SPYKE: Security ProxY with Knowledge-based intrusion prEvention

Sheng Wang[0000−0002−5821−0188], Rui Claro[0000−0003−0176−2720], and Miguel L. Pardal[0000−0003−2872−7300]

Instituto Superior Técnico, Universidade de Lisboa, Portugal
{sheng.wang,rui.claro,miguel.pardal}@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. These data sources will eventually upload data to the cloud. In this work we present SPYKE, 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: Intrusion Prevention 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*, *Google Home*, 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[1] has shown that the Amazon Echo can be infected after the user activates a malicious *skill*[2], effectively becoming a *spy device*, that sends voice recordings to an attacker.

Data leakage is very difficult to avoid, but it is possible to monitor communications through an *intermediary*, who can inform the user of the communications that are happening. In this paper we propose SPYKE – Security ProxY with Knowledge-based intrusion prEvention – an intermediate network device

[1] https://www.wired.com/story/amazon-echo-alexa-skill-spying/ accessed on May 16, 2018

[2] A skill is an application that can extend Amazon Alexa.

between the user devices and the cloud service providers. We implemented and evaluated the solution to show that it can handle a large number of device's rules, that it enforces 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 [4].

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 the device's MAC address. 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. A1 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.

Capability B1 is acquirable 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 goal was to build a prototype SPYKE, a trusted intermediary located in a private network. 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; and to block devices that are sending data to suspicious domains or IP addresses.

2 Background

In this section we present the smart home environment by referring devices, hubs, security monitors, and attacks.

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. For the purpose of collecting data from smart devices and controlling them, there are *smart hubs* like the Amazon Echo[3]. The authors of [6] present the ecosystem of Amazon Alexa. It consists of a Virtual Personal Assistant (VPA) which relies on the voice channel to communicate with users, and each device communicates with the hub which remains in constant communication with the server cloud. Kumar et al. [5] and Zhang et al. [7] highlight the security risks of Amazon Echo for using voice-controlled third-party skills.

Aside from smart hubs, *security monitors* control the network traffic. Davies et al. [2] 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 examples of security monitors are: *Pi-Hole*[4], *IoT Inspector*[5], *Fingbox*[6], *Google WiFi*[7], and *Bitdefender BOX 2*[8]: 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' authentication is 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 policy regarding the Internet access for each device. Regarding Network Intrusion Detection, Fingbox and Bitdefender Box notify the user the malicious devices. Finally, Pi-Hole and Bitdefender Box perform filtering due to the ability of blocking traffics that contain advertisement content.

Nowadays, the majority of network communications have encryption applied to provide confidentiality and integrity to the traffic content. However, Apthorpe et al. [1] showed how encrypted data is vulnerable due to the metadata of the traffic. They referred several attacks, namely *side-channel* attack, analyse the traffic peaks that may show an activity is being made, and *fingerprinting* attack, analyse the medatada of the traffic sensitive information, which can be performed by third-parties, e.g., ISP (Internet Service Providers). To mitigate,

[3] https://developer.amazon.com/alexa accessed on April 13, 2018
[4] https://pi-hole.net/ accessed on April 8, 2019
[5] https://iot-inspector.princeton.edu/ accessed on May 4, 2019
[6] https://www.fing.com/products/fingbox# info accessed on April 13, 2019
[7] https://support.google.com/wifi/answer/7168315?hl=en accessed on April 8, 2019
[8] 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

they presented *traffic shaping* where the traffic is modified to have a constant traffic rate to camouflage the characteristic traffic spikes.

Besides attacks from the third-parties, it is also possible to perform attacks to the private network. The authors of [3] presented a brute force attack to get the network password with WPA2 protocol used. 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 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.

3 Prototype

The SPYKE prototype system requires password authentication to accept the connection of new devices to the network. User intervention is required to grant this access, usually by providing the WPA2 password. 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. SPYKE also presents to the user all the destinations where devices have uploaded data.

Architecture

Regarding the system architecture, represented in Fig. 1, it is subdivided into two modules: *authentication*, where devices authenticate with the gateway in order to make requests; and *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

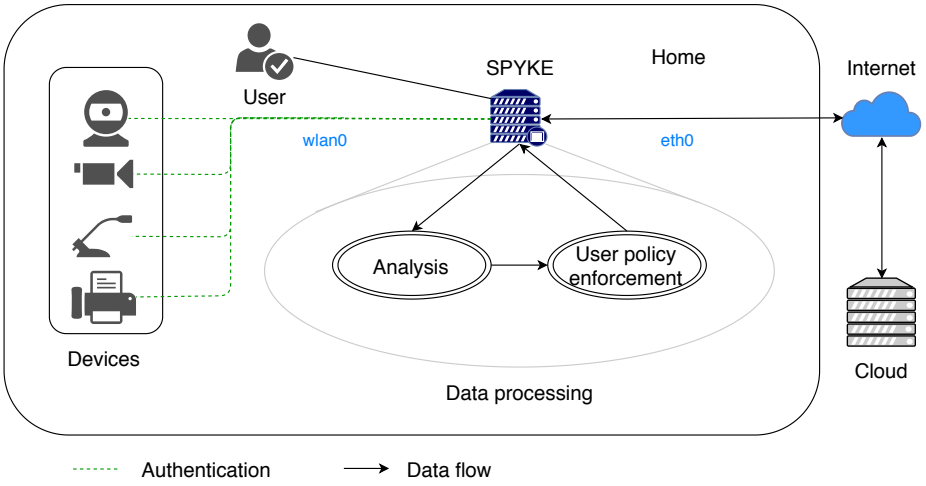

Fig. 1. SPYKE proposed architecture

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 also drops traffic that exceeded the maximum transfer quota and bandwidth defined by the user.

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.

Fig. 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 the user may allow or block the device. When the device's

status is changed to "ALLOWED": first, the information is updated to the database; a period is created; and `iptables`' rules are created and added to `iptables`. Furthermore, it occurs two following cases: 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; and whenever the end of periods is achieved, `iptables`' registers are extracted and stored in the database along with the period.

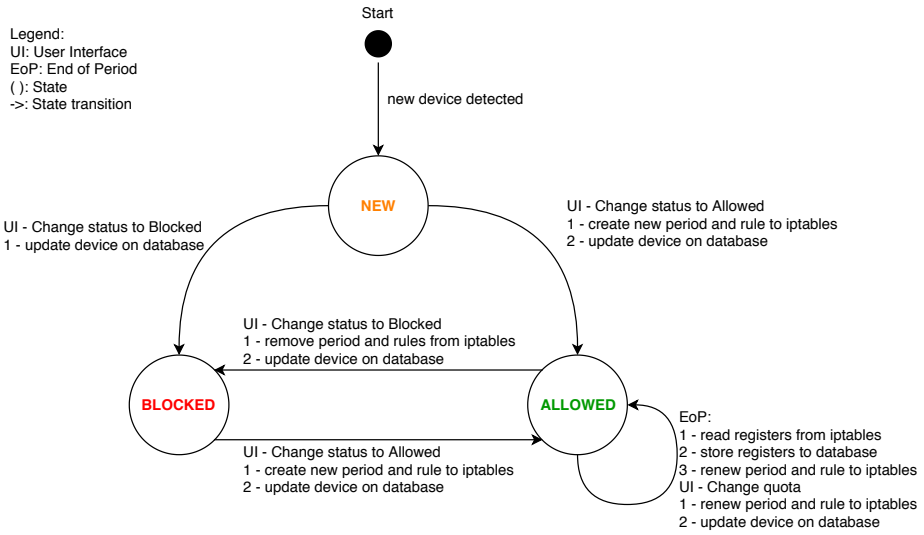

Fig. 2. State machine for a device managed by SPYKE

Fig. 3. represents the data flow of the entire system. First, devices authenticate themselves and obtain an IP address from DHCP server provided by `dnsmasq`[9]. 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 computer data storage.

Implementation

The proposed system was implemented on a Raspberry Pi 3b+[10] with Raspbian Stretch Lite as the Operating System. It provides a network interface card with

[9] http://www.thekelleys.org.uk/dnsmasq/doc.html accessed on April 8, 2019

[10] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ accessed on April 8, 2019

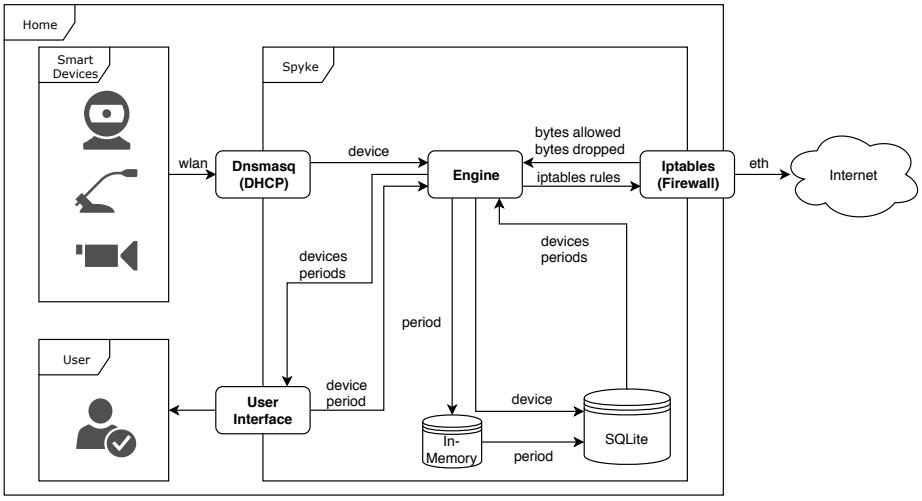

Fig. 3. Data Flow during SPYKE operation

protocols IEEE 802.11.b/g/n/ac WLAN (*wlan0*) and a 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 proving network to the devices and one other to provide the 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 iPerf[11] with an interval of 200 seconds.

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[12].

4.1 Performance experiments

The same environment and structure were used to measure and evaluate the SPYKE overhead in comparison with the baseline. The gateway maintains good performance whether SPYKE is running or not. In tables 2 and 3, respectively, we see that with 1 device, SPYKE transferred more data and has larger bandwidth than the baseline without firewall. We believe that this difference exists be-

[11] https://iperf.fr/ accessed on April 13, 2019

[12] Only changes in values greater than the error margin can be considered statistically relevant and not the effect of random errors

cause 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)

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)

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. Then, 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, we asked about the current weather and time to upload data. Fig. 4. 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. However, the bandwidth was set to 10 KBps, a period of 5 minutes and no quota limitation. Fig. 5. represents the result for 600 seconds (10 minutes). Even when the bandwidth is set to 10 KBps, it almost achieved 20 KBps due to the packet burst.

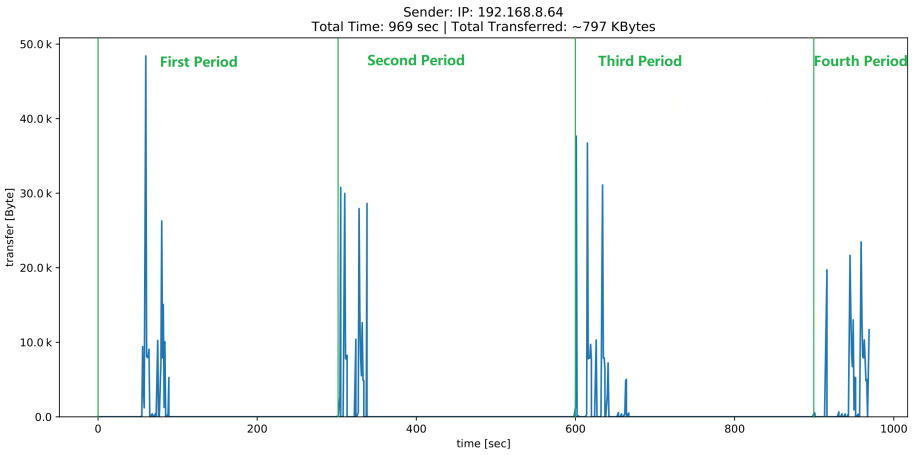

Fig. 4. Results for Amazon Echo with limited upload quota

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 two commercial devices. Furthermore, the user may be aware of a possible malfunctioning device, which behaves strangely and uploads data to undesired destination and able to block it. Additionally, the rule enforcement works as expected.

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 compilated by [4]. For this purpose, a Raspberry Pi was used with the `aircrack-ng`[13] tool installed.In addition, we used an external network interface, Alfa network AWUS036NHA[14] which provides Monitor Mode and Packet Injection. Then, the following attacks were performed:

Deauthentication and Disassociation attacks have the goal of ending a connection established between a device and an access point. We activated the external interface with the monitor mode and used `airodump-ng` to capture all the reachable traffic in the air. Then, we performed the attack by injecting deauthentication frame using `aireplay-ng` with the SPYKE and device's MAC addresses, then 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. However,

[13] https://www.aircrack-ng.org/ accessed on April 12, 2019
[14] https://www.alfa.net.my/webshaper/store/viewProd.asp?pkProductItem=15 accessed on April 12, 2019

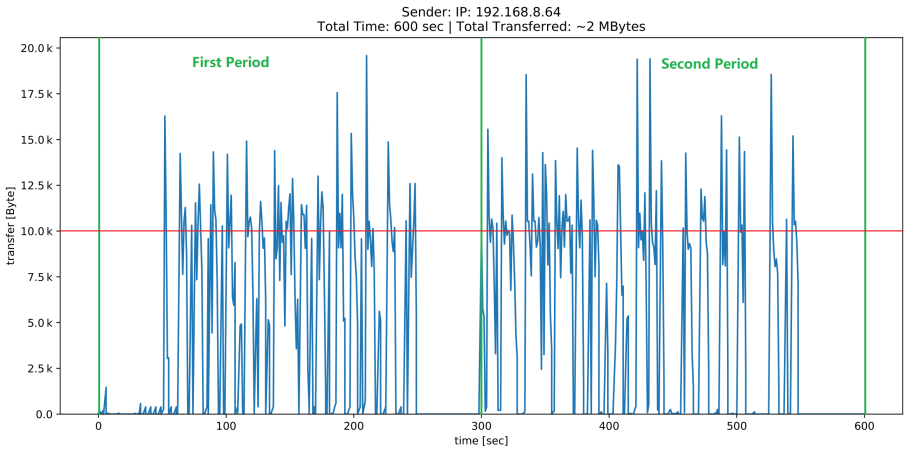

Fig. 5. Results for Amazon Echo with limited upload bandwidth

other devices with status "ALLOWED" did not lose the connection. For this reason, the attack is not completely effective against SPYKE.

Authentication Request Flooding attack is performed to slowdown, or even freeze an access point. We used the `mdk3`[15] tool to perform fake authentication request. We specified the MAC address of SPYKE and performed the attack. This attack was not effective due to the rules that limit 10 packets per second on the UDP connection on the ports 67 and 68. 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. 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. Meanwhile, `airodump-ng` is running and capturing packets that contains handshake packets. 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 successed. 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.

Spoofing attack is performed to fool an access point by disguising as a legitimate device. We used `macchanger` to change the MAC address to the legitimate device. After that, we connected to SPYKE with the valid 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. However, we were not able to send as many data as we wanted due to the rule enforcement. Furthermore, the user is notified of the destination where devices are connected to through the user interface. Then, the user can block the device.

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

Discussion Regarding the effectiveness against the attacks, SPYKE is able to defend against some DoS attacks.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 uploads 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 [4] 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 private network between devices and the Internet and provide visibility of communications. 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. We also did a security

assessment covering a set of availability attacks [4] 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.

Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019 (INESC-ID) and through project with reference PTDC/CCI-COM/31440/2017 (SureThing).

References

1. Apthorpe, N., Reisman, D., Sundaresan, S., Narayanan, A., Feamster, N.: Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. CoRR **abs/1708.05044** (2017), http://arxiv.org/abs/1708.05044
2. Davies, N., Taft, N., Satyanarayanan, M., Clinch, S., Amos, B.: Privacy mediators: Helping iot cross the chasm. In: Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications. pp. 39–44. HotMobile '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2873587.2873600, http://doi.acm.org/10.1145/2873587.2873600
3. Hafiz, M.M., Mohd Ali, F.H.: Profiling and mitigating brute force attack in home wireless lan. In: 2014 International Conference on Computational Science and Technology (ICCST). pp. 1–6 (Aug 2014). https://doi.org/10.1109/ICCST.2014.7045190
4. Kolias, C., Kambourakis, G., Stavrou, A., Gritzalis, S.: Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. IEEE Communications Surveys Tutorials **18**(1), 184–208 (Firstquarter 2016)
5. Kumar, D., Paccagnella, R., Murley, P., Hennenfent, E., Mason, J., Bates, A., Bailey, M.: Skill squatting attacks on amazon alexa. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 33–47. USENIX Association, Baltimore, MD (2018), https://www.usenix.org/conference/usenixsecurity18/presentation/kumar
6. Sturgess, J., Nurse, J.R.C., Zhao, J.: 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 (2018)
7. Zhang, N., Mi, X., Feng, X., Wang, X., Tian, Y., Qian, F.: Understanding and Mitigating the Security Risks of Voice-Controlled Third-Party Skills on Amazon Alexa and Google Home. ArXiv e-prints (May 2018)