SecWIR: Securing Smart Home IoT Communications via Wi-Fi Routers with Embedded Intelligence

Xinyu Lei, Guan-Hua Tu
{leixinyu,ghtu}@msu.edu
Michigan State University
East Lansing, Michigan

Chi-Yu Li
chiyuli@cs.nctu.edu.tw
National Chiao Tung University
Hsinchu City, Taiwan

Tian Xie, Mi Zhang
xietian1@msu.edu;mizhang@egr.msu.edu
Michigan State University
East Lansing, Michigan

ABSTRACT

Smart home Wi-Fi IoT devices are prevalent nowadays and potentially bring significant improvements to daily life. However, they pose an attractive target for adversaries seeking to launch attacks. Since the secure IoT communications are the foundation of secure IoT devices, this study commences by examining the extent to which mainstream security protocols are supported by 40 of the best selling Wi-Fi smart home IoT devices on the Amazon platform. It is shown that 29 of these devices have either no security protocols deployed, or have problematic security protocol implementations. Seemingly, these vulnerabilities can be easily fixed by installing security patches. However, many IoT devices lack the requisite software/hardware resources to do so. To address this problem, the present study proposes a SecWIR (**Sec**ure **Wi**-Fi **I**oT communication **R**outer) framework designed for implementation on top of the users' existing home Wi-Fi routers to provide IoT devices with a secure IoT communication capability. However, it is way challenging for SecWIR to function effectively on all home Wi-Fi routers since some routers are resource-constrained. Thus, several novel techniques for resolving this implementation issue are additionally proposed. The experimental results show that SecWIR performs well on a variety of commercial off-the-shelf (COTS) Wi-Fi routers at the expense of only a small reduction in the non-IoT data service throughput (less than 8%), and small increases in the CPU usage (4.5%~7%), RAM usage (1.9 MB~2.2 MB), and the IoT device access delay (24 ms~154 ms) while securing 250 IoT devices.

CCS CONCEPTS

• Security and privacy → Mobile and wireless security; **• Networks → Wireless access points, base stations and infrastructure**.

KEYWORDS

Wi-Fi, Smart Home, IoT, Security

1 INTRODUCTION

Motivation: With the increasing popularity and sophistication of wireless smart home IoT devices nowadays (e.g., smart sockets, smart bulbs, and smart cameras), smart home systems are gradually moving into the mainstream. The number of IoT devices is forecasted to grow from 115 million in 2018 to 320 million in 2020 with a compound annual growth rate of 40.65% [51]. Of the many different types of wireless IoT devices available (including cellular, ZigBee, and SigFox), the Wi-Fi-connected devices are particularly popular among home users. For example, according to the Amazon selling records, seven of top 10 best sellers of electrical outlet switches are Wi-Fi-connected devices. A recent report [54] also forecasts that the number of global Wi-Fi devices in homes will reach 17 billion in 2030 from 4 billion in 2019, and 60% of them are smart home devices. Compared with other types of IoT devices, Wi-Fi-connected devices have two significant advantages. First, they do not require the users to purchase additional IoT vendor home gateways/hubs, such as Samsung SmartThing Hub, to connect to and access them (i.e., they depend only on the users' existing Wi-Fi routers at homes). Second, Wi-Fi IoT devices are usually much cheaper than the alternatives. For example, a Samsung SmartThing outlet costs $35, whereas an Etekcity Wi-Fi smart outlet has a cost of less than $10. Thus, Wi-Fi-connected IoT devices offer users a high degree of convenience and functionality at only a moderate price.

In this study, we focus on the Wi-Fi-connected IoT devices that do not require any IoT security gateway/hub, since they are much more popular than those that need to pair with the gateway/hub in smart homes. Despite the many advantages which IoT devices can bring to daily life, they also offer an appealing target to malicious adversaries seeking to launch cyberattacks, such as phishing, identity theft, and distributed denial of service (DDoS). Consequently, the security of IoT communication is an important concern. Unfortunately, our study on 40 best-selling smart home IoT devices on the Amazon platform, yields a negative answer. We have two findings. First, many smart home IoT devices do not support any security protocols, and hence data confidentiality and integrity protection are not supported for the communications between these devices (referred to henceforth as **NonSecIoT** devices) and IoT servers. Second, while a small number of devices do offer some form of security protection, the related protocols are not compliant with standards and fail to protect the devices (referred to henceforth as **InSecIoT** devices) from malicious attacks. By exploiting these vulnerabilities, adversaries can launch a variety of attacks, including (but not limited to) remotely controlling users' appliances and capturing users' real-time images/videos, as shown in Figures 2 and 3, respectively. The results are summarized in Table 1.

At first glance, IoT vendors are guilty of ignoring security issues in marketing and distributing their products. Moreover, it seems intuitive that vendors could easily overcome the problem by simply patching their servers and shipped devices. In practice,

MobiSys '20, June 15–19, 2020, Toronto, ON, Canada
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7954-0/20/06…$15.00
https://doi.org/10.1145/3386901.3388941

however, the situation is far more complex. For example, of the 40 smartphone-controlled IoT devices mentioned above, 38 of the related smartphone control applications were found to communicate with the IoT servers through SSL/TLS security protocols. In other words, most IoT vendors do in fact support security protocols in their communication infrastructures between the IoT control applications and the IoT servers. The question therefore arises as to why most IoT vendors do not deploy any security protocols on the IoT devices themselves. There appear to be two possible reasons for this. First, *Wi-Fi-connected IoT devices simply lack the resources required to deploy mainstream security protocols*. For example, one popular IoT platform, Delta DFCM-NNN40, has only 256 KB flash memory [35]. Supporting SSL/TLS protocols for IoT devices requires as a minimum a lightweight SSL/TLS library and a list of trusted certificate authorities. The former library (e.g., WolfSSL [55]) needs around 20 KB-100 KB flash memory space, while the latter requires 250 KB [16]. That is, a minimum memory space of 270 KB is required, which exceeds the 256 KB available on the DFCM-NNN40 platform. Second, *not all IoT devices support remote software/firmware updates*. Consequently, it is not easy for IoT vendors to patch these security vulnerabilities of their devices without expensive recalls. We thus believe that there is a pressing need to develop novel approaches to secure these IoT devices; otherwise, they may be abused to launch a variety of cyberattacks.

Existing Technologies and their Limitations: The problem of securing IoT devices has been extensively examined in recent years. Broadly speaking, existing proposals for IoT device security can be categorized as either device-based approaches [24, 31, 33, 39, 52] or infrastructure-based approaches (e.g., IoT security gateways [15, 57], customized secure systems [36], in-hue secure manager [48], and customized securebox [27]). The device-based approaches require the after-market IoT devices to possess the capabilities required to deploy the proposed security mechanisms. Thus, they may not be suitable for all IoT device platforms; particularly those with resource constraints. Meanwhile, the infrastructure-based approaches typically require the users to purchase additional security hardware components, the cost of which may deter the users, to secure the IoT devices; hence, they also have only a limited practical applicability. Moreover, some IoT security gateways (e.g., Samsung SmartHome and Philips Hue) only support certain IoT devices.

Proposed Approach: We thus develop a framework designated as **SecWIR** (**S**ecure **W**i-Fi **I**oT communication **R**outer) to provide smart home IoT devices with secure IoT communications through commercial off-the-shelf (COTS) home routers. The development is based on two key rationales. First, most smart home IoT devices allow users to access them remotely through Internet via smartphone applications, and all of the IoT commands destined to, or the responses sent from, the smart home IoT devices pass through the user's home router. Consequently, the home router represents an ideal choke point from which to monitor and analyze all of the incoming and outgoing IoT traffic, and protect the associated IoT devices without modifying them. Second, by leveraging the embedded computing resources of the home router, the proposed mechanism does not require the users to purchase any additional security hardware, and hence the deployment cost is reduced; thereby improving the likely take-up of the proposed framework.

Our proposed framework has three key advantages over the existing solutions: (A1) SecWIR does not require IoT devices to be recalled or modified. (A2) SecWIR does not require users to purchase any additional IoT security gateways to secure their smart home IoT devices. It is still an infrastructure-based solution, but is built on the users' existing Wi-Fi routers. (A3) SecWIR offers substantial benefits to both IoT vendors and IoT users. In particular, IoT vendors can provide customers with secure IoT communications offered by SecWIR, and thus continue to use low-cost IoT platforms without sacrificing their profits. On the other hand, IoT users can keep using cheap IoT devices while securing them with SecWIR.

Challenges: To expand the IoT market and roll out various IoT applications, inexpensive IoT devices have become the mainstream. However, it is challenging to secure these low-cost IoT devices with limited resources through the proposed SecWIR framework. The design of SecWIR needs to address two key challenges. (1) COTS Wi-Fi routers are heterogeneous, and not all of them have sufficient resources to support IoT security functions. For high-end routers, deploying mainstream security protocols on top of the router seems technically straightforward. However, not all the resources of the routers can be used for IoT security functions since they usually need to support rich data services (e.g., DLNA media server, iTunes server, and ReadyCloud server [38]) which may consume many resources. For resource-constrained low-end routers, the problem is far more challenging. For example, the Linksys WRT400N router has 32 MB RAM for both its operating system and other local applications. After it is booted up, only 7 MB RAM is left for secure IoT communications. Furthermore, a user may have multiple IoT devices and it is technically difficult to secure a great number of devices and prevent user-perceived IoT access delays, due to resource constraints. (2) COTS Wi-Fi routers are designed to perform the efficient routing of data packets, not to implement and support security functions for IoT devices. Thus, largely preserving the original performance of the non-IoT data services while also securing IoT devices is far from trivial.

Our Solutions: To address these challenges, we propose four novel mechanisms for making more efficient use of the routers' resources to accomplish secure IoT communications. (1) *IoT-specific SSL/TLS tunneling* saves resources by only allowing NonSecIoT devices to use the secure IoT communications while preventing the access from other devices. The reason why SecWIR adopts the SSL/TLS tunneling mechanism instead of the IPSec tunneling is that the latter consumes more resources [9, 19] (§5.1.1). (2) *Priority-based SSL/TLS tunneling management with user-perceived IoT access augmentation* saves resources by using a suite of novel mechanisms including priority-based tunneling management, traffic-outlier detection, and cross-application detection. It provides secure IoT communications for a great number of IoT devices with only a limited number of active SSL/TLS tunnels, while largely preserving user experience of accessing IoT devices (§5.1.2). (3) *Stream security validation* secures InSecIoT devices with flawed security protocol implementations using a lightweight stream processing approach (§5.2). (4) *Resource monitoring* monitors the real-time resources (e.g., CPU and RAM) of the Wi-Fi routers and the user packet routing performance (e.g., the packet drop rate and throughput), and then dynamically assigns/frees resources to/from SecWIR as required (§5.3).

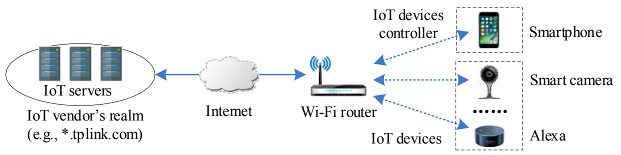


Figure 1: Wi-Fi smart home IoT service model.

Contributions: This study makes three key contributions:

- An investigation is performed into the mainstream security protocol support offered by 40 of the popular Wi-Fi IoT devices available nowadays, as grouped into four main application categories, namely remote control, automation, appliance, and security. The results show that 23 of 40 devices do not support any SSL/TLS protocols, while 6 devices have a flawed security protocol implementation.
- A framework, designated as SecWIR, is developed to secure Wi-Fi home IoT devices. For NonSecIoT devices, SecWIR provides secure communications with the IoT servers via a novel IoT-specific, resource-aware SSL/TLS tunneling method. For InSecIoT devices, SecWIR examines the compliance of the deployed security protocol standards and then effectively precludes communications between the InSecIoT devices and non-authentic IoT servers. With SecWIR, IoT users do not need to purchase expensive IoT devices to achieve secure IoT communications, and IoT vendors do not have to sacrifice profits for improving the user security.
- SecWIR is implemented on 5 COTS Wi-Fi home routers using OpenWrt, which is popular and has been used by more than 1300 router models with more than 200 vendors. It is shown that even on resource-constrained routers, SecWIR introduces only a small reduction of the non-IoT data service throughput (less than 8%), and a small increase of the CPU usage (4.5%~7%), the RAM usage (1.9 MB~2.2 MB), and the IoT access delay (24 ms~154 ms), while providing 250 IoT devices with secure IoT communications between them and their servers.

Paper organization: The remainder of this paper is organized as follows. §2 presents the related work. §3 introduces the Wi-Fi IoT service model and illustrates the security failings of Wi-Fi IoT device communications. §4 introduces the threat model, assumptions, and security guarantees underlying the development of the proposed SecWIR framework. §5 and §6 present the design details and the security analysis of the SecWIR framework, respectively. §7 implements and evaluates SecWIR, §8 discusses some potential issues and applications, and §9 concludes the paper.

2 RELATED WORK

IoT security vulnerabilities: Many works study IoT security vulnerabilities and fall into four main categories: IoT devices [8, 22, 26, 59], IoT infrastructure [45], IoT access technologies (e.g., cellular, Wi-Fi) [11], and IoT control applications [17, 18, 20]. For the IoT device security, Alrawi et al. [8] evaluated the security of 45 home-based IoT devices, with particular emphasis on vulnerable services, weak authentication, and problematic default configurations. For the infrastructure security, the authors in [45] studied the Philips Hue service infrastructure and devised a new attack vector to infect many IoT devices via the distribution of worms. For the IoT access security, Aras et al. [11] examined the vulnerabilities of LoRa IoT networks. Finally, several studies [17, 18, 20] explored the vulnerabilities of IoT control applications, which allowed malicious adversaries to control their victims' IoT devices. This study is motivated by two factors. First, ensuring secure IoT communications lies at the very heart of providing users with secure IoT services since, in the event of any security breach, adversaries can launch remote attacks against the IoT device, the IoT device owner, or even other people. Second, to the best of the authors' knowledge, the security of Wi-Fi home-based IoT devices has thus far received only relatively little attention from academia and industry. While Alrawi et al. [8] investigated the security of 45 home-based IoT devices, only 12 of these devices are Wi-Fi connected and most of them were not the best sellers on Amazon. Furthermore, Alrawi and his research group focused mainly on vulnerabilities detection, whereas the present study focuses on solution development.

IoT security solutions: Previous studies on IoT security can be categorized into either device-based approaches (e.g., [43, 47, 58]) or infrastructure-based approaches (e.g., [23, 27, 28, 36, 48, 49]). In the former type of approaches, several light-weight security protocols were proposed to enable the secure communications on resource-constrained devices, such as Lithe [43], LSec [47], and a RC4-based security protocol [58]. The infrastructure-based approaches can be broadly classified as either *active* or *passive*, depending on the particular approach taken. Solutions of the former type secure IoT devices by adding new proprietary security servers to create customized secure systems [36], security gateways [57], in-hue secure managers [48], customized securebox [27], cloud services [28], and privacy mediators [23]. The solutions of the latter type inspect the network traffic and provide post-attack detection or defense [49].

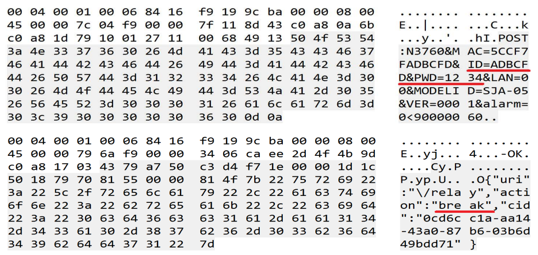
The present study differs from these approaches in three key regards. First, SecWIR provides secure IoT communications using the well-studied mainstream security protocols, whereas those proposed light-weight security protocols have not been fully studied yet and may have security vulnerabilities, e.g., using the insecure RC4. Second, SecWIR does not require users to purchase additional security equipment, but leverages instead their COTS Wi-Fi routers to facilitate the secure IoT communications. Third, SecWIR provides active real-time secure protection for the IoT communications and guarantees that adversaries cannot infer/decipher them or launch effective man-in-the-middle (MITM) attacks against IoT servers, IoT devices, and Wi-Fi routers.

3 WI-FI IOT SERVICE MODEL AND EMPIRICAL SECURITY STUDY

In this section, we first introduce the Wi-Fi-connected IoT service model and then present a security study on its IoT communication.

3.1 Wi-Fi-connected IoT service model

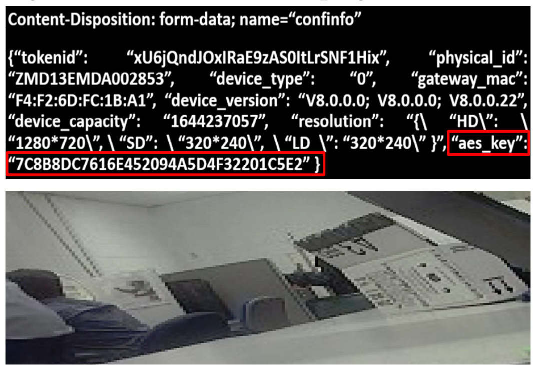
Figure 1 illustrates the service model which is commonly used by Wi-Fi-connected IoT devices nowadays. As shown, the devices, e.g., a smart camera and Amazon Alexa voice assistant, communicate with the IoT server in the vendor realm (e.g., *.tplink.com) via the Wi-Fi router and the owner communicates with the devices using a vendor-specific IoT application installed on a smartphone.



```
00 04 00 01 00 06 84 16  f9 19 9c ba 00 00 08 00   ........ ........
45 00 00 7c 04 f9 00 00  7f 11 8d 43 c0 a8 0a 6b   E..|.... ...C...k
c0 a8 1d 79 10 01 27 11  00 68 49 13 50 4f 53 54   ...y..'. .hI.POST
3a 4e 33 37 36 30 26 4d  41 43 3d 35 43 43 46 37   :N3760&M AC=5CCF7
46 41 44 42 43 46 44 26  49 44 3d 41 44 42 43 46   FADBCFD& ID=ADBCF
44 26 50 57 44 3d 31 32  33 34 26 4c 41 4e 3d 30   D&PWD=12 34&LAN=0
30 26 4d 4f 44 45 4c 49  44 3d 53 4a 41 2d 30 35   0&MODELI D=SJA-05
26 56 45 52 3d 30 30 30  31 26 61 6c 61 72 6d 3d   &VER=000 1&alarm=
30 3c 39 30 30 30 30 30  36 30 0d 0a               0<900000 60..

00 04 00 01 00 06 84 16  f9 19 9c ba 00 00 08 00   ........ ........
45 00 00 79 6a f9 00 00  34 06 ca ee 2d 4f 4b 9d   E..yj... 4...-OK.
c0 a8 17 03 43 79 a7 50  c3 d4 f7 1e 00 00 1d 1c   ....Cy.P ........
50 18 79 70 81 55 00 00  81 4f 7b 22 75 72 69 22   P.yp.U.. .O{"uri"
3a 22 5c 2f 72 65 6c 61  79 22 2c 22 61 63 74 69   :"\/rela y","acti
6f 6e 22 3a 22 62 72 65  61 6b 22 2c 22 63 69 64   on":"bre ak","cid
22 3a 22 30 63 64 36 63  63 31 61 2d 61 61 31 34   ":"0cd6c c1a-aa14
2d 34 33 61 30 2d 38 37  62 36 2d 30 33 62 36 64   -43a0-87 b6-03b6d
34 39 62 64 64 37 31 22  7d                        49bdd71" }
```

Figure 2: Hijacking attacks. Top: turn on a smart plug; bottom: turn off a smart plug.



```
Content-Disposition: form-data; name="confinfo"

{"tokenid": "xU6jQndJOxlRaE9zAS0ItLrSNF1Hix", "physical_id": "ZMD13EMDA002853", "device_type": "0", "gateway_mac": "F4:F2:6D:FC:1B:A1", "device_version": "V8.0.0.0; V8.0.0.0; V8.0.0.22", "device_capacity": "1644237057", "resolution": "{\ "HD\": \ "1280*720\", \ "SD": \ "320*240\", \ "LD \": "320*240\" }", "aes_key": "7C8B8DC7616E452094A5D4F32201C5E2" }
```

Figure 3: Spying attacks. Top: steal the AES encryption key of a camera; bottom: spy on victims.

Category	Type	Manufacturer	Model	Price	Num. of customer reviews@Amazon	Support security protocols?	Conform to security protocol standards?
Remote Control	Voice assistant	Amazon	Echo Dot	$50	117048	✓	✓
		Google	Home Mini	$49	NA	✓	✓
Automation	Smart socket	Etekcity	ESW01-USA	$10	3180	✗	–
		Belkin Wemo	F7C063	$30	10902	✗	–
		Geekbes	YM-WS-5	$9	222	✗	–
		TanTan	TANTANSMART01	$10	1006	✗	–
		TECKIN	SP10	$10	265	✗	–
	Smart strip	Foseal	1700-Joule	$33	70	✗	–
		GXA	ConsumerElec	$29	26	✗	–
		TONBUX	Powerstrip02	$30	331	✗	–
		KMC	B0781SVT8B	$25	51	✗	–
		mengyasi	B07216SSZY	$33	80	✗	–
	Smart bulb	Tp-Link	LB100	$30	2121	✗	–
		IVIEW	ISB600	$14	297	✗	–
		UPSTONE	YCL-1001	$13	152	✗	–
		Lotton	B075882X14	$17	106	✗	–
		LOHAS	B01MYQCXOH	$17	741	✗	–
	Thermometer	Honeywell	RTH6580WF	$86	2044	✗	–
		La Crosse	S85814	$32	224	✗	–
		Netatmo	NWS01-US	$127	813	✗	–
		Ecobee	ecobee4	$228	1098	✓	✓
		Emerson	ST55	$107	426	✓	✗
Appliance	Humidifier	DIKLA	B072TZDF76	$38	11	✗	–
		Essential	B07BF3MFH8	$37	74	✗	–
		RENPHO	B076VP1LPL	$30	365	✗	–
		ASAKUKI	B076F73M82	$38	81	✗	–
		Viva	B071XK49MN	$40	187	✗	–
Security	Smart camera	360	D503 HD	$40	181	✓	✓
		Zmodo	CS-S1U-WS-1	$40	4145	✓	✗
		YI Dome	720p HD	$35	4928	✓	✓
		EZVIZ Mini	CS-CV206	$40	710	✓	✗
		Funlux	CH-S1R-WA-Q3	$23	2706	✓	✗
		Logitech	961-000392	$90	407	✓	✓
		Amazon	1080p Full HD	$120	3532	✓	✓
		Nest	MAIN-99991	$166	6866	✓	✓
		Wyze	WYZEC2	$26	1481	✓	✗
	Smart video doorbell	Ding	AF-KSH001W	$60	1855	✓	✗
		AKASO	IPC010-US-NEW	$70	114	✓	✓
		Ring	720p HD	$100	27047	✓	✓
		SkyBell	SH02300SL	$148	1297	✓	✓

Table 1: Security analysis of 40 best-selling smart home Wi-Fi IoT devices at Amazon in four categories: remote control, automation, appliance, and security. –: Not Applicable.

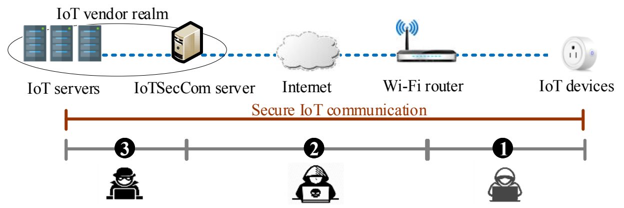


Figure 4: Schematic illustration of secure IoT communications provided by SecWIR and three possible attack sources.

3.2 Empirical security study on IoT communications

We conducted a security study on the security protocol support provided by 40 popular Wi-Fi smart home IoT devices which we purchased on the Amazon. The investigation focused on the SSL/TLS and IPSec protocols due to their widespread deployment. We observed two security vulnerabilities on the tested IoT devices, namely (V1) a lack of security protocol support and (V2) flawed certificate validation. The experimental results are summarized in Table 1.

(V1) A lack of security protocol support: Not all the Wi-Fi IoT devices support those studied security protocols. ***Validation:*** we used the `tcpdump` program on Wi-Fi routers to intercept all the IoT packets transmitted between IoT devices and IoT servers. Our study shows that 23 of the 40 devices do not support the studied cryptographic/security protocols (e.g., Etekcity, Belkin, Geekbes, and Tp-Link); that is, the packets are sent in plain-text.

(V2) Flawed Certificate Validation: Some Wi-Fi IoT devices have flaws in validating the IoT server's X.509 certificate [53] which is received during the establishment of an SSL/TLS connection with the server. Such flawed certificate validation can make the IoT devices suffer from various SSL/TLS MITM attacks. ***Validation:*** We first generated self-signed server certificates using the OpenSSL library and then provided the tested IoT devices with the fake server certificates via the SSLSplit [44] tool, which divides an SSL/TLS connection into two sub-connections and allows adversaries to conduct MITM attacks. Our study shows that 6 IoT devices mistakenly accept the forged server certificates.

Security Threats: By exploiting these two vulnerabilities, the adversary can launch a variety of IoT attacks. For example, the adversary can remotely control a victim's IoT devices. Figure 2 illustrates that we could generate IoT control messages to freely turn on/off the Etekcity smart socket, which does not support any security protocols. The adversary can also capture real-time images/videos from a victim's camera. Figure 3 demonstrates that by providing a security camera with a forged server certificate, we could discover the encryption key of the streaming video and then obtain the real-time images/videos of the victim's home from the camera.

4 THREAT MODEL, ASSUMPTIONS, AND SECURITY GUARANTEES

Threat Model: In this study, adversaries are people or organizations which launch remote attacks against victims (e.g., by taking control of their IoT devices). They are assumed to have the following capabilities. (1) They can *intercept*, *modify* or *inject* any messages in the public communication channels. Specifically, the secure IoT communication provided by SecWIR can be divided into three paths, namely the wireless path between the IoT device and the Wi-Fi router, the wired path between the Wi-Fi router and the entrance point of the IoT vendor realm, and that between the entrance point and the serving IoT server (see Figure 4). Adversaries

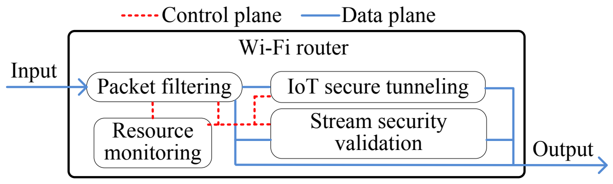


Figure 5: Architecture of SecWIR.

may launch attacks from either of these three paths. (2) The adversaries adhere to all cryptographic assumptions, e.g., an encrypted message cannot be decrypted without its decryption key.

Assumptions: SecWIR makes three basic assumptions, namely (1) IoT device vendors are willing to secure their shipped IoT devices for the sake of goodwill if the proposed remedies do not affect the existing IoT services or seriously degrade their profits; (2) the IoT vendor realms in which the IoT servers are deployed are secure; and (3) the owners of the IoT devices have at least one operational Wi-Fi home router which supports Wi-Fi Protected Access II/III (WPA2/3 [5]).

Security Guarantees: SecWIR aims to provide two security guarantees: (1) **secrecy** of all the IoT traffic exchanged between the IoT devices and the IoT vendors (i.e., the IoT packets always have ciphering protection); and (2) **integrity** of the IoT traffic such that even if the packets are intercepted, adversaries cannot use them to generate fabricated packets.

5 SECWIR DESIGN

In this section, we introduce the SecWIR design, a software-based security framework, for securing IoT devices using COTS Wi-Fi routers. SecWIR targets two particular types of IoT devices, namely NonSecIoT devices which lack support for standard security protocols, and InSecIoT devices which offer the security support, but have flawed implementations. SecWIR provides the NonSecIoT devices with mainstream security protocol support to facilitate secure communications with the IoT servers, and guards against insecure communications for the InSecIoT devices by identifying conflicts between the device behavior and the security protocol standards.

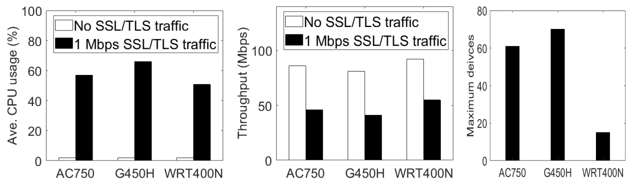
Figure 5 illustrates the architecture of SecWIR. It consists of four modules, namely (1) IoT secure tunneling, (2) stream security validation, (3) resource monitoring, and (4) packet filtering. As shown, all of the incoming packets to the Wi-Fi router first enter the packet filtering module. IoT packets are dispatched to either the IoT secure tunneling module or the stream security validation module, whereas non-IoT packets are forwarded directly to their destinations. The tunneling module enables the NonSecIoT devices to securely communicate with their IoT servers through IoT-specific, resource-aware SSL/TLS tunnels. Meanwhile, the validation module examines the SSL/TLS operations of the InSecIoT devices at run time using an efficient stream processing technique. It permits the SSL/TLS connections to be established only if their operations are verified as being compliant with the deployed security protocol standards. To prevent the normal operations of non-IoT devices from being affected, the resource monitoring module tracks the non-IoT device status and system resource statistics, and then dynamically allocate resources to the SecWIR framework in such a way as to maintain their normal operations.

We next elaborate on each module and then conduct a security analysis of the SecWIR framework.

5.1 IoT Secure Tunneling Module

To provide NonSecIoT devices with secure IoT communications, it is necessary to protect all of the packets transmitted between the IoT devices and the IoT servers. As described in §4, the secure IoT communication provided by SecWIR can be divided into three paths (see Figure 4). This section describes the tunneling module that protects the IoT traffic transferred from/to NonSecIoT devices in the second path (i.e., between the Wi-Fi router and the entrance point of the IoT vendor realm). Note that the first wireless transmission path and last wired path are both skipped here since it is assumed that the former is secured by the Wi-Fi security protocol (e.g., WPA2), and the latter is secure (see §4).

The aim of the secure tunneling module is to construct a secure SSL/TLS tunnel between the Wi-Fi router and the IoT vendor realm. SecWIR deliberately adopts an SSL/TLS tunnel rather than an IPSec tunnel since the latter protocol consumes more resources [9, 19]. The tunnel is built between two components, *IoTSecComClient* and *IoTSecComServer*, located at the two ends of the tunnel. IoTSecComClient is a SecWIR-specific security module built on the Wi-Fi router, while the IoTSecComServer is a standalone server deployed on the IoT vendor side.



(a) CPU usage (b) Throughput (c) Max IoT devices

Figure 6: Performance evaluation of conventional SSL/TLS tunnels on three routers: Buffalo G450H (400 MHz CPU/64 MB RAM), Tp-Link AC750 (580 MHz CPU/64 MB RAM), and Linksys WRT400N (680 MHz CPU/32 MB RAM).

Beyond Conventional SSL/TLS Tunneling: Seemingly, the proposed tunneling solution is similar to the conventional SSL/TLS tunneling technique. However, it presents three unique technical challenges. First, the delivery of SSL/TLS packets consumes more resources (e.g., CPU usage) than normal IP packets at the router. Figures 6a and 6b show that a 1 Mbps SSL/TLS data flow can consume more than 50% of the CPU usage and downgrade the throughput performance of non-IoT data services (e.g., FTP and DLNA) by up to 49% (from 81 Mbps to 41 Mbps); this may lead to a new security threat. For example, insider adversaries may launch DoS attacks by sending large amounts of junk data to the secure tunnel, thereby exhausting the resources of the Wi-Fi router. To defend such types of attacks, SecWIR should therefore prevent non-IoT devices from sending data to the tunnel. Second, in the IoT era, users may have multiple IoT devices, and using a separate SSL/TLS tunnel to protect each one of them may overload the Wi-Fi router and affect the performance of non-IoT devices. Our experimental result shows that an SSL/TLS connection consumes about 440 KB RAM using the

OpenSSL library v1.1.1e, which is not affordable for some resource-constrained routers to support a great number of IoT devices. For example, Linksys WRT400N can only support 15 IoT devices while no other services/applications are running on the router, as shown in Figure 6c. Third, the routers have only limited resources, and hence as the volume of non-IoT traffic increases, the resources available for SSL/TLS tunneling decrease and it becomes difficult to maintain the same level of user experience (e.g., the access time) for the IoT devices.

Therefore, to ensure the successful implementation of SecWIR, an efficient SSL/TLS tunneling approach is required. We thus propose two mechanisms, namely IoT-specific SSL/TLS tunneling (which addresses the first technical challenge) and priority-based SSL/TLS tunneling management (which focuses on the second and third challenges). These two mechanisms shall be applied together to addressing all the challenges.

5.1.1 IoT-specific SSL/TLS Tunneling. Access to the SSL/TLS tunnel is controlled based on the IoT device MAC address, as provided through a device registration procedure implemented using the management application of the Wi-Fi router. Many vendors (e.g., Netgear, Linksys, and TP-Link) provide such management applications [32] for users to control and monitor their Wi-Fi routers via their smartphones. In the SecWIR framework, whenever a new device associates with the Wi-Fi router, the SecWIR management application is enabled to confirm whether or not the new device is an IoT device by sending a notification message to the user. Only registered IoT devices are then permitted to access the SSL/TLS tunnel. Furthermore, to thwart MAC address spoofing attacks in which adversaries attempt to deceive the router by mimicking authorized devices, the remedy, such as [14], is additionally employed.

5.1.2 Priority-based SSL/TLS Tunneling Management. SecWIR needs to prevent the SSL/TLS tunneling module from downgrading the performance of the non-IoT devices associated with the router. Specifically, when the demand increases and the performance of the non-IoT devices deteriorates, tunneling resource is released to restore their performance. As the number of IoT devices increases, the spare resources at the router may become insufficient to establish all the required SSL/TLS tunnels. The IoT devices may thus experience long user-perceived access delays, which cannot be tolerated for certain IoT operations such as turning on a smart bulb.

A priority-based SSL/TLS tunneling management solution is thus proposed to establish, suspend, and tear down the SSL/TLS tunnels dynamically based on their priorities. To save resources, it is assumed that a tunnel can be used by multiple IoT devices belonging to the same vendor (e.g., all TP-Link IoT devices can communicate with the IoTSecComServer deployed in the TP-Link through the same SSL/TLS tunnel). This design is motivated by an observation that IoT vendors provide customers with different IoT control applications, e.g., TP-Link uses Kasa Smart, whereas Etekcity uses VeSync. It is not rare that users purchase IoT devices from only a few IoT vendors; otherwise, many IoT control applications need to be used. To preserve a similar IoT user experience when the router resources are sparse, the priorities of the IoT devices and tunnels are determined based on two factors: the IoT usage patterns and the waiting time. In particular, devices with a more frequent traffic pattern are assigned a higher priority and hence

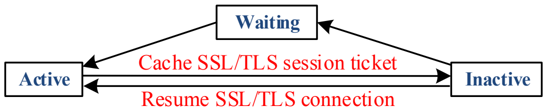


Figure 7: State transitions of an SSL/TLS tunnel.

the corresponding tunnel is canceled or suspended with a lower probability. As the time for which the IoT traffic waits to access the tunnel increases, the priority of the corresponding tunnel increases.

The priority-based tunneling management solution comprises three components: (1) a priority-based SSL/TLS tunneling management algorithm; (2) an IoT device priority function; and (3) an SSL/TLS tunnel priority function. The first algorithm manages the SSL/TLS tunnels based on the properties of the IoT devices and SSL/TLS tunnels, while the latter functions assign priorities to the IoT devices and SSL/TLS tunnels, respectively. We next elaborate on each of them.

Algorithm 1: SSL/TLS tunneling management

```
Initialization:
Randomly select some IoT devices to establish SSL/TLS tunnels until the active list
  reaches its capacity. The remaining unserved IoT devices are moved into the waiting
  list. Set slot counter $k$=1. For each SSL/TLS tunnel, set an active state timer $\tau = 0$ to
  record its active state duration. Set a minimum active time threshold $\tau_{min}$.
while (1) do
    Step 1: For each SSL/TLS tunnel, compute and update its priority value $p_c(k)$
      according to Equation (3), as well as update timer $\tau$.
    Step 2: Find the set $S$ of SSL/TLS tunnels whose active time $\tau > \tau_{min}$;
    Step 3: if $S = \varnothing$ then
        $k$++; continue;
    Step 4: Find the SSL/TLS tunnel (represented as $c_1$) in set $S$ with minimum
      priority; among the SSL/TLS tunnels with inactive or waiting state, find the
      SSL/TLS tunnel (represented as $c_2$) with maximum priority.
    Step 5: if $p_{c_1}(k) \geq p_{c_2}(k)$ then
        $k$++; continue;
    Step 6: if $p_{c_1}(k) < p_{c_2}(k)$ && $c_2$ is waiting then
        Step 7: Set $c_1$ to be inactive and move $c_1$ into the inactive list; set $c_2$ to be
          active and move $c_2$ into the active list, set $c_2$'s active state timer $\tau = 0$;
    Step 8: if $p_{c_1}(k) < p_{c_2}(k)$ && $c_2$ is inactive then
        Step 9: Swap the states and list types of $c_1$ and $c_2$, set $c_2$'s active state timer
          $\tau = 0$;
    Step 10: If the inactive list reaches its maximum capacity, the first-in IoT SSL/TLS
      tunnel is moved from the inactive list to the waiting list.
    Step 11: Sleep until the next time slot; k++;
```

(1) Priority-based Tunneling Management Algorithm: In SecWIR, the SSL/TLS tunnel state can be `Active`, `Inactive` or `Waiting` (see Figure 7). In the `Active` state, the SSL/TLS tunnel has been established and can be used immediately for the transfer of IoT packets. By contrast, in the `Inactive` state, the tunnel was established previously, but has now been suspended and cannot be used for packet transfer until an SSL/TLS connection resumption process has been performed to restore the tunnel. Note that SecWIR employs the ticket-based SSL/TLS session resumption mechanism specified in RFC5077 [46]. Finally, in the `Waiting` state, the tunnel has not yet been established, but some IoT devices have requested it to be set up for the transfer of their packets. The operational details of the tunneling management process based on these state transitions are shown in Algorithm 1.

In Step 1, the Wi-Fi router updates the priority and active time information for each SSL/TLS tunnel at the beginning. We set a minimum active time τ_{min} (e.g., $\tau_{min} = 1s$) to ensure that once an SSL/TLS tunnel turns to be `active`, it at least stays in the `active` state for τ_{min} time. In Steps 2-4, the router collects current priority and active time information to determine whether the state of each SSL/TLS tunnel needs to be changed or not. In Steps 5-9, the router handles two cases for the state update processes. In Step 10, the router handles the situation that the inactive list reaches its maximum capacity. In Step 11, the algorithm sleeps until the next time slot and runs again from the beginning.

(2) IoT device priority function: The secure tunneling module maintains a priority value for each IoT device, where the value is updated every T seconds (T is configurable and set to 0.1 s in this study). It implies that the states of SSL/TLS channels are updated every T s; a larger T leads to lower CPU usage but longer IoT device access time.). Let $p(k)$ be the priority function at the kth time slot. The function takes two factors into account, namely $p_u(k)$ and $p_w(k)$, where $p_u(k)$ is the usage frequency function, and $p_w(k)$ is the waiting time function. The priority function $p(k)$ for each IoT device is defined as

$$p(k) = x \times p_w(k) + y \times p_u(k), \tag{1}$$

where x and y are adjustable weight coefficients and $x + y = 1$ (they are set to 0.02 and 0.98, respectively, in this study). To reduce the additional IoT access delay, we give a higher weight to $p_u(k)$. The waiting time function, $p_w(k)$, is set to 0 whenever the IoT device has an active SSL/TLS tunnel assigned for its use; otherwise, $p_w(k) = p_w(k-1) + 1$. Meanwhile, the usage frequency function $p_u(t)$ is given by

$$p_u(k) = \begin{cases} 0, & k = 1 \\ \alpha p_u(k-1) + (1-\alpha) c_u(k), & k > 1 \end{cases} \tag{2}$$

where $c_u(k)$ is increased by one if any usage of the IoT device is detected during the kth time slot; otherwise, $c_u(k) = 0$. Note that $p_u(t)$ is updated using the exponential moving average (EMA) method [29]. SecWIR detects the IoT device usage to determine $c_u(k)$ using the following two approaches.

- *Traffic-outlier Detection.* The detection process is based on the frequency with which traffic peaks (outliers) are observed in the requested IoT traffic due to the execution of IoT control operations. In our study on the NonSecIoT devices, it is observed that while IoT devices are not being used, they periodically exchange small keep-alive messages (e.g., less than 170 bytes) with the IoT servers. However, while IoT devices are triggered to be used, the IoT servers transmit relatively large IoT command messages (e.g., more than 245 bytes) to the IoT devices. In SecWIR, the outlier detection process is performed using the algorithm proposed in [37].
- *Detection for Manual IoT Device Access.* If a user tries to access an IoT device once from the IoT vendor's control application on his/her smartphone during the kth time slot, the usage, $c_u(k)$, of all IoT devices belonging to the IoT vendor is increased by one. In practice, several techniques are available for performing this detection. For example, on Android phones, the `ActivityManager` module [6] can be used to retrieve the name of the foreground application. While an IoT device access is detected by the traffic-outlier detection, and the foreground application is an IoT control application, a potential manual IoT device access is detected.

(3) ***SSL/TLS tunnel priority function:*** The SSL/TLS tunnel priority is determined in accordance with the highest priority among all the IoT devices using the tunnel. In other words, the priority of a tunnel c in the kth time slot, is given by

$$p_c(k) = \max\{p_1(k), \ldots, p_n(k)\}, \tag{3}$$

where $p_1(k), \ldots, p_n(k)$ respectively represent priority values of n distinct IoT devices which share the same SSL/TLS tunnel c.

5.2 Stream Security Validation Module

This module aims to secure the communications of InSecIoT devices by examining whether the procedures used by the IoT device to establish secure channels are compliant with security protocol standards. For each non-compliant establishment procedure, the module may terminate the connection and notify the IoT device owner. The module addresses three common security issues: (1) insecure cryptographic cipher suites, (2) server certificate expiration, and (3) fake server certificates.

Implementing the stream security validation module is challenging for two reasons. First, current validation tools for checking security compliance do not support stream processing (i.e., packet-by-packet examination), but only batch processing. For example, SSLdump [50] outputs the server certificate for validation purposes only after an SSL/TLS connection has been established. Thus, there is no guarantee that insecure channels have not been established and used to send IoT packets to rogue servers. Second, the tools are not optimized for resource saving under resource constraints. For example, they may attempt to verify the same IoT server certificate multiple times within a short time period, thereby wasting the resources of Wi-Fi routers.

To address these problems, SecWIR incorporates two components within the validation module: (1) security standard validation and (2) hash-aided validation.

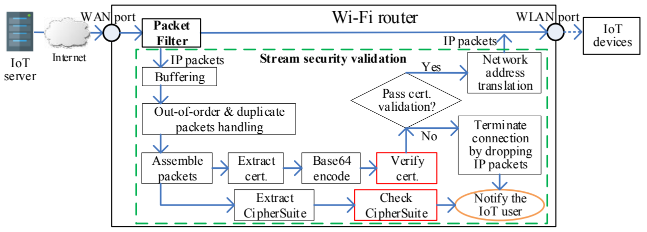


Figure 8: Security standard validation flow.

Security Standard Validation: As shown in Figure 8, a packet filter is used to dispatch all the packets containing SSL/TLS handshake messages to the validation module. The module extracts cryptographic information from the messages (e.g., ServerHello) and then examines whether an insecure cipher suite has been used by the IoT servers. The cipher suite is a set of algorithms that secure network communications; it usually contains key exchange, encryption/decryption, and message authentication code algorithms. Several algorithms have been reported as being insecure, including RC4, MD5, DES-CBC, and IDEA-CBC. When the cipher suite selected by the IoT servers contains insecure algorithms, the validation module can send a `warning` message to the device owner through the Wi-Fi router management application (e.g., [32]). Then, the module validates the device's server certificate in terms of its expiration date and the validity of the issuer.

Specifically, the validation process at this module consists of seven tasks: (1) buffering only the related packets; (2) dealing with packet retransmission and out-of-order delivery issues; (3) assembling messages related to a cipher suite; (4) extracting the server certificate from the cipher-related messages; (5) checking if any insecure algorithms are used in the messages (and sending a `warning` message to the device owner if necessary); (6) encoding the server certificate into a Base64 format, which is X.509-compatible, if a server certificate exists; (7) verifying the server certificate. If the certificate is determined to be valid, the module permits the tunnel establishment and then routes the IoT packets through this tunnel. In practice, the server certificate validation process may fail for two reasons, namely the certificate is expired; or the certificate is generated by a non-trusted CA. In both cases, the secure tunnel establishment process is terminated. Although the validation module currently considers only three common security issues, it can easily be extended to consider additional security issues if required.
Hash-aided Validation: In the empirical IoT study described in §3, it was found that some of the server certificates were verified many times. In some cases, this was due to the fact that the devices manufactured by the same IoT vendor connected to the same IoT server. In other cases, it was caused by some of the devices having only short-lived SSL/TLS connections, which were terminated as soon as the device request was served and reconstructed whenever a new request was received. The need to verify the server certificates repeatedly not only consumes the resources of the Wi-Fi router, but also delays the IoT access response time.

A hash-aided validation method was further designed to reduce the occurrence of repeated validation operations. In particular, whenever a server certificate is successfully verified, the validation module caches its hash value and this value is then referenced in any future validation process to check whether or not the certificate has been verified previously. Notably, the expiration time of the cached validation result is configurable in the proposed module.

5.3 Resource Monitoring

Wi-Fi routers are designed to efficiently route user packets rather than perform security functions for IoT devices. Thus, SecWIR should not affect the performance of non-IoT devices while securing IoT devices. Accordingly, the resource monitoring module aims to strike a balance between the non-IoT device performance and IoT security. In particular, it monitors the uplink and downlink packet drop status of non-IoT devices, and then dynamically allocates resources to SecWIR by adapting the number of maximum active SSL/TLS tunnels and the maximum data rate of each SSL/TLS tunnel. For example, when the Wi-Fi router starts to drop packets of non-IoT devices, the module gradually reduces the number of maximum active SSL/TLS tunnels.

6 SECURITY ANALYSIS

In this section, we examine how SecWIR fulfills the required security guarantees and explain how SecWIR defends against possible attack scenarios over the connections between the home Wi-Fi router and multiple IoT devices (C1), and between the IoTSecCom server deployed in the IoT vendor realm and a home Wi-Fi router of the IoT device owner (C2). Note, as described previously in §4, it is assumed that attacks do not occur within the IoT vendor realm.

6.1 Security Guarantees

SecWIR supports two security guarantees: **secrecy** and **integrity** for an IoT communication over the connections: C1 and C2.
C1: SecWIR supports the secrecy and integrity of the wireless communications between the IoT devices and the Wi-Fi router by enabling existing Wi-Fi security protocols (e.g., WPA2/WPA3 [1]). In particular, AES_128 and CCMP (CTR mode with CBC-MAC Protocol) are adopted by SecWIR for the secrecy and integrity protection, respectively.
C2: SecWIR supports the secrecy and integrity of the wired communications between the Wi-Fi router and IoT vendor realm using the SSL/TLS security protocols. In particular, SecWIR adopts a secure cipher suite consisting of ECDHE, ECDSA, AES_128, and CBC_SHA256, which contains a key exchange algorithm (ECDHE, Elliptic Curve Diffie-Hellman Ephemeral), a signature algorithm (ECDSA, Elliptic Curve Digital Signature Algorithm), a ciphering algorithm (AES128), and a message authentication code algorithm (SHA256). The first three algorithms guarantee the secrecy of communications, while the last algorithm guarantees their integrity.

6.2 Possible Attacks

In examining the robustness of SecWIR against malicious attacks, it is assumed that the attacks can be launched from either inside the home network (C1) or outside the home network (C2).
Inside Home Wi-Fi Attacks:

- *IoT Compromise/DoS/Side-Channel Attacks:* The IoT devices may suffer various forms of internal attacks, including compromising attacks (e.g, Mirai attacks [25]), DoS attacks (e.g., PING flooding), and side-channel attacks (e.g., inferring the IoT device usage by analyzing the intercepted IoT packets). All of these attacks rely on the adversary being able to access the IoT user's home Wi-Fi network. The empirical study of 40 common IoT devices described in §3 revealed two important observations: (1) all commands for IoT devices are sent by external IoT servers; and (2) all notifications/alerts/messages produced by IoT devices are sent to the external IoT servers. In other words, the IoT devices do not need to communicate with other internal hosts. Thus, to guard against internal attacks, SecWIR implements a security policy by which the IoT devices are prevented from communicating with any internal hosts other than SecWIR.
- *IoT Masquerading Attacks:* Adversaries may compromise non-IoT devices inside a victim's home Wi-Fi network and masquerade these devices as authentic IoT devices using a MAC address spoofing technique. They can send large quantities of spam data to the IoT server, thereby exhausting its resources. However, SecWIR can easily detect such attacks and report them to the IoT users since each device associated with the Wi-Fi router is assigned a unique security key (i.e., PTK, pairwise transient key) by the WPA2/WPA3 protocol. Furthermore, SecWIR maintains the registration of the MAC addresses of all the legitimate IoT devices (see §5.1.1). Thus, if multiple devices use the same MAC address as a registered IoT device but employ a different security key, a notification message is immediately sent to the IoT user.

- *Malicious/Compromised IoT Attacks:* Since SecWIR allows multiple IoT devices to share the same SSL/TLS tunnel, a compromised device can thus gain the access to the shared channel and may overwhelm the tunnel to hurt the performance of the other IoT devices. However, such attack damages are limited due to the following two reasons. First, as described previously, SecWIR can protect IoT devices from being remotely compromised. Second, even when an IoT device is compromised by non-cyber approaches (e.g., physical compromise), the attack can be mitigated by employing a per-device rate limit mechanism.
- *Denial-of-IoT-Service (DoIS) Attacks:* Adversaries may compromise the victim's non-IoT devices and then use these devices to generate huge volumes of non-IoT data traffic to consume the resources of the home Wi-Fi router and launch a DoIS attack against the user's IoT devices. However, SecWIR readily defends such attacks due to two reasons. First, the IoT user can assign a small amount of *guaranteed* resources to SecWIR. Since all of the security modules within SecWIR are designed to work efficiently with only limited resources, such an approach can substantially mitigate the impact caused by the DoIS attacks. Second, most COTS Wi-Fi routers support fair bandwidth sharing among the associated devices and hence adversaries are unable to occupy all the resources of the Wi-Fi router using compromised devices.

Outside Home Wi-Fi Attacks:

- *SSL/TLS Protocol Attacks:* Recently, researchers have exploited the vulnerabilities of SSL/TLS to develop various MITM attacks, including BEAST [21], CRIME [34], TIME [13], RC4 BIASES [42], SSL Renegotiation [60], and downgrade attacks [56]. However, since these attacks rely mainly on insecure security algorithms and problematic implementations, they can be easily thwarted by SecWIR. For example, BEAST, CRIME, TIME, RC4 BIASES, and SSL Renegotiation attacks can be addressed by enabling AES256, disabling TLS compression, enabling Encrypt-then-MAC authenticated encryption, disabling RC4, and using TLSv1.2, respectively.
- *Side-channel Attacks:* Researchers have demonstrated that adversaries can infer users' IoT usage by analyzing the encrypted IoT data [10]. However, SecWIR largely protects IoT users from such attacks by the means of the tunneling mechanism. Since multiple IoT devices share a TLS tunnel with the IoTSecCom server, it is difficult for adversaries to infer a particular IoT device usage due to the natural noise (IoT data) produced by the other IoT devices. Moreover, additional noises can be introduced by both IoTSecCom server and the SecWIR router to defend side-channel attacks.

7 SECWIR EVALUATION

This section describes the implementation and the evaluation of the prototype SecWIR framework.

Implementation: The SecWIR framework was written in Linux C and was implemented on top of OpenWrt/LEDE-powered Wi-Fi routers. OpenWrt/LEDE [4], a very popular operating system for Wi-Fi routers, has supported 235 Wi-Fi router vendors and 1362 models in its current release [41]. We upgraded all tested Wi-Fi routers to the latest stable OpenWrt/LEDE releases at the time of the paper submission (e.g., Linksys WRT400N uses v17.01.5, whereas Tp-Link AC1750 uses v19.07.2). The ***IoT Secure Tunneling*** module used the OpenSSL library [40] to establish, close, and restore

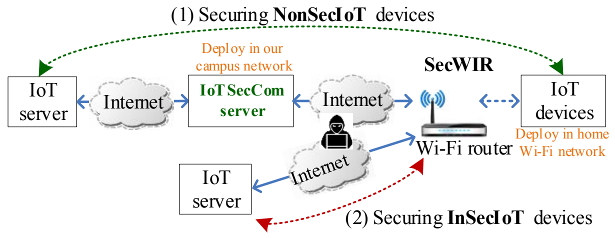
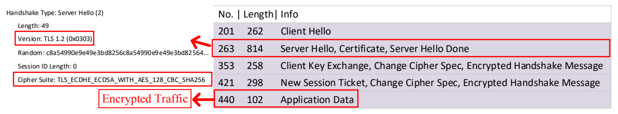

(1) Securing NonSecIoT devices
Deploy in our campus network
SecWIR
IoT server
Internet
IoT SecCom server
Internet
IoT devices
Wi-Fi router
Deploy in home Wi-Fi network
IoT server
Internet
(2) Securing InSecIoT devices

Figure 9: Experimental testbed.

SSL/TLS tunnels. In addition, the SSL/TLS session ticket followed the ASN.1 [12] representation standards. In the ***Stream Security Validation*** module, seven routines were developed to check the validity of the IoT server certificates and ensure they were compliant with the relevant security protocol standards (see Figure 8). In the event of validation failures, the related IP packets were dropped by configuring the Wi-Fi router's IP table [2]. The ***Resource Monitoring*** module used the Linux utilities to obtain real-time resource usage of the Wi-Fi router and packet delivery status. Specifically, the `Top` command [3] was used to retrieve CPU/RAM usage, while the `netstat` command was used to acquire TX/RX-DRP (the number of packets dropped) at specific WAN/WLAN interfaces.

7.1 Evaluation

The performance of the SecWIR framework was evaluated using the experimental setup shown in Figure 9 based on a Linksys WRT400N Wi-Fi router with a 680 MHz CPU, 32 MB RAM, and 8 MB flash memory. The experiments were commenced by evaluating the effectiveness and performance of the two SecWIR security modules: the IoT secure tunneling module and the stream security validation module. The performance overhead incurred by SecWIR was then evaluated for four additional routers in terms of the access delay, the throughput, the RAM usage and the CPU usage. In performing the experiments, the IoT devices were deployed in a tested home Wi-Fi network. To provide the NonSecIoT devices with secure IoT communications with their IoT servers, an IoTSecCom client was installed on the home Wi-Fi router, and an IoTSecCom server was installed on a campus network. An SSL/TLS tunnel was then established between the IoTSecCom client and the IoTSecCom server. For the InSecIoT devices, SecWIR checked whether the associated secure channel establishment process was compliant with the security protocol standards and notified the device owner if necessary.

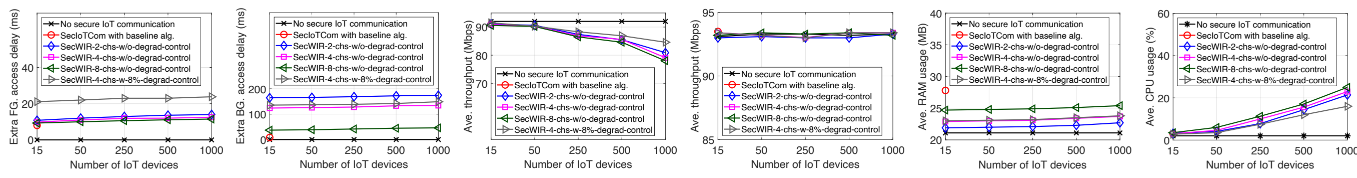

```
Handshake Type: Server Hello (2)
    Length: 49
    Version: TLS 1.2 (0x0303)
    Random: c8a54990e9e49e3bd8256c8a54990e9e49e3bd82564...
    Session ID Length: 0
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
```

No.	Length	Info
201	262	Client Hello
263	814	Server Hello, Certificate, Server Hello Done
353	258	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
421	298	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
440	102	Application Data

Figure 10: A TLS secure tunnel is successfully established.

7.1.1 *SecWIR Security Function Evaluation.*

IoT Secure Tunneling Module: We conducted an experiment to verify if the NonSecIoT devices can communicate with the IoT servers without any issues through the proposed secure IoT tunnels. There are 8 NonSecIoT test devices spanning four categories: socket (Geekbes and Etekcity), bulb (IView and TP-Link), humidifier (Essential and ASAKUKI), and strip (KMC and Teckin). Figure 10



(a) Extra FG. access delay **(b) Extra BG. access delay** **(c) Throughput(on-router services)** **(d) Throughput(off-router apps)** **(e) RAM usage** **(f) CPU usage**

Figure 11: Evaluation of secure IoT communication on Linksys WRT400N Wi-Fi router.

shows that a TLS secure tunnel was successfully established between the IoTSecCom server and IoTSecCom client, based on a CipherSuite consisting of ECDHE, ECDSA, WITH_AES_128, and CBC_SHA256. Note that the elliptic curve-based algorithms are deliberately chosen since they provide strong security even with limited key lengths, and hence are suitable for resource-constrained routers. We further launched the IoT hijacking attacks (as shown in Figure 2) against these IoT devices by sending them fake IoT commands. It was observed that all the fake IoT commands were discarded by SecWIR, and all the IoT devices could be accessed normally through the secure IoT tunnels.

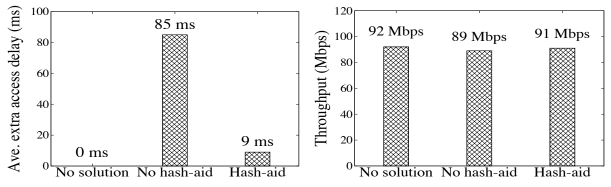
Stream Security Validation Module: The effectiveness of the stream security validation module in verifying the security of the channel establishment procedure was evaluated by deploying a server as an adversary between the tested InSecIoT devices and their IoT servers in order to intercept and modify the SSL/TLS messages and launch MITM attacks. In this experiment, we launched the spying attacks (as shown in Figure 3) and the CipherSuite downgrade attacks against two InSecIoT devices (i.e., Zmodo and Wyze security cameras) by sending them fake server certificates or fake ServerHello messages with a downgraded CipherSuite via our SSLSplit server. The experimental results are described in the following.

1) Expired & Forged Certificate Detection: The server intercepted the certificate sent by the IoT server and replaced it with a fake one (e.g., an expired certificate or a certificate issued by a non-trusted CA). The fake certificate was then forwarded to the tested InSecIoT device. The experimental result showed that the validation module could detect insecure server certificates and prevented the corresponding SSL/TLS connections from being established.

2) Insecure CipherSuite Detection: The server intercepted the ServerHello message sent by the IoT server and changed the CiperSuite selected by the IoT server to an insecure CiperSuite (e.g., using RC4 and MD5). A fake ServerHello message was then sent to the tested InSecIoT device. The experimental result showed that the validation module successfully detected the insecure CipherSuite usage and sent a warning message to the IoT user. Note that SecWIR does not explicitly forbid the use of an insecure CiperSuite but simply warn the user, since current IoT devices may have some restrictions which make them use those weak CiperSuites.

7.1.2 SecWIR Overhead Evaluation.

In evaluating the performance overhead of the SecWIR framework, four metrics were considered, namely the extra IoT device access delay incurred by SecWIR, the non-IoT traffic throughput, the RAM usage, and the CPU usage. Note that the extra IoT device access delay was defined as $t_1 - t_0$, where t_0 and t_1 represent the access



(a) Ave. extra access delay **(b) Throughput**

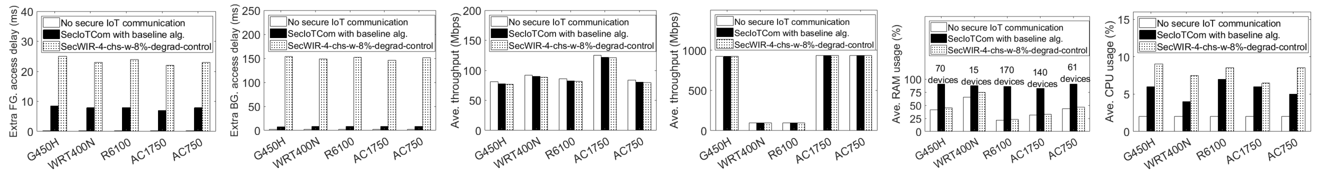
Figure 12: Evaluation of the stream security validation module on the Linksys WRT400N Wi-Fi router.

delays (i.e., response time) of the IoT device before and after running SecWIR, respectively. The throughput was measured using Netperf [30] to emulate the non-IoT traffic generated by on-router services (e.g., router-side FTP, DLNA, and iTune services) and off-router applications (e.g., accessing Internet from the user's laptop).

IoT Secure Tunneling Module: We evaluated this module for three different scenarios: (1) no secure IoT communication, (2) support of SSL/TLS using a baseline tunneling management algorithm, in which each IoT device established a dedicated SSL/TLS connection with its IoT server; and (3) support of SSL/TLS using the proposed priority-based tunneling management algorithm. The experiments were performed using different numbers of IoT devices, different numbers of active SSL/TLS tunnels, and different allowed degradation values of the maximum non-IoT traffic throughput.

1) Experiment Settings: We deployed four types of the NonSecIoT devices as described in §7.1.1. Commands were issued to the IoT devices with three different levels of usage frequency: (1) once per 1 min (25% of tested devices), (2) once per 10 mins (25% of tested devices), and (3) once per hour (the remaining tested devices). The baseline algorithm established and maintained as many SSL/TLS connections as the number of tested IoT devices when spare resources are sufficient. Note that, to evaluate the scalability of SecWIR, when the number of IoT devices is more than 16, an IoT device emulation server is deployed to emulate tested IoT devices based on their IoT traffic patterns, including both the foreground (FG) (e.g., IoT access commands) and background (BG) traffic (e.g., keep-alive messages).

2) Experimental Results: The results are shown in Figure 11. We have four observations. First, the CPU and RAM usage volumes of the priority-based algorithm are increased with an increasing number of active secure channels. For example, 2- and 8-active-secure-channel methods increase the CPU usage by 2.1% and 4%, respectively, to secure 250 IoT devices. However, more active secure channels lead to shorter IoT device access delays. Second, the RAM



(a) Extra FG. access delay (b) Extra BG. access delay (c) Throughput (on-router services) (d) Throughput (off-router apps) (e) RAM usage (f) CPU usage

Figure 13: Evaluation of secure IoT communication on five Wi-Fi routers: WRT400N (CPU: 680MHz, 32MB RAM), G450H (CPU: 400MHz, 64MB RAM), AC750 (CPU: 580MHz, 64MB RAM), R6100 (CPU: 560MHz, 128MB RAM), and AC1750 (CPU: 720MHz, 128MB RAM).

usage of the priority-based management algorithm is much less than that of the baseline algorithm. For example, compared with the non-secure communication scenario, the priority-based algorithm with 4 active secure channels only increases the RAM usage by 2.7 MB to support 1,000 IoT devices, whereas the baseline algorithm almost runs out of all the available RAM (7 MB) to support 15 IoT devices. Third, compared with the absence of the secure IoT communication, the baseline and priority-based SSL/TLS tunneling management algorithms increase the IoT device access delays by 8 ms and 8~175 ms (extra foreground delays: 8~24 ms; extra background delays: 35~175 ms), respectively. Although the SecWIR's priority-based tunneling management algorithm results in a longer access time, in practice, the increased access time compared to the case in which no security is deployed is unlikely to be perceived by the user since it is so short. Our study shows that SecWIR offers the IoT device access delay that is comparable to commercial IoT security gateways (less than 1 second, see §7.1.4). Fourth, when the resource monitoring module is not activated (e.g., SecWIR-2/4/8-chs-w/o-degrad-control), the non-IoT data throughout degradation caused by the priority-based management algorithm increases with an increasing number of IoT devices (i.e., from 0.5% with 15 devices to 15.2% with 1,000 devices). After activating the monitoring module (e.g., SecWIR-4-chs-with-8%-degrad-control), the throughput downgrade can be reduced to 8% by slowing down the speed of processing IoT traffic at SecWIR based on the introduction of a 10 ms sleep time.

Stream Security Validation Module: We next evaluate the performance of the stream security validation module.

1) Experimental Settings: Two tested smart cameras (i.e., Zmodo and Wyze) were connected to the Wi-Fi router. It was observed that each of the cameras established an SSL/TLS connection with the IoT server when being powered up. Therefore, the performance of the stream security validation module in examining the CiperSuite selected by the IoT servers and the IoT server certificates was evaluated by deliberately cycling the cameras on and off. In the experiment, a security standard examination request was generated every 6 seconds on average.

2) Experimental Results: Figure 12 shows the performance evaluation results. It is seen that the security standard examination based on the proposed hash-aided certificate validation process results in a lower extra IoT device access delay than that without hash-based optimization (i.e., 9 ms vs. 85 ms). In addition, the hash-aided validation process has only a small effect on the throughput of the Wi-Fi router. Notably, compared to the benchmark scenario in which secure IoT communications were not deployed, the stream security validation module was found to increase the RAM and CPU usage by less than 1 MB and 3% CPU, respectively, irrespective of whether or not the hash-aided optimization was employed. The results confirm that the validation module does not impose any significant performance overhead when securing InSecIoT devices.

7.1.3 *Performance Evaluation on Low-cost Wi-Fi Routers:*

The performance of SecWIR was further evaluated for the case in which the SSL/TLS tunneling module and stream security validation module were both enabled. In addition to the Linksys WRT400N Wi-Fi router, the experiments were also conducted on four other low-cost Wi-Fi routers, namely Buffalo WZR-HP-G450H ($35), Tp-Link AC750 ($30), Netgear R6100($50), and Tp-Link AC1750 ($50).

1) Experimental Settings: We conducted the same experiments as those previously evaluating the SSL/TLS tunneling module and stream security validation module while supporting 250 IoT devices. For the tunneling module, three mechanisms were evaluated: (1) no secure IoT communication, (2) support of SSL/TLS using a baseline tunneling management algorithm, in which each IoT device established a dedicated SSL/TLS connection with its IoT server, and (3) support of SSL/TLS using the proposed priority-based tunneling management algorithm using 4 active SSL/TLS tunnels and imposing the maximum 8% of non-IoT traffic throughput downgrade. For the validation module, a security standard examination request was generated every 6 seconds on average.

2) Experimental Results: Figure 13 shows the performance evaluation results. Among all the considered routers, the maximum foreground delay of the IoT device access is 25 ms, which is incurred by the Buffalo router. Furthermore, the Linksys router results in 8% of the maximum throughput degradation (92 Mbps $\longrightarrow$ 84.6 Mbps). Finally, SecWIR increases the CPU and RAM usage by 4.5%~7% and 1.9 MB~2.2 MB over all the five routers to secure 250 IoT devices, whereas the conventional SSL/TLS tunneling mechanism runs out of all the available RAM resources of the routers while supporting 80~235 fewer IoT devices than SecWIR. Overall, the results show that SecWIR provides IoT users with secure IoT communications even on COTS low-cost Wi-Fi routers and causes no significant degradation on non-IoT data service performance.

7.1.4 *Comparison with Samsung SmartThing Hub.*

Samsung SmartThing system is a popular smart home IoT system. To use Samsung SmartThing IoT devices, the user needs to first purchase a SmartThing hub and connects the hub with his/her home router, and then connect Samsung SmartThing IoT devices with the hub. Finally, the

IoT security infrastructure		SmartThing Hub	SecWIR
Hardware capabilities	CPU	528 Mhz	680 Mhz
	RAM	256 MB	32 MB
	Flash	4 GB	8 MB
Secure IoT communication	Security protocol	TLS v1.2	TLS v1.2
	key exchange algorithm	ECDHE	ECDHE
	Signature algorithm	RSA	ECDSA
	Ciphering algorithm	AES_256	AES_128
	Integrity algorithm	GCM_SHA384	CBC_SHA256
Performance	Time of accessing IoT devices	740-800 ms	700-750 ms
	Time of establishing TLS connections	115-462 ms	150-350 ms
	Time of validating a server cert.	25-120 ms	72-98 ms
	Time of re-validating a server cert.	25-120 ms	2-5 ms
Security	Defend IoT hijacking attack?	Yes	Yes
	Defend spying attacks?	Yes	Yes
	Reject forged/expired server certs.?	Yes	Yes
Price		$70	free*

∗: SecWIR relies on the IoT users' existing home router.
NOTE: The hardware capabilities and performance of SecWIR is based on Linksys WRT400N and Geekbes smart plugs.

Table 2: Comparison between using Samsung SmartThing IoT system and SecWIR.

user can access the IoT devices via the Samsung SmartThing application. Similar to SecWIR, the SmartThing hub plays the role of the IoT security gateway in this system. We thus compare SecWIR with the Samsung SmartThing hub. The comparison results are summarized in Table 2. We observe that both SecWIR and Samsung SmartThing provide comparable security functions (e.g., TLS v1.2 and AES encryption algorithms) and performance (e.g., accessing an IoT device only takes 0.8 s). However, SecWIR has two advantages over Samsung SmartThing. First, SecWIR supports IoT devices from various vendors, whereas the SmartThing hub is specific to Samsung IoT devices. Second, the cost of deploying a Wi-Fi-connected smart device is relatively inexpensive. A Samsung smart socket (GP-U9995JVLDAA) costs $35, whereas the Geekbes (YM-WS-5) smart socket takes only $9. Moreover, SecWIR does not require users to purchase additional IoT security gateways before securely using their smart home devices.

8 DISCUSSION

Incentives for IoT vendors: Providing NonSecIoT devices with secure IoT communications with their IoT servers inevitably requires the support of IoT vendors. However, as described previously in §3, it is reasonable to expect that most IoT vendors will be willing to deploy security mechanisms in their infrastructures in order to secure their IoT devices if the proposed solutions do not affect their existing IoT services or degrade their profit. Thus, the present study has deliberately developed standalone IoTSecCom servers for deployment in the IoT vendor realm (see Figure 4). In practice, the IoT servers do not need to be aware of this IoTSecCom server. In addition, SecWIR yields a win-win situation for the IoT vendors and IoT users. In particular, the IoT vendors can preserve their profit when deploying secure IoT communications since they can continue to employ low-cost IoT device platforms for their smart home devices, while IoT users do not need to purchase expensive IoT devices to ensure secure IoT communications, but can continue instead to use cheaper IoT devices supporting SecWIR.

Deploying SecWIR: To support SecWIR, the home Wi-Fi routers must be upgraded. Most Wi-Fi router vendors provide users with a web/app-based interface to manually upgrade their routers. Since SecWIR is deployed on top of the popular OpenWrt/LEDE system, which has supported 235 Wi-Fi router vendors and 1362 models, it is a relatively simple task for most router vendors to adopt SecWIR and release an appropriate update. Regarding the other Wi-Fi routers, we believe that it is not difficult for router vendors to adopt SecWIR since it is a low-overhead, resource-efficient security framework. Note that the deployment of SecWIR requires IoT users to have some knowledge of using web services or smartphone apps to upgrade their routers, if an automatic router upgrade is not supported.

Customizing OpenWrt/LEDE: Customizing OpenWrt/LEDE operating systems of home routers can be one option to give more resources to secure IoT devices. However, the available resources can still be exhausted rapidly by the inefficient, conventional SSL/TLS tunnel mechanism, when the number of IoT devices increases. Moreover, with many built-in services currently deployed on the COTS Wi-Fi routers (e.g., iTunes server), we believe that the proposed light-weight IoT security framework is still desirable.

Applying SecWIR to securing other IoTs: Although this study aims to secure IoT communication of the Wi-Fi-connected IoT devices, the techniques adopted by SecWIR can be also applied to other IoT technologies (e.g., LoRaWAN [7]). For example, the proposed priority-based SSL/TLS tunneling management and stream security validation modules can be deployed on LoRaWAN gateways to secure the IoT communications between LoRAWAN gateways and LoRAWAN customer IoT servers [7].

9 CONCLUSION

Wi-Fi smart home IoT devices are increasingly popular nowadays. The present study has shown that 29 of the 40 popular Wi-Fi IoT devices have no security protocols deployed, or contain problematic security implementations. By exploiting these vulnerabilities, adversaries can launch various cyberattacks against IoT device owners. The identified vulnerabilities stem from hardware/software limitations and/or imprudent security designs of the IoT devices. These limitations preclude the possibility of deploying security solutions on the devices themselves. Accordingly, this study has proposed an infrastructure-based solution, designated as SecWIR, for securing the IoT communications using COTS home Wi-Fi routers. Importantly, SecWIR provides the Wi-Fi IoT devices with full mainstream security protocol support without the need for any modifications of the existing IoT devices and IoT servers or the purchase of additional security hardware. It can enable IoT users to enjoy inexpensive, but still secure IoT devices without any substantial increase in the device access delay or the degradation of the non-IoT data service performance. As such, SecWIR plays a valuable role in paving the way for the further development and deployment of Wi-Fi smart home IoT technology.

10 ACKNOWLEDGMENTS

We greatly appreciate the insightful comments from our shepherd, Dr. Tianyin Xu, and the anonymous reviewers. This work is supported in part by the National Science Foundation under Grants No. CNS-1617627, PFI:BIC-1632051, CNS-1814551, and CNS-1815636. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors only and do not necessarily reflect those of the National Science Foundation.

REFERENCES

[1] An overview of wireless protected access 2 (wpa2). https://www.lifewire.com/what-is-wpa2-818352, 2017.

[2] Control network traffic with iptables. https://linode.com/docs/security/firewalls/control-network-traffic-with-iptables/, 2018.

[3] Linux top command. https://www.lifewire.com/linux-top-command-2201163, 2018.

[4] Openwrt/lede project. https://openwrt.org/, 2018.

[5] Wi-fi security. https://www.wi-fi.org/discover-wi-fi/security, 2020.

[6] ActivityManager. https://bit.ly/2o2pulx, 2018.

[7] Alliance, L. Lorawan: Low power wide area network. https://lora-alliance.org/about-lorawan, 2020.

[8] Alrawi, O., Lever, C., Antonakakis, M., and Monrose, F. Sok: Security evaluation of home-based iot deployments. In *IEEE Symposium on Security and Privacy (S&P)* (2019), pp. 1362–1380.

[9] Alshamsi, A., and Saito, T. A technical comparison of ipsec and ssl. In *AINA* (2005).

[10] Apthorpe, N., Reisman, D., Sundaresan, S., Narayanan, A., and Feamster, N. Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. *CoRR abs/1708.05044* (2017).

[11] Aras, E., Ramachandran, G. S., Lawrence, P., and Hughes, D. Exploring the security vulnerabilities of lora. In *IEEE International Conference on Cybernetics (CYBCONF)* (2017), pp. 1–6.

[12] ASN1. A layman's guide to a subset of asn.1, ber, and der. http://luca.ntop.org/Teaching/Appunti/asn1.html, 2018.

[13] Be'ery, T., and Shulman, A. A perfect crime? only time will only time will tell. https://media.blackhat.com/eu-13/briefings/Beery/bh-eu-13-a-perfect-crime-beery-wp.pdf, 2013.

[14] Benzaïd, C., Boulgheraif, A., Dahmane, F. Z., Al-Nemrat, A., and Zeraoulia, K. Intelligent detection of mac spoofing attack in 802.11 network. In *International Conference on Distributed Computing and Networking (ICDCN)* (2016), p. 47.

[15] Bonetto, R., Bui, N., Lakkundi, V., Olivereau, A., Serbanati, A., and Rossi, M. Secure communication for smart iot objects: Protocol stacks, use cases and practical examples. In *IEEE international symposium on a world of wireless, mobile and multimedia networks (WoWMoM)* (2012), pp. 1–7.

[16] Caextract. Mozilla ca certificate store in pem format. https://curl.haxx.se/docs/caextract.html, 2018.

[17] Celik, Z. B., Babun, L., Sikder, A. K., Aksu, H., Tan, G., McDaniel, P., and Uluagac, A. S. Sensitive information tracking in commodity iot. In *USENIX Security Symposium (USENIX Security)* (2018), pp. 1687–1704.

[18] Celik, Z. B., Tan, G., and McDaniel, P. D. Iotguard: Dynamic enforcement of security and safety policy in commodity iot. In *Network and Distributed Systems Symposium (NDSS)* (2019).

[19] Chawla, B. K., Gupta, O., and Sawhney, B. A review on ipsec and ssl vpn.

[20] Chen, J., Diao, W., Zhao, Q., Zuo, C., Lin, Z., Wang, X., Lau, W. C., Sun, M., Yang, R., and Zhang, K. Iotfuzzer: Discovering memory corruptions in iot through app-based fuzzing. In *Network and Distributed Systems Symposium (NDSS)* (2018).

[21] CVE. Beast attack. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3389, 2011.

[22] Dang, F., Li, Z., Liu, Y., Zhai, E., Chen, Q. A., Xu, T., Chen, Y., and Yang, J. Understanding fileless attacks on linux-based iot devices with honeycloud. In *International Conference on Mobile Systems, Applications, and Services (Mobisys)* (2019), pp. 482–493.

[23] Davies, N., Taft, N., Satyanarayanan, M., Clinch, S., and Amos, B. Privacy mediators: Helping iot cross the chasm. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications* (2016), ACM, pp. 39–44.

[24] Fernandes, E., Jung, J., and Prakash, A. Security analysis of emerging smart home applications. In *IEEE symposium on security and privacy (S&P)* (2016), pp. 636–654.

[25] Fruhlinger, J. The mirai botnet explained: How teen scammers and cctv cameras almost brought down the internet. https://www.csoonline.com/article/3258748/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html, 2018.

[26] Gollakota, S., Hassanieh, H., Ransford, B., Katabi, D., and Fu, K. They can hear your heartbeats: non-invasive security for implantable medical devices. In *ACM SIGCOMM Computer Communication Review* (2011), pp. 2–13.

[27] Hafeez, I., Ding, A. Y., Suomalainen, L., Kirichenko, A., and Tarkoma, S. Securebox: Toward safer and smarter iot networks. In *CAN@CoNEXT* (2016).

[28] Harris, A. F., Sundaram, H., and Kravets, R. Security and privacy in public iot spaces. In *International Conference on Computer Communication and Networks (ICCCN)* (2016), pp. 1–8.

[29] Holt, C. C. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting* (2004).

[30] Jones, R. Netperf. https://github.com/HewlettPackard/netperf, 2018.

[31] Kavalaris, S. P., and Serrelis, E. Security issues of contemporary multimedia implementations: The case of sonos and sonosnet. In *International Conference in Information Security and Digital Forensics (ISDF)* (2014), pp. 63–74.

[32] Komando. Best apps to control your router. https://www.komando.com/apps/368384/best-apps-to-control-your-router/all, 2018.

[33] Kumar, S., Hu, Y., Andersen, M. P., Popa, R. A., and Culler, D. E. Jedi: Many-to-many end-to-end encryption and key delegation for iot. In *Network and Distributed Systems Symposium (NDSS)* (2019).

[34] Mazerik, R. Beast vs. crime attack. https://resources.infosecinstitute.com/beast-vs-crime-attack/#gref, 2013.

[35] Mbed. Delta dfcm-nnn40. https://os.mbed.com/components/Delta-DFCM-NNN40/, 2018.

[36] Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A.-R., and Tarkoma, S. Iot sentinel: Automated device-type identification for security enforcement in iot. In *IEEE International Conference on Distributed Computing Systems (ICDCS)* (2017).

[37] Moshtaghi, M., Leckie, C., Karunasekera, S., Bezdek, J. C., Rajasegarar, S., and Palaniswami, M. Incremental elliptical boundary estimation for anomaly detection in wireless sensor networks. In *IEEE International Conference on Data Mining (ICDM)* (2011).

[38] Netgear. Readycloud at netgear. http://readycloud.netgear.com/client/en/welcome.html, 2020.

[39] Obermaier, J., and Hutle, M. Analyzing the security and privacy of cloud-based video surveillance systems. In *ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS)* (2016), pp. 22–28.

[40] Openwrt. Package: openssl-util. https://openwrt.org/packages/pkgdata/openssl-util, 2018.

[41] Openwrt. Supported routers. https://openwrt.org/toh/start, 2018.

[42] Paterson, K. On the security of rc4 in tls and wpa. http://www.isg.rhul.ac.uk/tls/, 2013.

[43] Raza, S., Shafagh, H., Hewage, K., Hummen, R., and Voigt, T. Lithe: Lightweight secure coap for the internet of things. *IEEE Sensors Journal* (2013), 3711–3720.

[44] Roethlisberger, D. Sslsplit. https://www.roe.ch/SSLsplit, 2018.

[45] Ronen, E., Shamir, A., Weingarten, A.-O., and O'Flynn, C. Iot goes nuclear: Creating a zigbee chain reaction. In *IEEE Symposium on Security and Privacy (S&P)* (2017).

[46] Salowey, J., Z. H. E. P., and Tschofenig, H. Transport layer security (tls) session resumption without server-side state. https://tools.ietf.org/html/rfc5077, 2008.

[47] Shaikh, R. A., Lee, S., Khan, M. A., and Song, Y. J. Lsec: lightweight security protocol for distributed wireless sensor network. In *IFIP International Conference on Personal Wireless Communications (PWC)* (2006), pp. 367–377.

[48] Simpson, A. K., Roesner, F., and Kohno, T. Securing vulnerable home iot devices with an in-hub security manager. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (2017).

[49] Sivanathan, A., Sherratt, D., Gharakheili, H. H., Sivaraman, V., and Vishwanath, A. Low-cost flow-based security solutions for smart-home iot devices. In *IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)* (2016).

[50] SSLdump. Ssldump home page. http://ssldump.sourceforge.net/, 2018.

[51] Statista. Unit shipments of wi-fi enabled smart home devices worldwide from 2016 to 2020 (in millions). https://bit.ly/2EhpNj1, 2016.

[52] Tian, Y., Zhang, N., Lin, Y.-H., Wang, X., Ur, B., Guo, X., and Tague, P. Smartauth: User-centered authorization for the internet of things. In *USENIX Security Symposium (USENIX Security)* (2017), pp. 361–378.

[53] Tuecke, S., Welch, V., Engert, D., Pearlman, L., and Thompson, M. Internet x. 509 public key infrastructure (pki) proxy certificate profile. Tech. rep., 2004.

[54] Weinschenk, C. Wi-fi installed base forecast to reach 17 billion by 2030, driven by the smart home. https://www.telecompetitor.com/wi-fi-installed-base-forecast-to-reach-17-billion-by-2030-driven-by-the-smart-home/, 2019.

[55] Wolfssl. Wolfssl memory usage. https://www.wolfssl.com/docs/benchmarks/, 2018.

[56] Wong, D. Downgrade attack on tls 1.3 and vulnerabilities in major tls libraries. https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2019/february/downgrade-attack-on-tls-1.3-and-vulnerabilities-in-major-tls-libraries/, 2019.

[57] Yu, T., Sekar, V., Seshan, S., Agarwal, Y., and Xu, C. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In *Workshop on Hot Topics in Networks (HotNets)* (2015).

[58] Zhang, C. N., Yu, Q., Huang, X., and Yang, C. An rc4-based lightweight security protocol for resource-constrained communications. In *IEEE International Conference on Computational Science and Engineering (CSE)-Workshops* (2008), pp. 133–140.

[59] Zhang, N., Demetriou, S., Mi, X., Diao, W., Yuan, K., Zong, P., Qian, F., Wang, X., Chen, K., Tian, Y., et al. Understanding iot security through the data crystal ball: Where we are now and where we are going to be. *arXiv preprint arXiv:1703.09809* (2017).

[60] Zoller, T. Tls / sslv3 renegotiation vulnerability explained. http://www.g-sec.lu/practicaltls.pdf, 2011.