Toward Securing Sensor Clouds

Apu Kapadia, Steven Myers, XiaoFeng Wang and Geoffrey Fox
School of Informatics and Computing
Indiana University, Bloomington
{*kapadia, samyers, xw7, gcf*}*@indiana.edu*

ABSTRACT

We aim to secure smart sensor networks, where computationally powerful sensing devices such as smartphones or cognitive radios interact with the cloud. In previous work, we have proposed a large-scale brokering framework, and we are researching several facets of securing sensors in the context of this framework. In this paper we discuss initial results for three portions of this effort, challenges that remain for secure sensor networks, and specific directions we are currently pursuing. In particular, we discuss our work on (i) Sensor risk assessment, relating to the possession and environment of the smartphone sensors, (ii) New malware threats and defenses installed on the sensor network proper, and (iii) Defense against the side-channel analysis on the Software-as-a-Service infrastructure.

KEYWORDS: Sensor Network, Brokered Network, Security, Wireless.

1 INTRODUCTION

With the increased pervasiveness of sensory devices for military and civilian uses comes the demand for effective processing of the large amounts of data they collect. This demand can only be met with the low-cost computing resources offered by today's cloud computing systems. Today's cloud can already support data-intensive computing at a low cost: for example, a large-scale computing task can be accomplished on Amazon's Elastic Compute Cloud (EC2) at an expense as low as 10 cents per CPU hour. So far little effort has been made in applying the ultra cost-effective cloud platform towards analyzing and managing sensor data. Recently, we have made the first step towards building a practical sensor cloud system [21]. Different from prior work on sensor networks [21], we assume that sensors communicate directly with a proxy or broker on a cloud. In our research, we consider a group of sensors organized as a hierarchical structure or some types of partitions, which communicate with their cloud proxies through wireless channels. The sensor platforms studied in our research are ones with multiple sensors that can each measure different properties of the environment. For example, we might have GPS for positioning, microphones for sound, laser-range finders for scanning surroundings, temperature indicators, wireless radios etc. We can imagine a host of different autonomous and manned devices that contain these sensors including vehicles, robots, smart-grid nodes, mobile computers, and smartphones. For each device, we have a number of different sensors that can provide different environmental readings on a near continuous basis, further these hosts all contain reasonable computational power and power supplies for continuous function. Finally, they all have reliable cellular network conductivities. We imagine that these hosts are continually collecting data from their environment, performing some level of data processing and publishing the outcomes to a cloud for further analysis or data storage. For the purposes of our studies, we examine modern Android smartphones as exemplar hosts in our work.

Figure 1, which has also been used in our prior paper [21], illustrates the structure of the system built in our research. A critical issue for this sensor-cloud computing environment is security and privacy. In [21], we summarize the security and privacy challenges we face when building a trustworthy sensor-cloud system, which come from the following perspectives:

- The environment in which sensors work can be compromised by the adversary. For example, the adversary can artificially reduce or raise temperatures to cause the sensors to collect improper data.
- Individual sensors can be vulnerable to attacks. This can happen when the adversary has physical access to the sensors, or remote access through propagating malware.
- Information flows within the cloud can be intercepted and stolen or modified by compromised cloud nodes.
- The cloud client can be infected by malicious code implanted by an adversary, which can lead to further security breaches within a sensor-cloud system.
- The communication channels between the sensors and the cloud, and between the client and the cloud are vulnerable

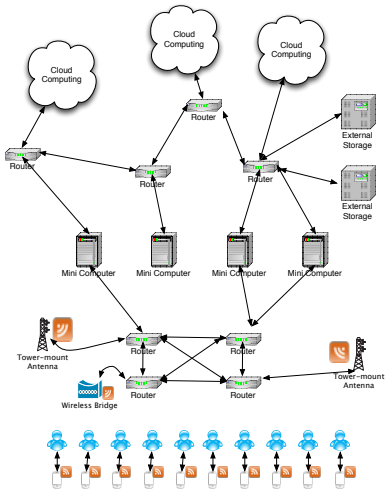

Figure 1. A depiction of the different components of the sensor and cloud-computing network. Android smartphones denote the sensors in the system, and are in the possession of individuals. The smartphones have some computational capacity, and transmit through WiFi or cellular services to a brokering network, running over traditional TCP/IP services. The brokering service can itself have computers performing filtering, processing and/or creating other mashups of sensor data.

to different types of attacks. Even when the data transferred over the channels is fully encrypted: side-channel information leaks constitute creditable threats.

Our prior research pinpoints a subset of issues within thE problem space that need immediately attention. Specifically, we investigated I) techniques for detecting anomalous use of sensors, particularly, when the adversary gains unauthorized physical access to smartphones; ii) we demonstrated that intelligent smartphone-based malware can be built to "understand" the context of a phone conversation and extract a small amount of high-value information from the context [33] (Given the small quantity of such sensitive information, the malware can deliver it to its master through covert channels, even without direct network access [33].); and iii) prior research shows that even in the presence of Wi-Fi encryption and HTTPS protection, the traffic features of the communication between sensors and the cloud, and between the cloud and its clients can easily be analyzed to infer highly-sensitive user data [6].

In this paper, we sketch recent progress and follow-up on previously discussed research plans on these fronts, including detection of anomalous use of sensors, and defenses against smartphone malware and side-channel leaks. We summarize below.

Sensor risk assessment. With the possibility of a vast number of sensors deployed throughout an environment on different hosts, with all the sensors feeding information in to a cloud computing infrastructure, there is the possibility for an adversary to attack a sensor, and thereby make its readings untrustworthy. There are several types of attacks on differing types of sensors one can imagine. These range from changing the environment of the sensor so its readings are faulty, to actively changing the logic in the sensor system itself. We are concerned with the former, as opposed to the latter in which traditional patching and anti-viral and anti-malware technologies are likely to be used. Therefore, what is needed is the ability of a sensor to measure changes to its normal operating environment, and report changes that seem to indicate a high risk that the sensor is not in an appropriate operating environment. Our research on contextual authentication and deauthentication of smartphones, has developed an layered sensing approach, whereby each sensor measures risks locally based on its own information and appropriate risk model. Individual sensor risks are then conglomerated together through a trained support vector machine, to a larger global risk view. In our application, the global risk is used to either deauthenticate or authenticate a user's smartphone, but the data could as easily be used to measure data-reliability and provenance from sensors on the phone.

Defense against sensory malware Our research on sensory malware, i.e., malware augmented by on-board and paired sensors, demonstrates how Trojan malware can spy on a user's phone conversations and steal their credit card information (for example) and send this information to a "malware master" [33]. Furthermore, our Trojan malware evades existing defenses.

We demonstrated that sensory malware is a potent threat with challenges remaining to defend against such attacks. Even though we presented a targeted solution to prevent sensory malware attacks, the general problem of knowing when sensors are taking legitimate readings of the environment and when they are actually causing a privacy breach is hard to solve.

Detection and quantification of side-channel information leaks Our prior research shows that the encrypted communication channels in sensor clouds are vulnerable to side-channel analysis. Specifically, a prominent feature of such systems is its extensive use of web applications, whose program logic is distributed over the client-side browser and the cloud-side web server. Our recent study [6] shows that such side-channel leaks present a serious threat to the Software-as-a-Service (SaaS) infrastructure: several popular web applications reveal such sensitive user data as family incomes, health records, investment strategies and others. Our research further shows that to mitigate this threat, we need to change the way the current web applications are being developed. In the follow-up work, we made the first step towards building a more secure web application. We propose a suite of new techniques [47] that automatically analyzes a web application

to detect its side-channel information problems and quantify such leaks. The outcomes of such an application informs the web developer of the presence of the problem and its seriousness. Based on such understanding, the developer can decide on the course of action that contributes to the improvement of the security quality of her applications.

The rest of the paper is organized as follows. Section 2 describes our recent work on sensor risk assessment. Section 3 presents a new detection mechanism we developed for mitigating the threat of sensory malware. Section 4 surveys our new technique for mitigating side-channel leaks. Section 5 describes the related research and Section 6 concludes the paper.

2 Sensor risk assessment

Our research on contextual sensor data shows that it is feasible for smartphones to continuously evaluate their sensors and based on appropriate risk-models calculate risk in real-time. While risk is measured in real-time, the risk models which must be learned through the use of appropriate machine learning algorithms, can be computationally intensive. This is not troubling, as training can either be offloaded to the cloud, or if security and/or privacy reasons interfere, then these computationally intensive tasks can be scheduled for a phone's low-use periods. For example, smartphones can train while the phone is charging during the evening. With relation to specific risks based on individual sensors, we have considered two alternate sensors: geo-locational data as is derived through a mashup of GPS, WiFi and Cellular Tower positioning, and determination of friends and strangers in the near proximity via short range Bluetooth radio.

We have shown that by labeling a small number of geographic sites that users frequent and developing a history of geo-locational positioning information, the phone can infer when it is supposed to be in certain locations. This modeling is done through the use of a third-order Hidden-Markov Model (HMM). Sequences of observations correspond to a discretization of location into a string on labeled known and unknown locations. Initially we performed experiments where time was discretized in to 30 minute segments. Using positional data from the MIT Reality Mining Project of Eagle and Pentland[14] , which contains positional data collected via phones for 100 users over a 9 month period, we trained the HMMs using traditional Baum-Welch and Viterbi HMM training algorithms. In fact, we train a separate HMM for each 4-hour segment of time during the day. This allows us to predict location based on a reasonably recent location history, but not so short as to be meaningless. Next, to calculate the risk of the phone's present location, we use an HMM trained over a 4-hour period immediately preceding the current time segment, and use it to predict—with the forward algorithm— the likelihood of the HMM having made the observed sequence of positions. We normalize the result, but essentially if the forward algorithm suggests that the HMM would produce a given sequence with low probability, then we predict high-risk, and vice-versa. Certain thresholds of risk are then set to determine if the phone should force authentication.

With such a risk calculation and forced authentication in place, we are now able to simulate loss and theft of the phones, and determine how effective our positional model is. Essentially, we simulate theft of phones by having their locations switch to a series of previously unknown locations that the phone does not frequent. To simulate loss of the phone, the phone is assumed to remain in its current location independent of the movements of its owner. While such simulations are first approximations to actual theft and loss, we believe that they appropriately simulate a large number of such cases. Actual data on the positioning of stolen phones is unavailable. Given such a model, we then determined thresholds for measured risk which should force authentication, allowing us to determine our type I and II errors. Some example users' Receiver Operating Characteristic (ROC) curves for theft are shown in Fig. 2. The right-most graph is atypical in performance, the other two more accurately report mean performance. Different lines depict Time of Theft plus an offset of 30 minute intervals (i.e. TOT+i depicts theft detection $i \cdot 30$ minutes after theft.

As can be seen in the ROC, given as little as 30 minutes, many users have reasonable theft detection. However, in practice 30 minutes may be too long of a period to detect theft, but with the 30 minute time segments use for training this is the fastest one might hope to detect theft or loss. Therefore, we are currently investigating the feasibility and efficacy of shortening time segments to 5 minute intervals to allow for a much more fine-grained detection. Issues involved with higher granularity are the computational intensity needed to train and compute with the much larger HMMs that need to be maintained, and the amount of history that needs to maintained to predict current location.

There are times when sensors are in new and unanticipated locations, for these times we buttress our geo-location sensor with a the Bluetooth sensor, used to detect the devices that are in close proximity to it. Roughly, the smartphone can learn which devices are frequently in close proximity to it, and learn that such devices are "friendly". The mere presence of such devices can indicate that there is a low risk that the phone has been stolen or lost. Examples could include your phone recognizing your car's presence, or two spouses phone's recognizing the presence of each other. In both scenarios, the presence of a trusted Bluetooth id suggests the risk that the phone has been lost or stolen is low. This scheme can also be used with other short-range wireless technologies, such as the 802.11 family of wireless networks. While maintaining Bluetooth radios in powered modes can increase the rate of power consumption, there are a large number of users who maintain power to their WiFi and Bluetooth radios throughout the day, without difficulty. In any event, we are using the smartphones

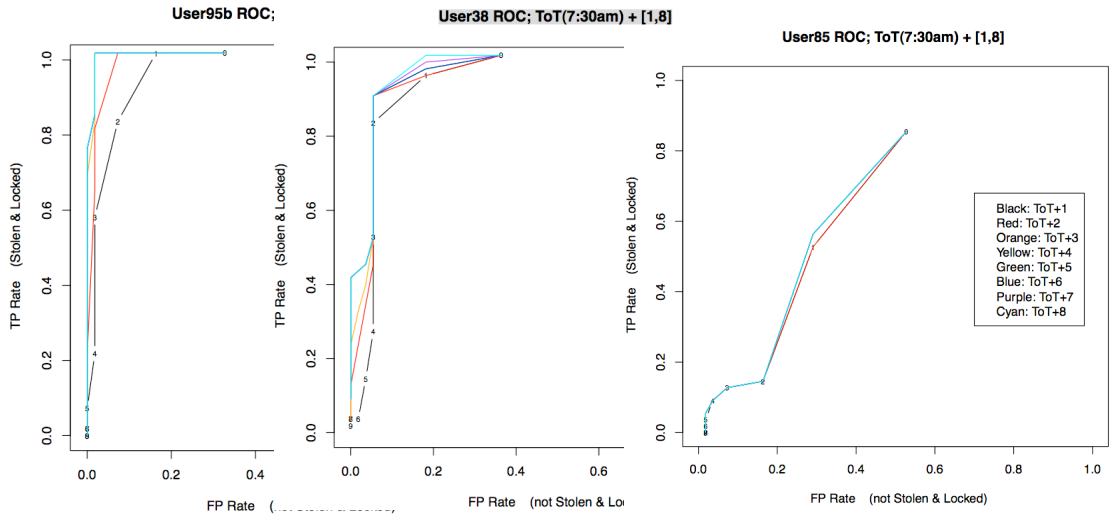

Figure 2. A depiction of three users' ROC curves.

as a test-bed platform for a large number of sensors, where power supplied to the radios need not be an issue.

In order to measure risk, we consider a two level white/grey-listing system. First, phones can whitelist certain Bluetooth devices, such that their presence will result in a low-risk measurement, no matter the presence of other devices. However, in the absence of such white-listed devices, a risk measurement must be made based on the devices that can be detected. In order to make such a measurement the phone constantly searches for Bluetooth identifiers and records their relative presences over time, defining a distribution D over Bluetooth identifiers: devices that are observed frequently have high probability mass, and vice-versa. Using this distribution, we can now think of those individuals that are frequently in proximity as more trusted, than those that are not. The entropy of this distribution, $H(D)$, tells us the average amount of risk we are exposed to on a daily basis. Thus to measure the risk of a given situation, we can measure the information presented by the identifiers (IDs) i, $-\log(\Pr(i))$, currently visible, and determine its distance from $H(D)$. To measure the probability of seeing a given id, we keep track of the amount of time all Bluetooth IDs have been seen in recent history (a month). We then consider the fraction of time a given id has been observed, presenting its probability. We use a heuristic to account for identifiers that have never previously been observed, according them a very small probability in the distribution that is not 0.

In order to punish and reward high and low risk scenarios, we consider this difference transformed under a logistic sigmoid. This gives us the following risk function:

$$Risk = \frac{1}{1 - e^{-\sum_{i \in \text{Observed ids}} (-\log(\Pr(i)) - H(D))}}.$$

In Figure 3 we show a sample of our risk prediction function, when used to predict risk for a given day in the Reality Mining Dataset. The upper (magenta) line depicts the relative risk predicted at any given point in time of the day based on the the grey-listing Bluetooth risk predictor (i.e., predicted by logistic sigmoid). The low risk time periods is consistent with the device being in the presence of devices that are frequently present. The high-risk periods indicate the presence of a number of individuals which the phone has rarely or never seen. The lower line indicates the relative probability mass of the observed identifiers at a given time.

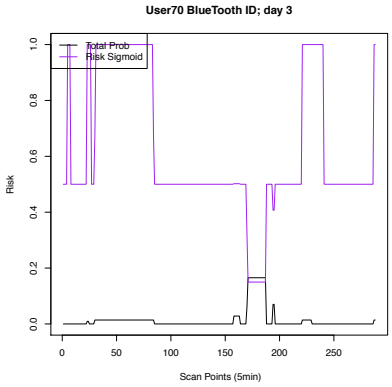

Figure 3. Depiction of a user's relative risk.

We are currently in the process of modeling theft and loss with the Bluetooth sensors, to measure the system's efficacy. We are again using the Reality Mining Dataset [14] which contains not only positional data on the individuals, but scans of nearby Bluetooth devices. Our final goal is to implement a global risk analyzer that will take the local

risk measurements from the individual sensors, and provide a better global risk measurement, then the individual sensors can provide. The expectation is that this will be done through the use of a trained Support Vector Machine (SVM).

3 Defense against Sensory Malware

Our research on sensory malware shows that is is feasible for Trojan malware to spy on user's speech-based communications through access to the microphone [33]. In particular we demonstrated how our Trojan can target a small amount valuable information such as a spoken credit card number or social security number and transmit this information to the "malware master". We showed how our sensory malware can evade existing defenses through the use of stealthy local processing of information and the use of covert channels so as to avoid suspicious permissions needed by the Trojan application. We thus demonstrated that sensory malware is a potent threat with challenges remaining to defend against such attacks.

Even though we presented a targeted solution to prevent sensory malware attacks, the general problem of knowing when sensors are taking legitimate readings of the environment and when they are actually causing a privacy breach is hard to solve. Our current defense against speech-based malware is to recognize that a phone call to a sensitive number (such as a bank or credit card company) is being made. If so, the reference monitor that controls when audio can be recorded and delivered to applications can preempt any ongoing recordings and replace it with blank audio. We showed that such a defense incurs minimal penalty and does not significantly delay outgoing phone calls [33].

To defend against sensory malware attacks in general, thus we propose techniques that take into account the *context* under which sensing is being performed. In our defense explained above, we can see that a phone call to a sensitive number is contextual information that is interpreted as an unsafe situation to be recording from the microphone. Similarly, the sensors can be used to detect various unsafe contexts for various sensors. We are currently exploring how to automatically infer such sensitive contexts for various sensors. For example, in our work (reported in this paper) for sensor risk assessment, we utilize various on-board sensors to assess the risk of theft. We propose various models to infer an unsafe or anomalous situation for the phone. In the same way, we hope to characterize normal situations when different sensors are used, and thus flag anonymous uses of sensors to the user. Our system can then take input from the user as to whether the access to a sensitive sensor is authorized and if so the system can continually refine its models for risk assessment.

We thus hope that we can build a generalized framework to learn and refine context based risk assessments of when it is relatively safe or unsafe to use different sensors. Again, "when" is not only a matter of time, but also various situations such as who is around, what motion is being sensed, what the camera can see, and so on. We believe this approach is an exciting avenue for research since sensor-based context are used to secure the use of sensors.

4 Mitigating Side-channel Leaks from Web Applications

Our research, as elaborated in our prior paper [6], indicates that mitigation of the side channel problem in web applications is nontrivial. Particularly, it is unlikely to have an application-agnostic solution to the problem. We evaluated the effects of two padding strategies on web applications, including *rounding* that rounds up packet sizes to the nearest multiple of certain bytes, and *random padding* that appends packets to a random length within a certain range. We found that the leaks within a famous online health information system cannot be completely subdued even after packets are rounded to 512 bytes, which incurs a network overhead of 32.3%. For a well-known tax application, even rounding packets to 2048 bytes, with an overhead of 38.10%, is insufficient for hiding the 7 income ranges disclosed from asymmetric execution paths, which actually cannot be covered by padding alone. More interesting is the observation that search engine leaks, which allows an eavesdropper to figure out the content of the input data by looking at the sizes of the packets carrying auto-suggestion lists [6], cannot be fixed by rounding, as the auto-suggestion lists are actually GZip-compressed by web servers, and some organizations decompress them for inspecting the packets while others let the users' browsers do the decompression: as a result, the web server cannot use rounding to protect both compressed and uncompressed contents transmitted in different recipients' Wi-Fi networks. On the other hand, random padding seems to have nothing but marginal effects on the inference attack on the images of an online investment application, because the eavesdropper can compare the traffic attributes of the same user's images collected from different rounds of client/server interactions to remove the randomness.

The first technical challenge is how to find the side-channel vulnerabilities within individual web applications. Detection of such vulnerabilities requires an in-depth analysis of information flows within the applications, and in some cases, acquisition of background knowledge on how they are used. In our recent paper [47], we present the first technique, call *Sidebuster*, for detecting and quantifying side-channel leaks. Based upon a set of "taint sources" labeled by the developer as sensitive, our techniques perform an *information-flow analysis* on source code of web applications to track the propagation of "tainted" data across a program's client/server components. Whenever the tainted data are found to be transmitted to the network through an encrypted channel, an information-leak evaluation is performed to understand whether the side-channel information of the channel, such as packet sizes and sequences, can be used to infer the content of the data. Whenever a branch condition is found to be tainted and its branches

contain client/server communications, Sidebuster evaluates whether the attributes of such communications reveal the sensitive condition. We also developed new techniques for analyzing GUI widgets, such as auto-suggestion lists, which are triggered by input events (e.g., letters being entered) to synthesize different user inputs into an integral variable (e.g., a query word) that the developer labels as a "taint target".

Every web application gives away some information through its side-channels. However, not all such information leaks are serious enough to warrant mitigation efforts. For example, people could live with disclosure of the lengths of their query words and some operations they performed, such as sending/receiving emails. A question, therefore, becomes how to quantify the private information that can be inferred from a side-channel vulnerability. This question can be answered through dynamic analysis, because the encrypted traffic of a web application is actually generated by its underlying web servers/browsers, whose source code is often beyond the access of the developer. To conduct such an analysis, we also come up with a design of a quantification technique that systematically re-runs selected portions of a web application to understand how the domain of a taint source or target can be partitioned by its side-channel leaks [47].

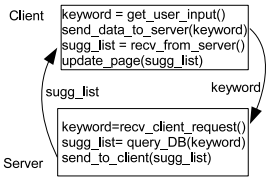

Figure 4. Data-flow example

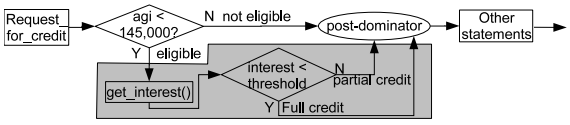

Figure 5. Control-flow example. The code within the dark boxes runs on the client side.

Figures 4 and 5 describe two examples. The first one is the code of a suggestion list. For simplicity, we describe the program using Java pseudocode. The program is split into the part that runs on a web server and the part that works in the client's browser.[1] Given a tainted variable `keyword`, a static analysis can identify the taint data to be propagated to the network through `send_data_to_server`. This function is then instrumented for further evaluation. Analysis of the server-side code also taints the suggestion list to be sent back to the client, which makes our analyzer instrument `sent_to_client` on the server side. Then, we can instrument related program statements for dynamic analysis, because in the absence of data, static analysis alone cannot determine whether the content of the tainted variable is inferable from attributes of the tainted web traffic. During the runtime of the application, the instrumented code works with the web server and browser to identify the sizes of the packets containing `keyword` letter(s) and those carrying their corresponding suggestion lists. The former leaks little information except the length of `keyword`, while the latter changes with the content of `keyword`: different contents lead to suggestion lists with different sizes. Such a side-channel leak is therefore identified. The example in Figure 5 is the simplified program logic for tax credit and deduction claims in a tax preparation application. Suppose that the variable, `agi`, is marked as a taint source. An information flow analysis tracks the taint data to the branch condition "`agi<145000`", and further identifies its *scope*, from the condition to a *post-dominator* where all branches converge. Checking the code within the scope, we find that the "not eligible" branch involves one round of client/server interactions, while the "eligible" branch has two, one for the deduction request/response and the other for entering the user's interest. This asymmetric structure then is detected to disclose the tainted condition.

White-box testing of web applications needs their source code and depends on the programming languages with which they are implemented. In follow-up research, we will study black-box testing that does not suffer from these constraints. Given that most high-profile web applications are actually closed source, black-box testing has become a popular choice for evaluating their security problems such as cross-site scripting (XSS) and SQL injection flaws. Although there are a large number of open-source and commercial black-box fuzzers, none of them check for side-channel leaks. A prominent property of a web application is that part of its program logic resides on the client side, and can therefore be used for guiding a fuzz test. As an example, let us consider a suggestion list with its input text box labeled as a taint source. Analyzing its JavaScript code reveals that the content of the text box will be sent to the server in response to every keystroke, which allows an automatic fuzzer to generate test keystrokes and evaluate the web flow vectors triggered by these inputs. Such an analysis can also uncover the changes of keystroke inputs resulting in different suggestion lists, which therefore also need to be tainted. The contents of the lists further inform the fuzzer of the options the user has, i.e., the legitimate values the input text box can take. This enables the fuzzer to generate test cases to evaluate whether these options can be distinguished from each other by their traffic attributes.

Another important direction we will pursue is to automatically transform a web application to remove the discovered side-channel leaks. In the case that side-channel leaks are actually caused by asymmetric execution paths (Figure 5), the application will be modified to either include fake state

[1] A Java program can be easily complied into AJAX code by the Google Web Toolkit (GWT)

transitions or move some of its states to its client-side component. Given the complexity of today's web applications and the presence of a large amount of legacy code, it is desirable for this code transformation to be supported by a suite of automatic tools, which we plan to develop. Based upon the side-channel vulnerabilities detected by white-box testing, padding policies can be specified for state transitions whose traffic attributes disclose the content of sensitive data flows. For the example in Figure 4, the policy can be to pad all responses that carry suggestion lists. Mitigation of control flow leaks can be more complicated. Consider Figure 5. We have two options: either adding a round of fake communications and delay on the "not eligible" branch or move the code related to a state transition, e.g., the program logic bounded by dash lines, to the browser. Such code splitting can be done automatically [9].

5 RELATED WORK

5.1 Mobile phone security and privacy

There has been some work in using sensors to establish context for different purposes on smartphones. The work of Peddemors et al. [30] uses past networking and sensor events to predict future network events. They give examples of predicting network availability. The ability to predict events is distinct from deviating from normal or prescribed behavior. Nonetheless they use the prediction of being at home or work, and for durations. Therefore, the system should be considered. Of particular problem is the complexity of computing predicted events, which would be too slow in our scenario.

The work of Tanviruzzaman et al. [37] is most similar to that discussed here. In their work, they suggest the use of a hierarchy of sensor information to establish authentication, and show some work on using accelerometer data on an iPhone to produce a biometric that can be used to authenticate to the phone. Jakobsson et al. [19] discuss the notion of implicit authentication of phones based on contextual data, and use call pattern data.

5.2 Sensory malware threats and defenses

Researchers have been investigating attacks and defenses related to sensory malware [5]. Work on proof-of-concept video malware shows how malware can capture video and transmit this video after suitable compression to lessen the burden on the network [45]. However such approaches are not stealthy because the amount of data sent over the network is still very large. We would like to assume that network access is limited completely using techniques such as Kirin [12], a security certification mechanism for applications on Android. Even in cases where a system such as Saints [28] is used to control the interaction between applications, we would like to study the use of covert channels to circumvent such mechanisms. While techniques such as behavioral detection of malware by monitoring system calls [3], and power consumption [25] attempt to detect malware on mobile platforms, we aim to study the limits of such detection techniques. Resources are limited on mobile devices, and malware could circumvent detection because of the inherent limitations placed on the detection techniques.

5.3 Side-channel information leaks

Side-channel leaks have been studied for a long time in different contexts. In addition to the information leaks that happen through electromagnetic signals (e.g., keystroke emanation [42]), shared memory/registers/files between processes (e.g., the recent discovery of the side-channel weakness in Linux process file systems [46]), CPU usage metrics, etc, the side-channel attacks are recently found to threaten cloud computing platforms like Amazon EC2 [31]. Examples of side-channel leaks through encrypted channels include the attack on the RSA secret keys used in OpenSSL [4], keystroke inference from SSH [34], analyses on phrases and sentences from the variable-bit-rate encoding in VoIP [44], and detection of movie titles in an encrypted video-streaming system [32]. For encrypted web communication, prior research shows that a network eavesdropper can fingerprint web pages using their side-channel characteristics [43, 7]. Such information leaks pose a threat to anonymity channels like Tor, MixMaster and WebMixes [36, 11, 2].

6 SUMMARY

We have outlined our research on secure sensor networks in the context of a high-level cloud based brokering architecture and highlighted various research challenges going forward. We outline research challenges associated with assessing the trustworthiness of the sensors based on environmental sensor data, detecting and defending against "sensory malware" on such sensors, and mitigating side-channel leaks when sensor devices communicate with the cloud. We believe these components of the overall cloud based sensor network architecture are the least trustworthy since they are out of the control of the cloud "back end." Thus, addressing these challenges will help protect the integrity of the sensing platforms, the privacy of users who carry mobile sensors, as well as the delivery of sensor data to the cloud. These protections will greatly contribute to trustworthy collection of sensor data from smart and mobile sensing devices.

REFERENCES

[1] Jani Mäntyjärvi, Mikko Lindholm, Elena Vildjiounaite, Satu-Marja Mäkelä, and Heikki Ailisto. *Identifying users of portable devices from gait pattern with accelerometers*, IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume 2, 2005.

[2] G.D. Bissias, M. Liberatore, D. Jensen, and B.N. Levine. "Privacy vulnerabilities in encrypted http streams," In proceedings of Privacy Enhancing Technologies Workshop (PET 2005), pages 1–11, 2005.

[3] A. Bose, X. Hu, K.G. Shin, and T. Park. "Behavioral detection of malware on mobile handsets," In MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services, pages 225–238, New York, NY, USA, 2008. ACM.

[4] D. Brumley and D. Boneh. "Remote timing attacks are practical," In proceedings of the 12th USENIX Security Symposium, pages 1–14, 2003.

[5] L. Cai, S. Machiraju, and H. Chen. "Defending against sensor-sniffing attacks on mobile phones," In MobiHeld '09: proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds, pages 31–36, New York, NY, USA, 2009. ACM.

[6] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow. In *Oakland '10: The 31st IEEE Symposium on Security and Privacy*, May 2010.

[7] H. Cheng and R. Avnur. "Traffic analysis of SSL encrypted web browsing," `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.1201\&rep=rep1\&type=url\&i=0`, 1998.

[8] J. Cheng, S.H.Y. Wong, H. Yang, and S. Lu. "Smartsiren: virus detection and alert for smartphones," In MobiSys '07: proceedings of the 5th international conference on Mobile systems, applications and services, pages 258–271, New York, NY, USA, 2007. ACM.

[9] Stephen Chong, Jed Liu, Andrew C. Myers, Xin Qi, K. Vikram, Lantian Zheng, and Xin Zheng. Secure web application via automatic partitioning. In *In SOSP'07*, pages 31–44. ACM Press, 2007.

[10] D. Dagon, T. Martin, and T. Starner. "Mobile phones as computing devices: The viruses are coming!" *IEEE Pervasive Computing*, 3(4):11–15, 2004.

[11] G. Danezis. "Traffic analysis of the http protocol over TLS," `http://homes.esat.kuleuven.be/~gdanezis/TLSanon.pdf`, as of Dec 2009.

[12] W. Enck, M. Ongtang, and P. McDaniel. "On lightweight mobile phone application certification," In CCS '09: proceedings of the 16th ACM conference on Computer and communications security, pages 235–245, New York, NY, USA, 2009. ACM.

[13] W. Enck, M. Ongtang, and P. McDaniel. "On lightweight mobile phone application certification," In CCS '09: proceedings of the 16th ACM conference on Computer and communications security, pages 235–245, New York, NY, USA, 2009. ACM.

[14] N. Eagle and A. Pentland, "Readlity mining: sensing complex social systems," Personal and Ubiquitous Computing, 10(4):1-268, Springer-Verlag, 2006

[15] W. Enck, M. Ongtang, and P.D. McDaniel. "Understanding android security," *IEEE Security & Privacy*, 7(1):50–57, 2009.

[16] W. Enck, P. Traynor, P. McDaniel, and T. La Porta. "Exploiting open functionality in sms-capable cellular networks," In CCS '05: proceedings of the 12th ACM conference on Computer and communications security, pages 393–404, New York, NY, USA, 2005. ACM.

[17] K. Farrahi and D.G. Perez. "Learning and predicting multimodal daily life patterns from cell phones," In J.L. Crowley, Y. Ivanov, C.R. Wren, D. Gatica-Perez, M. Johnston, and R. Stiefelhagen, editors, ICMI, pages 277–280. ACM, 2009.

[18] T. Iso and K. Yamazaki. "Gait analyzer based on a cell phone with a single three-axis accelerometer," In MobileHCI '06: proceedings of the 8th conference on Human-computer interaction with mobile devices and services, pages 141–144, New York, NY, USA, 2006. ACM.

[19] M. Jakobsson, E. Shi, P. Golle and R. Chow "Implicit Authentication for Mobile Devices" In the proceedings of HotSec'09: The 4th USENIX Workshop on Hot Topics in Security, Montreal, Canada, August 10–14, 2009.

[20] A. Kapadia, D. Kotz, and N. Triandopoulos. "Opportunistic Sensing: Security Challenges for the New Paradigm," In The First International Conference on Communication Systems and Networks (COMSNETS), January 2009.

[21] A. Kapadia, S. Myvers, X. Wang and G. Fox. "Secure Cloud Computing with Brokered Trusted Sensor Networks," In The 2010 International Symposium on Collaborative Technologies and Systems (CTS '10), May 2010

[22] C. Karlof, N. Sastry, and D. Wagner. "Tinysec: a link layer security architecture for wireless sensor networks," In SenSys '04: proceedings of the 2nd international conference on Embedded networked sensor systems, pages 162–175, New York, NY, USA, 2004. ACM.

[23] J.K. Lee and J.C. Hou. "Modeling steady-state and transient behaviors of user mobility: formulation, analysis, and application," In MobiHoc '06: proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing, pages 85–96, New York, NY, USA, 2006. ACM.

[24] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. "TinyOS: An operating system for sensor networks," In Ambient Intelligence. Springer Verlag, 2004.

[25] L. Liu, G. Yan, X. Zhang, and S. Chen. "Virusmeter: Preventing your cellphone from spies," In E. Kirda, S. Jha, and D. Balzarotti, editors, *RAID*, volume 5758 of Lecture Notes in Computer Science, pages 244–264. Springer, 2009.

[26] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. "Minisec: a secure sensor network communication architecture," In IPSN '07: proceedings of the 6th international conference on Information processing in sensor networks, pages 479–488, New York, NY, USA, 2007. ACM.

[27] Wired News. "Declassified NSA document reveals the secret history of tempest," `http://www.wired.com/threatlevel/2008/04/nsa-releases-se`.

[28] M. Ongtang, S.E. McLaughlin, W. Enck, and P.D. McDaniel. "Semantically rich application-centric security in Android," In ACSAC, pages 340–349. IEEE Computer Society, 2009.

[29] M. Ongtang, S. E. McLaughlin, W. Enck, and P. D. McDaniel. "Semantically rich application-centric security in Android," In ACSAC, pages 340–349. IEEE Computer Society, 2009.

[30] A. Peddemors, H. Eertink, and I. Niemegeers. "Predicting mobility events on personal devices", *Pervasive and Mobile Computing, Special issue on Human Behaviour in Ubiquitous Environments,* To Appear.

[31] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," In CCS '09: proceedings of the 16th ACM conference on Computer and communications security, pages 199–212, New York, NY, USA, 2009. ACM.

[32] T.S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno. "Devices that tell on you: privacy trends in consumer ubiquitous computing," In SS'07: proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, pages 1–16, Berkeley, CA, USA, 2007. USENIX Association.

[33] Roman Schlegel, Kehuan Zhang, Xiaoyong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. "A Stealthy and Context-Aware Sound Trojan for Smartphones," In the 18th Annual Network & Distributed System Security Symposium (NDSS '11), San Diego, CA, February 6–9, 2011.

[34] D.X. Song, D. Wagner, and X. Tian. "Timing analysis of keystrokes and timing attacks on SSH," In SSYM'01: proceedings of the 10th conference on USENIX Security Symposium, pages 25–25, Berkeley, CA, USA, 2001. USENIX Association.

[35] G.E. Suh, J.W. Lee, D. Zhang, and S. Devadas. "Secure program execution via dynamic information flow tracking," In ASPLOS-XI: proceedings of the 11th international conference on Architectural support for programming languages and operating systems, pages 85–96, 2004.

[36] Q. Sun, D.R. Simon, Y.M. Wang, W. Russell, V.N. Padmanabhan, and L. Qiu. "Statistical identification of encrypted web browsing traffic," In SP '02: proceedings of the 2002 IEEE Symposium on Security and Privacy, page 19, Washington, DC, USA, 2002. IEEE Computer Society.

[37] M. Tamviruzzaman, S.I. Ahamed, C.S. Hasan, and C. O'brien. "ePet: when cellular phone learns to recognize its owner," In SafeConfig '09: proceedings of the 2nd ACM workshop on Assurable and usable security configuration, pages 13–18, New York, NY, USA, 2009. ACM.

[38] The Tor Project. *Tor: anonymity online.* `http://www.torproject.org/`, 2009.

[39] P. Traynor. "Securing cellular infrastructure: Challenges and opportunities," *IEEE Security & Privacy*, 7(4):77–79, 2009.

[40] P. Traynor, W. Enck, P. McDaniel, and T.L. Porta. "Mitigating attacks on open functionality in sms-capable cellular networks," In *MobiCom '06: proceedings of the 12th annual international conference on Mobile computing and networking*, pages 182–193, New York, NY, USA, 2006. ACM.

[41] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, P.D. McDaniel, and T.F. La Porta. "On cellular botnets: measuring the impact of malicious devices on a cellular network core," In E. Al-Shaer, S. Jha, and A.D. Keromytis, editors, *ACM Conference on Computer and Communications Security*, pages 223–234. ACM, 2009.

[42] M. Vuagnoux and S. Pasini. "Compromising electromagnetic emanations of wired and wireless keyboards," In *proceedings of the 18th USENIX Security Symposium*, pages 1–16, Montreal, Canada, 2009. USENIX Association.

[43] D. Wagner and B. Schneier. "Analysis of the SSL 3.0 protocol," In *WOEC'96: proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 4–4, Berkeley, CA, USA, 1996. USENIX Association.

[44] C.V. Wright, L. Ballard, S.E. Coull, F. Monrose, and G.M. Masson. "Spot me if you can: Uncovering spoken phrases in encrypted voip conversations," In *SP '08: proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 35–49, Washington, DC, USA, 2008. IEEE Computer Society.

[45] N. Xu, F. Zhang, Y. Luo, W. Jia, D. Xuan, and J. Teng. "Stealthy video capturer: a new video-based spyware in 3g smartphones," In *WiSec '09: proceedings of the second ACM conference on Wireless network security*, pages 69–78, New York, NY, USA, 2009. ACM.

[46] K. Zhang and X. Wang. "Peeping Tom in the Neighborhood: Keystroke Eavesdropping on Multi-user Systems," In *USENIX Security '09: proceedings of the 18th USENIX Security Symposium*, Montreal, Canada, 2008. USENIX Association.

[47] K. Zhang, Z. Li, R. Wang, X. Wang and S. Chen. "Sidebuster: Automated Detection and Quantification of Side-channel Leaks in Web Application Development," In *CCS '10: the 17th ACM Conference on Computer and Communications Security*, Chicago, USA, 2010, ACM.