and misuse user personal zone data

15.5.3. Structure

Attack: Denial of Service	Origin: OWASP	Obstacle: PZH offline
Exploit: Unverified Password Change	Origin: CWE	Obstacle: Password recovery process attacked

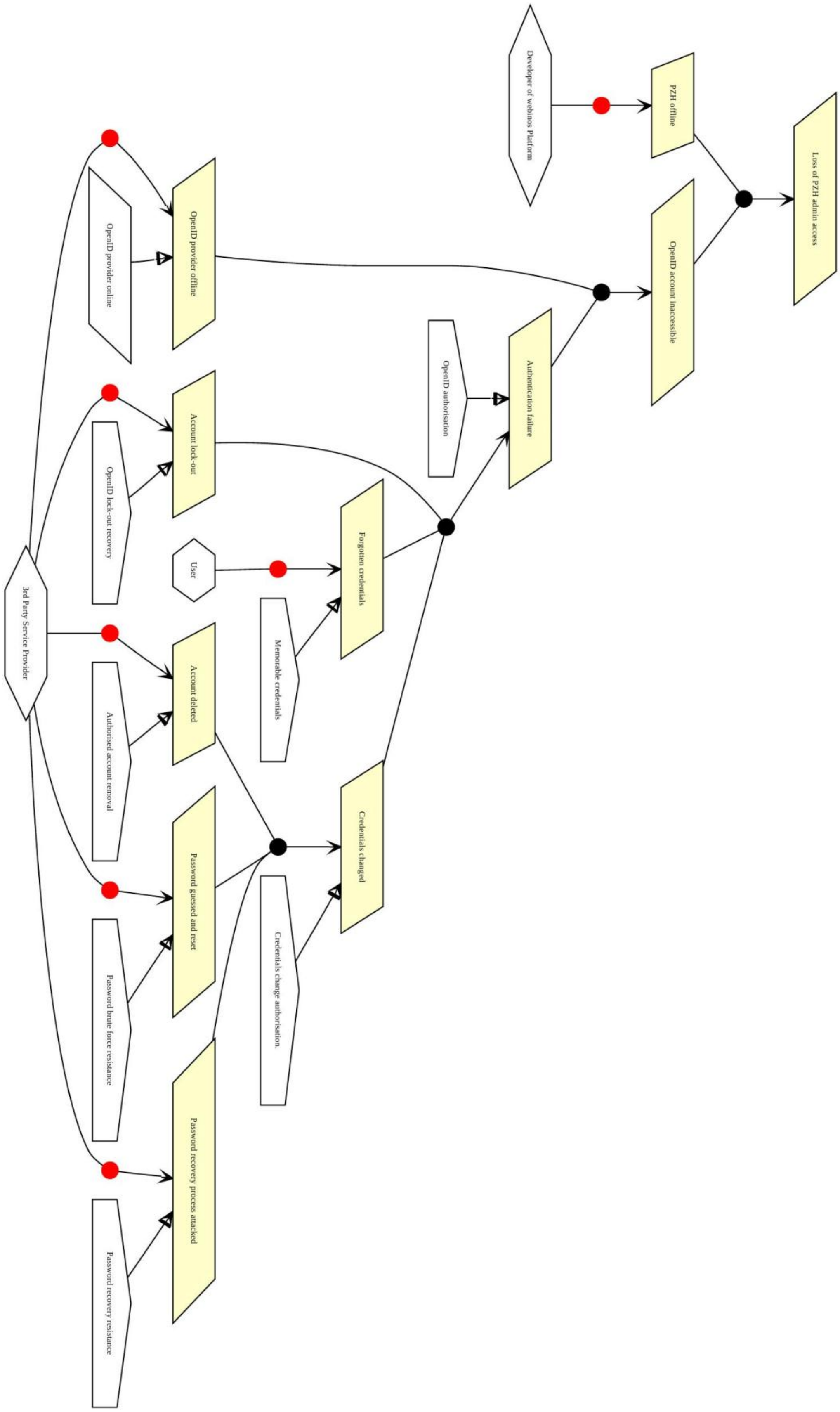
15.5.4. Participants

Attacker	Motives	Capabilities (Value)
Frankie	Thrill-seeking	Knowledge/Methods(Medium), Software(Medium)

15.5.5. Collaboration

Target	Personal Zone Administration Console
Exploit	Identity Provider

15.5.6. Implementation


Developer of webinos Platform
PZH offline
Loss of PZH admin access
OpenID account inaccessible
OpenID provider offline
OpenID provider online
Authentication failure
OpenID authorisation
Account lock-out
OpenID lock-out recovery
Forgotten credentials
User
Memorable credentials
3rd Party Service Provider
Account deleted
Authorised account removal
Password guessed and reset
Password brute force resistance
Credentials changed
Credentials change authorisation.
Password recovery process attacked
Password recovery resistance

Frankie is attempting to gain access to personal accounts on a range of popular websites, including email services and social networks. He intends to deface public records of some people (those he knows and has taken offense to) and maybe to expose a few people who he thinks are arrogant and deserve to be knocked down a peg or two. To do this, he is exploiting password recovery features and guessing commonly used "secret phrases" for openid accounts based on user names he knows, as well as some he has found on public forums. This resets user passwords and, as a consequence, means that the real users can't authenticate against the OpenID providers. Unfortunately for the victims, this also means that they have lost access to their webinos personal zone administration interface. Later on, Frankie realises that some of his victims are running webinos, and he makes use of his access to their accounts to (in some cases) delete files and settings, and in other cases secretly add his own device to their personal zones.

15.5.7. Obstacles

Obstacle	Category	Definition
Account deleted	Vulnerability	The OpenID account was deleted by an attacker impersonating the user
Account lock-out	Vulnerability	The OpenID account has been "locked out" due to too many failed authentication attempts
Authentication failure	Vulnerability	The OpenID authentication process refuses to authenticate the user
Credentials changed	Vulnerability	The OpenID account credentials have been changed by an attacker
Forgotten credentials	Vulnerability	The webinos user has forgotten their credentials
Loss of PZH admin access	Vulnerability	The PZH administration interface cannot be reached
OpenID account inaccessible	Vulnerability	The OpenID process does not successfully authenticate the correct user to the hub
OpenID provider offline	Vulnerability	The OpenID provider is unavailable
Password guessed and reset	Vulnerability	The OpenID account password was guessed by an attacker and then reset
Password recovery process attacked	Vulnerability	The OpenID account recovery procedure has been exploited to reset the password
PZH offline	Vulnerability	The PZH is not accessible over the internet

15.6. Exploit network bandwidth

15.6.1. Intent

Exploit webinos to gain access to network provider resources.

15.6.2. Motivation

Security Goal	Value	Description
Integrity	Medium	Ethan's ability to manipulate media preferences and other personal or log data should be considered a significant attack.

15.6.3. Structure

Attack: Repudiation Attack	Origin: OWASP	Obstacle: None
Exploit: Permissive convergence preferences	Origin: D2.8	Obstacle: Spoof network settings message

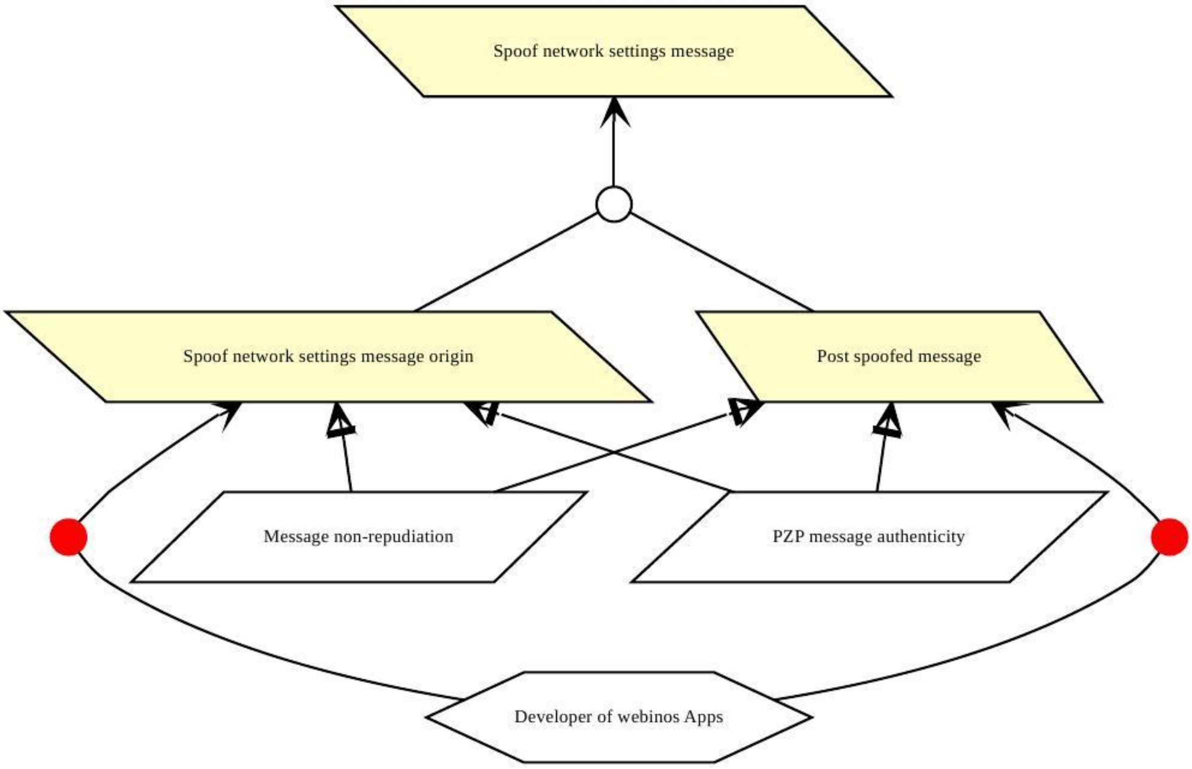
15.6.4. Participants

Attacker	Motives	Capabilities (Value)
Ethan	System resource theft	Knowledge/Methods(Medium), Software(Medium), Technology(Medium)

15.6.5. Collaboration

Target	Personal Media Preferences
Exploit	Personal Media Preferences

15.6.6. Implementation


Spoof network settings message
Spoof network settings message origin
Post spoofed message
Message non-repudiation
PZP message authenticity
Developer of webinos Apps

Ethan has discovered that, invariably, webinos users have over-permissive convergence settings between devices used in the home and, with by altering the right settings, it is possible to change network service settings. This is especially useful given that some network providers, as part of their business model, allow users to change their bandwidth settings online via webinos. This is a boon for Ethan's botnet business the convergence affordances allow him to easily increase the bandwidth available to his botnet clients.

A number of botnet client machines were running webinos, so he tried connecting to one of them and running a customised webinos web service client to discover any available webinos services. Sure enough, he discovered a text input service running on a mobile phone. Ethan was able to easily connect to this and manipulate the machine owner's "use maximum bandwidth" setting. If he was lucky, the machine owner's network provider would automatically increase this home's bandwidth without the owner ever finding out.

15.6.7. Obstacles

Obstacle	Category	Definition
Post spoofed message	Accountability Threat	Post spoofed message
Spoof network settings message	Accountability Threat	Spoof network settings message sent to network provider.

Spoof network settings message origin	Integrity Threat	Spoof network settings message origin.

15.7. Man In The webinos Browser

15.7.1. Intent

Ethan wants to steal application data, either to find credit card details, account details or to sell-on to advertisers or spammers

15.7.2. Motivation

Security Goal	Value	Description
Confidentiality	High	Ethan intercepts confidential application data

15.7.3. Structure

Attack: Man-in-the-browser attack	Origin: OWASP	Obstacle: Application data intercepted
Exploit: Inclusion of Functionality from Untrusted Control Sphere	Origin: CWE	Obstacle: Widget renderer supports extensions

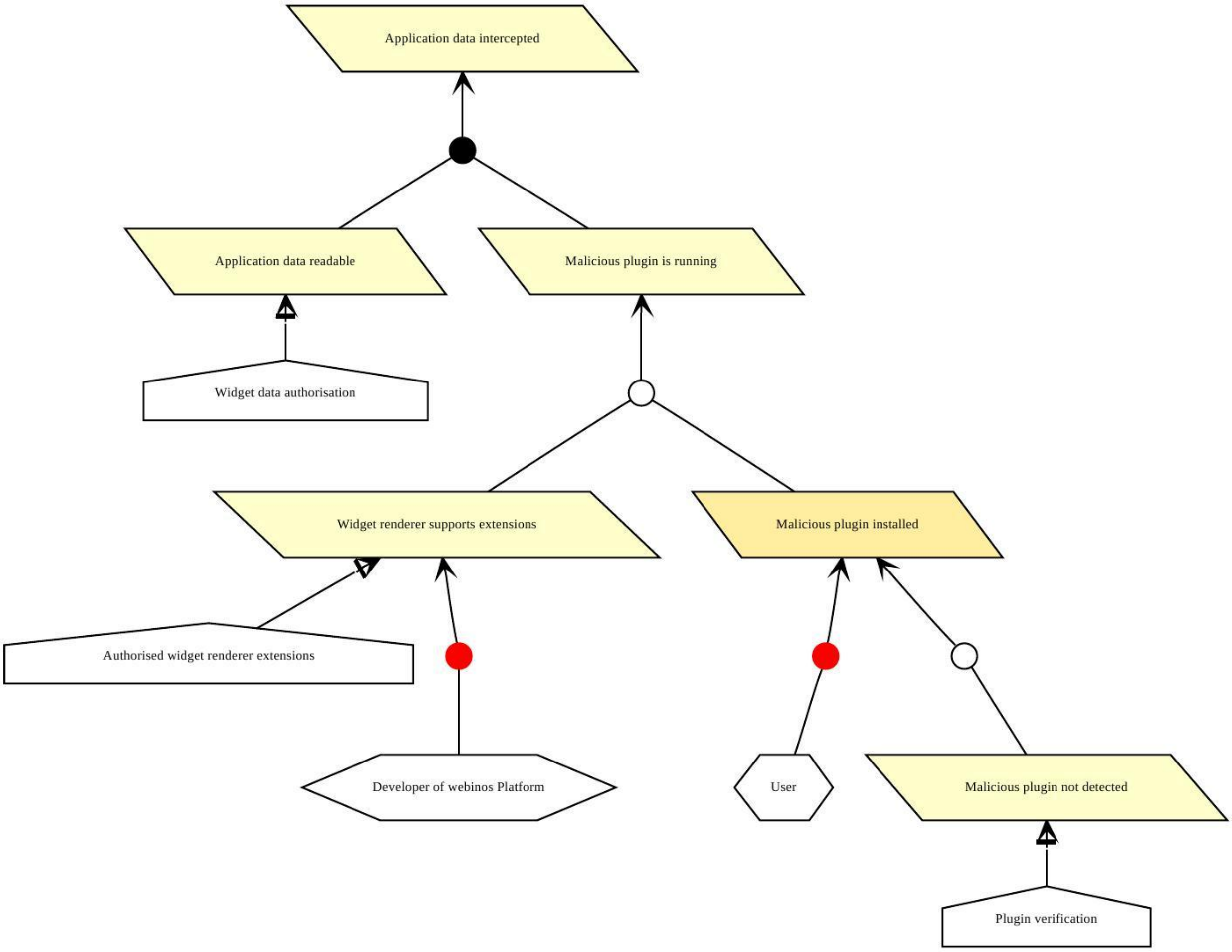
15.7.4. Participants

Attacker	Motives	Capabilities (Value)
Ethan	System resource theft	Knowledge/Methods(Medium), Software(Medium), Technology(Medium)

15.7.5. Collaboration

Target	Application Data
Exploit	Rendering Engine

15.7.6. Implementation


Application data intercepted
Application data readable
Malicious plugin is running
Widget data authorisation
Widget renderer supports extensions
Malicious plugin installed
Authorised widget renderer extensions
Developer of webinos Platform
User
Malicious plugin not detected
Plugin verification

Ethan develops a browser plugin which allows him to intercept all messages between the webinos widget renderer and the PZP, giving him access to application data.

15.7.7. Obstacles

Obstacle	Category	Definition
Application data intercepted	Vulnerability	Application data from webinos widgets is intercepted in the widget renderer
Application data readable	Vulnerability	Application data from webinos widgets is readable outside of the widget renderer
Malicious plugin installed	Vulnerability	The user installs (or suffers from a driveby download) a malicious plugin
Malicious plugin is running	Vulnerability	Malware in the form of a malicious widget renderer plugin is running

Malicious plugin not detected	Vulnerability	Malware plugin not detected.
Widget renderer supports extensions	Vulnerability	The widget renderer supports plugins or extensions

15.8. Overlay network facilitated relay attack

15.8.1. Intent

Exploit webinos overlay network to commit fraud.

15.8.2. Motivation

Security Goal	Value	Description
Accountability	High	David relies on seamless convergence and a lack of traceability to carry out his attack.
Confidentiality	Medium	David is interested in obtaining confidential information available to the webinos runtime.

15.8.3. Structure

Attack: NFC replay attack	Origin: D2.8	Obstacle: Malicious personal zone enrolment
Exploit: System data trust	Origin: D2.8	Obstacle: Spoof message origin

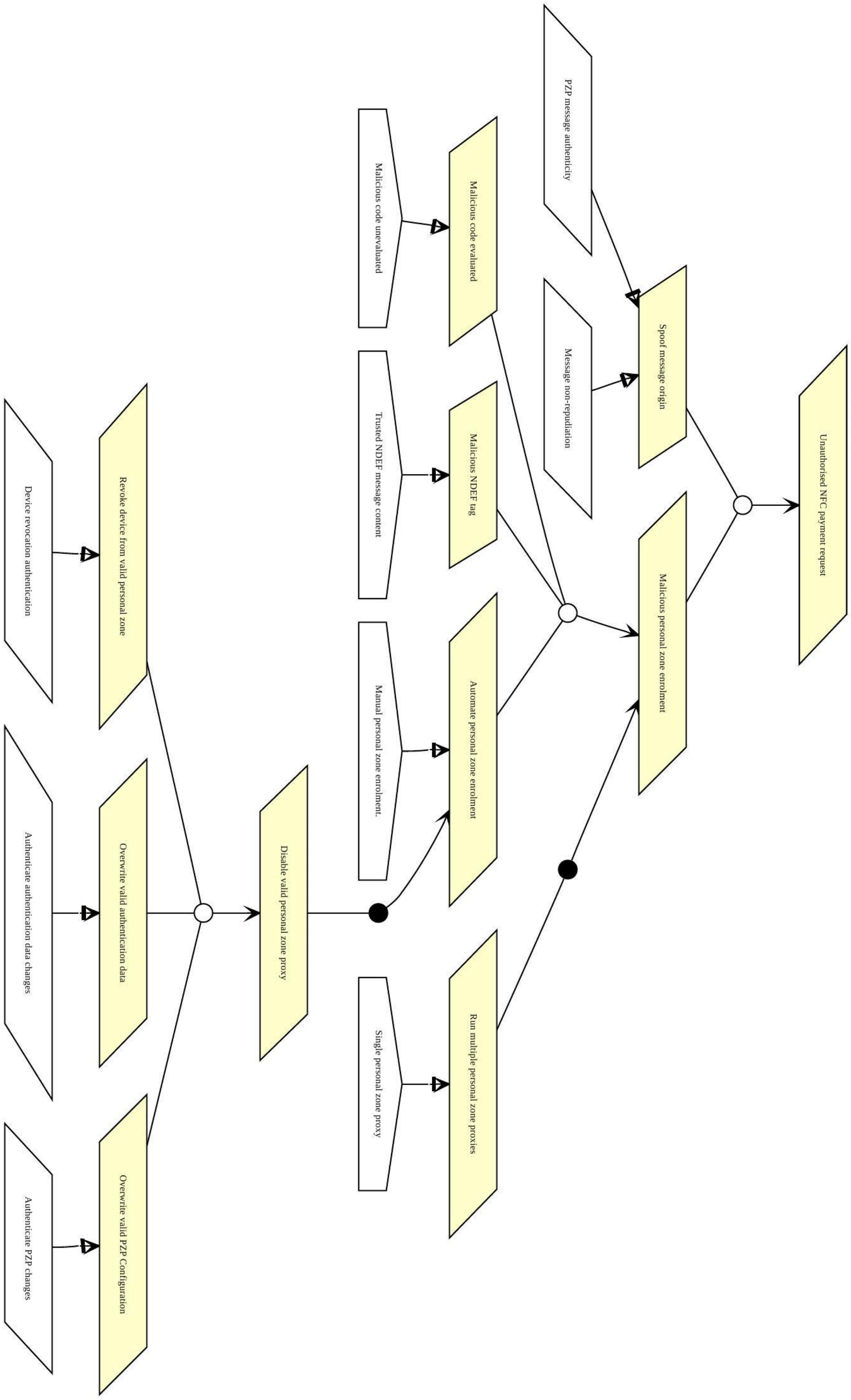
15.8.4. Participants

Attacker	Motives	Capabilities (Value)
David	Data theft	Resources/Personnel and Time(Medium)

15.8.5. Collaboration

Target	Device API, Privacy Preferences
Exploit	Analytics Data, Device API

15.8.6. Implementation


Unauthorised NFC payment request
Spoof message origin
Malicious personal zone enrolment
PZP message authenticity
Message non-repudiation
Malicious code evaluated
Malicious NDEF tag
Automate personal zone enrolment
Run multiple personal zone proxies
Malicious code unevaluated
Trusted NDEF message content
Manual personal zone enrolment.
Single personal zone proxy
Disable valid personal zone proxy
Revoke device from valid personal zone
Overwrite valid authentication data
Overwrite valid PZP Configuration
Device revocation authentication
Authenticate authentication data changes
Authenticate PZP changes

As Alice was walking away from the bar with her drinks, she narrowly avoided dropping one of her glasses as man brushed past her. "Sorry", he mumbled as he walked briskly towards the toilets.

David devised a new scam which took advantage of the recent take-off of NFC based mobile payment, and the new webinos platform that could link different apps and devices together. The scam involved dropping a leech mobile phone into someone's bag which contained a webinos enabled NFC phone. The leech would attempt to get the victim's phone to join his personal zone which, David estimated, would not be too difficult. Once in place, David could then purchase things via NFC, while webinos routed the request to the victim's NFC reader via the personal zone overlay network. David tinkered around with different devices, settings, and applications, and was quite surprised at how easy this scam appeared to be. Consequently, he would need to make the most use of this exploit quickly before other people got in on the act.

David's plan was to find somewhere where there would be lots of young but unsuspecting people who might not notice something being put into their bags. David decided a nightclub on a Friday night would be a good bet. As David entered the club at shortly after midnight, he noticed a large group of people mingling near the bar. As he approached, he noticed a large women trying to carry multiple drinks with a hand-bag slung around her shoulder. Confidently, with his leech device in hand, David walked towards this woman.

15.8.7. Obstacles

Obstacle	Category	Definition
Automate personal zone enrolment	Accountability Threat	A device is enroled into a personal zone without the use of an out-of-band channel.
Disable valid personal zone proxy	Accountability Threat	A valid device personal zone proxy is disabled
Malicious code evaluated	Vulnerability	Event handler evaluates malicious JSON content.
Malicious NDEF tag	Integrity Threat	An NFC tag contains NDEF message encapsulating malicious code.
Malicious personal zone enrolment	Accountability Threat	A device is enroled into a personal zone without the knowledge of the device owner.
Overwrite valid authentication data	Integrity Threat	Overwrite valid authentication data.
Overwrite valid PZP Configuration	Integrity Threat	Overwrite valid PZP Configuration file.
Revoke device from valid personal zone	Accountability Threat	Revoke device from valid personal zone

Run multiple personal zone proxies	Accountability Threat	Valid and malicious personal zone proxies run on a single device.
Spoof message origin	Integrity Threat	Messages sent from a source leech device to a destination device have the leeched device as an origin.
Unauthorised NFC payment request	Accountability Threat	A request is received for an NFC payment from an unauthorised device.

15.9. Policy evasion through Browser APIs

15.9.1. Intent

Jimmy creates a webinos application which collects location data for a social application. Jimmy believes the app is much better if location tracking is enabled and so uses every source of location data available.

15.9.2. Motivation

Security Goal	Value	Description
Unobservability	Low	Jimmy can observe user location

15.9.3. Structure

Attack: Accessing Functionality Not Properly Constrained by ACLs	Origin: CAPEC	Obstacle: Browser API authorised
Exploit: Missing Authorization	Origin: CWE	Obstacle: None

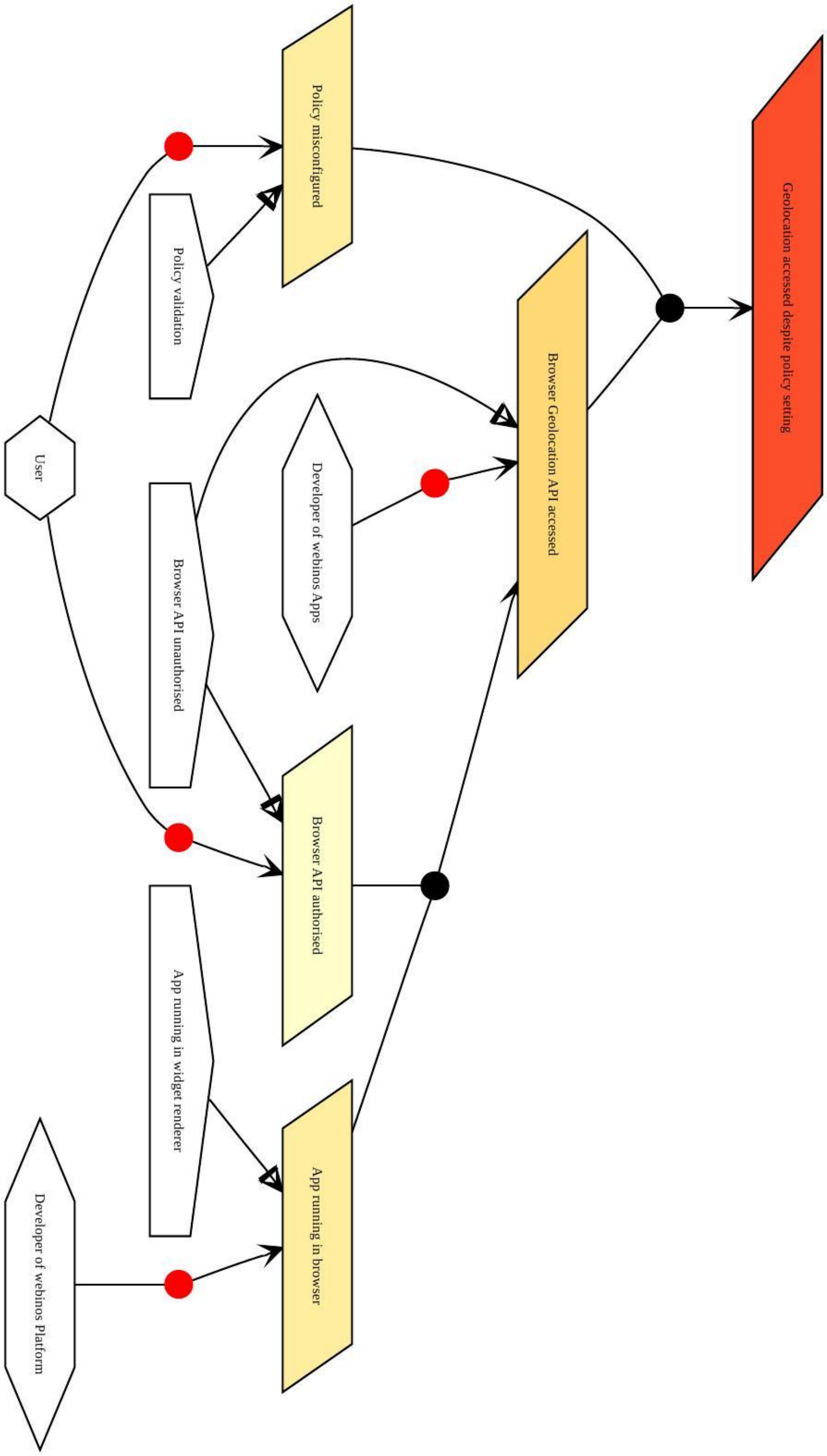
15.9.4. Participants

Attacker	Motives	Capabilities (Value)
Jimmy	Money	Knowledge/Methods(Medium), Software(Medium)

15.9.5. Collaboration

Target	Personal Data
Exploit	Browser

15.9.6. Implementation


Geolocation accessed despite policy setting
Policy misconfigured
Browser Geolocation API accessed
Policy validation
User
Browser API unauthorised
Developer of webinos Apps
Browser API authorised
App running in widget renderer
App running in browser
Developer of webinos Platform

Jimmy creates a legitimate and generally non-malicious web application which the end user installs. However, Jimmy designs it so that it accesses both webinos APIs and browser APIs, depending on what is available at the time. When the user turns off access to geolocation using webinos, Jimmy's app automatically invokes the browser API instead which has already been authorised.

15.9.7. Obstacles

Obstacle	Category	Definition
App running in browser	Vulnerability	The application is running in a browser
Browser API authorised	Vulnerability	
Browser Geolocation API accessed	Vulnerability	The application can access the browser-supplied geolocation API
Geolocation accessed despite policy setting	Vulnerability	An application is able to access user Geolocation details despite policy settings explicitly specifying that this shouldn't be possible.
Policy misconfigured	Vulnerability	The user believes they have disabled geolocation when, in fact, they haven't

15.10.PZH Pharming

15.10.1. Intent

Ethan wants to steal user account details for personal zones in order to create a botnet of webinos devices

15.10.2. Motivation

Security Goal	Value	Description
Confidentiality	High	Ethan intercepts confidential user credentials

15.10.3. Structure

Attack: Pharming	Origin: CAPEC	Obstacle: PZH Admin page spoofed by attacker
Exploit: UI Misrepresentation of Critical Information	Origin: CWE	Obstacle: User does not check PZH admin URL

15.10.4. Participants

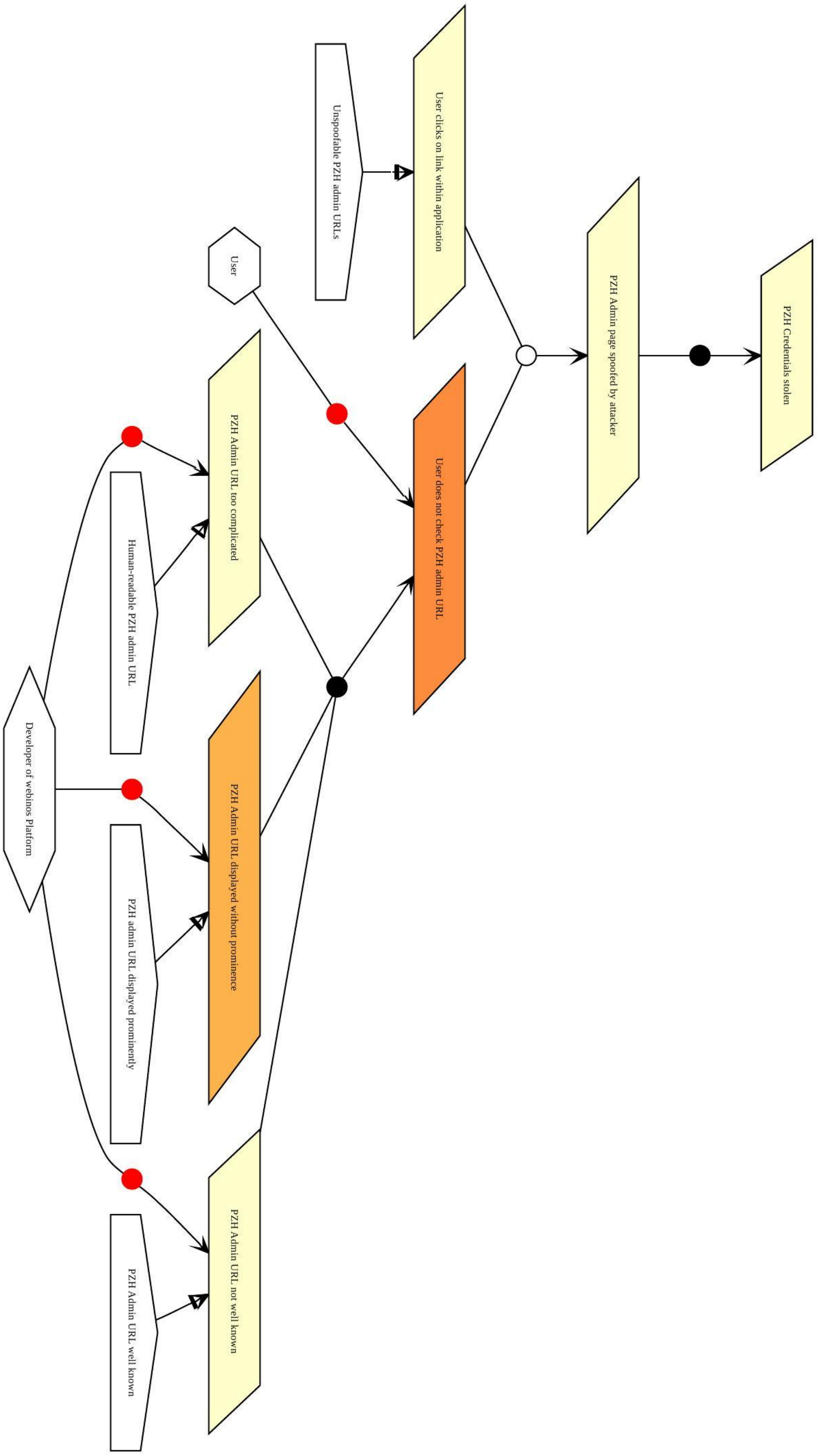
Attacker	Motives	Capabilities (Value)

Ethan	System resource theft	Knowledge/Methods(Medium), Software(Medium), Technology(Medium)

15.10.5. Collaboration

Target	Identity Provider
Exploit	Personal Zone Administration Console

15.10.6. **Implementation**


Unspoofable PZH admin URLs
User clicks on link within application
User
PZH Admin page spoofed by attacker
PZH Credentials stolen
PZH Admin URL too complicated
Human-readable PZH admin URL
User does not check PZH admin URL.
Developer of webinos Platform
PZH Admin URL displayed without prominence
PZH admin URL displayed prominently
PZH Admin URL not well known
PZH Admin URL well known

Ethan creates a web application which offers a link to the user's PZH administration console. This link actually directs them to a spoofed version of their page, featuring a MITM attack on the credentials they would enter into their identity provider's webpage.

15.10.7. Obstacles

Obstacle	Category	Definition
PZH Admin page spoofed by attacker	Vulnerability	The user's PZH admin page is impersonated by another webpage
PZH Admin URL displayed without prominence	Vulnerability	The PZH admin URL bar is not visible or easy to miss
PZH Admin URL not well known	Vulnerability	The user does not know the correct URL
PZH Admin URL too complicated	Vulnerability	The PZH admin URL is complicated and easy to misread
PZH Credentials stolen	Vulnerability	The user types their PZH administration page credentials into an attacker's webpage
User clicks on link within application	Vulnerability	A malicious application offers a link to the user's PZH admin page, which is in fact directed to a malicious webpage hosted by the attacker.
User does not check PZH admin URL	Vulnerability	The user does not check that the PZH admin page URL is what it should be

15.11.Steal In-Car Data

15.11.1. Intent

Use an open webinos session for unauthorised transfer of data between devices in different personal zones.

15.11.2. Motivation

Security Goal	Value	Description
Accountability	High	The attacker can access user active session of the service. If it is a service with a fee, the attack can spend user money.
Confidentiality	Medium	The attacker can access user session data.

15.11.3. Structure

Attack: Session hijacking	Origin: D2.8	Obstacle: None
Exploit: Automatic login	Origin: D2.8	Obstacle: Malicious personal zone enrolment

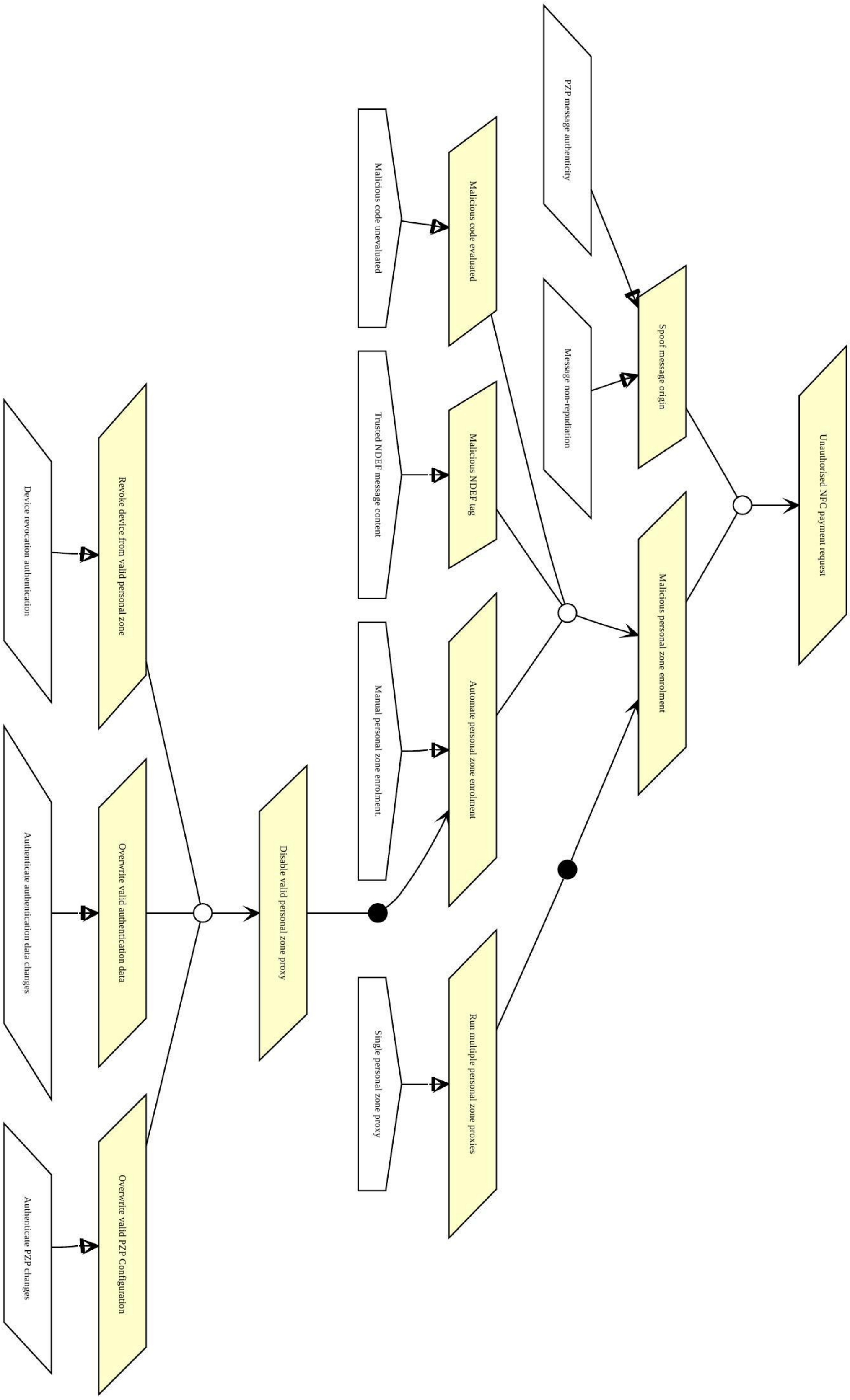
15.11.4. Participants

Attacker	Motives	Capabilities (Value)
David	Data theft	Resources/Personnel and Time(Medium)

15.11.5. Collaboration

Target	Personal Data
Exploit	Browser

15.11.6. Implementation


Unauthorised NFC payment request
Spoof message origin
Malicious personal zone enrolment
PZP message authenticity
Message non-repudiation
Malicious code evaluated
Malicious NDEF tag
Automate personal zone enrolment
Run multiple personal zone proxies
Malicious code unevaluated
Trusted NDEF message content
Manual personal zone enrolment.
Single personal zone proxy
Disable valid personal zone proxy
Revoke device from valid personal zone
Overwrite valid authentication data
Overwrite valid PZP Configuration
Device revocation authentication
Authenticate authentication data changes
Authenticate PZP changes

Justin gives David, the mechanic, his car keys so David can carry out his scheduled MOT.
David at the moment has not other customers, so he decides to fiddle with the in-car system.
The computer requires no authentication so he can enter and nose about navigation history. He realizes that the web browser is logged in a music service with fee. He listens to the music waiting for the next motorist coming.
After listening few songs, he uses his own webinos enabled smartphone to get the Justin's personal zone device list via Justin's PZH. From the list he selects the in-car system. When on the in-car system touch screen appears the confirmation dialog box, he taps on OK and the devices are now connected. David can exit from the car and access Justin's music service form his own smartphone.
When Justin comes back to the parking, David disconnects the smartphone form the music service and from Justin's personal zone and returns the car to its owner.

15.11.7. Obstacles

Obstacle	Category	Definition
Automate personal zone enrolment	Accountability Threat	A device is enroled into a personal zone without the use of an out-of-band channel.
Disable valid personal zone proxy	Accountability Threat	A valid device personal zone proxy is disabled
Malicious code evaluated	Vulnerability	Event handler evaluates malicious JSON content.
Malicious NDEF tag	Integrity Threat	An NFC tag contains NDEF message encapsulating malicious code.
Malicious personal zone enrolment	Accountability Threat	A device is enroled into a personal zone without the knowledge of the device owner.
Overwrite valid authentication data	Integrity Threat	Overwrite valid authentication data.
Overwrite valid PZP Configuration	Integrity Threat	Overwrite valid PZP Configuration file.
Revoke device from valid personal zone	Accountability Threat	Revoke device from valid personal zone
Run multiple personal zone proxies	Accountability Threat	Valid and malicious personal zone proxies run on a single device.
Spoof message origin	Integrity Threat	Messages sent from a source leech device to a destination device have the leeched device as an origin.
Unauthorised NFC	Accountability	A request is received for an NFC payment from an

payment request	Threat	unauthorised device.

15.12.Steal webinos Session

15.12.1. Intent

Exploit open sessions to steal personal data for financial gain

15.12.2. Motivation

Security Goal	Value	Description
Anonymity	Low	The malicious request associated with the CSRF may involve harvesting small amounts of data which, collectively, could lead to the disclosure of a subject's identity.
Confidentiality	Medium	The attack allows the malicious code access to session data, some of which may be long-lived.

15.12.3. Structure

Attack: Cross-Site Request Forgery	Origin: OWASP	Obstacle: None
Exploit: Session Fixation	Origin: OWASP	Obstacle: Malicious code evaluated

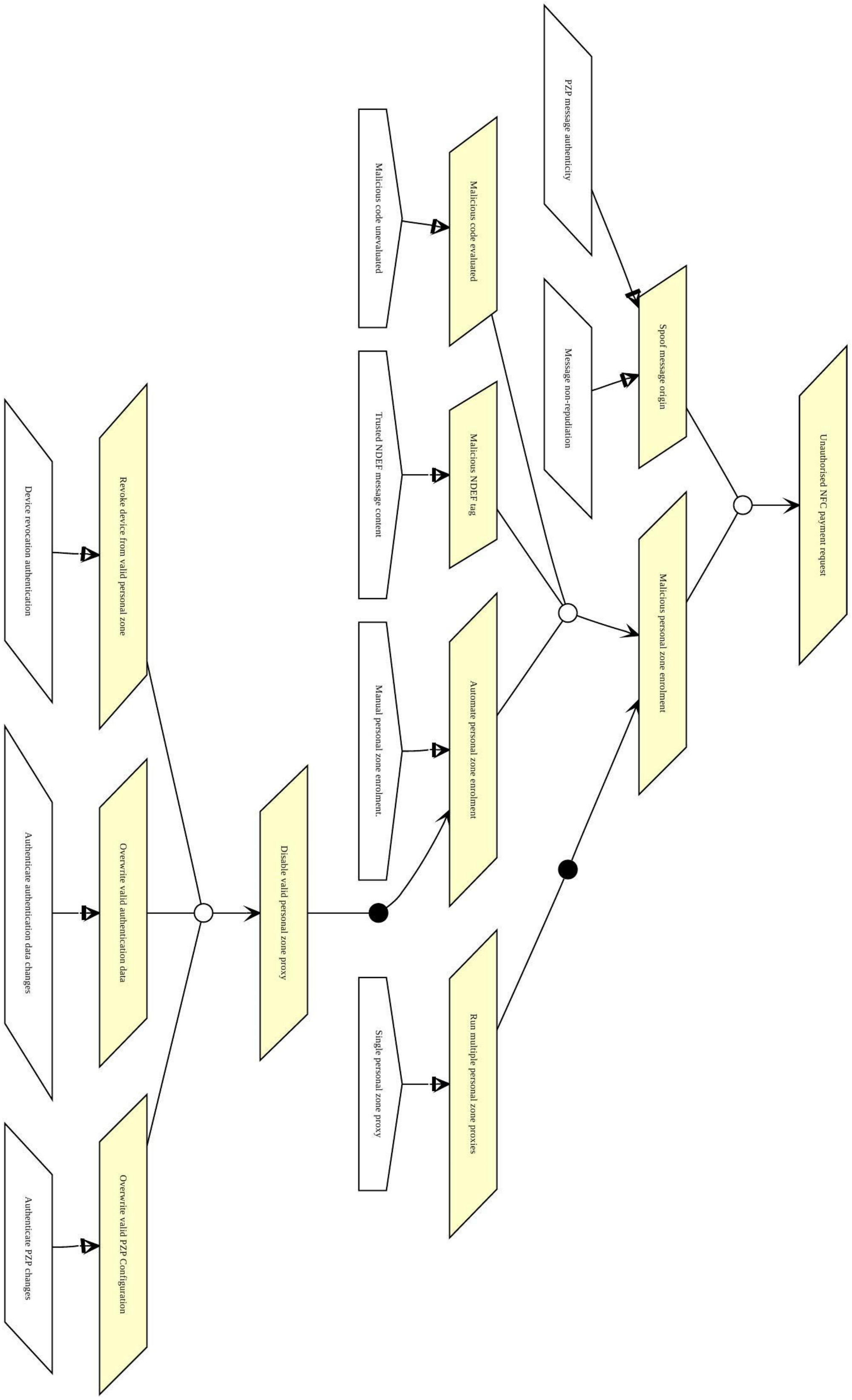
15.12.4. Participants

Attacker	Motives	Capabilities (Value)
Ethan	System resource theft	Knowledge/Methods(Medium), Software(Medium), Technology(Medium)

15.12.5. Collaboration

Target	Personal Data
Exploit	Session

15.12.6. **Implementation**


PZP message authenticity
Malicious code unevaluated
Malicious code evaluated
Spoof message origin
Message non-repudiation
Trusted NDEF message content
Malicious NDEF tag
Unauthorised NFC payment request
Malicious personal zone enrolment
Device revocation authentication
Revoke device from valid personal zone
Manual personal zone enrolment.
Automate personal zone enrolment
Authenticate authentication data changes
Overwrite valid authentication data
Disable valid personal zone proxy
Single personal zone proxy
Run multiple personal zone proxies
Authenticate PZP changes
Overwrite valid PZP Configuration

As a new source of income, Ethan has opened an account with an affiliate stock photo service. Ethan carries out some rudimentary google hacking to identify forums that might be open to stored XSS vulnerabilities. Eventually, Ethan finds a forum that is used by a Webinos enabled camera app that synchronises images with a cloud based photo sharing service.

Ethan develops an XSS/CSRF exploit by loading malicious code into the application's blog page. On viewing the blog, the script obtains the Webinos session data and, via some local and external javascript and php, sends the session data to another site which uses the Webinos APIs, close a device's session, re-opens the session to the device, obtains the images, and submits these to the affiliate stock photo service using the service providers web services API. Ethan realises the exploit might only work on certain devices, but he thinks that something is better than nothing...

Several days later, Justin is taking some snaps of some his friends at a bar using a recently downloaded camera application that syncs his images to the cloud based provider. On his way home, Justin browses the community developed help data in the application. Justin doesn't realise that, by checking this help data, his photos are about to net Ethan some additional income.

15.12.7. Obstacles

Obstacle	Category	Definition
Automate personal zone enrolment	Accountability Threat	A device is enroled into a personal zone without the use of an out-of-band channel.
Disable valid personal zone proxy	Accountability Threat	A valid device personal zone proxy is disabled
Malicious code evaluated	Vulnerability	Event handler evaluates malicious JSON content.
Malicious NDEF tag	Integrity Threat	An NFC tag contains NDEF message encapsulating malicious code.
Malicious personal zone enrolment	Accountability Threat	A device is enroled into a personal zone without the knowledge of the device owner.
Overwrite valid authentication data	Integrity Threat	Overwrite valid authentication data.
Overwrite valid PZP Configuration	Integrity Threat	Overwrite valid PZP Configuration file.
Revoke device from valid personal zone	Accountability Threat	Revoke device from valid personal zone
Run multiple personal zone proxies	Accountability Threat	Valid and malicious personal zone proxies run on a single device.

Spoof message origin	Integrity Threat	Messages sent from a source leech device to a destination device have the leeched device as an origin.
Unauthorised NFC payment request	Accountability Threat	A request is received for an NFC payment from an unauthorised device.

15.13.Test footprinting

15.13.1. Intent

15.13.2. Motivation

Security Goal	Value	Description
Accountability	High	Ethan looks for test code which provides unauthorised resource access

15.13.3. Structure

Attack: Locate and Exploit Test APIs	Origin: CAPEC	Obstacle: Test API enabled
Exploit: Allocation of Resources without Limits or Throttling	Origin: CWE	Obstacle: Unrestricted request specification

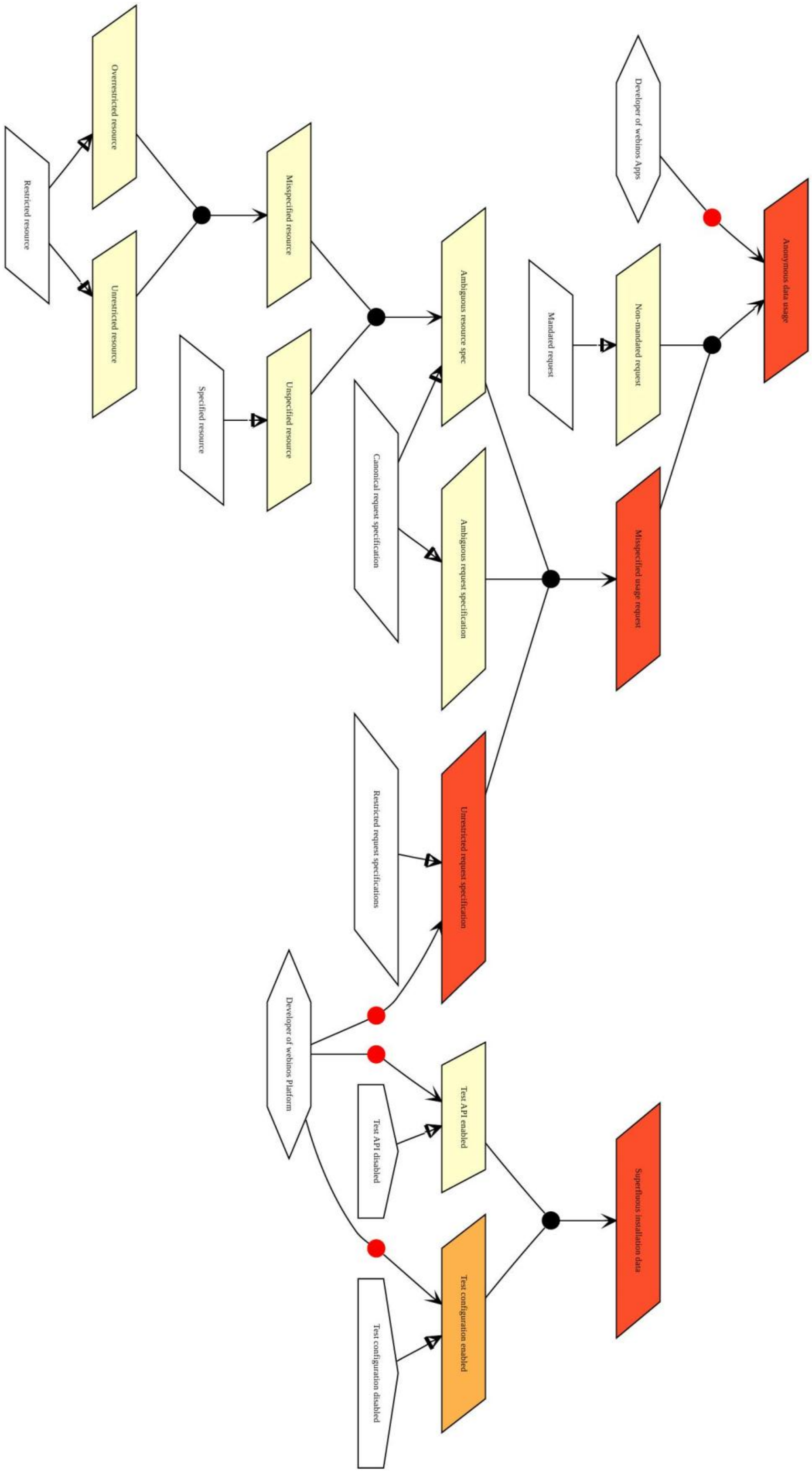
15.13.4. Participants

Attacker	Motives	Capabilities (Value)
Ethan	System resource theft	Knowledge/Methods(Medium), Software(Medium), Technology(Medium)

15.13.5. Collaboration

Target	Application Data
Exploit	Access Request

15.13.6. Implementation


Restricted resource
Overrestricted resource
Unrestricted resource
Misspecified resource
Specified resource
Unspecified resource
Ambiguous resource spec
Canonical request specification
Ambiguous request specification
Mandated request
Non-mandated request
Developer of webinos Apps
Anonymous data usage
Misspecified usage request
Restricted request specifications
Unrestricted request specification
Developer of webinos Platform
Test API enabled
Test API disabled
Test configuration enabled
Test configuration disabled
Superfluous installation data

Ethan finds apps with superflous test code which allows him to exploit test data in runtime.

15.13.7. Obstacles

Obstacle	Category	Definition
Ambiguous request specification	Vulnerability	Request specification is ambiguous.
Ambiguous resource spec	Vulnerability	Resource specification is ambiguous
Anonymous data usage	Vulnerability	Users do not specify how they will use data they are requesting.
Misspecified resource	Vulnerability	Resource is misspecified.
Misspecified usage request	Vulnerability	Usage request is misspecified.
Non-mandated request	Vulnerability	Request is non-mandated.
Overrestricted resource	Vulnerability	Resource is overly restricted compared to its specification.
Superfluous installation data	Vulnerability	webinos application installation contains superfluous data and code.
Test API enabled	Vulnerability	Test API is enabled in operating environment.
Test configuration enabled	Vulnerability	Test and sample configuration data is enabled in the operating environment.
Unrestricted request specification	Vulnerability	User agent grants access to all network resources.
Unrestricted resource	Vulnerability	Resource contains no restrictions.
Unspecified resource	Vulnerability	Resource is unspecified.

15.14.Exploiting transitive permissions

15.14.1. Intent

Frankie wants to spy on people through a web application, as he thinks it might be fun and might be able to catch people in embarassing situations

15.14.2. Motivation

Security Goal	Value	Description

Confidentiality	Medium	Frankie can access personal data normally protected by webinos policies

15.14.3. Structure

Attack: Data Interception Attacks	Origin: CAPEC	Obstacle: Installed App exposes communication interface
Exploit: Improper Preservation of Permissions	Origin: CWE	Obstacle: Installed App given API permissions

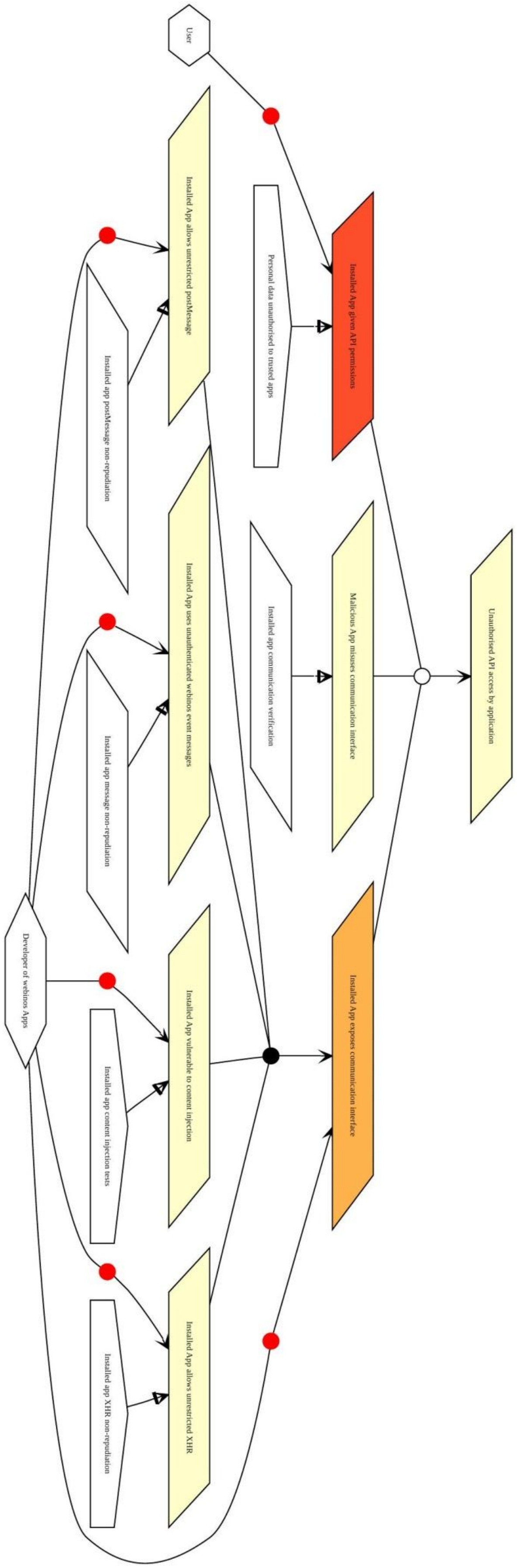
15.14.4. Participants

Attacker	Motives	Capabilities (Value)
Frankie	Thrill-seeking	Knowledge/Methods(Medium), Software(Medium)

15.14.5. Collaboration

Target	Personal Data
Exploit	Device API

15.14.6. Implementation


User
Installed App allows unrestricted postMessage
Installed app postMessage non-repudiation
Personal data unauthorised to trusted apps
Installed App given API permissions
Installed App uses unauthenticated webinos event messages
Installed app message non-repudiation
Installed app communication verification
Malicious App misuses communication interface
Unauthorised API access by application
Developer of webinos Apps
Installed App vulnerable to content injection
Installed app content injection tests
Installed App exposes communication interface
Installed App allows unrestricted XHR
Installed app XHR non-repudiation

Frankie creates an application which appears to be legitimate and only requests minimal permissions. Even if untrusted, many users are likely to install it and trust the webinos security framework to protect it once it has been uploaded to an app store. However, the application makes use of an unprotected communication interface with another application (which has access to many APIs) to read user data. Frankie makes sure that his app uploads this data to is servers...

15.14.7. Obstacles

Obstacle	Category	Definition
Installed App allows unrestricted postMessage	Vulnerability	The trusted application is part hosted and allows arbitrary postMessages from other origins
Installed App allows unrestricted XHR	Vulnerability	The trusted application is part hosted and allows arbitrary XHR from other origins
Installed App exposes communication interface	Vulnerability	A trusted app exposes communication interface to other applications
Installed App given API permissions	Vulnerability	A trusted app is installed and given permission to access several APIs with access to personal data
Installed App uses unauthenticated webinos event messages	Vulnerability	The trusted application will receive and process event messages from other apps without verifying the authenticity of the other applications
Installed App vulnerable to content injection	Vulnerability	The trusted application is part hosted and vulnerable to a content injection attack, allowing another site to abuse the permissions of that site
Malicious App misuses communication interface	Vulnerability	A malicious application is able to communicate arbitrarily with a trusted application
Unauthorised API access by application	Vulnerability	A webinos application is able to access an API despite restrictions imposed by the policy system through calling another, trusted application

15.15.Linkability through findServices

15.15.1. Intent

Jimmy wants to be able to re-identify users of different applications in order to sell data to analytics and marketing companies who run advertising networks. If he can collect data about users on applications this can be used to better target adverts.

15.15.2. Motivation

Security Goal	Value	Description
Unlinkability	Low	Jimmy can connect actions of users of different applications

15.15.3. Structure

Attack: Cross Site Identification	Origin: CAPEC	Obstacle: None
Exploit: Information Exposure Through Persistent Cookies	Origin: CWE	Obstacle: Apps share usage data

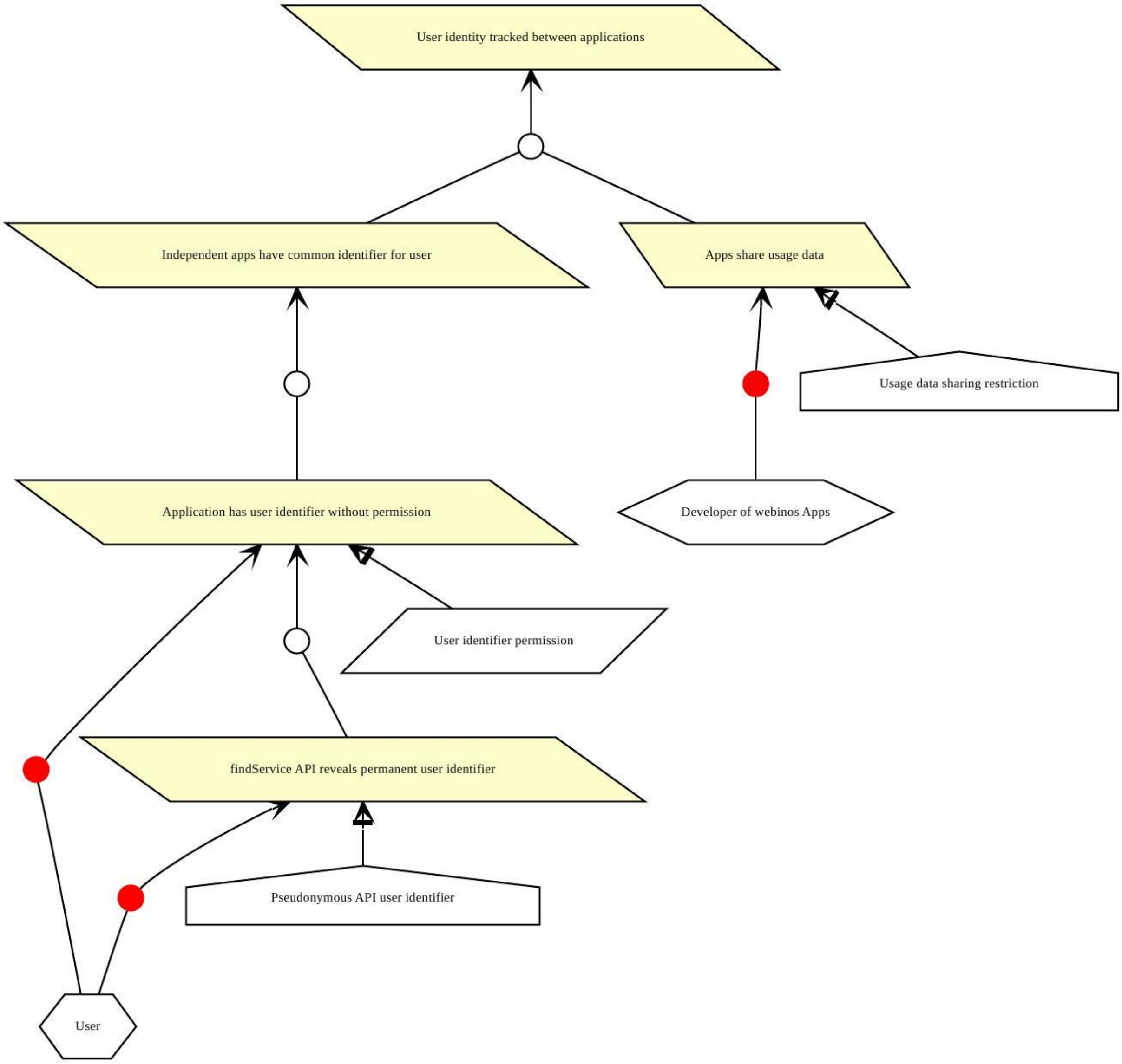
15.15.4. Participants

Attacker	Motives	Capabilities (Value)
Jimmy	Money	Knowledge/Methods(Medium), Software(Medium)

15.15.5. Collaboration

Target	Personal Data
Exploit	Device API

15.15.6. Implementation


User identity tracked between applications
Independent apps have common identifier for user
Apps share usage data
Usage data sharing restriction
Application has user identifier without permission
Developer of webinos Apps
User identifier permission
findService API reveals permanent user identifier
Pseudonymous API user identifier
User

Jimmy creates a legitimate application which uses various unimportant APIs, inluding the discovery API. Through the discovery API he is able to retrieve identifiers for webinos services hosted in the user's personal zone. These are persistent. Jimmy combines these identifies with records about the user's activity on the application and sells them to an online analytics firm who combines this information with other instances of these service identifiers.

15.15.7. Obstacles

Obstacle	Category	Definition
Application has user identifier without permission	Vulnerability	One application is installed and is able to obtain a user identifier without requesting access to an API for this.

Apps share usage data	Vulnerability	The two application share usage data, either directly or through an intermediary analytics service
findService API reveals permanent user identifier	Vulnerability	The output of the findServices API is an effective user identifier.
Independent apps have common identifier for user	Vulnerability	Two or more independent applications have been installed in the personal zone and have the same user identifier.
User identity tracked between applications	Vulnerability	An application is able to successfully re-identify a user of a different application despite them restricting access to personally-identifying APIs.

15.16.Identity theft with webinos messaging

15.16.1. Intent

Ethan attempts to steal usernames and passwords of webinos users in order to then phish their friends by sending them scam emails.

15.16.2. Motivation

Security Goal	Value	Description
Confidentiality	Medium	The loss of email credentials

15.16.3. Structure

Attack: Mobile Phishing (aka MobPhishing)	Origin: CAPEC	Obstacle: SMS intercepted and relayed
Exploit: Insufficiently Protected Credentials	Origin: CWE	Obstacle: None

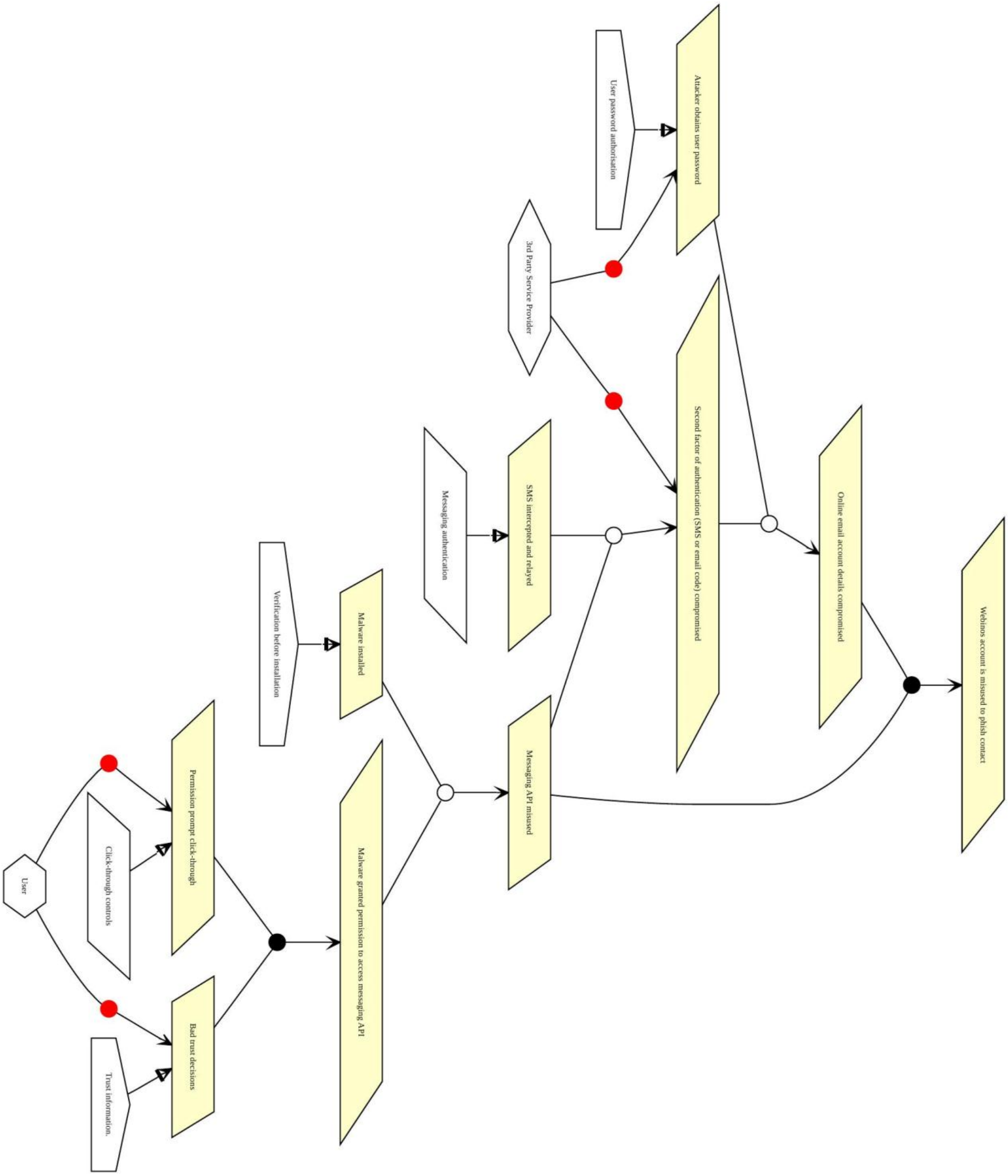
15.16.4. Participants

Attacker	Motives	Capabilities (Value)
Ethan	System resource theft	Knowledge/Methods(Medium), Software(Medium), Technology(Medium)

15.16.5. Collaboration

Target	Application Data
Exploit	Access Request

15.16.6. Implementation


User
Click-through controls
Trust information.
Permission prompt click-through
Bad trust decisions
Malware granted permission to access messaging API
Verification before installation
Malware installed
Messaging authentication
SMS intercepted and relayed
Messaging API misused
3rd Party Service Provider
User password authorisation
Attacker obtains user password
Second factor of authentication (SMS or email code) compromised
Online email account details compromised
Webinos account is misused to phish contact

Ethan obtains user account details and uses malware to defeat 2-factor authentication

15.16.7. Obstacles

Obstacle	Category	Definition
Attacker obtains user	Vulnerability	The attacker gains the user's login password. This may

password		be through the password recovery system, compromise of a re-used password on another site, or otherwise.
Bad trust decisions	Vulnerability	The user makes a bad decision to trust a piece of malware. This could be because they have insufficent information available to make a better decision.
Malware granted permission to access messaging API	Vulnerability	The user grants a piece of malware inappropriately high permissions
Malware installed	Vulnerability	A piece of malicious software is installed by the end user
Messaging API misused	Vulnerability	Malware is given access to the messaging API which is then used to send messages and receive presumed-private messages
Online email account details compromised	Vulnerability	The attacker gains knowledge of the user's account details which are sufficient for him to log-in and impersonate them
Permission prompt click-through	Vulnerability	Users click 'ok' to all permission prompts and therefore do not realise they are granting inappropriate permission
Second factor of authentication (SMS or email code) compromised	Vulnerability	The attacker gains credentials provided by the second authentication factor (an SMS containing a short token, for example)
SMS intercepted and relayed	Vulnerability	SMS messages are intercepted by malware with a subscription to the messaging API. The authentication token is then forwarded to the attacker
Webinos account is misused to phish contact	Vulnerability	An attacker uses their ability to log in to the user's email account to impersonate them and send phishing emails to their contacts.

15.17.Request footprinting

15.17.1. Intent

Glean an understanding of what resources are available on a device by eavesdropping on requests.

15.17.2. Motivation

Security Goal	Value	Description
Confidentiality	Low	Ethan wants to get a better understanding of what resources are under policy control.

15.17.3. Structure

Attack: Network Eavesdropping	Origin: OWASP	Obstacle: Eavesdrop request enforcement channel
Exploit: Missing XML Validation	Origin: OASIS	Obstacle: Missing XML document validation

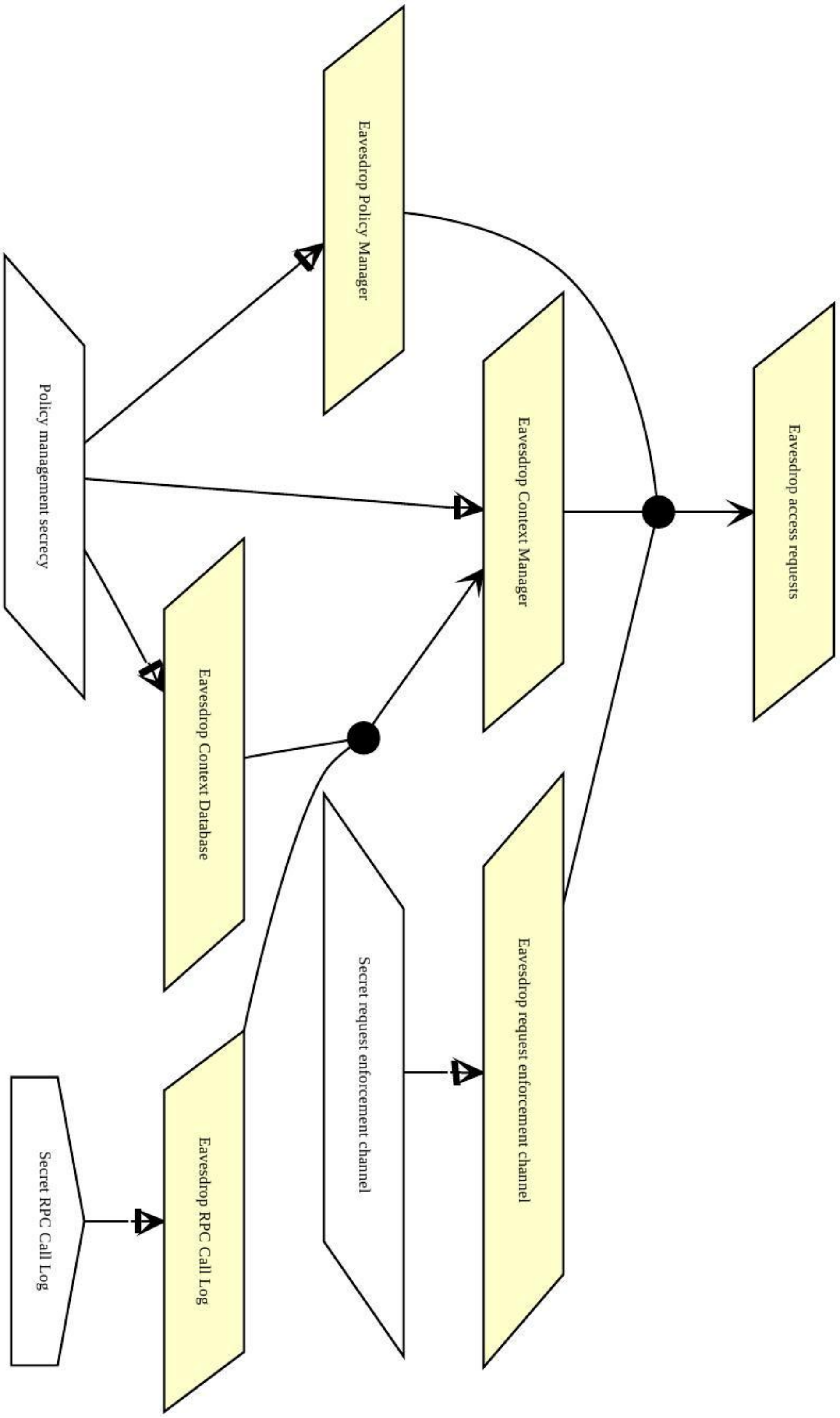
15.17.4. Participants

Attacker	Motives	Capabilities (Value)
Ethan	System resource theft	Knowledge/Methods(Medium), Software(Medium), Technology(Medium)

15.17.5. Collaboration

Target	Application Data
Exploit	Application Data

15.17.6. Implementation


Policy management secrecy
Secret RPC Call Log
Eavesdrop Context Database
Eavesdrop RPC Call Log
Eavesdrop Policy Manager
Secret request enforcement channel
Eavesdrop Context Manager
Eavesdrop request enforcement channel
Eavesdrop access requests

Ethan eavesdrop on traffic to and from the policy manager to find possible policy management vulnerabilities.

15.17.7. Obstacles

Obstacle	Category	Definition
Eavesdrop access requests	Confidentiality Threat	Eavesdrop access requests
Eavesdrop Context Database	Confidentiality Threat	Eavesdrop use of policy management in Context Database
Eavesdrop Context Manager	Confidentiality Threat	Eavesdrop use of policy management in context manager
Eavesdrop Policy Manager	Confidentiality Threat	Eavesdrop software making use of the Policy Manager
Eavesdrop request enforcement channel	Confidentiality Threat	Eavesdrop access request enforcement channel.
Eavesdrop RPC Call Log	Confidentiality Threat	Eavesdrop logged use of policy management in RPC call log

16 Attack Patterns Not Included in the Architectural Risk Analysis

Due to time constraints, the following attacks are documented but have not been included in the full risk analysis. They are lacking obstacle models and connection to system goals and architectural models. We intend to add them to the CAIRIS system in the final year of the project.

16.1. Misidentification of a PZH through disconnected TLS and HTTPS certificates.

16.1.1.1 Description

During enrolment of a PZP to a PZH, the user visits their PZH web interface to collect a short authentication token. This web interface is authenticated using standard web PKI and DNS. The enrolment process then allows the PZP to connect to the same provider and present the token. However, the PZP authentication process currently uses a different certificate (the PZH certificate not the PZH provider web interface). This is an opportunity for a man-in-the-middle attack. The same flaw may be present when a PZH is first instantiated.

16.1.1.2 Details

Attacker	Ethan
Intent	Automatically enrolling people to the wrong personal zone provider in order to capture and sell user data and credentials
Motive	Money
Target asset	Personal data
Exploit asset	PZP
Target threat	CAPEC-272: Protocol Manipulation
Exploit vulnerability	CWE-297: Improper Validation of Host-specific Certificate Data
Known uses	None
Related patterns	None

16.1.1.3 Consequences

This attack is based on a protocol flaw, and would be a vulnerability in all deployments. As a result, any webinos user could be susceptible and there could be widespread loss of data and credentials, affecting the security of personal information and privacy loss.

16.1.1.4 Mitigations

Make sure that the initial enrolment process identifies the PZH's TLS certificates. Or, modify the underlying TLS certificate to use the PZH provider's certificate rather than a different certificate.

16.2. Malicious service misuse

16.2.1.1 Description

Applications use the findService() call to find services of a particular type. It is expected that applications will then present these services to the user for them to select.

However, an application doesn't need to obey the user's intentions and might invoke a different service to the one selected. Indeed, this might be because the user is intentionally *not* selecting a service for security reasons. E.g., the user may select a storage service with only media files, but the application may then access files elsewhere.

The policy system can only handle this if policies refer to services at a finer granularity rather than API types + devices.

One motivation for this attack could be corporate espionage: an application used for a business purpose could attempt to find out extra information from the end user.

16.2.1.2 Details

Attacker	**Jessica**
Intent	Obtain private and confidential user information from APIs based on user anti-preferences
Motive	Money
Target asset	Personal data
Exploit asset	Discovery API and Policy manager
Target threat	
Exploit vulnerability	
Known uses	None
Related patterns	None

16.2.1.3 Consequences

Users unintentionally share information they had explicitly not wanted to share or give an application access to.

16.2.1.4 Mitigations

Restrict applications on the *service* rather than API level. Provide visual clues as to which services an application may access.

16.3. The webinos botnet.

16.3.1.1 Description

Webinos is used to create a botnet. A botnet is often used to gain personal data (credit card information, for example) en masse, or perform DDoS attacks on target websites and web services. There are many possible ways this could be done:

- An implementation flaw is exploited in the widget renderer, which abuses its trusted relationship with the PZP to make arbitrary API requests and access remote services.
- A method of enrolling rogue devices into personal zones is discovered which allows an attacker to join every personal zone. They can then install applications in every personal zone.
- A PZP implementation flaw (perhaps a native API implementation) is exploited by an application in order to install native malware on the device. Because this API is commonly available, the malware spreads rapidly.
- Native device malware (outside of webinos) steals PZP keys and credentials and is able to hijack the personal zone.

16.3.1.2 Details

These details concentrate on the possibility of an implementation flaw.

Attacker	Ethan
Intent	Create a large and powerful botnet and sell it to cybercriminals for DDoS or data theft.
Motive	Money
Target asset	Personal data and the device runtime
Exploit asset	PZP software, widget renderer, device software
Target threat	CAPEC-8: Buffer Overflow in an API Call

Exploit vulnerability	CWE-75: Failure to Sanitize Special Elements into a Different Plane (Special Element Injection)
Known uses	None
Related patterns	None

16.3.1.3 Consequences

Loss of personal or payment data, loss of availability for a website or service. Users being marked as 'malicious' by network providers identifying the rogue traffic.

16.3.1.4 Mitigations

- Careful design and implementation of APIs and protocols.
- Sanitise all user or application-provided input
- Isolate rendering components from privileged components.
- Defence-in-depth: recommend that users have anti-malware tools and do not install arbitrary applications.
- Review device enrolment implementation
- Use hardware-based key storage, such as a Trusted Platform Module

16.4. TLS certificate management errors

16.4.1.1 Description

The webinos platform might contain a buggy TLS certificate handling implementation, resulting in authentication errors. E.g., accepting all certificates or all hostnames, or not verifying the full certificate path. This could allow a remote attacker to make them connect to invalid PZH providers.

16.4.1.2 Sources

ACM CCS Paper: "Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security" (to appear) by Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumgärtner, Bernd Freisleben, Matthew Smith. 2012

ACM CCS Paper: "The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software." (to appear) by by M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, V. Shmatikov, and D. Boneh

16.4.1.3 Details

Attacker	Ethan
Intent	Use invalid TLS certificate checking to make users connect to the wrong personal zone hub and therefore

Motive	Money
Target asset	Personal data
Exploit asset	PZP connection manager
Target threat	CAPEC-98: Phishing
Exploit vulnerability	CWE-296, CWE-297, CWE-599. E.g., CWE-296: Improper Following of Chain of Trust for Certificate Validation
Known uses	None
Related patterns	None

16.4.1.4 Consequences

User data and identity compromised.

16.4.1.5 Mitigations

Review TLS implementations, enforce correct certificate handling.

16.5. Remote JavaScript includes

16.5.1.1 Description

Taken from You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions

Remotely including javascript in web applications is dangerous as it is a trust assumption many developers don't consider. Attacking one common inclusion provider could attack multiple applications.

In this attack, we consider that Ethan might exploit a web server hosting a popular JavaScript library imported by many trusted applications.

Another option would be for an insider working on a popular JavaScript library to insert an application-specific attack. this might be fulfilled by Irwin.

16.5.1.2 Details

Attacker	Ethan
Intent	To gain access to multiple user devices to create a botnet or access personal data residing on a personal zone
Motive	Money

Target asset	Personal data, personal zone
Exploit asset	Application
Target threat	CAPEC-444: Malicious Logic Insertion into Product Software via Externally Manipulated Component, CAPEC-96: Block Access to Libraries (related)
Exploit vulnerability	CWE-114: Process Control
Known uses	None
Related patterns	None

16.5.1.3 Consequences

All applications using and importing this library are compromised: their access to APIs may be misused by the library editor to gain access to personal data and resources. Users may suffer from data loss, identity theft (applications could be overwritten to steal user data) and more.

16.5.1.4 Mitigations

Disabling remote script invocation from external origins. Further, a developer could specify that scripts may only be loaded if their integrity matches an included checksum.

16.6. Misidentification of users requesting inter-zone certificate exchange

16.6.1.1 Description

Frankie pretends to be 'Bob' (one of Clara's friends) when requesting access to Clara's personal zone. He does this at the certificate exchange stage exploiting a potential lack of authentication of user identities. Frankie does this because he is obsessed with Clara and wants to find out more about her interests and track her activities to arrange 'accidental meetings'.

16.6.1.2 Details

Attacker	Frankie
Intent	Access personal data, track activities
Motive	Stalking
Target asset	Personal data
Exploit asset	Certificate manager, PZH admin

Target threat	CAPEC-196: Session Credential Falsification through Forging
Exploit vulnerability	CWE-287: Improper Authentication
Known uses	None
Related patterns	None

16.6.1.3 Consequences

Users may unwittingly add someone to their list of 'known users' based on an invalid identification process and authentication. This could result in data disclosure to unwanted entities which could cause embarrassment, privacy loss, device damage, and more. This process could form the basis of a large-scale attack on all webinos users, adding invalid users to their personal zones.

16.6.1.4 Mitigations

The webinos zone certificate exchange process requires users requesting access to authenticate through their OpenID credentials which will tell the PZH their email address. There are several attacks on this process, but it should be possible to specify only OpenID providers who are trusted, and make sure that the email address domain matches the OpenID provider domain name. We also rely on users knowing each other's OpenID email address or account URI. Alternative implementations might share user identity via social networks (e.g. a Facebook plugin).

16.7. Intra-zone privacy: not sharing applications across the whole zone

16.7.1.1 Description

Justin has a PC and a smartphone. He shares his PC with his girlfriend, sometimes, and so is careful to make sure that no embarrassing content is ever displayed. However, on his smartphone he installs more risqué applications and has a less savoury browsing history.

Unfortunately, webinos synchronises his settings and application policies across all devices on his personal zone. When his girlfriend goes to change some settings on his webinos-enabled PC, she realises he has a "strip poker" application installed. A long argument ensues...

16.7.1.2 Details

Attacker	None - misusability case. Arguably Justin's girlfriend or the webinos platform itself.
Intent	None - Justin is attempting to maintain his privacy and his girlfriend may accidentally impose on that
Motive	None - synchronisation of application identity and policy is used as part of the personal zone security system

Target asset	Application data and Application identity
Exploit asset	PZP
Target threat	No clear mapping to CAPEC or OWASP
Exploit vulnerability	Nothing perfect, but related - CWE-200: Information Exposure, CWE-612: Information Exposure Through Indexing of Private Data, CWE-668: Exposure of Resource to Wrong Sphere
Known uses	None
Related patterns	None

16.7.1.3 Consequences

User embarrassment and privacy loss.

16.7.1.4 Mitigations

Control synchronisation. Do not synchronise application manifests unless necessary. Allow users to purge local data and settings of private information. Restrict UI access to zone synchronisation data.

16.8. Unintentional privacy loss while sharing personal zone service addresses during inter-zone communication

16.8.1.1 Description

Two webinos users (Alice and Justin) connect their personal zones and request access to each others services. By requesting access, however, they share the address information of their applications and services. So Justin shares his device and application name and Alice shares her device and service identity.

In most cases this is not a problem, but it may result in embarrassment if it reveals the fact that Justin is in a location Alice wasn't expecting - e.g., at home when he said he was away on holiday.

16.8.1.2 Details

Attacker	None: another user (e.g., Justin)
Intent	None
Motive	Loss of personal context
Target asset	Personal data

Exploit asset	inter-zone communication
Target threat	None
Exploit vulnerability	None
Known uses	None
Related patterns	None

16.8.1.3 Consequences

Embarrassment.

16.8.1.4 Mitigations

Remove addressing information after it leaves the personal zone. However, this might not be reasonable if Alice (in this case) wants to know where Justin is accessing her data from.

17 Complete List of Actions Defined in the Trust Model

Action	Description
Access control via policy enforcement	This component is trusted to enforce policies by mediating access control requests
Adding personal zone devices	This component is trusted to add new devices to the personal zone
Authenticating devices against device certificates	This component is trusted to authenticate a connecting device against a set of trusted device certificates
Authenticating identity providers	This component is trusted to identify trustworthy identity providers and their messages as part of the OpenID process
Authenticating the widget runtime and web browser	This component is trusted to authenticate connections from widget runtimes and web browsers
Authenticating users to local device	This component can authenticate a device user to the device using local mechanisms
Authenticating user's online identities	This component can authenticate a personal zone user's identity as defined by their OpenID.
Authenticating web domains	This component is trusted to authenticate domain name TLS/SSL certificates against trusted roots
Certificate and signature processing	This component is trusted to check certificates and signatures for validity and against trusted roots
Communicating with remote PZPs	This component is trusted to communicate with other PZPs across the personal zone
Communicating with the local PZP	This component is trusted to make connections to a PZP on the same device as the component
Content isolation	This component can isolate content from external, unauthorised entities
Correct API use (after permission has been granted)	This component is trusted to use an API in a suitably safe, secure and privacy-preserving way.
Creating sessions	This component can create sessions with another component
Displaying information to the local user	This component is trusted to communicate information to the user

Editing XACML policies (local)	This component is trusted to make changes to XACML policies affecting the *local* device
Editing XACML policies (zone-wide)	This component is trusted to make changes to XACML policies which may affect *all* devices in the personal zone
Enforcing cross-origin restrictions	This component is trusted to enforce policies relating to origin restrictions, such as WARP, CSP and same-origin
Fulfilling data handling policies	This component is trusted to fulfil data handling obligations
Handling device keys	This component is trusted to use PZP private keys
Identifying applications	This component can obtain an application's identity information
Installing web applications	This component is trusted to install web applications
Maintaining platform integrity	This component is trusted to maintain the integrity of a platform (preventing malware and corruption)
Maintaining sessions	This component is trusted to maintain a communication session with another entity
Name resolution	This component is trusted to resolve the names of entities into their addresses
Process isolation	This component is trusted to isolate individual running processes
Protecting itself against runtime attacks	This component is trusted to keep itself safe against runtime attacks
Rendering applications	This component is trusted to render application content and display it to the end user
Revocation of devices	This component is trusted to revoke devices from accessing the personal zone
Routing	This component is trusted to route messages within the webinos personal zone
Sandboxing applications	This component is trusted to limit the capabilities of applications to only functionality which webinos makes available
Self-imposed least privilege	The component is trusted to request only the capabilities it

	uses and nothing more
Storing data – integrity	This component is trusted to store data in a way that will protect it from unauthorised modification
Storing data – confidentiality	This component is trusted to store data in a way that will protect it from unauthorised access
Storing keys	This component is trusted to store private keys
Taking user input	This component is trusted to receive and process user input
Untrusted Input validation	This component is trusted to receive external input and validate it to protect the platform from attack
Validating application content	This component is trusted to validate the content of applications against integrity checks