SPYKE: Security ProxY with Knowledge-based intrusion prEvention

Sheng Wang
Instituto Superior Técnico, Universidade de Lisboa, Portugal
sheng.wang@tecnico.ulisboa.pt

ABSTRACT

In the near future, the Internet of Things (IoT) will be a reality and there will be many sensors in our smart homes, for example. These data sources will eventually upload data to the cloud. In this work we present SPYKE (Security ProxY with Knowledge-based intrusion prEvention), a network intermediary that stands between IoT devices and the Internet, that provides visibility to which communications are taking place between devices and remote servers; and it also has the ability block and limit connections. We evaluated SPYKE with respect to the performance and security. It has low performance overhead and is effective against a set of important attacks. SPYKE is available as an open-source project and is deployable in inexpensive, off-the-shelf hardware like the Raspberry Pi.

KEYWORDS

Privacy Protection; Intrusion Detection System; Internet of Things; Spyware

1 INTRODUCTION

In line with the Internet of Things (IoT) trend, many consumers are transforming their homes into smart homes. As a consequence, data from sensors situated into devices are increasing in numbers. These devices are connected to the Internet, so these data are usually sent to remote cloud servers for data mining purposes.

Companies have been proposing different smart home systems, namely: *Amazon Echo*[1], *Google Home*[2], *SmartThings*[3], among others. These systems use a *hub* as an intermediary device to control the smart home and provide a good user experience. However, there are significant weaknesses that can be used to expose private data. For example, a recent report[4] has shown that the Amazon Echo can be infected after the user activates a malicious *skill*[5], and, as a result it continues sending empty requests that maintain the connection established. This way, it is effectively turned into a *spy*, that sends voice recordings to an attacker.

The majority of systems already use data encryption to hide the data content. Even so, they cannot hide their identities, because the destination IP address and the source IP address are needed to make the communication. Data leakage is very difficult to avoid, but it is possible to have an *intermediary*, who can inform the user of the communications that are happening as it is intercepting them.

To provide added privacy protection we propose SPYKE – Security ProxY with Knowledge-based intrusion prEvention – an intermediate network device between the user (and his devices) and the cloud (service providers). Our system is situated on the user's side in order to intercept the data before it is sent to the Internet.

In this article, we evaluate SPYKE and show that it can handle a large number of device's rules, that enforce limitations on outgoing packets, with no significant degradation in performance. In addition, we assessed the network security by testing the system with a set of well-known attacks described by authors of [5].

Attacker model

For a smart home device to upload data to the Internet through SPYKE, it has to possess a valid password and the explicit user permission. An attacker may exploit this feature by having a valid password and having the device identity. In this model, we consider the following capabilities to model different types of attackers:

- A1 Record any Ethernet frames on the air.
- A2 Inject Ethernet frames with a given source and destination MAC addresses.

Capability A1 can be acquired by an attacker by snooping anywhere near SPYKE using a network interface with the monitor mode ability. A2 can be acquired by using a network interface with the packet injection ability. By having the capability to record and inject Ethernet frames the attacker may perform several attacks:

- B1 Disconnect the legitimate devices.
- B2 Discover the network password.
- B3 Spoof the legitimate device.

[1] https://developer.amazon.com/alexa accessed on April 13, 2018
[2] https://store.google.com/?srp=/product/google_home accessed on April 13, 2018
[3] https://www.smartthings.com accessed on April 13, 2018
[4] https://www.wired.com/story/amazon-echo-alexa-skill-spying/ accessed on May 16, 2018
[5] A skill is an application that can extend Amazon Alexa.

Capability B1 can be performed by having the MAC addresses of legitimate devices and of SPYKE. B2 is very hard to acquire due to the fact that it needs to perform password guessing. Having the legitimate device's MAC address and the valid password, the attacker gains the B3.

Objectives

The objectives stated were to be able:

- to monitor communications between every single device in the network and the Internet;
- to limit the total transfer and the bandwidth;
- to block devices that are sending data to suspicious domains or IP addresses;
- and to block traffic from new or unknown devices by default, i.e., enforce a whitelist access policy.

2 BACKGROUND

In this section we define *middlebox*, an intermediary box situated between the user and the remote server. And present the smart home environment by referring devices, hubs, security monitors, and attacks.

Middlebox

A middlebox is an intermediary device that is situated in the communication flow of a user and a remote server. The main functionality of a middlebox is to intercept, inspect, filter or modify traffic instead of only performing packet forwarding. A middlebox can be a physical device or any software that has the mentioned functionality. One example is a *Network Address Translation (NAT)* box that allows several private network devices to share a single public IP address to access the Internet, because NAT has the ability to masquerade a private IP address to the public IP address and vice versa. Another middlebox example is a *firewall*. A firewall can monitor the network traffic by filtering incoming and outgoing packets. It needs a set of rules that defines whether to allow or block specific traffic.

If a middlebox is correctly deployed, it can improve the performance by dropping unwanted packets and performing an efficient distribution of packets through devices. It can also improve security by blocking packets that may compromise the private network.

Smart Home

A smart environment is the result of the composition of a set of smart devices. These smart devices are usually connected to each other to allow a better user experience. However, devices report data about the user. Examples of smart devices that report home status are smart sensors, e.g., motion sensor reports whenever someone passes by, smoke detector reports the user whenever it detects smoke. Some devices can also perform actions like a coffee machine, a smart plug, or a smart light.

For the purpose of collecting data from smart devices and controlling them, there are *smart hubs* like SmartThings, Amazon Echo, and others.

The authors of [7] present some of those smart hubs' ecosystem, namely SmartThings and Amazon Alexa. In SmartThings' ecosystem, each device communicates with a hub which remains in constant communication with the server cloud. The communication is encrypted to provide privacy and data integrity. Moreover, the user uses a smart application to manage devices through the server cloud. In Amazon Echo's ecosystem, the Virtual Personal Assistant (VPA) Alexa relies on the voice channel to communicate with users, i.e., voice-controlled hub. It is similar to SmartThings as each device communicates with a hub that remains in constant communication with the server cloud.

Kumar et al. [6] and Zhang et al. [8] highlight the security risks of Amazon Echo and Google Home for using voice-controlled third-party skills. They explained *voice squatting* attack and in addition, Zhang et al. also explained *voice masquerading* attack. In the first one, the attacker makes a skill which invocation is phonetically similar to the original skill. Then, the user may activate it and give the attacker sensitive information that is supposed to use on the original skill. In the last one, the attacker again makes a simple skill to be executed by the user. And when the user is executing the attacker's skill, he may want to switch or activate another skill without interrupting the current one. In this case, the attacker's skill may fool the user by saying that the wanted skill is activated and waiting for the user to give sensitive information.

To mitigate the first threat, they built a skill-name scanner that is able to convert the invocation name of a skill into a phonetic expression to measure the phonetic distance between two different skills. For the second threat, a context-sensitive detector was built upon the VPA infrastructure. It consists of two components which are the *Skill Response Checker* and the *User Intention Classifier*. The first one captures suspicious skill responses that a malicious skill may craft such as a fake skill recommendation mimic. The second one is based on the user's intention, i.e., it examines the information flow of the opposite way.

Aside from smart hubs, there are *security monitors* which control the home network. The difference is that a security monitor has more features and functionalities to control the network traffic.

Davies et al. [3] presented an example of security monitor, the *Privacy Mediator* situated in a *Cloudlet* which is a small data center located between the Internet and devices and it is within the trust domain of the end user. It can be

installed on a high-end WiFi access point or can be physically installed in homes, schools or small business. The raw information, obtained by devices, is converted and then aggregated and obfuscated by Privacy Mediator before sending it to the Internet. Moreover, the authors mentioned that the information flow is very important for the user and also the remote server, so having a good set of data redaction and privacy policy enforcement is needed. Privacy Mediator also has user policies to configure which devices have access to the Internet.

More security monitors are: *Pi-Hole*[6], *IoT Inspector*[7], *Fingbox*[8], *Google WiFi*[9], and *Bitdefender BOX 2*[10]:

Pi-Hole is designed to block ads in smart home's devices using as standard hardware a Raspberry Pi. It requires devices to set the DNS to use Pi-hole IP address as it works as it performs DNS filtering to block the ad domains.

IoT Inspector is designed to monitor the home network and inspect IoT devices. It collects and show to users the information of devices.

Fingbox is used to check, restrict and block every device that is connected to the network. In addition, it can block devices that are near but not connected to the network yet.

Google WiFi is intended to provide fast and seamless WiFi connectivity to the home. To use it, it needs a modem, broadband connection to an Internet Service Provider (ISP), a Google Account and Google Wifi app for remote control through a smartphone.

Bitdefender Box 2 has the intention to secure the smart home network. It provides the antivirus to all the network, instead of being in a computer.

Table 1 compares the mentioned systems with SPYKE that we are proposing. Regarding the authentication, Google WiFi, Bitdefender Box and SPYKE provide authentication by WiFi using the password protocol WPA2 (WiFi Protected Access). Other systems' authentications are granted with an existing router, i.e., any device that is connected to the router, can connect to them. In case of Pi-Hole, devices need to know its IP address and set it as DNS server so it be used as the intermediary. Except for Pi-Hole and IoT Inspector, all systems can be defined by the user (user policy) regarding the Internet access for each device, and also the bandwidth. Regarding Network Intrusion Detection, Fingbox and Bitdefender Box notify the user who is trying to connect the system. Finally, Pi-Hole and Bitdefender Box perform filtering due to the ability of blocking traffics that contain advertisement content. SPYKE was designed to implement network intrusion detection and filtering, as a matter of time we moved them to the future work.

Nowadays, the majority of network communications have encryption applied to provide privacy and integrity in the traffic content. Nevertheless, encryption can no longer guarantee privacy protection. Third-parties, e.g., Internet Service Providers (ISP) have access to the network traffic and so, they may be able to analyze the traffic.

Apthorpe et al. [1] showed how encrypted data is vulnerable because of the metadata of the traffic to the attacker. They referred several attacks using these metadata against the encrypted communications. Mostly, *side-channel privacy* attack and *fingerprinting* attack which can be performed by ISP. The first attack comes with the idea of analyzing the traffic pattern, i.e., during a period of time, how traffic behaves, whenever a peak of traffic at a certain time may show an activity is being made. The second attack takes the advantages of metadata by analyzing the traffic information, e.g. sender IP address, destination IP address, timestamp, among other traffic sensitive information.

In order to mitigate these attacks, they showed three solutions: the first one is to retain all traffic in the private network by blocking the traffic for going to outside, however, the majority of smart devices like Amazon Alexa need Internet access to be functional; the second one is to tunnel traffic to another place, in order to gain anonymity, again, this solution cannot handle the vulnerability of side-channel attack; and the last one is the traffic shaping, in order to protect against the side-channel attack, the traffic remains in a constant traffic rate to camouflage the traffic spikes.

Besides attacks from the third-party, it is also possible to perform attacks to the physical network. The majority of the existing routers provide Wireless LAN (WLAN) access with password protection protocol. At the initial stage, it used Wired Equivalent Privacy (WEP) as the security protocol. However, WEP had several flaws due to its *Stream Cipher* using RC4 (Rivest Cipher 4) and *Error Detection* using CRC-32 (Cyclic Redundancy Check - 32 bit) checksum. The authors of [5] provided several attacks to discover the IV (Initialization Value) used on the WEP. WEP was replaced with the WPA and WPA2, new password protection protocols.

Even though WPA2 provides a much better protection, it may not be considered completely secure. The authors of [4] presented a brute force attack to get the network password even with WPA2 protocol used. This attack uses several tools, namely `aireplay-ng`, `airodump-np` and `aircrack-ng`. The idea is to keep watching traffic from the air, meanwhile, send some deauthentication frame to an authenticated device. Then, the device will try to reauthenticate to the access point

[6]https://pi-hole.net/ accessed on April 8, 2019
[7]https://iot-inspector.princeton.edu/ accessed on May 4, 2019
[8]https://www.fing.com/products/fingbox#info accessed on April 13, 2019
[9]https://support.google.com/wifi/answer/7168315?hl=en accessed on April 8, 2019
[10]https://download.bitdefender.com/resources/media/materials/box/v2/user_guide/BOX_UserGuide_v2_en_.pdf accessed on April 8, 2019

Table 1: Comparison of Smart Home Monitors

	Pi-Hole	IoT Inspector	Fingbox	Google WiFi	Bitdefender Box 2	SPYKE
Authentication	N	N	N	Y	Y	Y
User policy	N	N	Y	Y	Y	Y
Network Intrusion Detection	N	N	Y	N	Y	N*
Filtering	Y	N	N	N	Y	N*
Open-Source	Y	Y	N	N	N	Y

which makes possible to the attacker to obtain the WPA password handshake traffic. Finally, having the password handshake traffic, it is possible to perform a brute force attack to determine the network password.

The easier way to mitigate this attack is to use a difficult password and change the password often. However, the authors presented two different techniques. The first one is to map all devices MAC address into the access point, to prevent access from unknown devices. Nevertheless, this can lead to MAC address spoofing attack. The attacker may use `Macchanger` to change the MAC address of the device and authenticate himself as a legitimate device. The second one is to increase the periodicity of transmitting the *beacon frame*, this frame serves to announce the presence of the WLAN. Then, it reduces its frame broadcasting frequency and makes very difficult to perform deauthentication flooding attack. More details about the attack are presented by Čisar et al. in [2].

3 PROTOTYPE

The SPYKE prototype system requires password authentication to accept connection requests from new devices. User intervention is required to grant access to the Internet. In addition, the prototype system provides a level of control on the uploading of packets, i.e., it controls the quantities of packets that can be uploaded. After permission is granted, SPYKE also presents to the user all the destinations where devices have uploaded data.

Architecture

Regarding the system architecture, represented in Figure 1, it is subdivided into two modules. The first one is *authentication*, where devices authenticate with the gateway in order to make requests. The second one is *user policy enforcement* that SPYKE uses for providing privacy protection.

Authentication. A device needs to authenticate itself to prove that it belongs to the home network. The SPYKE gateway performs the wireless authentication using WPA2 as the password authentication protocol. This means that a device can authenticate itself by showing it knows the WiFi password. However, knowing the password of a home network can be an easy task for an attacker, either because the gateway is using a default password, or because the user simply shared the WiFi password. So the device permission to access the Internet is not only controlled by knowledge of the password, but also the explicit user's approval on the user interface, i.e., whitelist access policy.

To distinguish all connected devices, the gateway uses Dynamic Host Configuration Protocol (DHCP) server to assign a different IP address as identity to each authenticated device. The IP address assignment is based on the unique MAC address provided by devices.

Data Processing. After the authentication is established, and the permission granted by the user, data that is uploaded and downloaded by a device passes through the gateway. This data processing is represented into two processes: *Analysis* and *User policy*. Data is analyzed at the first step by understanding the packet metadata, e.g., the sender and receiver IP addresses and content size. Then, the user has access to see connected devices, and destination IP addresses of connections made by each device. The proposed system allows the user to check how much data has passed and how much had been dropped. Regarding the user policy, the user may define which device has the access to the Internet, how many data the device can upload, and the available bandwidth. After the data analysis, the proposed system compares data with the user defined policies. It drops packets that came from unknown devices and ones that are previously blocked by the user. In addition, it drops also traffic that exceeded the maximum transfer quota or bandwidth defined by the user. Finally, all allowed traffic goes to the Internet.

The SPYKE prototype requires intervention from the user. Whenever a device tries to connect to SPYKE, it should authenticate itself providing the correct password. Once a device is authenticated to the gateway, its information like unique MAC address, assigned IP address, and hostname is recorded and presented in the user interface. If the user approves the device, the gateway starts to give the device access to the Internet. After that, it starts recording the packet's destination IP address and total packets have been sent within a period of time.

Figure 2 shows the lifecycle of a device in SPYKE. When a new device is detected by the system, its status becomes "NEW" and the information is stored in the database. Then

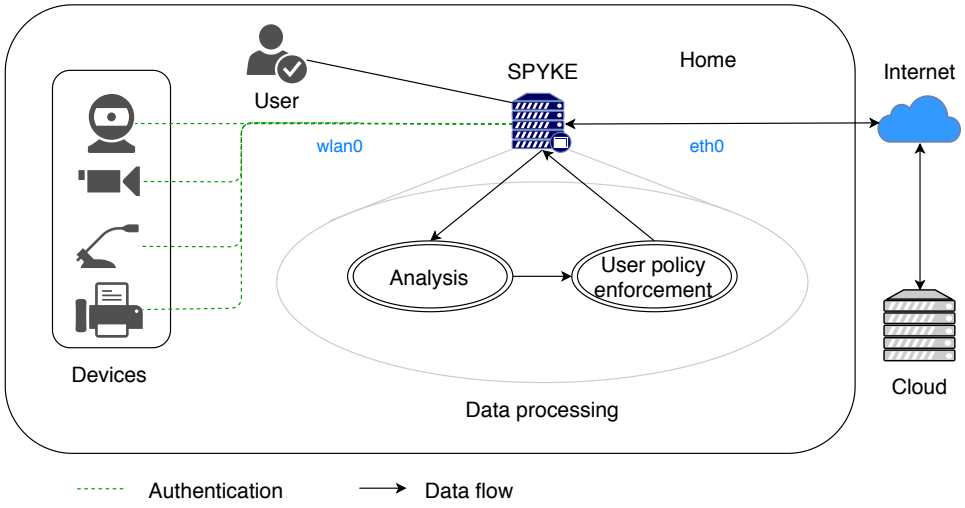

Figure 1: SPYKE proposed architecture

the user may allow or block the device. In the first case, the device's status is changed to "BLOCKED" and the information is stored in the database. For the second case, the device's status is changed to "ALLOWED", the information is stored in the database and in addition a period is created and stored to the database and `iptables`' rules are created and added to `iptables`.

Whether the device is "ALLOWED" or "BLOCKED", the user may still allow or block the device but never change back to "NEW". Furthermore, when the device's status is set to "ALLOWED", it occurs two following cases: the first case, the user may change the device's maximum bandwidth, maximum transfer quota, or period, hence the device is updated, information is stored in the database and the `iptables`' rules are updated; the second case is whenever the end of periods is achieved, `iptables`' registers are extracted and stored in the database within the period and a new period is created.

Figure 3 shows the Data Model, composed by two tables, Device and Period, with an one-to-many relationship.

In the Device's table, the *id* is the MAC address that is considered as the *primary key*. In the Period table, the *id* is composed by start and end time, and the associated MAC address from the device, creating a *composite key*.

When a new device is detected, SPYKE stores information containing variables like the MAC address and name provided by the device, the IP address provided by DHCP server, the status as new, quota and bandwidth values as zero (0) with corresponding unit as KiloByte (KB), and the period as zero with corresponding unit as minute (m).

After the user changes the status of the device to "ALLOWED", a Period is created and stored with the associated device's MAC address, the start time and end time.For example, if a device is registered at 6:00 with a period of 5 minutes, the value of *start_time* will be 6:00 and the value of *end_time* will be 6:05. If the period is not defined, then it will be set to 1 hour by default.

Figure 4 represents the data flow of the entire system. First, devices authenticate themselves and obtain an IP address from DHCP server provided by `dnsmasq`[11]. The engine stores the device information in the database and waits for the user's approval. After the user's approval, the engine adds rules on `iptables` allowing the access of the device to the Internet, and adds the defined period to the In-Memory data storage that relies on main memory of the computer data storage.

The periods of each device are stored in the database whenever it achieves the end of the period, and then new periods are created, and `iptables`' rules are renewed. Meanwhile, the user can access the information about bytes allowed and dropped by each period. The user can also decide whether to allow or block any devices as well as modify the value of period or limit.

Implementation

The proposed system was implemented on a Raspberry Pi 3b+[12] with Raspbian Stretch Lite as the Operating System.

The Raspberry Pi 3B+ uses Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC, providing a processor with a frequency of 1.4 GHz, and uses LPDDR2 SDRAM that provides 1 GB of RAM. The Raspberry Pi 3B+ provides a network interface card with protocols IEEE 802.11.b/g/n/ac WLAN (*wlan0*) that provides 2.4 GHz and 5 GHz bands. It also provides

[11] http://www.thekelleys.org.uk/dnsmasq/doc.html accessed on April 8, 2019
[12] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ accessed on April 8, 2019

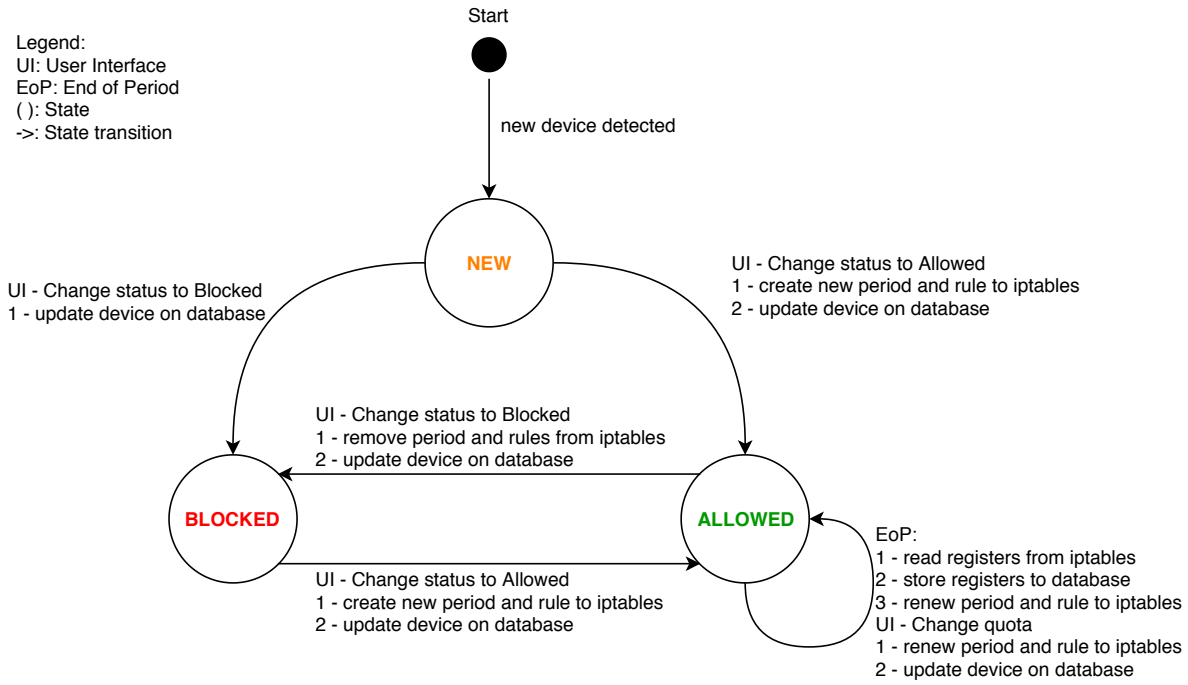

Figure 2: State machine for a device managed by SPYKE

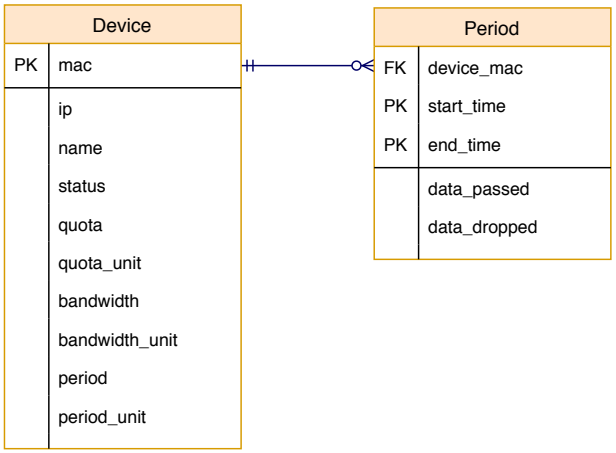

Figure 3: Data Model of SPYKE database

another network interface via cable with Gigabit Ethernet (*eth0*) up to 300 Mbps.

Despite being implemented into a Raspberry Pi, it can be implemented in any off-the-shelf hardware that has two network interfaces, one for providing the network to the devices and another to provide Internet access.

4 EVALUATION

The SPYKE prototype was evaluated in regard to performance and security. First, we measured the baseline performance without running SPYKE. The experiment measured using an iPerf[13] from the Client to the Server with an interval of 200 seconds. The results are transformed into a graphical representation using MatplotLib[14].

To quantify the random errors in measurements, the program runs were repeated several times. At least 30 runs, so that calculation can assume a normal distribution of the samples, according to the Central Limit Theorem[15].

4.1 Performance experiments

The same environment and structure were used to measure and evaluate the SPYKE overhead in comparison with the baseline. Table 2 and Table 3 show the average of the total transfer and bandwidth of thirty experiences consecutively. This table compares all the average of all experimental evaluations that have been done for evaluating the system's performance.

The gateway maintains good performance whether SPYKE is running or not. In tables 2 and 3, respectively, we see that

[13] https://iperf.fr/ accessed on April 13, 2019
[14] A Python's library that plots data result in a graphical representation: https://matplotlib.org/ accessed on April 8, 2019
[15] Only changes in values greater than the error margin can be considered statistically relevant and not the effect of random errors

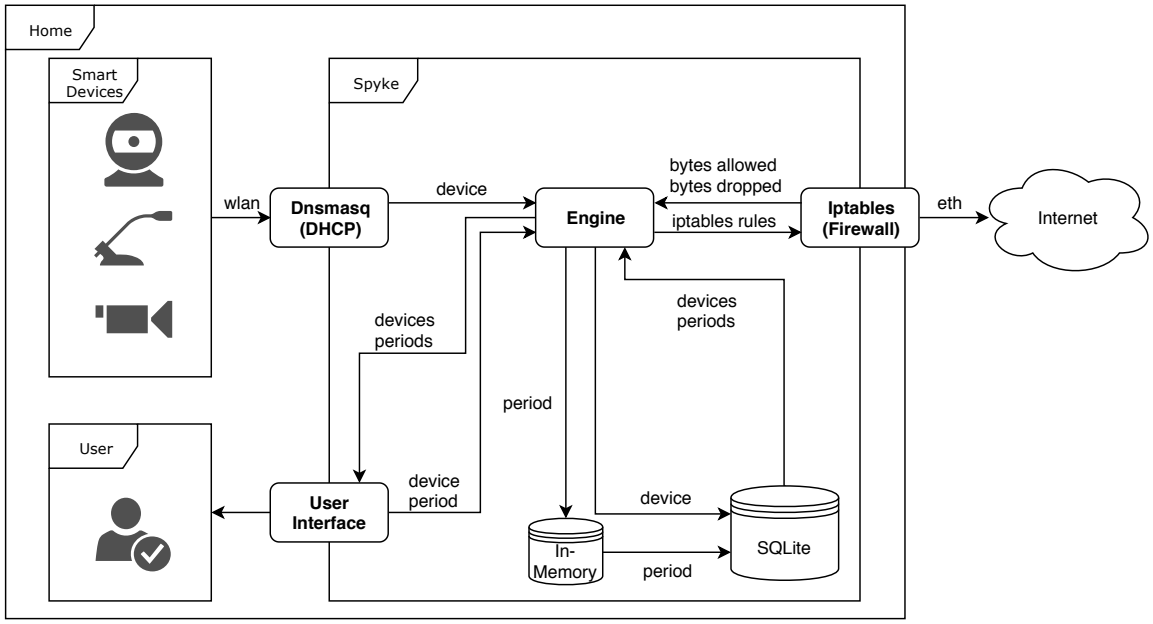

Figure 4: Data Flow during SPYKE operation

with 1 device, SPYKE transferred more data and has larger bandwidth than the baseline without firewall. We believe that this difference exists because SPYKE blocks connections by default, so it increases performance slightly by forwarding only the allowed connections.

By increasing the number of `iptables` rules, i.e. 10 000 devices, it started dropping the performance. Nevertheless, for a smart home environment, SPYKE is capable to handle a very large number of devices.

Table 2: Average total transfer results for the performance experiments. Thirty runs were performed for each experiment.

	Total Transfer (MB)	Minimum Total Transfer (MB)	Maximum Total Transfer (MB)
Normal	1010	1001 (-09)	1016 (+06)
1 device	1045	1020 (-25)	1064 (+19)
100 devices	1020	1002 (-18)	1024 (+04)
1 000 devices	1023	1022 (-01)	1024 (+01)
10 000 devices	980	968 (-12)	993 (+13)

One of the important features of SPYKE is blocking upload traffic from unknown devices by default, and allowing the user to set limits on the usage for known devices. After the performance evaluations, rules were enforced to limit the bandwidth as well as quota for a defined period of time for all upload data.

Table 3: Average bandwidth results for the performance experiments. Thirty runs were performed for each experiment.

	Bandwidth (MBps)	Minimum Bandwidth (MBps)	Maximum Bandwidth (MBps)
Normal	5.050	5.000 (-0.050)	5.080 (+0.030)
1 device	5.230	5.100 (-0.130)	5.320 (+0.090)
100 devices	5.100	5.010 (-0.090)	5.120 (+0.020)
1 000 devices	5.115	5.110 (-0.005)	5.120 (+0.005)
10 000 devices	4.900	4.840 (-0.060)	4.960 (+0.060)

Regarding the limit of maximum transfer per period, several experiments were performed on the Amazon Echo Dot 3 with 200 KB as the quota during 5 minutes and no limitation on bandwidth. During this period of time, the test user asked about the current weather, time to upload data. Figure 5 shows the experience for 969 seconds, which is more than 16 minutes, this covers four periods. The 200 KB limit over 4 periods was enforced and a total 797 KB of data was transferred as expected.

About the bandwidth, a similar experiment was performed with the Amazon Echo. But this time, the bandwidth was

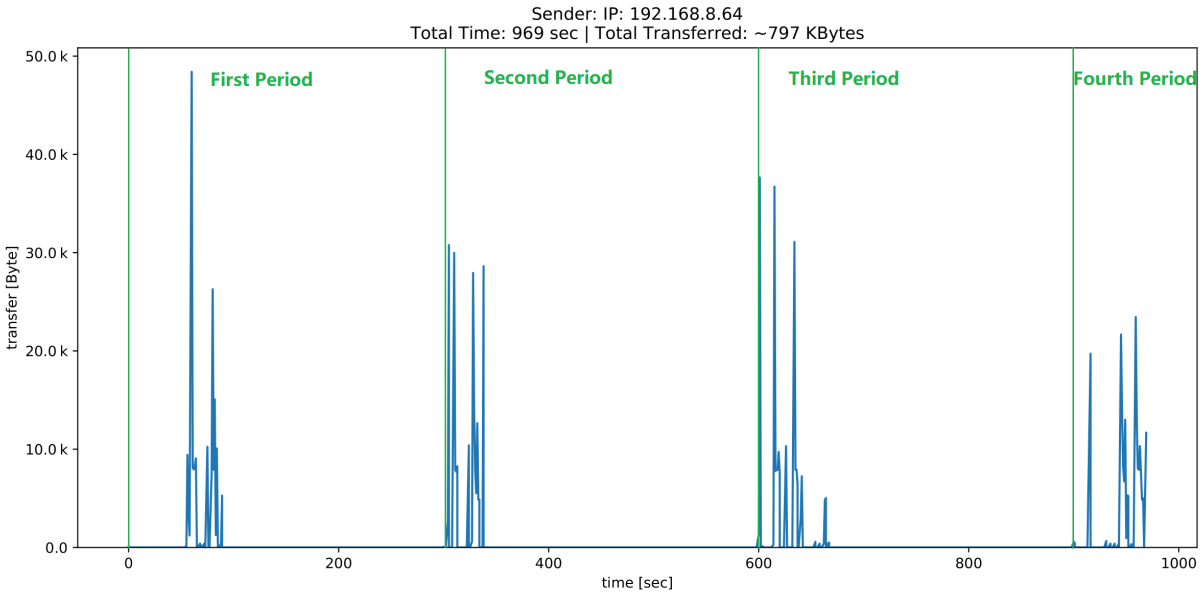

Figure 5: Results for Amazon Echo with limited upload quota

set to 10 KBps, a period of 5 minutes and no quota limitation. Figure 6 represents the result for 600 seconds (10 minutes). Even when the bandwidth is set to 10 KBps, it almost achieved 20 KBps due to some occasional packet bursts.

Discussion

The performance experiments showed that the prototype overhead is very small, a large number of device rules are supported, and we verified the operation with commercial devices. Furthermore, the user may be aware of a possible malfunctioning device, which behaves strangely and uploads data to an unknown or undesired IP address and the user can block it. Additionally, the rule enforcement works as expected except for bandwidth, which may achieve almost double of the device value due to the packet burst.

4.2 Security assessment

We evaluated the effectiveness of the SPYKE prototype against well-known attacks. The assessment started with a list of availability attacks compiled by [5].

SPYKE uses WPA2 as the password security protocol. However, any password security protocol is vulnerable to a Dictionary attack. Other relevant attack is Spoofing attack.

For this purpose, a Raspberry Pi was used again with the `aircrack-ng`[16] tool installed to assess network security. In addition, an external network interface, Alfa network AWUS036NHA[17] was used in order to use the `aircrack-ng`, because it has Monitor Mode and is able to do Packet Injection.

For checking the effectiveness of the proposed system, the following attacks were performed: Deauthentication and Disassociation, Authentication Request Flooding, Dictionary, and Spoofing attacks.

Deauthentication and Disassociation attacks are attacks with the goal of ending a connection established between a device and an access point. We activated the external interface to monitor mode and used `airodump-ng` to capture all the reachable traffic in the air, for obtaining the information needed. Then, we performed the attack by injecting deauthentication frame using `aireplay-ng` with the MAC address of the connected device and SPYKE, and the device lost the connection. Furthermore, we performed the Deauthentication Broadcast attack, that sends an unlimited number of deauthentication frames to make SPYKE busy and ignoring other devices. SPYKE was not able to identify the attack. However, other devices with status "ALLOWED" did not lose the connection.

The same experiment were tested again by setting 500 ms as the beacon interval, which is a solution mentioned by [4]. With some difficulty, `airodump-ng` was able to find the MAC address of the connected devices and SPYKE. The connected device with status "ALLOWED" will not be able to connect to the Internet while the attack is performing. Though, other

[16] https://www.aircrack-ng.org/ accessed on April 12, 2019

[17] https://www.alfa.net.my/webshaper/store/viewProd.asp?pkProductItem=15 accessed on April 12, 2019

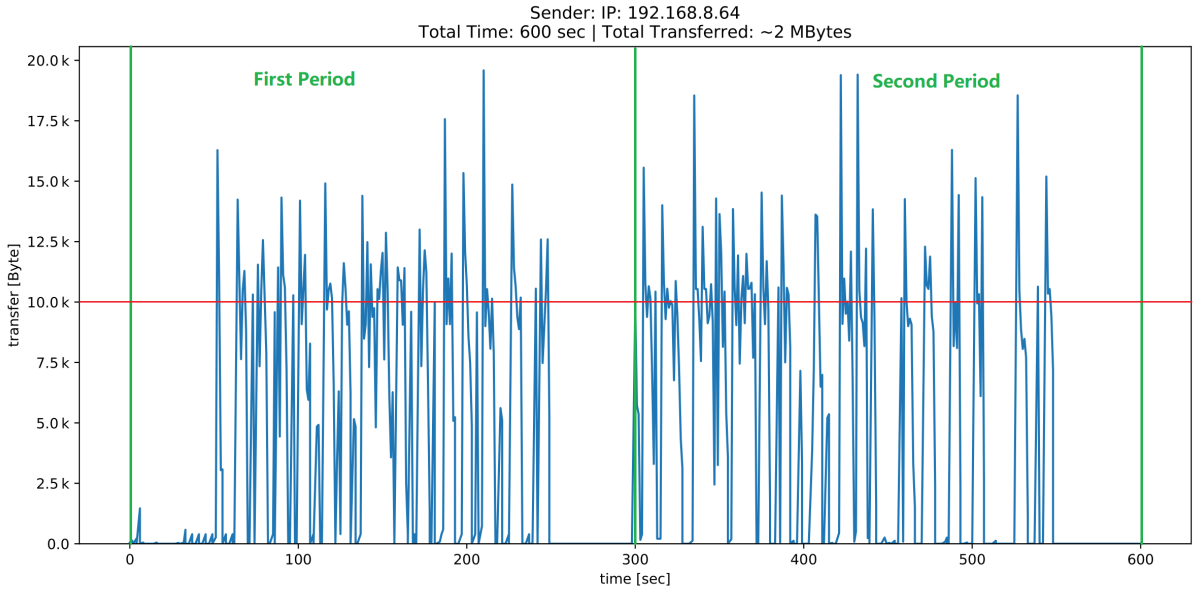

Figure 6: Results for Amazon Echo with limited upload bandwidth

devices are still connected and have the Internet accessed, in case they are allowed. For this reason, the attack is not completely effective against SPYKE.

Authentication Request Flooding attack is performed to exhaust, slowdown, or even freeze an access point. We used the `mdk3`[18] tool to perform fake authentication request. We specified the MAC address of SPYKE and performed the attack. After some minutes, the device lost the connection as well as all other connected devices. To prevent devices disconnecting from the system, we added rules on `iptables` to limit 10 packets per second on the UDP connection on the ports 67 and 68. Then, the same experiment was done again. This time, SPYKE did not disconnect devices. For this reason, we considered SPYKE is able to protect against this attack.

Dictionary attack is performed to find out the password of an access point. In most cases, it is used to crack WPA and WPA2 encryption schemes. First, we run `airodump-ng` to capture the packets. Then we deauthenticated a connected device. Once the device is disconnected, it will try to reconnect again. Meanwhile, `airodump-ng` is running and capturing packets and stores in a pcap file. Finally, we used `aircrack-ng` with lists of password as input to the handshake packets. If the list contains the correct password the attack is accomplished.

There are several ways of preventing this attack, e.g. use a strong password, which is difficult to be guessed. SPYKE is effective against this attack if the WiFi password is well defined.

[18]https://tools.kali.org/wireless-attacks/mdk3 accessed on April 13, 2019

Spoofing attack is performed to fool an access point by disguising as a legitimate device. We used `macchanger` to change the unique MAC address of the network interface to a legitimate device. After that, we connected to SPYKE with the correct password. Then, we grant the same permission as the legitimate device. Nevertheless, the connection is not stable due to two devices trying to connect SPYKE with the same identity.

SPYKE limits the bandwidth and quota of each device. For this reason, we were not able to send much data. Furthermore, the user is notified, through the user interface, of the destination where devices are connected to. Then, the user can block the device.

Since we can find out the MAC address of any connected device by using `airodump-ng`, we can perform Spoofing attacks to all connected devices. One way to exclude completely the attacker from the network is by changing the WiFi password regularly.

Discussion

Regarding the effectiveness against the attacks, SPYKE is able to defend against some DoS attacks, namely deauthentication broadcast, dissociation broadcast and authentication request flooding, but it cannot eliminate DoS attacks on the devices. SPYKE cannot protect against the brute force attack on the password. However, even if an attacker accesses the home network, he has no access to the Internet.

An attacker may falsify his identity by using the MAC address of a legitimate device, and upload data to the Internet.

In this case, SPYKE cannot block it immediately, but it will eventually block it when it overcomes the maximum quota allowed defined by the user. In addition, the user can be informed by the user interface that the device is uploading data to unknown IP address, and block it.

Table 4 shows availability attacks [5] that is within the SPYKE coverage. The majority of attacks are performed in the data link layer, with the goal to compromise the device connectivity. This means that attacks which are not covered by the proposed system are focused on disconnecting devices. The SPYKE prototype covered attacks that have the intention to freeze or slowdown, such as Deauthentication Broadcast, Disassociation Broadcast and Authentication Request Flooding.

Table 4: SPYKE availability attacks coverage

Attack	Covered by SPYKE
Deauthentication	×
Disassociation	×
Deauthentication Broadcast	✓
Disassociation Broadcast	✓
Block ACK Flood	×
Authentication Request Flooding	✓
Fake Power Saving	×
Clear To Send Flooding	×
Request To Send Flooding	×
Beacon Flooding	×
Probe Request Flooding	×
Probe Response Flooding	×

5 CONCLUSIONS

SPYKE was designed to be located in a smart home network between smart devices and the Internet. It provides visibility of communications between each device and the Internet. Additionally, SPYKE has the ability to enforce rules to block and limit connections. Furthermore, it is available as an open-source project and deployable in an inexpensive off-the-shelf hardware such as Raspberry Pi.

We evaluated the SPYKE prototype. The results showed good performance and effective rule enforcement. The prototype was able to handle connections even when adding 10 000 devices rules to the system. Beyond that, it was tested with commercial devices like Amazon Echo and TP Link Smart Electrical Plug. The rule enforcement is applicable to all connected devices, upload packets are indeed reduced according to the limitation set. We also did a security assessment covering a set of availability attacks [5] and SPYKE was able to prevent a set of relevant DoS and Spoofing attacks.

In regard to future work, we propose the following:

Traffic Shaping The side-channel attack compromises the user's privacy, because an attacker may figure out which routine the user has by analyzing the traffic spikes. Traffic Shaping [1] can be used to produce a constant traffic rate and camouflage traffic spikes.

Intrusion Detection System SPYKE should perform packet content analysis and detect if the device's end-point is a threat. For example, detect a smart plug that is uploading audio or video, that is is not suppose to be able to do. This module can be implemented by using an existing open source IDS like Snort, Suricata or Zeek. In addition, the detection should go beyond knowledge-based rules that detect known attacks, and add Machine-Learning-based techniques, to detect anomalies that may be unknown attacks.

Incoming Traffic: We focused more on limiting and checking the outgoing traffic, because we want to block the devices that upload sensitive data to undesired third parties. However, the same architecture and implementation can be extended to handle incoming traffic.

REFERENCES

[1] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic. *CoRR* abs/1708.05044 (2017). arXiv:1708.05044 http://arxiv.org/abs/1708.05044

[2] P Čisar and S Maravić Čisar. 2018. ETHICAL HACKING OF WIRELESS NETWORKS IN KALI LINUX ENVIRONMENT. *Annals of the Faculty of Engineering Hunedoara* 16, 3 (2018), 181–186.

[3] Nigel Davies, Nina Taft, Mahadev Satyanarayanan, Sarah Clinch, and Brandon Amos. 2016. Privacy Mediators: Helping IoT Cross the Chasm. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications (HotMobile '16)*. ACM, New York, NY, USA, 39–44. https://doi.org/10.1145/2873587.2873600

[4] M. M. Hafiz and F. H. Mohd Ali. 2014. Profiling and mitigating brute force attack in home wireless LAN. In *2014 International Conference on Computational Science and Technology (ICCST)*. 1–6. https://doi.org/10.1109/ICCST.2014.7045190

[5] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis. 2016. Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset. *IEEE Communications Surveys Tutorials* 18, 1 (Firstquarter 2016), 184–208. https://doi.org/10.1109/COMST.2015.2402161

[6] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. 2018. Skill Squatting Attacks on Amazon Alexa. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 33–47. https://www.usenix.org/conference/usenixsecurity18/presentation/kumar

[7] Jack Sturgess, Jason R C Nurse, and Jun Zhao. 2018. A capability-oriented approach to assessing privacy risk in smart home ecosystems. In *Living in the Internet of Things: Cybersecurity of the IoT Conference*. IET.

[8] N. Zhang, X. Mi, X. Feng, X. Wang, Y. Tian, and F. Qian. 2018. Understanding and Mitigating the Security Risks of Voice-Controlled Third-Party Skills on Amazon Alexa and Google Home. *ArXiv e-prints* (May 2018). arXiv:cs.CR/1805.01525