Methods of Detection of HTTP Attacks on a Smart Home Using the Algebraic Matching Method

Viktor Horbatiuk[1], Serhii Horbatiuk[1]

[1]*Glushkov Institute of Cybernetics, 40 Akademika Hlushkova Avenue, Kyiv, 03680, Ukraine*

Abstract

All international and domestic spheres of production and service are developing at a frantic pace, and in modern life it is no longer possible to imagine any enterprise or government institution without connecting to the Internet and using cloud services. The development of digital technologies forces the application of innovative solutions in everyday life and entertainment. In our modern age with society's current dependence on high-tech gadgets and the Internet, we can definitely mark the emergence of smart home technology. In this regard, interest in private information on the network is growing, more approaches to attacks are appearing, cybercrime is becoming more organized, and its level is increasing. This work aims to show the types of cyber attacks on smart homes, as well as tools and methods for their detection, in particular, the method of mathematical comparison, which provides an opportunity to create stable web applications and services, taking into account the requirements for their security and reliability.

Keywords

cyber security, HTTP attacks, smart home, attack detection, algebraic approach, algebraic matching, attack formalization, security properties.

1. Introduction

A smart home is a system of sensors and devices, combined into a single system, capable of performing actions and solving certain everyday tasks without human intervention. The Internet of Things (IoT) is the mechanism that currently powers smart homes. Today, HTTP traffic dominates the Internet. All programmable devices, smart appliances and devices in today's smart homes are connected to the Internet. Data centers are experiencing high volumes of HTTP traffic, and many businesses are seeing more and more of their revenue from online sales. However, as its popularity grows, so do its risks, and like any protocol, HTTP is vulnerable to attack. Attackers use various attack methods to obtain user data or create a denial of service on web servers. Such attacks are done to gain some benefit or profit or just for fun.

A cyberattack is an attack by cybercriminals using one or more computers against one or more computers or networks. A cyberattack can maliciously shut down computers, steal data, or use a compromised computer as a launching point for other attacks.

The issue of cyber security is very important because it protects all categories of data from theft and damage. This includes confidential data, personal information, protected health information, personal information, intellectual property, data, and government and industry information systems.

Without cyber security programs, your home will not be able to protect itself from data breach attempts, making it an irresistible target for cybercriminals. Risks are increasing due to global connectivity and the use of cloud services such as Amazon Web Services to store sensitive data and personal information. The widespread misconfiguration of cloud services, combined with increasingly organized cyber criminals, means that the risk of your home being affected by a successful cyber attack or data breach is increasing. Smart home service providers can no longer rely solely on off-the-shelf cybersecurity solutions such as antivirus software and firewalls, cybercriminals are becoming smarter, and their tactics are becoming more resistant to conventional cyber defenses.

13th International Scientific and Practical Conference from Programming UkrPROGP'2022, October 11-12, 2022, Kyiv, Ukraine
EMAIL: viktor.gorbatiuk@gmail.com (A. 1); gorbatiuk_sergiy@i.ua (A. 2)
ORCID: 0000-0001-7544-0260 (A. 1); 0000-0001-6834-4211 (A. 2)

In fact, our society is more technologically dependent than ever before, and there is no sign of this trend slowing down. That is why the importance of cyber security is growing. Data leaks in smart home systems that have a high level of integration with social networks can lead to identity theft. Sensitive information such as social security numbers, credit card details and bank account details are now stored in the cloud storage services Dropbox or Google Drive.

2. Overview of attacks in smart homes

Whether a home has a full smart home system or just a set of smart devices, we must evaluate security as the total sum of the security of each device. 40.8% of smart homes have at least one device vulnerable to cybersecurity threats. At the same time, 31.4 percent are at risk due to unpatched software vulnerabilities. The only way to protect yourself from the potential threat is to pay more attention to the smart home devices that are installed, and attacks on such devices are not much different from conventional network attacks.

Methods of network attacks are classified as "passive" and "active". Passive attacks are the interception of data on the way to the recipient. Active attacks are a network attack in which a hacker tries to make changes to data on the target object or data en route to the target. They are divided into "forgery", "change of message" and "denial of service". For a more detailed explanation, we consider a simple list of three categories [7].

Reconnaissance attacks are attacks to gather general information. Snooping (also known as "tracking" or "gathering information") is simply access to private information. This information can be used to advantage, for example, to obtain company secrets that will help in your own business or in making decisions about buying shares. It can also be used for active attacks such as blackmail. These attacks can be carried out through both logical and physical approaches, information is collected through network scanning or through social engineering and physical surveillance. Some common examples of reconnaissance attacks include packet sniffing, pinging, port scanning, phishing, social engineering, and Internet information requests. We can consider them further by dividing them into two categories, logical and physical. [13]

Logical reconnaissance includes everything done in the digital world and does not require human action on the other side to complete an reconnaissance attack. For example, ping scans and port scans are two methods of detecting whether a system is connected and what it is looking for on the network. The answer from a port scan might be to detect if an IP address is listening on port 443 for HTTPS traffic. This lets the hacker know if they can use HTTPS for their purposes.

Network man-in-the-middle (MITM) attacks occur when malicious parties intercept traffic passing between networks and external data sources or within the network. In most cases, hackers achieve man-in-the-middle attacks by using weak security protocols. They allow hackers to pose as a relay or proxy account and manipulate data during real-time transactions.

Unverified user data can put organizational networks at risk of SQL injection attacks and the injection of malicious SQL code. In this network attack method, external parties manipulate forms by sending malicious codes instead of the expected data values. They compromise the network and gain access to sensitive data such as user passwords [12].

Physical reconnaissance goes beyond what a network administrator can control. There are elements that will never be fully secured, such as places and security elements such as cameras, door locks or security guards. However, this may affect the physical security of the network.

Access attacks require some intrusion capability. These can include anything as simple as obtaining the account holder's credentials to connect the equipment directly to the network infrastructure. Often these access attacks can be compared to reconnaissance as logical or physical, logical through the network and physical which leans more towards social engineering.

Logical access attacks, such as brute-force attacks or validating network passwords using tables or dictionaries, tend to generate a lot of network traffic and can be easily detected by even a non-experienced network monitor. It is for this reason that most logical access attacks are usually carried out after enough data or authority has been obtained. There is also a tendency to resort to the passive side of the attack, like a man-in-the-middle attack, to try to gather more information.

There is also such a group of attackers as ransomware. As a result of the attack, attackers encrypt data access channels while holding the decryption keys - a model that allows hackers to extort money from affected organizations. Payment channels usually include untraceable cryptocurrency accounts. While cyber security agencies don't prevent criminals from paying, some organizations continue to do so as a quick fix to recover access to data.

When talking about data modification attacks, most people think of an attacker changing the content of emails to be malicious or changing the numbers in an electronic bank transfer. While such high-level data modification attacks are possible, there are more subtle ways to modify data. For example, if you could intercept a wireless transmission and change the address (IP address) field of a message, this could cause the message to be forwarded over the Internet to you instead of to the recipient. Why is this done? Since the message in the link is encrypted and you cannot read the content, if you can transmit it over the Internet, you will receive a decrypted version. The IP header is easier to attack because it is in a known format.

Masquerading is the term when an attacking network device pretends to be a valid device. This is an ideal approach if the attacker wants to remain undetected. If the device can successfully fool the target network into verifying it as an authorized device, the attacker gains all the access rights that the authorized device set during login. Also, there will be no security warnings.

Physical access is access to equipment or access to people. Social engineering is very dangerous and difficult to defend against simply because users are usually the weakest link in cyber security. The simplest type of social engineering attack is sending phishing emails designed to trick someone in this way, or installing credential-logging programs on a person's computer with access. Even cyber security professionals can be vulnerable to such attacks simply because they live among the humans that they are and we are not perfect and make mistakes.

Denial of Service (DoS) is very different from the other categories in both technique and purpose. While others give the attacker additional privileges, a DoS attack usually blocks everyone, including the attacker. The goal of a DoS attack is to harm the target by preventing the network from functioning. Downtime means that the network cannot receive any traffic. This can happen due to a power failure or the network being flooded with unnecessary traffic that prevents the network from functioning. Both have historically occurred without any malicious intent, and both can be prevented with physical and logical blockers.

3. Attacks on smart houses based on vulnerability of HTTP protocol

5.1. HTTP request smuggling

HTTP request smuggling (HRS) refers to techniques for interfering with the normal processing of a sequence of HTTP requests by a website.

Most modern web applications use HTTP proxy servers between application logic and users. Users send requests to a "back-end server", also commonly referred to as a reverse proxy or load balancer. This server forwards user requests to one or more internal servers that execute application logic. It is because of this communication structure, when the upstream and internal servers do not agree on the details of the HTTP protocol, that the HRS attack becomes possible.

Request smuggling vulnerabilities are considered critical because they allow attackers to bypass security measures. The attacker then gains unauthorized access to sensitive information and directly compromises other users.

According to the HTTP protocol specification, there are two methods of indicating the end of an HTTP request, which are used as a vulnerability:

- Using the Transfer-Encoding header: chunked - sends the request body in chunks separated by a newline character. As a rule, the fragment ends with 0.
- Using the Content-Length header - defines the size of the request body in bytes. Attackers use both headers in the same request, hiding the second request in the body of the first request. In this way, the second request is "smuggled". There are three main ways to exploit this vulnerability:

- CL-TE: the external server uses the Content-Length header and the internal server uses the Transfer-Encoding header. The external server processes the Content-Length header and determines that this request is of a length in bytes that will include a smuggled message. This request is forwarded to the internal server, which processes the Transfer-Encoding header and treats the message body as chunked encoding. It processes the first chunk, which is said to be of zero length, and is therefore treated as the end of the request. The following bytes of the smuggled message remain unprocessed and the backend server will treat them as the start of the next request in the sequence.
- TE-CL: the external server uses the Transfer-Encoding header and the internal server uses the Content-Length header. The external server processes the Transfer-Encoding header and treats the message body as chunked encoding. It processes the first fragment that has a length of bytes that will include the smuggled message before the beginning of the new line. It then processes the second chunk, which is of zero length and is therefore considered to be the end of the request. This request is forwarded to the backend server, which processes the Content-Length header and determines that the request body has a number of bytes that will not include a smuggled message. Subsequent bytes remain raw and will be treated by the backend server as the start of the next request in the sequence.
- TE-TE: both the external and internal servers support the Transfer-Encoding header, but one of the servers can be forced not to handle it by processing the header in some way.

5.2. Attacks with HTTP host headers

Such attacks are carried out against vulnerable websites that manipulate the host header value in an unsafe manner. If the server implicitly trusts the Host header and is unable to properly validate or avoid it, an attacker can use this approach to inject malicious data that manipulates server-side behavior. Attacks that involve injecting data directly into the host header are often known as "host header injection" attacks.

Since the Host header is actually controlled by the user, this practice can lead to a number of problems. If the input is not properly shielded or validated, the Host header is a potential vector for exploiting a number of other vulnerabilities.

An attacker cannot make the victim's browser display the wrong host in a useful way. However, if the target server uses a web cache, it is possible to turn this harmless vulnerability into a dangerous one that persists by convincing the cache to provide a poisoned response to other users. To create a web cache poisoning attack, you need to get a response from the server that reflects the data that was sent. The main goal is to make an attack by storing a cache key that will be matched against other users' requests. If successful, the next step is to cache this malicious response. As a result, this cached response will be sent to all users who try to visit the corresponding page. Stand-alone caches usually include a Host header in the cache key, so this approach usually works best on integrated application-level caches. However, the methods discussed earlier can sometimes allow you to poison even offline web caches.

Password reset poisoning is a technique in which an attacker manipulates a vulnerable website to create a password reset link that points to a domain they control. This behavior can be used to steal the secret tokens needed to reset arbitrary users' passwords and ultimately compromise their accounts.

5.3. Slow HTTP-attack

Such an attack is one of the types of DDoS attacks. A slow HTTP attack exploits the working mechanisms of the HTTP protocol, in which the protocol requires that each request from a client be fully accepted by the server before it can be processed. If the HTTP request is incomplete, or if the transfer rate is very slow, the server remains busy waiting for the rest of the data. If the server keeps too many busy resources, it creates downtime.

Slow HTTP attack is a software-level DoS attack in which a large number of incomplete HTTP requests are sent. These attacks are more difficult to trace than classic Dos attacks because:

- These attacks do not consume huge amounts of bandwidth.
- They aim to create a resource constraint within the program by focusing on the weakest link in the program.
- These attacks usually use the https protocol as a transport to hide their true origin.

Slow HTTP attacks are mainly of three types:

1. Slow headers (or Slowloris): In a slow header attack, the attacker triggers the action using a tool called Slowloris or similar. This tool opens a connection, then sends HTTP headers to support the process, but never completes the request. Thousands of HTTP POST connections are made, sending HTTP headers very slowly, forcing the target web server to keep the connection open. Slowloris takes all the resources of the target web server in this way only, blocking requests from legitimate clients.
2. Slow Body (aka R-U-Dead-Yet): The slow body attack works the same as the slow header attack. An attacker using a tool called R-U-Dead-Yet or similar sends a POST request body that does not expire. The attack phase begins by making an initial TCP connection to the target web server. It then sends the HTTP POST header first, just like a normal connection would. The header contains information about the size of the body of the data packet that will be sent next. The attacker sends the message body at a very low speed. The connection remains open, causing the victim's web server to wait for a long time. New similar connections are created in large numbers, using all server resources until the creation of legitimate connections becomes impossible.
3. Slow Read: In a slow read attack, attackers send valid TCP-SYN packets to open a connection to the target server. Next, it starts requesting the document from the target server. After the download starts, the attacker starts to slow down the reading of the received packets. This attack will continue and take all the resources of the target server. Slow-read attacks do not require forging requests to keep a session open for a long time.

As described above, the attacks are performed using a software script that is capable of transmitting a partial packet data request, keeping multiple open connections to the victim's web server. The following HTTP header is periodically sent to support the attack, but intentionally never completes. The attack forces the victim's web server to provide all of its resources to the attacker, ultimately causing legitimate users to be denied connections. An attacker doesn't need massive bandwidth to shut down a victim's web server, just a large number of connections.

Each message line ends with the CRLF character. CRLF stands for CR (carriage return) and LF (line feed). CRLF is a non-printable character. The request message ends with an empty line, the last line having two CRLF characters. The two together are used to indicate an empty line.

In a slow HTTP attack, there will never be an empty string. The attacker intentionally does not send the end-of-request CRLF character. The following request message will result in a slow HTTP attack due to having only one CRLF tag at the end, which means that the header is incomplete and the server must wait for the header to complete.

5.4. HTTP request splitting

Similar to a request smuggling attack, this attack is made possible by the presence of intermediate components, which may include: a load balancer, a reverse proxy, a web caching proxy, an application firewall, a web browser, and so on. Regardless of role, they must maintain consistent, consistent HTTP communication state across components. However, including unexpected data in the HTTP header allows an attacker to craft a complete HTTP message that is displayed by a client web browser or an internal web server, regardless of whether the message is part of a request or a response.

When an HTTP request contains unexpected CR and LF characters, the server may respond with an output stream that is interpreted as "splitting" the stream into two different HTTP messages instead of one. CR is a carriage return, also represented by %0d or \r, and LF is a line feed, also represented by %0a or \n.

In addition to CR and LF, other valid/RFC-compliant special characters and unique character encodings may be used, such as HT (horizontal tab, also represented by %09 or \t) and SP (space, also represented by + or %20).

These types of unverified and unexpected data in HTTP message headers allow an attacker to control a second "detached" message to create attacks such as server-side request forgery, cross-site scripting, and cache poisoning attacks.

4. Overview of attack detection tools

An analysis of the current state of technology for solving information protection problems is carried out, and we will analyze the tools used to prevent attacks and identify vulnerabilities related to cyber security in smart homes.

4.1. Tools for detecting DDoS attacks

Fastnetmon is a common open source package that offers a service running on a Linux server [6]. It is a very high-performance DDoS detector built on several packet capture mechanisms [8]. It supports a number of capture mechanisms such as port mirroring, NetFlow, sFLOW, IPFIX, etc. to feed it information about incoming traffic. It can detect an attack on specific IP addresses on the network based on bandwidth, number of packets per second or number of flows. You can define and configure these parameters based on the attack profile. The next part is to tell the router to drop malicious traffic and the appropriate BGP blackhole or BGP flow rules to mark that particular route. Fastnetmon offers options to determine how long an IP address remains blocked and when it can be allowed again. It has reliable support for all leading network providers and has unlimited scalability thanks to its flexible design. You can integrate FastNetMon into any existing network without any changes or additional equipment!

A framework called HADEC is designed to detect live high-speed DDoS attacks that occur at the network and application layers, such as TCP-SYN, HTTP GET, UDP, and ICMP [23]. The framework consists of two main components: a discovery server and a capture server. Real-time DDoS detection begins with a capture server responsible for capturing real network traffic and passing the log to the detection server for processing. Detection evaluates the incoming packet for UDP, ICMP, and HTTP to detect an attack if the outgoing connection exceeds a predefined threshold. The proposed detector provides low-cost solutions for financial institutions, as well as small and medium-sized companies [4].

A detection method called D-FACE is used to detect four types of traffic: legitimate user, low-speed, high-speed, and flash traffic [22]. The detection uses the entropy difference that contains the normal traffic flow, while the entropy value of the source IP is the detection matrix to calculate the attack. Discovery begins by extracting the appropriate header that classifies the network into a unique network flow. The separation of low traffic, high traffic and flash event traffic is based on the comparison of the current speed of the incoming traffic in each time window and on the basis of the information traffic value.

There is also a method that detects an HTTP DDoS attack using a machine learning approach to distinguish botnet from legitimate users in detecting attack traffic, authentic traffic, and flash traffic [16]. The proposed system is deployed as a proxy and checks user behavior instead of monitoring all traffic. The proposed work detects the source of a botnet and examines user behavior to detect a malicious request to a web server.

The matrix of machine learning with the biological algorithm of bats allows for quick and early detection of HTTP DDoS attacks [15]. The work involved time slots instead of user sessions and packet patterns to create a detection algorithm. Timeslot uses a machine learning matrix to assign a maximum number of sessions to a single time slot and calculate the number of sessions per time slot to detect a DDoS attack at the application layers. The matrix also accounts for the two HTTP GET request pages. The frequency with which users access a web page and the time interval between the request of the first page and the second page are determined to monitor user behavior.

Another tool is cloud-based HTTP DDoS detection using a statistical approach with a covariance matrix [1]. The detection implemented two algorithms, known as training and testing, to recognize different types of HTTP attacks based on attack behavior. A training algorithm was used to construct common patterns of network traffic, and a testing algorithm was used to determine the types of traffic

received. The results obtained from this study were evaluated using a confusion matrix to measure the performance of detecting and delivering results of indoor and outdoor cloud environments.

4.2. Server-based intrusion detection tools

WWWstat [3] is primarily a program for collecting web server usage statistics. This program does not perform intrusion detection by itself, but its output can be used for manual intrusion detection by checking abnormal usage statistics.

Autobuse [24] is a framework for analyzing firewall log files and web server logs. It analyzes log entries for known attacks and reports them through several mechanisms, such as email.

Logscanner [25] is a framework for analyzing log files that can include functions. It automatically contacts the person responsible if needed and feeds logs to user-designed functions.

Swatch [5] analyzes UNIX syslog files in the same way as other tools, grouping similar entries to automate processing.

CyberCop Server [17] is a commercial intrusion detection tool formerly known as WebStalker. This tool includes features to monitor web server activity based on policies defined by the server operator, but does not provide log file analysis.

Snort is an open source network intrusion prevention and detection system (IDS/IPS) developed by Sourcefire. Snort can perform protocol analysis and content search/matching. It can be used to detect various attacks and probes such as buffer overflows, hidden port scans, CGI attacks, SMB probes, OS identity attempts, and more. Snort also has real-time alerting capabilities, including alerting mechanisms for the syslog, a user-defined file, a UNIX socket, or WinPopup messages for Windows clients. Snort has three main uses: a direct packet analyzer such as tcpdump, a packet logger (useful for debugging network traffic, etc.), or a full-fledged network intrusion prevention system [11].

Fail2Ban scans log files such as /var/log/auth.log and bans IP addresses that have too many failed login attempts. This is done by updating the system firewall rules to reject new connections from these IP addresses for a certain period of time. Fail2Ban is ready to read many standard log files, such as for sshd and Apache, and can easily be configured to read any log file you choose for any selected error. While Fail2Ban is able to reduce the frequency of incorrect authentication attempts, it cannot eliminate the risk that weak authentication presents. Configure services to use only two-factor or public/private authentication mechanisms if you really want to secure services [9].

FuzzDB was created to increase interest in the probability of occurrence and detection of security conditions through dynamic application security testing. This is the first and most comprehensive open dictionary of malicious injection patterns, predictable resource locations, and regular expressions for matching server responses. Attack Patterns - FuzzDB contains comprehensive lists of useful attack bootstrap primitives for testing malicious injections. These patterns, categorized by attack type and, if applicable, by known platform type, in which issues such as OS command injection, directory listings, directory traversal, source access, file download traversal, authentication traversal, XSS, etc. http header crlf injections, SQL injection, NoSQL injection, and others. As an example, FuzzDB catalogs 56 patterns that can potentially be interpreted as a null byte, and contains lists of commonly used methods such as "get, put, test" and name-value pairs, and then initiates debug modes [10].

Using these template tools it is possible to avoid HTTP Request Smuggling:

1. To disallow requests with headers TE, CL and double CL. It helps to avoid attacks CL:CL, as well as TE:CL and CL:TE. Any request that includes both headers will receive HTTP 400 answer. HTTP-requests with couple of CL headers with value of different length may be also corrected with HTTP 400 answer.
2. To disallow wrong TE headers. To avoid TE:TE attacks and support correct processing of couple of TE values, some types of headers have to be denied:

- Headers without space in front of value, «chunked»
- Headers with extra spaces
- Headers with ending message symbols in the beginning
- Headers that include «chunk» value instead of «chunked» (server will normalize it as chunked coding)
- Headers with couple of spaces in front of «chunked» value

- Headers with single or double quoted values
- Headers with CRLF symbols in front of values, «fragments»
- Headers with invalid symbols

To detect and prevent exploitation of known and common vulnerabilities, the OWASP organization has defined a common set of rules known as the OWASP Core Rule Set [18] (OWASP CRS). OWASP CRS is widely used by large organizations such as Akamai, Azure, CloudFare, Fastly, and Verizon. The task of OWASP CRS is to provide a set of general attack detection rules that, when passed to the MODSECURITY web application firewall, provide a basic level of protection for any web application. OWASP CRS implements a negative model where rules are designed to detect known attack patterns [21].

There are also so-called network intrusion detection tools. Such systems detect intrusions by intercepting packets from the network and applying a set of signatures. Examples of this family of tools include Network Flight Recorder [20], Bro [19], RealSecure [14] or NetRanger [2].

5. Application of the algebraic method for detecting and resisting attacks

With the growth of hacker tricks and the complexity of attacks, the basic, common methods and tools that were used to protect traditional information technologies from cyber attacks eventually become unable to completely prevent the successful penetration of malicious programs into the system. Therefore, new approaches are needed. Although the systems are protected by IT security tools, attackers still find a way to gain unauthorized access and compromise them through cyber attacks. These cyberattacks must be detected as quickly as possible with an acceptable false alarm rate, and must be identified and isolated. Thus, there is an urgent need for an effective cyber attack detection system as an integral part of cyber infrastructure that can accurately detect cyber attacks in a timely manner so that countermeasures can be quickly taken to ensure the availability, integrity and privacy of systems. The new challenges that arise in security requirements challenge traditional mathematical tools. Therefore, it is recommended to use an algebraic approach to solve big data problems and other artificial intelligence approaches such as machine learning.

In V.M. Glushkov Institute of Cybernetics, among the methods of mathematical modeling, insertion modeling is widely used, which, with the help of algebraic methods, makes it possible to ensure safety and security. Algebraic models are also used to analyze the behavior of all involved agents in order to check their influence and ability to perform their task as well as the ability of the entire system to function stably and smoothly.

For the formal description of the model, the specifications of the algebra of behaviors are used, and the formal methods of verification are based on the methods of symbolic modeling and automatic theorem proving. In systems with an arbitrary number of agents, the algebraic approach and insertion modeling allow us to prove or disprove the properties of such a system. We can generate different interaction scenarios of agents or groups of agents using an abstract formal application model. Such scenarios have a symbolic form and are illustrated by counterexamples. The generated symbolic scripts provide a complete picture of the behavior, and as a result can be used for testing during the build phase of the software.

For complex distributed systems with many agents, insertion modeling is one of the effective methods for building models and simulating the interaction of agents with the environment. The main concept of inertial modeling is the creation of a clear hierarchy from the environment to the agents included in these environments, the interaction of agents with environments of different levels, their mutual influence on each other, and changes in the behavior of a group of agents when the environment changes. The environment can act as an agent, which can also be immersed in another environment. In such systems, states are defined by attribute values, and agents are viewed as attribute transition systems. Agents are described by a set of attributes that define the type of agent, and environment attributes are associated with global attributes that are known to all agents.

The algebra of behaviors is a two-sort algebra over the set of behaviors and actions of agents. Behavior is described with the help of behavioral equations consisting of behavioral expressions, which in turn contain operations - "." (prefixing), “+” (indeterminate choice), “;” (sequential

composition of behaviors), "||" (parallel composition). Actions of agents are determined using preconditions and postconditions in terms of the corresponding theory and illustrated by the process component. An example of protocol formalization and attack is shown below.

5.1. Formalization of HTTP protocol and simple attack

The most common attacks on smart homes are attacks that, if successful, give complete control over the network of devices. Considering the simplicity and high probability of not being noticed, we can emphasize the "man-in-the-middle" attack among a number of others. We will take it for analysis.

«Man-in-the-middle» attack (MITM) is a general term meaning that attacker acts as an intermediary in the transmission of data between the user and the program — either by eavesdropping or impersonating one of the parties, imitating a normal exchange of information.

This allows the attacker to intercept information and data from any party and send malicious links or other information to all nodes in the network in a stealthy manner that is very difficult to detect. The information obtained during the attack can be used not only to control the devices, but also include stealing the personal data of all users, unauthorized transfers of funds or changing passwords. Additionally, such an attack can be used to gain a foothold inside a protected network during the infiltration stage of an Advanced Persistent Threat (APT).

ARP spoofing is the process of associating an attacker's MAC address with the IP address of a legitimate user on a local network using forged ARP messages, which is one approach in man-in-the-middle attacks. As a result, the data sent by the user to the IP address of the host is transmitted to the attacker, which is very convenient in centralized systems where all control rests on the controller (hub) of the smart house.

Address Resolution Protocol (ARP) is a protocol that allows you to find a specific device on a network. ARP converts Internet Protocol (IP) addresses to Media Access Control (MAC) addresses and vice versa. Most often, devices use ARP to communicate with a router or gateway that allows them to connect to the Internet. Hosts maintain the ARP cache, a table of comparisons between IP addresses and MAC addresses, and use them to connect to a destination on the network. If a host does not know the MAC address for a particular IP address, it sends an ARP request packet asking other machines on the network for the corresponding MAC address. The ARP protocol was not designed with security in mind, and not only does it not verify that the response to an ARP request actually comes from an authorized party, it also allows hosts to receive ARP responses even if they never sent the request. This is a weakness in the ARP protocol that makes an ARP spoofing attack possible.

ARP spoofing has a clear sequence of actions of the attacker, and includes the following steps:

- The attacker scans the target network for the IP and MAC addresses of the hosts. This step is not directly related to this attack, but is necessary in preparation for the attack.
- Selects its target, i.e. the target IP and MAC address. As a target, it is advisable to choose a router or smart home controller through which all traffic passes.
- Creates structurally correct responses to the ARP protocol request. The response packet to all network nodes contains the attacker's MAC address and the target's IP address. Multiple responses are generated for the target with the attacker's MAC address and the IP address of every other node.
- Sends generated ARP responses over the local network to the appropriate addressees.
- Each of the nodes on the local network, including the target, receives and caches ARP packets.
- Data from all network nodes to the target and vice versa now reaches the attacker.
- To preserve his anonymity and system performance, the criminal must forward the received data to their original addressees.

We consider the HTTP protocol as the interaction of agents in network environments. Each agent has an IP name and is defined by an enum type IP_NAME containing all possible IP names. The value of the attribute is a character string, for example: 192.168.1.1. Accordingly, each agent has a

network address, which is also determined by a set of enumerated type MAC_NAME. The value of the attribute is also symbolic, for example: 00:00:5e:00:53:af.

We consider the agent type NODE, which is defined by its attributes, namely:

- IP:IP_NAME – IP address of the agent.
- list_IP: (int) -> IP_NAME – the functional attribute of the agent, containing the IP of agents from the table in which the addresses of all agents to which a message was once sent or received are recorded.
- M – the number of rows in the table, or the number of addresses contacted by the agent.
- MAC:MAC_NAME – MAC address of the agent.
- list_MAC: (int) -> MAC_NAME – MAC addresses of all agents in the table

Each NODE agent has a name – a1,a2,…

An environmental agent can be defined as either honest or criminal. When interacting, agents perform actions corresponding to the message exchange protocol. Such are the following actions, which are parameterized by the corresponding values of the attributes.

The SendRequest(x, z) action sends a Request message, where x is the sender agent, z is the IP address of the recipient with the corresponding MAC address. The record is sent only to those recipients who are in the list, that is, there is a prerequisite for the action, the Request(x.IP, u) message is sent to the MAC address u if such an IP exists in the sender's table.

SendRequest(x, z) = (Exist i:int)(z == x.list_IP(i) && (1 <= i <= x.M))-> “send Request(x.IP, list_MAC(i))” 1

GetRequest, the agent receives a request Request(y, u), where y is the sender's agent IP, u is the recipient's MAC address. This action also assumes that the Request is received only by the agent whose MAC address matches the second parameter of the notification. In the same action, the corresponding agent sends a response to the request - Response to the MAC address that it found in its list according to the sender's IP.

GetRequest = (Exist x:NODE, y:IP_NAME, u:MAC_NAME,i:int) (y == x.list_IP(i)) && (1 <= i <= x.M) && (u == x.MAC) -> “receive Request (y,u), send Response(x.IP, x.list_MAC(i))” 1

Similarly, we define the protocol for the sender receiving a response to the request, namely the Response notification.

GetResponse = (Exist x:NODE, u:MAC_NAME, y:IP_NAME) (u == x.MAC) -> “receive Response (y,u)” 1

Action NoSendRequest(x,z) is an action in which z is not in the address list of agent x and the notification is not sent.

NoSendRequest(x, z) = (Forall i:int)(z != x.list_IP(i) && (1 <= i <= x.M))-> “” 1

In case the address is not in the agent's list, it queries all agents in the network to identify the desired one and sends its address

SendARPRequest(x,z) = (Forall y:NODE) -> “send Broadcast(x.IP, x.MAC, z)” 1

Receiving a Broadcast message by agent x, which has received a request for its address, occurs using the GetARPRequest action. In the same action, the agent sends a message about its MAC address to the sender's address.

GetARPRequest = (Exist x:NODE, y:IP_NAME, u:MAC_NAME, z:IP_NAME) (z == x.IP) -> ”receive Broadcast(y, u, z), send ARPResponse (x.IP, x.MAC, u)” 1

The agent that searched for the address receives the ARPResponse and adds it to its list.

GetARPResponseExist = Exist(x:NODE, y:IP_NAME, z:MAC_NAME, u:MAC_NAME, i:int) (x.MAC == u) && (x.list_IP(i) = y) && (1<=i<=x.M) -> “receive ARPRequest(y,z,u)” (list_MAC(i) = z)

GetARPResponseNew = Exist(x:NODE, y:IP_NAME, z:MAC_NAME, u:MAC_NAME) (Forall(i:int) (x.list_IP(i) != y) && (1<=i<=x.M)) && (x.MAC == u) -> “receive ARPRequest(y,z,u)” (x.M = x.M + 1; x.list_IP(M + 1) = y; list_MAC(M + 1) = z)

The behavioral equation representing this protocol will be the following parallel composition of agents:

B0 = B1(a1,a2) || B1(a1,a3) || … || B1(a2,a1) || (a2,a3) || … ,

B1(x,z) = AgentRequest(x,z).B1,

AgentRequest1(x,z) = (SendRequest(x, z.IP).GetRequest.SendResponse.GetResponse + NoSendRequest(x,z.IP).SendARPRequest(x, z.MAC).GetARPRequest.SendARPResponse.(GetARPResponseNew + GetARPResponseExist)

The equation does not take into account the loss of signal and the absence of a node with the requested IP.

A malicious agent can take advantage of the opportunity to send false data and pretend that its MAC address matches the IP name we are requesting. This is done in order to intercept notifications sent to this agent.

In this way, we remove the prerequisite in the GetARPRequest action z == x.IP,

GetARPRequest = (Exist x:NODE, y:IP_NAME, u:MAC_NAME, z:IP_NAME) -> ”receive Broadcast(y, u, z), send ARPResponse (x.IP, x.MAC, u)” 1

Then the condition that the agent is an intruder is determined by the inequality when executing the GetARPRequest protocol. Thus, we can record the behavior of the attacker with the following pattern:

X = Z. GetARPRequest, where the rule violation condition will be written in the action template (z != x.IP) -> ”” 1

We have given the simplest formalization of the protocol to illustrate the possibility of an attack. The entire protocol is a behavioral equation to be solved with respect to X using symbolic modeling. In this way, we determine that the result of the attack is achievable and we will get a path leading to this result, namely a sequence of appropriate actions. In this way, we will determine whether the protocol prevents this attack or not. To prevent this attack, you need to insert a check. There are three abnormal attacks that can be used as a test:

1. In an attack, the response is sent without a request, so it is necessary to check whether a request was sent

GetARPResponseExist = SendARPRequest -> …
GetARPResponseNew = SendARPRequest -> …

2. During the attack, it is not checked whether the sender sends his address in such response, so you need to compare the provided address with the sender's address

GetARPResponseExist = (y.MAC == z) -> …
GetARPResponseNew = (y.MAC == z) -> …

3. The address book should not contain two identical addresses

GetARPResponseExist = Forall(i:int) (x.list_MAC(i) != x.list_MAC(i+1)) -> …
GetARPResponseNew = Forall(i:int) (x.list_MAC(i) != x.list_MAC(i+1)) -> …

Then the behavioral equation will have no solution.

5.2. Detection of attacks by algebraic matching

Algebraic matching is a method of identifying potential vulnerabilities in a code or system model by comparing the behavior model of such a system with an attack pattern. The method is based on dynamic analysis of behavior by solving behavioral equations.

The model is given by the system of equations in the algebra of behaviors, and the attack is given by the pattern of behavior. At the same time, it is necessary to find a set of behavioral scenarios in a given system of behavioral equations that correspond to a given template or lead to it from the initial behavior.

This task can be divided into two subtasks:

1. To find a sequence of actions corresponding to a given pattern, which is reduced to solving behavioral equations, the solution of which is a set of behavioral scenarios corresponding to the pattern or a set of behavioral scenarios starting with the initial action of the initial behavior and leading to the behavior of the template in a sequence of other actions.

2. Proving the reachability of a scenario using symbolic modeling in cases where there are no attributes that make such a scenario possible. In this simulation, the simulation environment is compared with the premise of the action in the template.

When designing any system, it is recommended and even necessary to conduct simulations of all possible attacks in order to understand their probability. When an attacker tries to attack the network,

the security mechanism can recognize potentially dangerous actions during operation, but it is possible to assess under what conditions the attack will be successful only during model development.

6. Conclusion

Thus, we have reviewed most of the currently known attacks on smart homes, as well as tools for their detection. A large number of such tools are widely used on a commercial scale and have proven themselves quite well, showing high efficiency. But despite this, there are situations when security measures are better implemented at the system design stage, so the search and use of new approaches remains relevant.

One of these approaches is algebraic methods of mathematical modeling. We applied the Algebra of Behavior method to simulate a "man-in-the-middle" attack in a smart home network and verified the possibility of using it to simulate network attacks. It can be effective both for modeling attacks and the network as a whole, which allows you to detect problems that were not even foreseen. We plan to consider and model other attacks in order to prove the feasibility of the method and its practical effectiveness.

7. References

[1] Aborujilah and S. Musa, “Cloud-based DDoS HTTP attack detection using covariance matrix approach,” Journal of Computer Networks and Communications, vol. 2017, Article ID 7674594, (2017) 1-8.

[2] Cisco Systems Inc. NetRanger – Enterprise-scale, Real-time, Network Intrusion Detection System, 1998. URL: http://www.cisco.com/warp/public/751/netranger/netra_ds.htm

[3] R. Fielding, wwwstat: Httpd logfile analysis software, November 1996. URL: http://www.ics.uci.edu/pub/websoft/wwwstat/

[4] Ghafar A. Jaafar, Shahidan M. Abdullah, Saifuladli Ismail “Review of Recent Detection Methods for HTTP DDoS Attack” Journal of Computer Networks and Communications, 2019.

[5] S. E. Hansen, E. T. Atkins, Automated system monitoring and notification with swatch. In Proceedings of the seventh Systems Administration Conference, LISA ’93, 1993.

[6] Anurag Bhatia, Ultra fast automated DDoS detection & mitigation, 2017. URL: https://anuragbhatia.com/2017/10/networking/isp-column/ultra-fast-automated-ddos-detection-mitigation/

[7] Classification of Attacks. URL: http://etutorials.org/Networking/802.11+security.+wi-fi+protected+access+and+802.11i/Part+I+What+Everyone+Should+Know/Chapter+4.+Different+Types+of+Attack/Classification+of+Attacks/

[8] Fastnetmon. URL: https://fastnetmon.com/

[9] Findbestopensource. Fail2Ban. URL: https://www.findbestopensource.com/product/fail2ban-fail2ban

[10] Findbestopensource. FuzzDB. URL: https://www.findbestopensource.com/product/fuzzdb-project-fuzzdb

[11] Findbestopensource. Snort. URL: https://www.findbestopensource.com/product/snort

[12] Cyber EDU, Network attacks and network security threats explained. URL: https://www.forcepoint.com/cyber-edu/network-attack

[13] TripWire, 3 Types of Network Attacks to Watch Out For, 2022. URL: https://www.tripwire.com/state-of-security/vulnerability-management/3-types-of-network-attacks/

[14] Internet Security Systems, Inc. RealSecure, 1997. URL: http://www.iss.net/prod/rsds.html

[15] I. Sreeram and V. P. K. Vuppala, “HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm,” Applied Computing and Informatics, 2017

[16] K. Singh, P. Singh, and K. Kumar, “User behavior analytics-based classification of application layer HTTP-GET flood attacks,” Journal of Network and Computer Applications, vol. 112, (2018) 97–114.

[17] Network Associates Inc. Cybercop server. 1998. URL: http://www.nai.com/products/security/cybercopsvr/index.asp
[18] OWASP. Owasp modsecurity core rule set project. URL: https://www.owasp.org/index.php/
[19] V. Paxson, Bro: A system for detecting network intruders in real-time. In Proceedings of the 7th USENIX Security Symposium, 1998.
[20] M. J. Ranum, K. Landfield, M. Stolarchuk, M. Sienkiewicz, A. Lambeth, E. Wall, Implementing a generalized tool for network monitoring. In Proceedings of the Eleventh Systems Administration Conference LISA 97, 1997.
[21] Rodrigo Martınez, Enhancing web application attack detection using machine learning, Instituto de Computaci´on, Facultad de Ingenier´ıa Universidad de la Rep´ublica, Uruguay
[22] S. Behal, K. Kumar, and M. Sachdeva, "D-FACE: an anomaly based distributed approach for early detection of DDoS attacks and flash events," Journal of Network and Computer Applications, vol. 111, (2018) 49–63.
[23] S. Hameed and U. Ali, "HADEC: hadoop-based live DDoS detection framework," EURASIP Journal on Information Security, vol. 2018, (2018) 1-11.
[24] G. Taylor, Autobuse. Internet, 1998. URL: http://www.picante.com/gtaylor/autobuse/
[25] C. Tuininga, and R. Holak, Logscanner. 1998. URL: http://logscanner.tradeservices.com/index.html