IoTAMU: Protecting Smart Home Networks via Obfuscation and Encryption

Youngjun Park
School of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, USA
Email: youngjun.park@afit.edu

Richard Dill
School of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, USA
Email: richard.dill@afit.edu

Barry E. Mullins
School of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, USA
Email: barry.mullins@afit.edu

***Abstract*—In the changing landscape where an increasing number of organizations deploy smart devices to their networks, one of the greatest challenges they face is security. While the use of Internet of Things (IoT) has enabled new capabilities, such as ease of access, remote control, and interoperability, it has also introduced new attack vectors. For example, due to the limited hardware capacity, IoT devices lack the additional computational resources required for security, such as data encryption. As a result, gaining access to the data associated with the IoT devices becomes almost trivial assuming the adversary has physical access to the device or logical access to the network. Unfortunately, the production of the IoT devices cannot be effectively regulated without a governing policy, leaving the burden to secure the devices to the end users. To help mitigate the vulnerabilities stemming from the hardware limitations of IoT devices, we present Internet of Things Active Management Unit (IoTAMU), a defensive model to obscure the sensitive data sent over Wi-Fi. As a proof of concept, we first show that the video stream created by one of the most popular IoT cameras being sold on Amazon can be recreated via passive sniffing. Then, we present an automated tool to extract the video stream from network traffic. In 100 percent of test cases, the tool was able to extract a recognizable video stream from captured network traffic. Finally, we propose IoTAMU, a central management agent which acts as the network proxy for the vulnerable IoT devices to both obfuscate the network traffic by mimicking real devices, and to serve as an encryption agent for the devices with limited computational capacity. The model requires minimum set up for the users, and is compatible with any device that is configurable over Wi-Fi. IoTAMU will help pioneer easily deployable user-end security agents to protect the confidentiality in smart home networks.**

Keywords–Internet of Things (IoT); data security; network obfuscation; Wi-Fi camera.

I. INTRODUCTION

The term Internet of Things (IoT) represents a wave of embedded technologies with the added functionality of connectivity. Since its inception, IoT has infiltrated numerous public sectors in Industrial Control Systems (ICS), cities, healthcare, and government [1]; according to the International Data Corporation, the IoT industry is projected to reach 1.2 trillion dollars by the year 2022 [2]. However, the rapid growth of the industry is rivaled by the increasing number of vulnerabilities that are discovered in the devices. As the devices continue to be deployed in critical infrastructures and national institutions, investigating secure policies for the use of IoT becomes one of the priorities for organizations such as the U.S. Department of Defense [3].

Embedded systems found in automotives, Supervisory Control and Data Acquisition (SCADA) systems, among others, were originally designed to function as closed systems. By connecting those systems to the Internet, the devices presented numerous vulnerabilities that cannot be easily defended. The dangers of these design deficiencies were highlighted in a 2015 study on the infotainment system found in modern vehicles, which discovered a vulnerability that allowed an adversary to gain remote control of the vehicle [4]. Security experts have recognized the security flaws in IoT devices and have investigated potential attack surfaces to help the manufacturers and the users to mitigate them, including the effort by the Open Web Application Security Project (OWASP) [5]. However, the hardware limitations of the devices often become the bottleneck for meaningful security measures such as encryption, which requires large computational power. Security is therefore left in large part to the end users.

Currently, one of the most common modes of communication for IoT devices is Wi-Fi [1]. While the security of Wi-Fi has improved after transitioning from the Wired Equivalent Privacy (WEP) standard to Wireless Protected Access 2 (WPA2) [6], there still exists vulnerabilities that allow an adversary to gain access to the network through publicly available password cracking tools such as Aircrack-ng [7] and Cain [9]. One of the vulnerabilities of many IoT devices is that they send and receive data in the clear, allowing an attacker with Wi-Fi access to passively sniff the network traffic.

In this study, we reverse engineer one of the most popular wireless cameras on Amazon to illustrate its vulnerability to eavesdropping. Numerous studies, including those of [10]-[12], have demonstrated vulnerabilities that exist in network cameras. In particular, Ostrom and Sambamoorthy [11] showcase a series of attacks that can be launched against IoT cameras via Address Resolution Protocol (ARP) cache poisoning, a common technique to eavesdrop on network traffic between hosts [13]. This study highlights the pervasiveness of eavesdropping via passive network scans 10 years after the DEFCON talk.

Over the years, researchers have sought out ways to mitigate the inherent security threats present in IoT networks. These approaches include Local Area Network (LAN) management schemes via Software Defined Networks (SDN) [14][15], deployment of encryption gateways [16][17], and obfuscating network traffic by sending crafted traffic [18]. While the SDN approach prevents a compromised device or a malicious host from further attacks, it does not prevent a bystander from passively eavesdropping on the network traffic. Using encryption gateways prevents an adversary from eavesdropping on sensitive data such as those of IoT cameras. However, the proposed methods of [16] utilize a cloud architecture, which does not provide an end-to-end protection of the communication. The recently proposed edge computing approach of [17] implements security agents with greater computational capacity on edge devices such as a wireless router. But the framework requires the modification of the IoT device's existing protocols. Lastly, the authors of [18] demonstrate the feasibility of IoT device fingerprinting from encrypted Wi-Fi traffic; they are able to infer the duration and time in which a user is present in a smart home. They defeat device fingerprinting and information leakage in a smart home by spoofing Wi-Fi traffic to mimic the IoT devices using a Raspberry Pi. This study presents IoTAMU, a defense a model that couples encryption agents and network traffic spoofing to enhance the confidentiality of an IoT network.

This research provides the following contributions:

- We exploit an eavesdrop vulnerability in a popular IoT camera
- We present an automated tool to extract the H.264 video stream from network traffic
- We introduce Internet of Things Active Management Unit (IoTAMU): a data confidentiality model for IoT networks that performs network traffic obfuscation and application level encryption

The rest of the paper is structured as follows: Section 2 describes the threat model in which the experiment was designed, and Section 3 presents the vulnerabilities found in an IoT camera and the results of reverse engineering its proprietary protocol. Section 4 presents the design of IoTAMU. Finally, Section 5 concludes the paper and discusses future work.

II. Threat Model

The threat model of this study consists of the following:

- A smart home network set up by a user which consists of a central router acting as the Access Point (AP) [19] to connect different IoT devices via Wi-Fi secured with WPA2
- A user accessing the video feed from an IoT camera at a remote location via an application provided by the vendor
- An adversary in proximity to the smart home who has gained access to the network by cracking the WPA2 preshared key and passively sniffing the network [20][21]

Other modes of wireless communications for IoT such as Zigbee [22], and Bluetooth [23] exist, but they are out of scope of this study.

After gaining access to the network, the adversary passively sniffs the network traffic and analyzes the data without detection with Wireshark [24]. Because many IoT devices send unencrypted data over the network [5], the adversary collects sensitive information without authorized access to the device. As highlighted in [25], there are numerous attacks that an adversary can perform after gaining access to a target network. However, this study focuses on compromise of data confidentiality as a result of eavesdropping in an unprotected network.

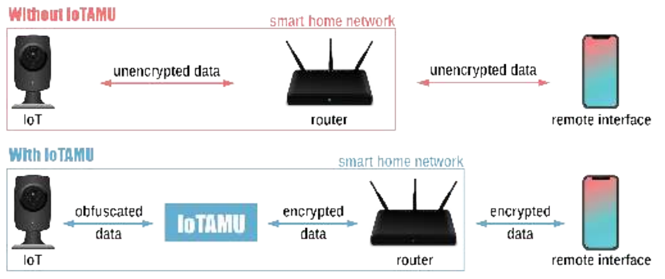

Figure 1. Communication in smart home networks with and without IoTAMU.

The primary goal of IoTAMU is to protect sensitive data in transit to and from an IoT device. Figure 1 depicts a smart home network with and without the use of IoTAMU. As previously mentioned, a typical smart home network is vulnerable to eavesdropping as it exchanges unencrypted data. IoTAMU is a security agent located between the devices and the router, encrypting their communication. It is paired with a decryption agent on the other end of the communication that sits on the device the user uses to interact with the IoT devices. The IoTAMU also performs periodic spoofing to deter device fingerprinting and obfuscate the network data from sniffers.

III. Investigating the vulnerabilities of an IoT camera

As a proof of concept, we investigate the vulnerability of the network protocol for Wansview Wireless 1080P Security Camera model Q3 being sold on Amazon [26]. As of September 18, 2019 it holds the Amazons Choice label on the website for the keywords "wi-fi baby monitor" with more than 3,000 customer reviews of average 4.1 out of 5-star rating scale. A comprehensive list of tools used in this work is summarized in Table 1.

A. *Experimental Setup*

As shown in Figure 2, the IoT camera is connected to the router via Wi-Fi and communicates with the vendor application (Wansview) running on an Android device (Samsung Galaxy S8) over the 4G network provided by the cellular provider (T-Mobile in this case). In proximity to the smart home is an adversary on a laptop (Lenovo Thinkpad) running Kali as its operating system. An Alfa wireless card is connected to the laptop to capture the network traffic in the smart home. The captured data is then passed into Wireshark for decoding and analysis. There are methods publicly available to gain root access to the camera with its admin credentials [27], which has been verified by the authors. However, this study focuses on passive network sniffing which does not require direct interaction with the device.

B. *Sniffing the Network Traffic from IoT*

To sniff the network traffic, the Alfa card connected to the laptop was first set to monitor mode via Airmon-ng [7]. Then, Airodump-ng was used to identify the Internet Protocol (IP)

TABLE I. LIST OF TOOLS USED

Name	Version	Description
Motorola Router	MG7540	Router that connects the IoT camera to the Internet via Wi-Fi
Lenovo Thinkpad	W541 I7-4910MQ (Kali 2018.4)	Laptop used for sniffing network traffic
Wansview Wireless Camera	Q3S (X Series)	IoT camera
Samsung Galaxy S8	SM-G950U1 (Android version 9)	Smartphone to remotely control the camera
Alfa Card	AWUS036NHA	Wireless network interface controller to send and receive 802.11 traffic
Wireshark	2.6.8	Software used for packet analysis
Airodump-ng	1.5.2	Software used to capture network traffic
Aireplay-ng	1.5.2	Software used to inject network traffic
Airmon-ng	1.5.2	Software used to configure the wireless card for network sniffing
Wansview	1.0.16	Mobile application to interact with IoT Camera
Python	3.5.2	Programming language used for automated H.264 video extraction

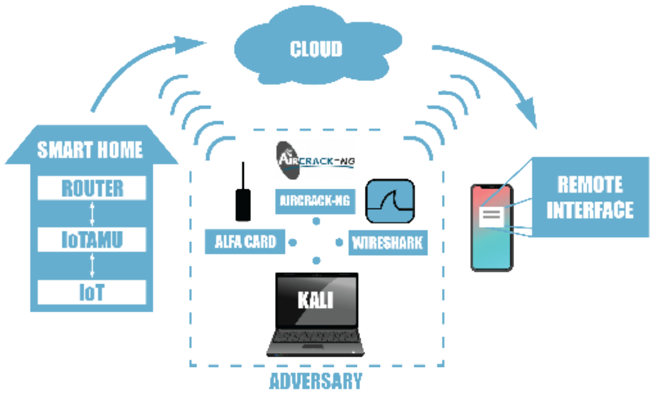

Figure 2. An overview of the experimental setup.

[8] address of the router acting as the AP. Next, Airodump-ng is used again to record network traffic associated with the target router, and Aireplay-ng was used to send spoofed deauthentication messages to the IoT camera to capture the WPA 4-way handshake between the router and the camera. The messages in the handshake are used by Wireshark along with the WPA2 preshared key to decode the encrypted Wi-Fi messages. For the purpose of this experiment, it is assumed that the adversary has gained access to the WPA2 preshared key. While gathering network data, the user in a remote location accessed the camera feed through the mobile application. After a period of time, the sniffer was stopped and the recorded data was viewed in Wireshark for analysis. In order to view the encrypted Wi-Fi data in Wireshark, the preshared key for WPA2 was input under the IEEE 802.11 decryption key setting.

C. Decoding the Video Stream Protocol

When the video stream was initiated from the remote user, the camera first performed a Domain Name System (DNS) [28] lookup of its cloud server followed by a series of network discovery protocols including Simple Service Discovery Protocol (SSDP) [29]. Its primary transport protocol was User Datagram Protocol (UDP), which is often used for transportation of time-sensitive data like video streams [19].

Determining the initial start time of video stream was clear within Wireshark, highlighted by the jump in the length of payload from mostly sub-100 range to 1,032 bytes. At first glance, Wireshark was unable to determine the type of data being transmitted. However, exporting one of the 1,032 payload and analyzing its entropy showed a steep downward slope suggesting that the payload was not encrypted (Figure 3). Upon further inspection, the first packet of the stream contained the file signature of a JPEG image, which can be recognized by the characters “JFIF”. The subsequent packets showed that the video was transferred as an H.264 [30] stream, suggesting that the initial JPEG image corresponds to the thumbnail image shown in the Android application.

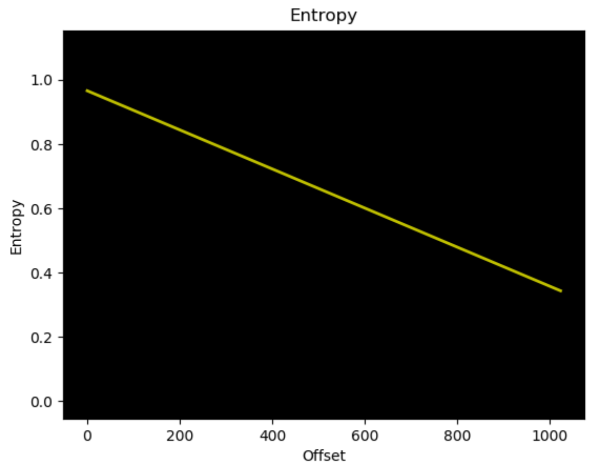

Figure 3. Entropy of a sample payload during video stream.

Once the video stream began, on top of the stream data, the UDP payloads also contained various control information consisting of a 4-to-8-byte block header and an optional block that varied from 0 to 40 bytes. Figure 4 illustrates the breakdown of the proprietary protocol in all packets. The first byte of the payload was a fixed value of `0xf1`, which was followed by either `0xd0`, `0xd1`, `0xe0`, `0xe1`. The control byte `0xd0` was used in payloads with stream data, whereas `0xd1` served as acknowledgement packets analogous to the Transmission Control Protocol (TCP) [19] counterpart. `0xe0`, and `0xe1` were sent out by both the server and the client, always followed by a 2-byte zero padding, representing a keep alive signal to leave the stream open to prevent replay attacks. The next two varying bytes represented the length of the payload following the two bytes (i.e., length of UDP payload in bytes - 4 bytes). They were followed by a `0xd100` for any data part of the JPEG image, `0xd101` for H.264 video stream, and `0xd102` for MPEG Audio Data Transport Stream (ADTS) [31] data. The final two bytes in the 8-byte header represented

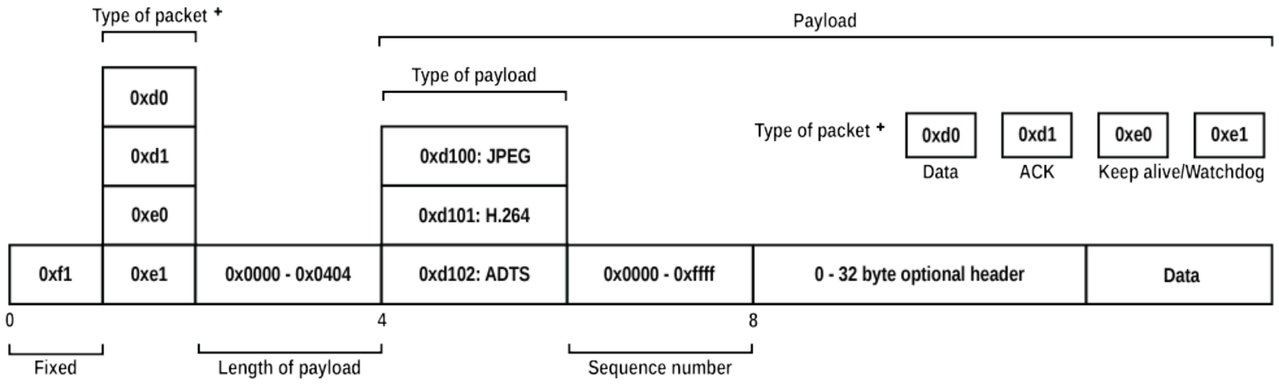

Figure 4. Breakdown of a UDP packet payload during camera stream.

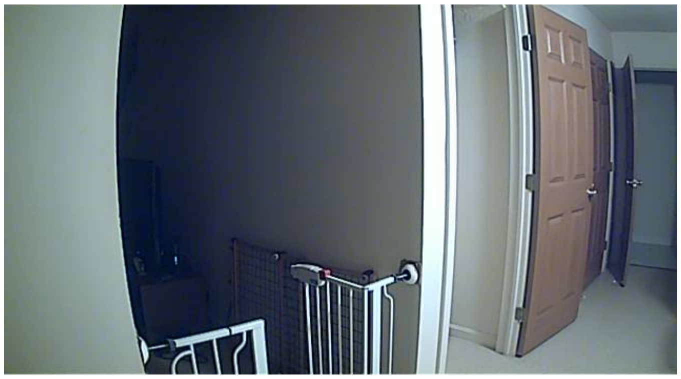

Figure 5. A snapshot of the reconstructed video stream.

the sequence number of the data to follow in a data stream packet, or the number of packets being acknowledged in an ACK packet.

There were a few variations of the optional overhead following the 8-byte header depending on the type of packet. However, for the purpose of extracting the video feed, we were only required to determine that the optional header for the packet containing the JPEG header was 8 bytes, and the optional header for the packet containing the H.264 and the audio header was 32 bytes. Any stream data directly following the initial headers did not contain optional headers. There were, however, optional 32-byte headers for H.264 packets that were not the first in the series of packets. These headers could be distinguished by the `0x55aa` sequence following the first 8-byte header, and pertinent data could be correctly extracted by filtering for the specific sequence.

The aforementioned header information was used to build an automated H.264 videos stream extraction tool written in Python [32] to recreate the video stream (Figure 5) from a pcap capture file without having physical access to the camera or its credentials. The tool was able to extract a recognizable video stream from the pcap file in 100 percent of the test cases.

Due to the inconsistent nature of Wi-Fi traffic and the unreliability of UDP, the reconstructed video feed was not a perfect replication of the video data stream sniffed en route to the mobile application. As previously identified in [5], the eavesdrop vulnerability found in this research showcases its pervasiveness in IoT devices. This security flaw can easily be taken advantage by a malicious insider or a determined adversary; it warrants further research to mitigate this vulnerability.

IV. IoTAMU Design

To mitigate the eavesdrop vulnerability created by unencrypted application-level traffic, we propose IoTAMU, a central network gateway for IoT. Its setup in a typical smart home is depicted in Figure 6. In an end-to-end communication involving a smart home, as shown in Figure 2, there are two ends of traffic an adversary can capture: data exchanged between the IoT device and the cloud, and those between the remote interface and the cloud. However, the easier of the two end hosts is the smart home end of the communication, because the IoT devices are often stationary, and remain static in the network.

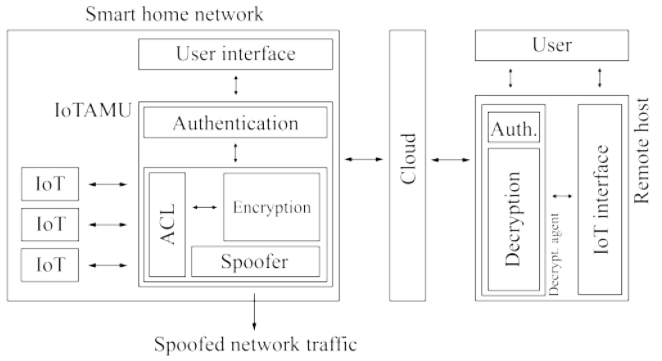

Figure 6. An overview of the IoTAMU model in a smart home network.

IoTAMU will protect this vulnerable end of the communication by serving as the network proxy to all IoT devices present in the network to encrypt their application level traffic before forwarding it to the router. The proxy can be easily set up in the network by designating it as the default gateway for the IoT devices through the Dynamic Host Configuration Protocol (DHCP) [33] in the LAN. The encryption agent within IoTAMU will be paired with a decryption agent living on the other host as a background process, often with enough computational capacity to perform encryption and decryption (e.g., computer, smartphone) by itself. IoTMU will consist of the following capabilities: (1) Authentication, (2) Access control list (3) Encryption, and finally (4) Spoofing.

The encryption agents will be accessed and configured (i.e., add or remove devices) through an authentication mechanism via username and password set by the user. In addition, to ensure only the intended devices are communicating with it, IoTAMU will store an access control list based on the network signatures of the IoT devices such as Media Access Control (MAC) address [34], IP address, etc. This does not in fact prevent an adversary from spoofing one of the IoT devices to communicate with the gateway. But since the intention of the gateway is data confidentiality, its vulnerability to spoofing will not affect its functionality.

Similar to the approach taken by [16], the encryption can be performed through Public Key Encryption (PKI). This requires the exchange of keys between IoTAMU in the smart home network and each of the encryption agents residing in the other end host. The encryption agents for the end hosts can take the form of a smart phone application that runs in the background or an executable on a computer. Initially, each participating hosts will exchange their public keys in a certificate signed by a common entity whose public key will be preinstalled on the agents. All subsequent traffic will be encrypted and decrypted using the private key and the public key of the end hosts. The use of PKI will prevent a bystander from intercepting a common key to decrypt the messages. Although taking an approach like SSL, which uses PKI to exchange session keys and using symmetric keys for later exchanges may lessen the computational demand, it is vulnerable to man-in-the-middle attacks [13], which defeats the purpose of IoTAMU.

Finally, the IoTAMU will periodically send out spoofed Wi-Fi traffic to mimic certain device types. Spoofing network traffic substantiated by the research in [18] will help fortify the network against fingerprinting and information leakage in a smart home. It will also protect the unencrypted data being exchanged between IoTAMU and the IoT devices by concealing the actual communication among spoofed traffic. Creation of spoofed packets can be variations on the following criteria:

- Length of the payload in the network
- Frequency and timing of the packets sent
- Spoofing unused IP address in the network
- Spoofing a plausible MAC addresses of certain vendors to mimic device types

Using the aforementioned criteria, a central IoT gateway design will help secure a smart home network without changing any existing protocols or relying on the vendors for security. The latency and the packet overhead imposed by encryption and the effect of periodic spoofing on network congestion is left for future investigation.

V. Conclusion

This paper discusses the vulnerabilities of IoT devices in a smart home network. We demonstrated the eavesdrop vulnerability in the Wansview IoT camera by reverse engineering its proprietary communication protocol, and by creating an automated tool to extract the H.264 stream created by the camera. The proposed design of IoTAMU mitigates this vulnerability for the IoT devices in smart homes through encryption and spoofing. As it does not rely on any existing protocols, IoTAMU can be implemented for the varying protocols utilized by the IoT devices to easily deploy in smart home networks. Development of a Wireshark dissector for the proprietary protocol, the implementation of IoTAMU, and the analysis of its performance is left as future work.

Acknowledgment

The views expressed are those of the authors and do not necessarily reflect the official policy or position of the Air Force, the Department of Defense, or the U.S. Government.

References

[1] P. P. Ray, “A Survey on Internet of Things Architectures,” J. King Saud Univ. - Comput. Inf. Sci., vol. 30, no. 3, pp. 291-319, 2018.

[2] M. Torchia and M. Shirer, “IDC Forecasts Worldwide Technology Spending on the Internet of Things to Reach $1.2 Trillion in 2022,” International Data Corporation, 2018, URL: https://www.idc.com/getdoc.jsp?containerId=prUS43994118 [accessed: 2019-09-18].

[3] United States Government Accountability Office, “INTERNET OF THINGS Enhanced Assessments and Guidance Are Needed to Address Security Risks in DOD,” 2017, URL: https://www.gao.gov/assets/690/686203.pdf [accessed: 2019-10-21]

[4] C. Valasek and C. Miller, “Remote Exploitation of an Unaltered Passenger Vehicle,” Black Hat USA, pp. 1-91, 2015.

[5] Open Web Application Security Project, “IoT Attack Surface Areas,” 2015, URL: https://www.owasp.org/index.php/IoT_Attack_Surface_Areas [accessed: 2019-09-18].

[6] A. Sari and M. Karay, “Comparative Analysis of Wireless Security Protocols: WEP vs WPA,” Int. J. Commun. Netw. Syst. Sci., vol. 08, no. 12, pp. 483-491, Dec. 2015.

[7] “Aircrack-ng,” 2018, URL: https://www.aircrack-ng.org [accessed: 2019-09-18]

[8] J. Postel “Internet Protocol,” RFC 791, 1981, URL: https://tools.ietf.org/html/rfc791 [accessed: 2019-09-18]

[9] “Cain and Abel,” 2014 [Online]. Available from: https://www.oxid.it/cain.html [accessed: 2019-09-18]

[10] C. Heffner, “Exploiting Surveillance Cameras Like a Hollywood Hacker,” no. February, p. 1-30, 2013.

[11] J. Ostrom and A. Sambamoorthy, “DEFCON 17: Advancing Video Application Attacks with Video Interception, Recording, and Replay,” 2009, URL: https://www.youtube.com/watch?v=QcsQ6UzMJiU [accessed: 2019-09-18]

[12] N. Kalbo, “I Got My EyeOn You - Security Vulnerabilities in D-Links Baby Monitor,” 2018, URL: https://dojo.bullguard.com/dojo-by-bullguard/blog/i-got-my-eyeon-you-security-vulnerabilities-in-baby-monitor/ [accessed: 2019-09-18]

[13] E. Skoudis and T. Liston, Counter hack reloaded: a step-by-step guide to computer attacks and effective defenses, Prentice Hall Press, 2005.

[14] M. Miettinen et al., “IoT Sentinel Demo: Automated Device-Type Identification for Security Enforcement in IoT,” Proc. - Int. Conf. Distrib. Comput. Syst., pp. 2511-2514, 2017.

[15] D. Soteris et al., “SDN-driven protection of smart home WiFi devices from malicious mobile apps,” Proc. 10th ACM Conf. Secur. Priv. Wirel. Mob. Networks, pp. 122-133, 2017.

[16] C. Doukas, I. Maglogiannis, V. Koufi, F. Malamateniou, and G. Vassilacopoulos, “Enabling data protection through PKI encryption in IoT m-Health devices,” IEEE 12th Int. Conf. Bioinforma. Bioeng., pp. 25-29, 2012.

[17] R. Hsu, J. Lee, T. Q. S. Quek, and J. Chen, “Reconfigurable Security: Edge Computing-based Framework for IoT,” IEEE Network., vol. 32, no. 5, pp. 92-99, 2018.

[18] S. M. Beyer, B. E. Mullins, S. R. Graham, and J. M. Bindewald, "Pattern-of-Life Modeling in Smart Homes," IEEE Internet Things J., vol. 5, no. 6, pp. 5317-5325, Dec. 2018.

[19] J. F. Kurose and K. W. Ross, Computer networking: a top-down approach, 7th ed. New Jersey: Pearson, 2017.

[20] O. Nakhila and C. Zou, "Parallel Active Dictionary Attack on IEEE 802.11 Enterprise Networks," in MILCOM IEEE Military Communications Conference, 2016, pp. 265-270.

[21] A. Abdelrahman, H. Khaled, E. Shaaban, and W. S. Elkilani, "WPA-WPA2 PSK Cracking Implementation on Parallel Platforms," in 2018 13th International Conference on Computer Engineering and Systems (ICCES), 2018, pp. 448-453.

[22] Zigbee Alliance, "What is Zigbee?," 2018, URL: https://www.zigbee.org/what-is-zigbee/ [accessed: 2019-09-18]

[23] Bluetooth SIG, "Protocol Specifications," 2019, URL: https://www.bluetooth.com/specifications/protocol-specifications/ [accessed: 2019-09-18]

[24] "Wireshark," 2019, URL: https://www.wireshark.org/ [accessed: 2019-09-18]

[25] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Internet of Things (IoT): Taxonomy of security attacks," in 2016 3rd International Conference on Electronic Design, 2016, pp. 321-326.

[26] "Wansview Wireless 1080P Security Camera, WiFi Home Surveillance IP Camra for Baby/Elder/Pet/Nanny Monitor, Pan/Tilt, Two-Way Audio & Night Vision Q3-S," Amazon, 2019, URL: https://www.amazon.com/Wansview-Wireless-Security-Surveillance-Monitor/dp/B075K89NTR/ref=sr_1_14?keywords=wi-fi+baby+monitor&qid=1568920347&s=gateway&sr=8-14 [accessed: 2019-09-18]

[27] J. Wedell, "Rooting a cheap IP camera (Wansview K2)," Ramblin Wedells, 2017, URL: https://jonwedell.com/rooting-a-cheap-ip-camera/ [accessed: 2019-09-18]

[28] P. Mockapetris, "Domain names - implementation aqnd Specification," RFC 1035, 1987, URL: https://www.ietf.org/rfc/rfc1035.txt [accessed: 2019-09-18]

[29] A. Donoho et al., "UPnP Device Architecture," UPnP Forum, 2015, URL: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v2.0.pdf [accessed: 2019-09-18]

[30] "H.264: Advanced video coding for generic audiocisual services," International Telecommunication Union, 2019. [Online]. Available: https://www.itu.int/rec/T-REC-H.264 [accessed: 2019-09-18]

[31] "White Paper on AAC Transport Formats" International Organisation for Standardisation, 2014, URL: https://mpeg.chiariglione.org/white-papers [accessed: 2019-09-18]

[32] "Python," Python, 2019, URL: https://www.python.org/ [accessed: 2019-09-18]

[33] R. Droms "Dynamic Host Configuration Protocol," RFC 2131, 1997, URL: https://tools.ietf.org/html/rfc2131 [accessed: 2019-09-18]

[34] "Standard Group MAC Addresses: A Tutorial Guide," IEEE Standards Association, 2019, URL: https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tutorials/macgrp.pdf [accessed: 2019-09-18]