Untangling the Knot: Breaking Access Control in Home Wireless Mesh Networks

Xin'an Zhou
xzhou114@ucr.edu
University of California, Riverside
Riverside, California, USA

Juefei Pu
jpu007@ucr.edu
University of California, Riverside
Riverside, California, USA

Keyu Man
kman001@ucr.edu
University of California, Riverside
Riverside, California, USA

Zhiyun Qian
zhiyunq@cs.ucr.edu
University of California, Riverside
Riverside, California, USA

Srikanth V. Krishnamurthy
krish@cs.ucr.edu
University of California, Riverside
Riverside, California, USA

ABSTRACT

Home wireless mesh networks (WMNs) are increasingly gaining popularity for their superior extensibility and signal coverage compared to traditional single-AP wireless networks. In particular, there is a single gateway node and multiple extender nodes that cooperate to provide wireless coverage. We observe that there is no comprehensive research conducted on the security aspects of the control plane of such networks. For example, this decentralized architecture enables each extender node to independently authenticate wireless clients by synchronizing access control policies from the gateway node. However, this synchronization unexpectedly opens an attack surface which has not been scrutinized.

In our research, we conduct an empirical study investigating devices and protocols of six popular home wireless mesh network vendors, focusing on the attack surface introduced by the access policy synchronization. Interestingly, we find that the exact protocols used to support such functionalities vary by vendors, despite the existence of the EasyMesh standard that vendors could opt-in. Furthermore, we find a number of serious security flaws, including but not limited to malicious clients retaining their network access indefinitely and direct compromises of gateway and extender nodes in some cases. These issues arise due to the lack of coordination across different layers of protocols that work together to support the control plane. We reported all of our findings to the affected vendors and they have acknowledged the issues and are working towards fixes (most of the vendors have released patches). Finally, we discuss trade-offs in different existing designs, suggest alternative solutions, and summarize lessons learned from the research.

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**; **Security protocols**; **Cryptanalysis and other attacks**.

KEYWORDS

Wireless mesh network; Wi-Fi; 802.11; security protocols

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3670380

ACM Reference Format:
Xin'an Zhou, Juefei Pu, Keyu Man, Zhiyun Qian, and Srikanth V. Krishnamurthy. 2024. Untangling the Knot: Breaking Access Control in Home Wireless Mesh Networks. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3670380

1 INTRODUCTION

The past decade has witnessed a remarkable revolution of Wireless LAN (WLAN) technologies. Thanks to innovations in wireless standards [7] and techniques (like MIMO [1]), 802.11 wireless networks now provide tremendously faster access speeds and better signal coverage. Due to transmission power limitations imposed by different regulations [33], scalably improving wireless signal coverage requires deploying multiple interconnected wireless nodes. One approach is to hardwire multiple access points, but physical infrastructure limitations can render the solution cumbersome. This makes wirelessly inter-connected mesh networks (i.e., wireless mesh networks) a popular alternative. As a result, a growing number of users have begun to deploy home wireless mesh networks, with a compound annual growth rate of more than 10% in the past five years [2, 3].

Historically, enforcing access control for single-AP wireless networks has been a relatively straightforward task [13, 82]. In contrast, the emergence of multiple access points in wireless mesh networks demands inter-AP communication to support various functionalities such as access policy synchronization. In particular, there are multiple different types of nodes in wireless mesh networks and they have non-identical functionalities and privilege levels (see Section 2.1). For example, in addition to providing wireless coverage, a gateway node manages other extender nodes through inter-AP control plane protocols. Each extender node can independently authenticate wireless clients by synchronizing access control policies with the gateway, also by means of control plane protocols.

To our knowledge, such inter-AP control plane protocols have not been scrutinized for security. It is not well understood how these protocols are used by real-world home wireless mesh networks (hereafter "mesh networks") to enforce security and access control, and whether they are practically secure. To this end, we seek to answer the following research questions:

RQ1: What practical control plane protocols, once considered opaque, are employed in modern mesh networks?

RQ2: How do access points in mesh networks establish mutual trust, which is essential for the security of their control plane?

RQ3: Do protocols and mechanisms utilized by different access points enforce access control *consistently and in a timely way* for mesh networks?

To answer these questions, we first need to identify the relevant protocols and the corresponding programs that carry out the protocols. Unfortunately, we find that there was an absence of a *unified* standard relevant to the control plane protocols regarding how access points authenticate each other and perform common operations such as access policy synchronization. This makes our investigation challenging, due to the fact that vendors have rolled out their own custom implementations. Moreover, this not only introduces diverse security risks, but also makes mesh devices from different vendors incompatible. Subsequently, there has been an effort to standardize inter-AP control plane protocols, i.e., EasyMesh [7], including the operations regarding access policy synchronization. Unfortunately, we find that the focus of the standard is to support inter-operability instead of ensuring security.

In this paper, we conduct a detailed analysis of real-world mesh devices from six popular vendors. We identify a common control plane functionality that involves network access policy synchronization (NAPS). Such a functionality plays a key role in controlling and maintaining the mesh network. Our analysis reveals that implementations that either follow the EasyMesh standard or otherwise have an array of serious, previously unknown vulnerabilities. For example, a wireless client as an attacker can inject or steal access policies (e.g., Wi-Fi passphrases), defeat access control completely and even compromise the security of access points (i.e., obtain shell access).

To help understand and improve the security of NAPS in mesh networks, we summarize the flaws into a number of important security properties that are missing. It is important to note that even though there exist security features at the lower layer (e.g., MAC-layer), they are not properly integrated into the operation of NAPS, leading to classic security issues in a cross-layer fashion. For example, there is a lack of a mechanism to bootstrap trust in four of the NAPS protocol designs we examine, because upper layers make implicit and incorrect assumptions about lower layers, which do not hold in practice. In particular, a concrete flaw pattern we found consequently is that an access point is unable to properly distinguish (possibly malicious) wireless clients from other access points.

In summary, our research is the first empirical study of home wireless mesh networks that demonstrates challenges in designing a collection of protocols that span multiple layers to achieve a common functionality (NAPS). Our research improves the understanding of how flaws in one protocol can influence another for home wireless mesh networks. For example, in one of the mesh networks, the encryption keys of the NAPS protocol layer are leaked through another layer (Section 7.1).

Finally, to improve mesh network security, we also take the first steps towards systematically understanding the inherent challenges in the design space of mesh networks, identify trade-offs for different existing designs, and propose practical defenses.

We make the following contributions:

- We conduct the first systematic study of control plane protocols of real-world home wireless mesh networks. Specifically, we identify novel attack surfaces regarding the network access policy synchronization (NAPS) functionality of mesh networks.
- We analyze real-world mesh devices from six popular vendors and discover various access control vulnerabilities leading to powerful attacks. We find that oftentimes, the flaws stem from the lack of coordination and understanding across different protocol layers.
- We identify key challenges in designing secure NAPS operations for mesh networks. Based on the insights with respect to the identified vulnerabilities, we design practical defenses.
- We open source the attack tools (accessible at https://github.com/seclab-ucr/CCS24Mesh) to enable future research into the security of wireless mesh networks.

Ethical Considerations. In this research, we acquired all the mesh networks from the consumer market. We carefully designed experiments and carried out attacks all in a controlled environment (i.e., in a local residential setting) to avoid any potential harm.

We have reported all discovered vulnerabilities to the relevant vendors and standard communities. All of them have acknowledged the problems, and the majority have patched the vulnerabilities. More details are in Section 8.3.

2 BACKGROUND

In this section, we first briefly review the network architecture of mesh networks. Next, we describe the common workflow of making use of these mesh networks. Finally, we introduce Network Access Policy Synchronization, an integral functionality that is essential for (and exists in) all modern mesh networks.

2.1 Network Architecture of Wireless Mesh Networks

A typical home wireless mesh network is composed of (1) a single *gateway node* (a central access point with the highest privileges) connecting the wireless mesh network to upstream networks (e.g., the Internet), (2) several *extender nodes* (a second, less privileged type of access points) connected to the gateway node via wireless links, or via other extender nodes (and multiple wireless links), improving wireless signal coverage, and (3) standard 802.11 wireless clients connecting either to the gateway node directly, or to any extender node, for network access (Figure 1).

Access points (i.e., the gateway node and extender nodes) are connected to each other via one or more *backhaul links* (i.e., wireless links that serve as "trunks" of a mesh network). Wireless clients (e.g., mobile phones and personal computers) are connected to access points through *fronthaul links* (i.e., wireless links that provide network access). Network traffic generated by wireless clients can be forwarded to the upstream network or other wireless clients by traversing both fronthaul and backhaul links. To control network access to a home mesh network at the medium access control (MAC) layer, two types of identities/credentials are essential: backhaul link identities/credentials and fronthaul link identities/credentials. Backhaul link identities/credentials (i.e., backhaul SSIDs and passphrases) are held by a pair of access points to

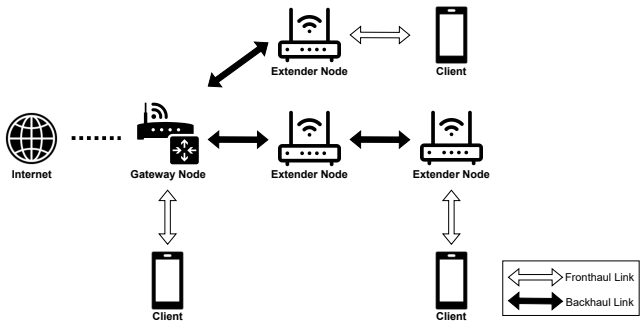

Figure 1: The Architecture of Home Wireless Mesh Networks

establish a *direct* inter-access-point layer-2 wireless backhaul link via a four-way handshake of WPA2/3 [73, 75], which acts as the underlying communication channel of the control plane. Backhaul SSIDs are typically not broadcasted and hidden from the public, and are different from fronthaul SSIDs which are assigned by network owners. Similarly, backhaul passphrases are also different from fronthaul passphrases, and are either pre-configured or set by trusted users during network setup (Section 5). Fronthaul link identities/credentials directly control wireless clients' access to a mesh network. In other words, standard 802.11 wireless clients use fronthaul link credentials (e.g., WPA2/WPA3 passphrases) to connect to gateway/extender nodes via fronthaul SSIDs and obtain network access.

For better usability, home wireless mesh networks maintain a single/shared fronthaul link credential (i.e., the same "Wi-Fi passphrase") that is used for accessing every gateway/extender node. Maintaining the fronthaul link credential across the mesh network requires the credential to be synchronized via Network Access Policy Synchronization (NAPS) among access points (described below). Moreover, extender nodes use the same IP subnet as wireless clients by default. For the devices we have surveyed, firewalls are either non-existent or not enabled by default.

2.2 Common Workflow of Home Mesh Networks

There are two types of users associated with home mesh networks: *network owner* and *guest user*.

Network Setup. A network owner sets up her mesh network by (1) connecting a gateway node first with an upstream network (e.g., with an ISP-provided internet modem), (2) enrolling other extender nodes into the mesh network, (3) designating universal fronthaul link credentials (e.g., WPA2/WPA3 passphrases) for the mesh network, (4) establishing management credentials (e.g., "administrator passwords" for web-based management interfaces). The network owner can also modify the management settings later.

Network Access. To obtain network access to the mesh network (e.g., for Internet services), a guest user asks for the mesh network identity (i.e., fronthaul SSID) and fronthaul link credentials from the network owner. In this way, the wireless client of the guest user can authenticate and associate with a fronthaul link of any gateway/extender node. Network packets can therefore travel between the wireless client and upstream networks via a fronthaul link, and potentially through multiple backhaul links as well.

Access Revocation. To revoke network access from these guest users, a network owner can rotate fronthaul link credentials (e.g., by changing network settings via the management interface of her mesh network). In this way, guest users will lose access to the mesh network, because all access points will apply new fronthaul link credentials after Network Access Policy Synchronization (see below), which blocks previous guest users at the wireless link/MAC layer.

2.3 Network Access Policy Synchronization (NAPS)

A secure and trustworthy mesh network needs consistent and timely enforcement of network access policies. In a home mesh network, the gateway node is the authority that establishes network access policies, and multiple extender nodes work in conjunction with the gateway node to enforce such network access policies.

We define Network Access Policy Synchronization (NAPS) to be the functional unit that is responsible for the aforementioned security-sensitive network access control management. Specifically, during the NAPS operation, fresh network access policies (e.g., fronthaul SSIDs, fronthaul link credentials, management credentials of network owners, etc.) are transmitted from a gateway node to extender nodes, utilizing inter-access-point backhaul links. In this way, the gateway and extender nodes can apply the fresh network access policies.

Empirically, NAPS is implemented using *in-band* control plane protocols, and different implementations of NAPS often use authenticated and encrypted network protocols to provide isolation for the control plane; the encryption used by NAPS enables its traffic to "securely" traverse the layer-2 wireless links just like any other data traffic. Since encryption is end-to-end, naive man-in-the-middle attacks (e.g., via ARP spoofing attacks) will not easily compromise the security of the control plane (e.g., because decrypting/injecting such control plane data will be difficult given the cryptographic protections).

3 THREAT MODEL

We consider an *in-network* attacker (i.e., tier-1 clients in Figure 2) who leverages her own fronthaul link credentials (e.g., WPA2/WPA3 passphrases provided by a hostess) to obtain temporary access to a target mesh network. Such an attacker does not know the backhaul link identities/credentials. The extender nodes are considered tier-2 nodes with higher privileges than clients. Finally, the gateway node is considered tier-3 because it can instruct extender nodes, e.g., by distributing access policies to them.

We assume that the in-network attacker can exploit traffic redirection techniques (e.g., ARP poisoning/spoofing attacks) to inspect and possibly manipulate inter-access-point traffic. We also assume this attacker can read any software/hardware documentation, and obtain the firmware for the target mesh network. We do not assume this attacker can physically manipulate the target mesh network.

An in-network attacker aims to achieve privilege escalations, such as (1) retaining her mesh network access, so that even if the network owner changes the fronthaul link credentials, she can

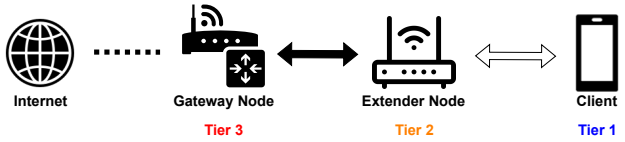

Figure 2: Privilege Tiers of Home Wireless Mesh Networks

continue to gain access, and (2) compromising the integrity of the mesh network, e.g., distributing malicious access policies to extender nodes, or even obtaining shell access to access points.

4 OVERVIEW

Goals. We aim to study real-world NAPS operations and their protocols, distill expected security properties, and understand whether they hold in practice. To this end, we (1) identify NAPS programs, (2) analyze NAPS programs for their authentication and key management (AKM) across different network layers, (3) formulate the security expectations/properties of NAPS which were unclear before our study, (4) categorize weaknesses into distinct flaw patterns.

Challenges. Analyzing NAPS functionalities is challenging in practice. We define *NAPS programs* as the specific application programs that implement the access policy synchronization functionality. We observe that NAPS programs of different vendors utilize disparate protocols. These protocols work at different layers (e.g., IEEE 1905 at layer-2.5 versus application/TCP/IP layer) and have distinct security properties and guarantees. As Figure 3 shows, the backhaul link keys/credentials are used for secure communication at the link/MAC layer. On the other hand, NAPS protocols are implemented at higher layers which will have their own security mechanisms and secret keys as well. It is unclear whether and how the NAPS interacts with the link/MAC layer, and what the security implications are. In addition, most of the NAPS programs lack source code. It is therefore difficult to understand and even identify the right programs responsible for NAPS.

Workflow. Methodologically, we identify NAPS programs by first facilitating dynamic debugging (i.e., trying to get full access to gateway/extender nodes by rooting/jailbreaking them). This can be done either by (1) directly obtaining shell access through physical debugging interfaces, or (2) exploiting classic types of vulnerabilities in traditional attack surfaces (e.g., command injection via the management interface reachable by an admin user only). Note that the sole purpose of applying these two testing techniques is to allow us to identify NAPS programs and perform whitebox dynamic testing, ultimately facilitating vulnerability discovery relating to the novel attack surface exposed by NAPS.

Once we find the NAPS programs, we can inspect the protocols they speak and categorize the cryptographic network protocols used by these NAPS programs into two types: standardized or proprietary. For both types of protocols, we directly examine real binary code using disassemblers and decompilers to understand how they handle authentication, key exchange, and confidentiality/integrity/freshness protection. For standardized protocols (e.g., SSH [80], TLS-SRP [66], EasyMesh [7]), we also examine protocol specifications and relate them to concrete implementations.

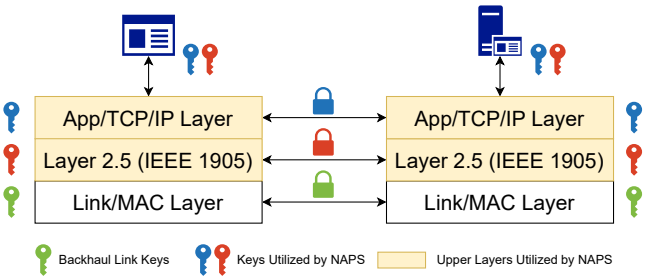

Figure 3: NAPS Clients/Servers Use Traffic Encryption at Different Network Layers

Security Expectations and Insights. As mentioned previously, within a mesh network, there are nodes with differing privilege levels. Therefore, the first important security expectation/challenge is that *every node must be able to differentiate the node that they are talking to with respect to their roles and privilege levels.* This ensures that the access policies received from higher privilege nodes are authentic (not faked by less privileged nodes) and are not leaked to unintended nodes (i.e., client nodes). Second, since NAPS operates at higher layers on top of the link/MAC layer, it is important to ensure that *NAPS protocols should perform proper authentication collaboratively with the lower layer.* Third, *compromising one protocol layer's credentials should not compromise the security of another layer.*

Summary of Security Flaw Types. Unfortunately, we find the above security expectations do not hold in any of the six real-world mesh devices we tested. Note that since these protocols vary in layer, type, and complexity (many are proprietary and ad-hoc), it is difficult to summarize them narrowly based on low-level vulnerability types. Instead, we categorize the security flaws into two broad and orthogonal types based on their cross-layer interactions and impacts:

Type I: *missing cross-layer trust.* In this type, we consider protocol stacks where the lower-layer (link/MAC) trust anchor does not facilitate the establishment of trust for NAPS in upper layers. This flaw alone directly leads to the failure to distinguish access points from wireless clients. For example, some vendors use separate TLS certificates for NAPS that are not pre-configured or signed under PKI.

Type II: *cross-layer trust compromise.* In this type, we consider protocol stacks where the lower-layer trust does help the NAPS layer establish trust. However, we find flaws located in one layer can affect another layer (e.g., creating more attack surfaces and enabling further compromises) and sometimes the flaws directly originate from unsafe behaviors across layers.

Roadmap. In the following sections, we first demystify how access points in mesh networks establish/bootstrap layer-2 wireless links, which involves certain forms of initial trust (Section 5). Then, we showcase that trust anchors can sometimes be missing for NAPS at higher layers, and how attackers can exploit such *missing cross-layer trust* for four NAPS protocols (Section 6). Then, Section 7 reports further in-depth security analyses (along with strong attacks) of three more real-world NAPS protocols used by legitimate access points. It shows that even if cross-layer trust exists, because

Table 1: Generation and Distribution Methods of Backhaul SSIDs and Passphrases for Six Home Wireless Mesh Networks

Vendor	Generation Method	Distribution Method
Netgear Orbi	Unique for every bundle	Pre-configured
ASUS AiMesh	Derived from trust anchors	Pre-configured
TP-Link Deco	Derived from trust anchors	Pre-configured
Linksys	Unique for every bundle	By trusted users
Wyze	Unique for every bundle	By trusted users
AmpliFi	Unique for every bundle	By trusted users

of insufficient cross-layer cooperation, real-world NAPS protocols cannot enforce access control consistently and in a timely way and have unfortunately become novel attack surfaces.

5 INTER-AP TRUST ESTABLISHMENT AT LAYER 2

A foundation for NAPS to operate securely between access points is to first establish the initial wireless layer-2 links between access points (i.e., "backhaul/trunk links" in Figure 1). This means the shared backhaul link identity and credentials (i.e., backhaul SSID and passphrases) should be established and known to both the gateway node and extender nodes. Such secrets should not be leaked to adversaries, i.e., client nodes that have the lowest privilege in the mesh network. By studying six of the most popular mesh networks, we observe that the generation and distribution methods of backhaul SSIDs and passphrases, as being either (1) the gateway node and extender nodes are all pre-configured with a hard-coded backhaul SSID, and hard-coded backhaul passphrases that are distributed with each mesh device bundle sold together (e.g., Netgear Orbi), or (2) gateway/extender nodes are pre-configured, again for each device bundle sold together, with a hard-coded and shared secret. The backhaul SSID/passphrases are then derived from the secret using a fixed one-way derivation function (e.g., ASUS AiMesh and TP-Link Deco), or (3) trusted users (e.g., the network owner) are involved in establishing the backhaul links via out-of-band methods including Bluetooth (e.g., Linksys, Wyze, and AmpliFi). We summarize the behaviors of the products from different vendors in Table 1.

For example, ASUS AiMesh Wi-Fi Systems [4] employ a design that uses a pre-configured shared secret that is a static 16-byte key, called `amas_bdlkey`. `amas_bdlkey` is then hashed using SHA-256 to derive the backhaul SSID, and the second-order SHA-256 hash serves as the backhaul passphrase.

TP-Link Deco Mesh Wi-Fi Systems [68] pre-configure each gateway/extender node in the same Deco bundle with a shared secret: a pair of 512-bit RSA public/private keys. Then, TP-Link designed a one-way function to (a) extract every seventh byte of that RSA *public* key to form a 16-byte backhaul SSID, and to (b) extract every eleventh byte of that RSA *private* key to form an 8-byte backhaul passphrase.

Wyze, Linksys, and AmpliFi mesh networks, however, leverage the network owner's companion mobile app to securely establish initial trust and connectivity between a new extender node and the gateway node during network setup. These methods are considered secure owing to their near-field characteristics, e.g., Bluetooth is used. While utilizing pre-configured credentials or leveraging trusted users for backhaul link establishment can enhance security, we point out that such backhaul link credentials are seldom rotated/refreshed. In fact, vendors in the first two categories have completely hard-coded backhaul passphrases and therefore cannot rotate (unless there is a firmware update that changes the hard-coded key values and forces a reset). For vendors in the third category, even though the backhaul passphrases can change, they would require a reset of the network which is cumbersome (a user would have to set up the network again with the companion app). This means that the backhaul passphrases are a critical secret that should be well-guarded. Otherwise, an attacker who manages to extract the backhaul passphrases can impersonate gateway or extender nodes and persist in the network indefinitely, as long as the passphrases are not changed.

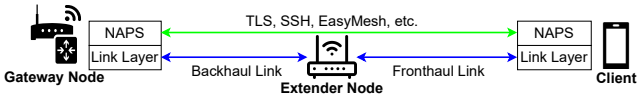

Figure 4: NAPS Layers Operate Independently of the Link Layer

6 MISSING CROSS-LAYER TRUST

From the previous section, we can see that vendors have chosen simple methods to establish the layer-2 links. Even though these methods sacrifice interoperability due to pre-configured and hard-coded secrets, they do achieve the goal of establishing the link-layer trust securely. Unfortunately, after analyzing the NAPS protocols in the upper layers, we find that the link-layer trust does not always carry over to the upper layers. Specifically, as shown in Figure 4, the NAPS layer operates completely independently of the link layer. In some cases, the secrets derived on backhaul links are not used to bootstrap the upper layer protocols. This means that a malicious client node does not need to worry about the backhaul links and simply speaks the upper layer protocols such as TLS and SSH to access the NAPS interface on the gateway or extender node. Furthermore, even though the upper layer protocols are seemingly secured with encryption and authentication (e.g., TLS), their trust anchors are not properly established. For example, we see self-signed certificates being used for TLS connections in products from some vendors.

We now briefly summarize the results of our analysis of NAPS protocols for different vendors. As shown in Table 2, we list the NAPS protocols implemented by each vendor. We can see that sometimes a vendor will implement more than one protocol to support NAPS operations. For example, Netgear Orbi implements both a custom protocol (SOAP[1]-over-TLS) and the EasyMesh standard. We also list the "flaw type" for each protocol. Type I indicates that the specific protocol does not leverage any trust anchors already established in layer 2, which will be covered in this section. Type II

[1]Simple Object Access Protocol.

Table 2: Summary of Measurement Results for NAPS Protocols of Six Mesh Networks

Vendor	NAPS Protocol	Flaw Type	Configuration Transmitted	Attacker gets shell access?
Netgear Orbi	SOAP over TLS	I	(1)★Δ, (2)★Δ	Yes
	EasyMesh	I	(1)★	No
ASUS AiMesh	Custom encrypted protocol	II	(1)★Δ, (2)★Δ	Yes
TP-Link Deco	TCP over Dropbear SSH	II	(1)★Δ, (2)★Δ	Yes
Linksys	TLS-SRP	II	(1)★Δ, (2)★Δ	Yes
Wyze	MQTT with TLS	I	(1)★, (2)★	Theoretically Possible
	EasyMesh	I	(1)★	No
AmpliFi	WebSocket with TLS	I	(1)★Δ, (2)★Δ	No

I. Missing cross-layer trust, II. Cross-layer trust compromise
(1) fronthaul/backhaul SSIDs and passphrases, (2) (hashes of) management credentials,
★: leaked, Δ: writable/controllable by an attacker

is to be covered in Section 7. As we can see, there are three vendors and five protocol implementations suffering from the Type I flaw.

We also describe the result of the flaw in the last two columns. For example, we show that the EasyMesh implementation of Netgear Orbi is subject to its configuration (or access policy) being leaked to a malicious client node. Next, we dive into the details of each vendor and its protocol implementation.

6.1 Case Study: Netgear Orbi Mesh Wi-Fi Systems

We describe the flaw of the custom NAPS protocol in Netgear Orbi as a case study. It employs SOAP-over-TLS where SOAP is a lightweight messaging protocol. Together with TLS, the messages are supposed to be transmitted securely.

However, we identify that the initial trust used to establish layer 2 (Section 5), is not carried to SOAP-over-TLS. Each extender node in an Orbi mesh network generates a self-signed TLS certificate locally for the SOAP service, and does not have the self-signed certificate registered at the gateway node during the initial network setup. As a result, when performing NAPS, it can be inherently challenging for the gateway node to verify the authenticity of extender nodes, and such TLS encryption that protects the transmitted access policies is only *opportunistic* [28].

Thus, Netgear Orbi came up with a remedy: to perform inter-access-point authentication inside SOAP. This is done by the gateway node's sending an MD5 hash of the string "`NETGEAR_Orbi_`<MAC_G>`_`<MAC_E>`_password`" for inter-access-point SOAP authentication. Interestingly, the string does not in fact contain any secret. MAC_G and MAC_E are simply the last three public bytes of the gateway and an extender node's `br0` interface addresses, respectively. An attacker can craft the same MD5 hash easily (noting that the "`password`" string represents eight static ASCII characters found in the SOAP handler programs of the Netgear Orbi firmware and is not a secret). After such an insecure authentication, the gateway node can send/push authoritative access policies via SOAP to (the perceived) extender nodes (e.g., to synchronize the network owner's management passwords, fronthaul/backhaul SSIDs and passphrases, etc., with the extender nodes). We point out that such a design represents challenges in establishing inter-access-point trust, and can be broken by active in-network attackers.

PoC Attack. We, acting as an in-network attacker, successfully launched an active ARP-based man-in-the-middle attack to intercept inter-access-point SOAP-over-TLS traffic. This was possible because the gateway could not distinguish an attacker from a legitimate access point: the gateway did not have the correct certificates of extender nodes. We could thus setup a fake SOAP-over-TLS server and redirect benign SOAP-over-TLS traffic to this fake server to wiretap fresh fronthaul/backhaul SSIDs and passphrases before they reach extender nodes. As a result, we demonstrated successfully that the attacker could regain network access even after being disassociated from the mesh network due to fronthaul passphrase changes.

We also successfully masqueraded as the gateway node and connected to SOAP interfaces of extender nodes. We could calculate the aforementioned MD5 hash (i.e., the SOAP credential) by querying our local ARP cache to obtain MAC_G and MAC_E, and defeat inter-access-point authentication. As a result, we could arbitrarily manipulate any extender node's management passwords. We showed that we could continue to use the attacker-controlled management password to unlock a `telnet` debugging interface (a hidden service/functionality listening on UDP port 23) [49] and obtain shell access on extenders, which was done by reverse-engineering the specific unlocking algorithm of that hidden service and constructing a magic packet using the management password. Finally, we can manipulate an extender node's DNS server settings, among other network configurations.

6.2 Case Study: Wyze Mesh Wi-Fi Systems

Wyze mesh networks also struggled to differentiate between legitimate access points and potentially malicious wireless clients *across network layers*, and employed a design reliant on the idea of "security through obscurity". Technically speaking, Wyze Mesh Wi-Fi Systems did not correctly establish inter-access-point trust for NAPS purposes. As a result, an in-network attacker can masquerade as a legitimate access point and read network settings arbitrarily.

To perform NAPS, a legitimate extender node maintains an encrypted MQTT connection (protected by TLS-PSK [39]) with the gateway node. The MQTT (i.e., Message Queuing and Telemetry Transport) protocol is a widely used lightweight IoT messaging protocol. Through the publish-subscribe messaging paradigm, MQTT

"publisher" clients can send IoT control messages to MQTT "subscriber" clients through MQTT servers/brokers [81].

For Wyze mesh networks to push fresh access policies to extender nodes, the gateway node acts as an MQTT publisher, and all extender nodes act as MQTT subscribers. Once new access policies are available on the gateway node, they are immediately sent by a user-space agent program to the MQTT channel under a specific MQTT topic, and will be received by all MQTT subscribers subscribed to that specific topic.

Surprisingly, we discover that the MQTT server on the gateway node uses TLS-PSK credentials found in the Wyze mesh network firmware. The firmware is published on Wyze's official website and used by all Wyze mesh devices of the same model (not just the ones we purchased and tested). This means all Wyze mesh networks share a common key for MQTT connections. This allows an in-network attacker with knowledge of this key to break inter-access-point authentication and impersonate an access point.

We are also able to spot a command-and-control interface in the user-space agent program, where the gateway node sends terminal commands via MQTT for extender nodes to execute. This theoretically can allow an attacker to execute arbitrary commands and achieve remote code execution on extender nodes. However, realizing such an attack requires the attacker to understand vendor-specific MQTT command formats, which requires extra engineering efforts.

PoC Attacks. We, as an in-network attacker, first performed an ARP spoofing attack to block the MQTT traffic between legitimate extender nodes and the gateway node. In this way, legitimate extender nodes could not receive fresh access policies and would not switch the fronthaul link SSID/credentials. As a result, the attacker could prevent itself from being disconnected. After that, we connected to the MQTT server on the gateway node using TLS-PSK credentials found in the Wyze mesh network firmware. We then maliciously subscribed to the MQTT topic for NAPS and steal fresh fronthaul/backhaul SSIDs and passphrases, evading access control effectively.

6.3 Case Study: AmpliFi Mesh Wi-Fi Systems

The problem of missing cross-layer trust also extends to AmpliFi mesh networks. Due to the lack of a unified standard that guides the design and security of NAPS, AmpliFi has chosen WebSocket with TLS as its proprietary protocol for transmitting configurations between access points. However, the initial trust to establish backhaul links is not applied cross-layer. Each access point generates its own TLS certificate locally for the WebSocket service, similar to Netgear Orbi. More specifically, access points do not verify the validity of the certificates and the design is therefore effectively achieving opportunistic encryption. However, such inter-access-point TLS connections are clearly vulnerable to active man-in-the-middle attacks (e.g., ARP spoofing/poisoning attacks), leaking sensitive control plane data such as backhaul and fronthaul passphrases.

PoC Attacks. We, as an in-network attacker, first launched an active ARP-based man-in-the-middle attack to inspect the WebSocket traffic of AmpliFi mesh networks: we setup a fake WebSocket server (with TLS) and intercepted inter-AP WebSocket traffic, while using another TLS connection to pass intercepted cleartext payloads to the original/real WebSocket server. The data encapsulated in TLS sessions (e.g., backhaul and fronthaul passphrases) were then decrypted into cleartext MessagePack formats [24]. By dynamically inspecting/modifying these cleartext data, we were able to steal/manipulate the backhaul and fronthaul passphrases of the target mesh network, effectively evading network access revocations based on fronthaul passphrase renewals.

6.4 Case Study: Analyzing Wi-Fi EasyMesh

In the prior sections, we analyzed three proprietary NAPS protocols and performed strong end-to-end attacks to demonstrate their security weaknesses. Admittedly, as the "secret sauces" of mesh networks to enforce uniform access control, these custom NAPS protocols have rendered mesh networks more usable. However, as a result, interoperability is broken since access points of different vendors cannot inter-operate.

As first attempts to solve this problem, Wi-Fi EasyMesh is a new standard that tries to improve interoperability for mesh networks. Essentially, Wi-Fi EasyMesh tries to define a standardized approach for the critical NAPS feature (among other functionalities), so that EasyMesh-certified products from different vendors can be compatible with each other and enforce uniform access control in a heterogeneous mesh network.

Similar to other NAPS protocols, Wi-Fi EasyMesh enables network access policies to be transmitted from the authoritative gateway node to other extender nodes. This is done *in-band* using logical IEEE 1905 Ethernet frames over inter-access-point wireless layer-2 links. By studying the specification documents of EasyMesh and relating them to concrete implementations, we point out that even standardized approaches have failed to properly define and solve the security challenges for the critical NAPS functionality.

At a high level, EasyMesh lacks authentication and has practically failed to distinguish between malicious wireless clients and legitimate access points, possibly for the benefits of mesh network extensibility. Consequently, an in-network attacker (i.e., a wireless client) can directly follow EasyMesh protocol steps and consistently pull/steal fresh network access policies (e.g., including fronthaul/backhaul SSIDs and passphrases) from EasyMesh IEEE 1905 interfaces. The attacker can thus evade access control revocation mechanisms based on fronthaul/backhaul key renewals.

Specifically, NAPS in the EasyMesh specification is performed in the following way: An extender node, using the role of an "enrollee" in EasyMesh specification, sends its own "enrollee" public key, nonce value and MAC address to the gateway node, with a Wi-Fi Simple Configuration (WSC) M1 message type [40] of IEEE 1905, a layer-2.5 protocol [44]. The gateway node uses the role of a "registrar" in EasyMesh specification. The "registrar" will then reply with its own "registrar" public key, nonce value, and "Encrypted (access point) Settings", as WSC M2 messages. "Encrypted Settings" are essentially fresh access policies (e.g., fronthaul/backhaul SSID and passphrases) encrypted with "KeyWrapKey", which is derived from aforementioned enrollee/registrar nonces, enrollee MAC address, and an ephemeral Diffie-Hellmann key. That Diffie-Hellmann key per se is derived equivalently from either (1) the private key of the enrollee and the public key of the registrar, or (2) the private key of the registrar and the public key of the enrollee. In a

nutshell, the "enrollee" can thus directly use its own private key to decrypt the "Encrypted Settings" and obtain fresh access policies. The root cause is that there lacks trust anchors for such layer-2.5 and only opportunistic encryption is provided for transmitting access policies.

PoC Attacks. We repurposed the WPSCrack [77] tool for decrypting "Encrypted Settings" in WPS M2 messages, which was carried in IEEE 1905 frames sent by victim gateway nodes. This was possible because the attacker could generate her own malicious public/private key pair, and provide this malicious "enrollee public key" to pull/steal fresh network access policies, such as fronthaul/backhaul SSID and passphrases, and then use this malicious "enrollee private key" to derive the aforementioned Diffie-Hellmann key, with which "KeyWrapKey" could be finally derived to decrypt "Encrypted Settings". Since there was no limitation on the number of trials, we verified that an attacker could continuously perform a key-stealing attack to obtain fresh passphrases before they were forced to disconnect from the network (due to passphrase changes). This means that the attacker would regain network access to the target mesh network. This attack is potent against all mesh network products that enable EasyMesh, including Netgear Orbi and Wyze in our study.

7 CROSS-LAYER TRUST COMPROMISE

In the previous section, we identified flaws that arise due to missing cross-layer trust. In this section, we study the remaining protocol stacks where the trust is indeed carried over from the lower layer to the NAPS layer. Yet, we managed to identify both design and implementation flaws of NAPS protocols, compromising the cross-layer trust. In other words, we demonstrate how a flaw in one protocol layer can lead to compromises of another layer. Some of the flaws are directly caused by unsafe behaviors across layers. First of all, our study shows that all NAPS control planes use cryptographically enforced isolation, rather than logical separation/isolation (like VLAN). This means that a malicious client node can still reach the protocol endpoints and attempt to participate in NAPS operations and speak the NAPS protocols. Second, we study the new attack surface where an in-network attacker (i.e., client node) can *reach* NAPS endpoints and *disrupt* NAPS protocols by guessing cryptographic keys and/or exploiting software security vulnerabilities. As a result, access control of mesh networks can be totally evaded. In this section, we report three in-depth security analyses of real-world NAPS protocols and showcase their security weaknesses with strong end-to-end attacks. We summarize the results in Table 2 — protocols with Type II flaws.

7.1 Case Study: ASUS AiMesh Wi-Fi Systems

ASUS AiMesh Wi-Fi Systems [4] use a custom inter-access-point cryptographic protocol to perform NAPS. As we know, rolling/creating custom crypto protocols is generally risky and difficult to get right. After we analyze the details, we find that the keys used for NAPS can be effectively learned with a very low computational cost. This is due to a cross-layer flaw where certain information leakage occurs in layer 2.

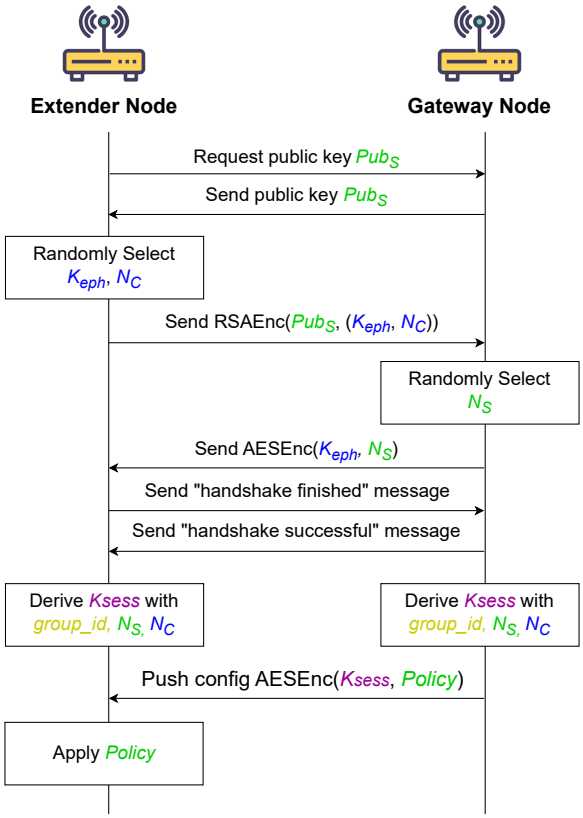

Figure 5: Protocol Steps of the ASUS AiMesh Protocol

Specifically, `cfg_server` and `cfg_client` are a pair of network programs available on a gateway node and an extender node respectively. They perform authenticated and encrypted NAPS (Figure 5). First, the client dials a clear-text TCP connection to the server to request its public key (denoted as Pub_S). Then, the client selects an ephemeral 256-bit AES key (denoted as K_{eph}) and a 64-bit client nonce (denoted as N_C), and encrypts them using Pub_S. The encrypted content is then sent to the server. On receiving K_{eph} and N_C, the server selects its own server nonce (denoted as N_S), encrypts it using K_{eph}, and sends it to the client. The client will then obtain N_S, and send an *unauthenticated* "handshake finished" message code to the server. Finally, the server will return a "handshake successful" message code to the client. So far, we point out there is no real authentication, because an attacker can masquerade as the gateway node and respond with its own public key and establish a valid session.

After such an *unauthenticated* handshake, the server and the client both use a pre-shared credential called *group_id* (a 128-bit random-looking value), as well as N_C and N_S, to derive a session key with SHA-256. In this way, they can perform authenticated and encrypted inter-access-point NAPS afterwards. Thus, the secrecy of *group_id* should be strictly protected. Otherwise, an in-network attacker might use *group_id* to authenticate to the gateway node and masquerade as an extender node to steal network access policies, or to decrypt and manipulate inter-access-point NAPS traffic.

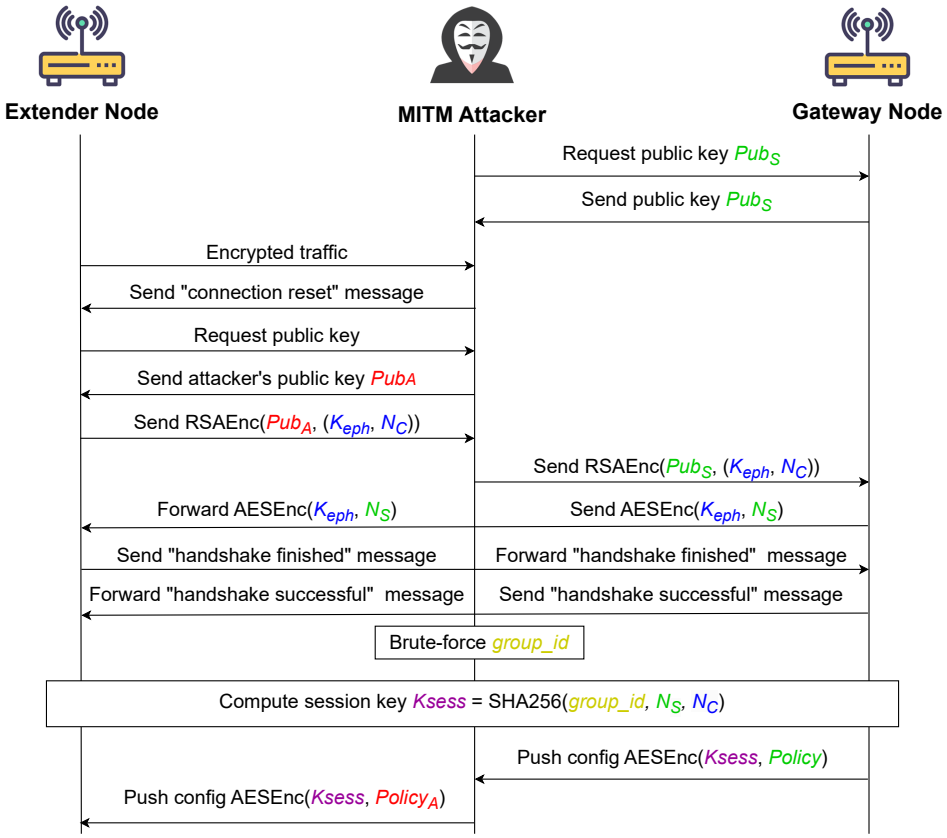

Figure 6: Attacking ASUS AiMesh Protocol

We discover that, unfortunately, *group_id* is not well protected. First, by analyzing the NAPS programs, we find that *group_id* is generated by performing an MD5 hash on the concatenation of (1) the gateway node's `br0` interface address, and (2) a private unix timestamp in seconds (denoted as T_{pri}) of ASUS mesh's initial setup time or the most recent factory-reset time. An in-network attacker can passively obtain (1) by querying their operating system's ARP cache for the gateway node's MAC address. However, without knowing (2), an attacker will be unable to easily guess the correct value for *group_id*. If the attacker has to brute force all possible timestamp values within a year, it would be over 30 million (near 2^{25}) possibilities or less than 25 bits in entropy. This may seem like something within the feasible range for brute force attacks. However, the only oracle for success is going through the protocol step by step, which is an online process and is time-consuming. Our experimental result suggests that it will take at most 23 days to finish the brute force if the mesh network was set up back in Jan 2020 (when the devices were first available on the market).

However, by analyzing `cfg_server`, we find that a special hash value of *group_id* is broadcasted in clear text by the gateway node in 802.11 beacon frames, which effectively creates an offline oracle for much more efficient brute force attacks. Specifically, the gateway node first combines a public 32-bit timestamp (also broadcasted in clear text, denoted as T_{pub}) with *group_id* using a bitwise `AND` operation. Then, the combined value is hashed using SHA-256. Finally, the first 16 bytes of that SHA-256 hash (denoted as H_{GID}), and T_{pub}, are broadcasted by the gateway node in clear text inside the 802.11 management beacon frame's vendor-specific information element. This design possibly helps ASUS AiMesh access points in a bundle to identify each other over wireless channels but the hash is effectively an offline oracle for guessing *group_id*. It is worth noting that this is an interesting cross-layer interaction where *group_id* is a secret used in both layer 2 and NAPS (ASUS AiMesh). This exposes a bigger attack surface of the secret which ultimately allows the attacker to learn it.

PoC Attacks. By sniffing ASUS AiMesh 802.11 beacon frames, we could obtain T_{pub} and H_{GID} embedded in 802.11 vendor-specific information elements. Then, we simply enumerated all possible values of T_{pri}, and thus all possible values of *group_id*, and compared *group_id* hashes to the broadcasted H_{GID} to infer the correct value of *group_id*. The space of enumeration was modest, because the release dates of ASUS AiMesh products were in recent years, and the granularity of T_{pri} was in seconds. Note that the computation was conducted completely offline, without interacting with any access points, thereby being fast. End-to-end experiments showed that using `python`'s multiprocessing library to perform on-demand

enumeration required no more than 20 seconds on the commodity device MacBook Pro (16-inch, 2023) even if the mesh network was set up back in Jan 2020. This made brute-force very much feasible for practical attacks.

After correctly guessing *group_id*, we could perform two types of strong attacks that break access control. First, we could disguise as an extender node and use authenticated and encrypted connections to pull the fresh network access policies from the `cfg_server` instance of the gateway node. End-to-end experiments showed that, we could pull fresh network access policies (including the fresh fronthaul passphrases) right before the new network access policies were applied, and stay in the network indefinitely.

Second, we could disguise as the gateway node to distribute malicious network access policies to a victim extender node. To this end, we first performed an ARP-based man-in-the-middle attack against a victim extender node to intercept the TCP traffic between the victim extender node and the gateway node. We then exploited an unauthenticated "reset" op-code of `cfg_server` and `cfg_client` to terminate a benign encrypted NAPS connection between the victim extender node and the gateway node. The victim extender node would then start over all protocol steps and pull Pub_S from the attacker, and finally build an "authenticated" and encrypted connection with the attacker since the attacker could know *group_id* (Figure 6). The attacker could then distribute malicious network access policies ($Policy_A$) to that victim extender node, such as to install a malicious SSH management public key, which could only be installed by network owners in a benign scenario. We could then gain malicious root access to the extender node by using the malicious SSH management private key, which was a management functionality of ASUS mesh networks. After this, we could continue to collect/modify/inject wireless traffic at that extender node, to modify the network owner's management credentials (password hashes for the web management interface) to attacker-controlled values, and even leak the layer-2 trust anchor "`amas_bdlkey`" and non-rotating backhaul SSID/passphrases. In other words, the more functionalities there were on the extender nodes, the more damage the attacker could inflict.

Attacking Vendor Mitigation. We reported this attack to ASUS soon after the discovery. ASUS developed an experimental patch that effectively uses two independent secrets for layer 2 and the NAPS layer. While it is a step in the right direction, the patch is ineffective for devices that are already compromised. Specifically, instead of trying to prevent secrets from being leaked, ASUS tried to invent another new secret key (called "`cfg_key`") to secure NAPS connections. In the patched network setup process, the extender node will have the chance to fetch this newly invented NAPS secret key `cfg_key` with backhaul passphrases. After that, inter-access-point NAPS connections will be encrypted using the new `cfg_key`, abandoning the use of *group_id*. Although this mitigation can prevent new attacks (because `cfg_key` will not be known to a new attacker), it cannot stop existing/previous attackers: they can still attack the network setup mechanism, using compromised backhaul passphrases to query the new key `cfg_key`. After that, the attacker can still decrypt/inject access policies. This shows that establishing and maintaining cross-layer trust for mesh networks is inherently a challenging problem. This problem is related to Post-Compromise Security [15] – how to protect a system after the secret has been compromised. We will discuss this issue and propose a defense in Section 8.2.

7.2 Case Study: Linksys Velop Mesh Wi-Fi Systems

Linksys Velop Mesh Wi-Fi Systems use the Secure Remote Password (SRP) protocol [79] and related mechanisms for inter-access-point NAPS. SRP, as a zero-knowledge password proof protocol, effectively allows an extender node to securely authenticate to the gateway node. Different from traditional authentication paradigms, SRP requires a server to authenticate a client with a "verifier" that is tied to the client's correct password but is not using or storing the correct password directly. This way, even if the attacker manages to compromise the SRP server, the attacker cannot obtain client passwords.

In the case of the SRP protocol on Linksys, an extender node's SRP credential is an SRP username and password preassigned by the gateway node during the network setup. The gateway node holds a cryptographic verifier which is tied to the extender node's password. In order to pull network access policies from the gateway node, an extender node utilizes a network client program `sct_client` to dial a TLS-SRP [20, 21] connection (a protocol that uses purely the username and password for authentication) to the corresponding server program `sct_server` on the gateway node. In this way, the extender node can authenticate to the gateway node and build an encrypted channel with it, so that the network access policies can be transmitted securely.

Unfortunately, the NAPS interfaces on Linksys Velop gateway nodes have also created a new attack surface for in-network attackers. When authenticating extender nodes, the SRP server on the gateway node will query an internal database with the SRP username sent through the TLS-SRP connection by an extender node. After that, the SRP server will check if the provided SRP password is mathematically bound to the cryptographic verifier stored in that internal database for that SRP username. During the database query process, the SRP username was not properly sanitized, and was passed as a command line argument. An in-network attacker can thus perform a pre-authentication command injection (not SQL injection) and obtain shell access to the gateway node.

Even worse, the gateway node was also found to store SRP passwords in clear text, which is undesirable. Originally, for SRP to fully enjoy its theoretical security properties, SRP servers should never store client passwords.

PoC Attacks. We successfully launched a command injection attack by tainting the `srpuser` parameter of OpenSSL's `s_client` command with malicious terminal commands. This could effectively steal both usernames and passwords by retrieving the corresponding database files. An attacker could then masquerade as an extender node with compromised SRP usernames and passwords. Since these SRP usernames and passwords never rotate, an attacker could consistently pull fresh network access policies from the gateway node and obtain the fresh fronthaul passphrases. The in-network attacker could thus evade access revocations based on fronthaul passphrase renewals, and stay indefinitely in the mesh network.

We further confirmed that an in-network attacker could even spawn a reverse shell using the command injection, through which

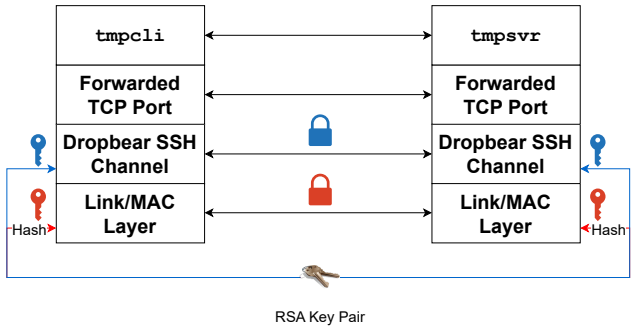

Figure 7: TP-Link Deco's Inter-access-point NAPS Relies on Insecure Dropbear SSH Channels.

we could continue to steal non-rotating backhaul SSID/passphrases, to collect/modify/inject wireless traffic, and even modify fronthaul SSID/passphrases and management passwords, defeating access control completely.

7.3 Case Study: TP-Link Deco Wi-Fi Systems

TP-Link Deco mesh networks utilize Dropbear SSH to protect inter-access-point NAPS. These SSH connections are established on top of pre-configured RSA keys, which are used to derive the backhaul SSIDs and passphrases (see Section 5). In other words, Dropbear SSH (at the application layer) uses the same trust anchor as layer-2 wireless backhaul links. However, the strength of the trust anchor is weak and brute-forceable, breaking the access control at both layers.

Specifically, as shown in Figure 7, to protect inter-access-point NAPS communication, in TP-Link Deco mesh networks, an extender node and the gateway node authenticate to each other with the pre-configured RSA private key (Section 5), through Dropbear SSH connections over backhaul links. On every access point, the Dropbear SSH server enables SSH port forwarding, so that other user-space network programs on that access point can multiplex onto this underlying Dropbear SSH channel, and reap security benefits (i.e., these user-space programs need not "reinvent the wheel" and build custom authentication/encryption for inter-process communication like NAPS).

An extender node's NAPS client (i.e., a network-facing program called "`tmpcli`") thus pulls network access policies from the NAPS server on the gateway node by relaying the client's clear text and unauthenticated TCP connection over that Dropbear SSH connection, utilizing SSH port forwarding. The gateway node can also proactively connect to extender nodes' NAPS servers (by connecting to extender nodes' Dropbear SSH servers first) and push network access policies. We point out that with this design, the attack surface is minimized because the NAPS server "`tmpsvr`" will not directly open public TCP ports to in-network attackers. Inter-access-point authentication, NAPS data confidentiality/integrity and freshness can be achieved through multiplexing the underlying Dropbear SSH connections between access points.

Nevertheless, we find the RSA public/private key pair has only 512 bits, which is far too weak, especially given the product was launched in 2020. Furthermore, Dropbear SSH servers on all access points (in the same Deco bundle) use that same public key (i.e., the trust anchor) for SSH host key, meaning that once the corresponding private key is obtained by an attacker, they will be able to maliciously authenticate to all access points. Such weak trust anchors allow targeted offline brute-force attacks: an in-network attacker can first directly masquerade as an access point to fetch the SSH host public key from a target access point's Dropbear SSH server (following the SSH authentication protocol), and then brute-force to guess the weak private key.

Even worse, a number of command injection vulnerabilities exist in TP-Link Deco's custom NAPS server implementation "`tmpsvr`". We suspect the inputs are not sanitized because the developers falsely assumed the underlying Dropbear SSH tunnel somehow protects "`tmpsvr`", making it not directly reachable.

PoC Attacks. We conducted an experiment in which an in-network attacker first connected to the Dropbear SSH server and requested the 512-bit Dropbear SSH host public key. Then, using this 512-bit SSH public key, we were able to brute-force the 512-bit private key within four days using commodity hardware and GGNFS/MSIEVE [47, 50], a software application designed to effectively factor big numbers and thus brute-force RSA private keys when knowing weak RSA public keys. Consequently, we successfully masqueraded as a legitimate access point and maliciously authenticated to the gateway node's Dropbear SSH server, and consistently pulled fresh network access policies, including fronthaul link passphrases. In other words, we showed that we could always obtain fresh fronthaul link passphrases before they reached legitimate extender nodes. Even if the attacker lost layer-2 wireless access because a new passphrase was applied, the attacker could use the new passphrase to reconnect to the mesh network. Furthermore, since the backhaul link credentials were derived from such an RSA public/private key pair (Section 5), we, as an attacker, could also calculate these static backhaul link credentials, and therefore join the network as an extender node, even if the attacker somehow lost the most recent fronthaul passphrase. Being an AP node allowed the attacker to then query the access policy from the gateway node, which includes fronthaul link SSID/passphrases, and continue to stay in the network as a wireless client.

In addition, we successfully launched command injection attacks against the NAPS server (after authenticating to Dropbear SSH servers using the brute-forced RSA private keys) by exploiting unsanitized parameters passed to `popen()`. These attacks directly led to shell access to gateway/extender nodes. This access allowed us to further collect/modify/inject wireless traffic directly at that target access point, and manipulate fronthaul link credentials used by a target access point, among other things.

8 DEFENSES AND DISCUSSION

8.1 Balancing Security with Extensibility: Future Designs

As one can now realize, correctly establishing cross-layer trust and enhancing cross-layer cooperation are two key points of improving security for mesh networks. However, by reflecting on various real-world designs, we find it inherently challenging to balance security with extensibility for mesh networks.

For example, the case studies of TP-Link Deco and ASUS demonstrate that using pre-configured trust anchors to establish both layer-2 wireless links and inter-access-point NAPS authentication can admittedly benefit security, but can possibly hurt mesh network extensibility: without auxiliary mechanisms (e.g., Bluetooth) that can securely pass existing trust anchors to new extender nodes, the mesh network will not be wirelessly extensible. On the other hand, if one favors extensibility by not using trust anchors (recall the case studies of EasyMesh, Netgear Orbi, Wyze, and AmpliFi), security will be compromised, since in this case wireless clients can never be effectively distinguished from access points across different network layers. One might naturally wonder: Is it possible for future solutions to improve security without hurting extensibility? And if the answer is positive, what do future solutions look like?

We answer this question optimistically. We first observe that, most of the vendors already have some kind of mechanism to setup the layer-2 communication securely (e.g., for establishing the backhaul passphrases). One can technically piggyback on such mechanisms to establish the NAPS-layer trust as well. We discuss two such existing negotiation mechanisms below as examples.

Trusted Users Can Help Dynamically Establish Cross-Layer Trust. During setup, a mesh network can possibly leverage trusted users, like network owners, to bootstrap inter-access-point trust across different network layers. For example, Push-button Configuration (PBC) is originally a usable security technique [36] to build an encrypted layer-2 wireless link between two wireless clients/APs. At its core, PBC can be seen as a general technique to allow two access points to negotiate shared secrets over-the-air under the trust of users. In a future solution, PBC can be re-purposed: by involving trusted users to press buttons at both access points, pair-wise trust anchors can be dynamically established for NAPS while the layer-2 trust is being established. It is worth mentioning that for best practices, it is better to have independent trusts established at the link layer and NAPS layer. This can avoid the risks of vulnerabilities due to cross-layer interactions (e.g., ASUS AiMesh).

Leveraging Out-of-Band Methods. During a mesh network setup, out-of-band channels (e.g., Bluetooth) may help establish cross-layer trust anchors between access points. For example, in a future design, the network owner may simply place trusted access points near each other during network setup, and extender nodes can be configured to receive an authoritative credential (i.e., the trust anchor) from the gateway node. After such an initial configuration and user confirmation, these out-of-band pairing interfaces can be switched off to minimize security risks [78]. These out-of-band pairing interfaces can be temporarily reactivated by users whenever new mesh devices need to be added (which happens infrequently).

Besides the approach to establish strong trust at the NAPS layer and prevent unauthorized client nodes from accessing certain interfaces, we also present an orthogonal, stricter approach:

Reducing Attack Surfaces with Network Isolation. We can set up network isolation at lower layers to prevent wireless clients from reaching the NAPS interfaces of the gateway and extender nodes, effectively reducing the attack surface. One potential isolation mechanism is VLAN. However, it is not widely available in home wireless mesh devices. Therefore, we propose creating separate backhaul SSIDs for the data plane and control plane traffic respectively. As illustrated in Figure 8, a single wireless access point can broadcast two backhaul SSIDs "Backhaul-User" and "Backhaul-NAPS" on the same frequency band using virtual SSIDs [12]. The SSID "Backhaul-User" forwards users' data traffic only, while "Backhaul-NAPS" will be used for inter-AP NAPS traffic only. There shall not be layer-2 frame forwarding or layer-3 routing between these two subnets. In this case, wireless attackers can not reach NAPS interfaces by IP addresses because the NAPS interfaces belong to a different, isolated IP subnet. The downside of such an isolation-based solution is that it may prevent wireless clients from accessing NAPS functionality completely. Nevertheless, from the access policies that are synchronized, we did not find any example where there is a need for this information to be distributed to a wireless client.

Summary. The first two methods (PBC and out-of-band methods) introduce additional complexity, costs (e.g., requirement of Bluetooth interfaces), and security risks. In contrast, network isolation reduces security risks by minimizing the attack surface but may break legitimate network management functionalities.

8.2 Post-Compromise Recovery

We observe that many of the vulnerabilities leak hard-coded secrets (e.g., backhaul passphrases or the original trust anchors like `amas_bdlkey` in Section 5), and are therefore difficult to recover from, even if vendors decide to issue patches. Realistically speaking, it is unlikely that existing vendors will radically change their respective choices of NAPS protocols. This means that they will have to execute the recovery using the same compromised NAPS channel.

One naive idea is to have the mesh vendor push an update to change the hardcoded secrets to different values for each bundle they sold independently. However, this means a different firmware image needs to be prepared for each bundle independently, which is a known challenging problem in the domain of automated software diversity [38].

Another idea is to leverage their original trust anchors to establish new trust anchors, which can be made unknown to attackers, and recover from the compromised trust anchors. We use Figure 9 to illustrate the idea.

Solution. Without losing generality, all selected mesh networks can use the pre-configured/trusted backhaul link passphrases to derive another credential that serves as the initial/bootstrapping trust anchor of NAPS, using a key derivation function (KDF) [52, 55]. Such a KDF is embedded in the updated firmware. We assume the KDF is not known to the attacker before the patch is applied. To achieve this, in practice, the firmware image should be encrypted and decryptable by only the mesh devices (e.g., via keys stored in the hardware to prevent reverse engineering). Moreover, KDF can be protected with practical techniques like code obfuscation [34] to significantly hinder reverse engineering efforts.

After that, the vendor can try to let legitimate access points negotiate new trust anchors from the initial/bootstrapping trust anchor. This can be done with, for example, password-authenticated key exchange (PAKE) protocols over their respective untrusted NAPS protocols. PAKE protocols (e.g., Dragonfly [75]) can allow two parties to securely establish a new shared secret with nonces so that it is not predictable. In the mesh network scenario, using

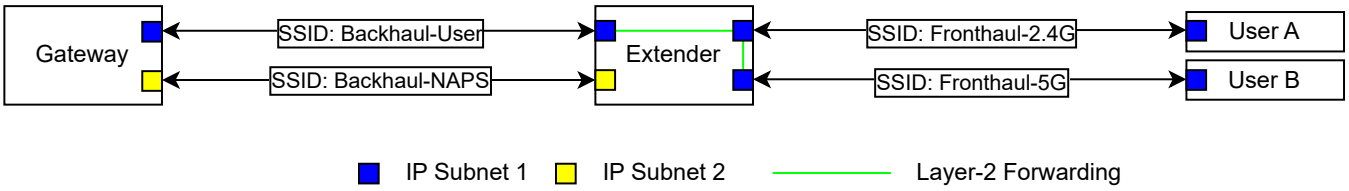

Figure 8: NAPS Protocols Can Use a Separate Backhaul SSID for Security.

Figure 9: Using Original Trust Anchors to Bootstrap New Trust Anchors for NAPS

PAKE for new trust anchor establishment has the advantage that a man-in-the-middle, without knowing the "bootstrapping trust anchor", is unable to guess/obtain the new trust anchor.

For example, SOAP APIs of Netgear Orbi can possibly be extended to enable such a new trust anchor's negotiation/agreement. The negotiated new trust anchor can then be used to derive session keys, via inter-access-point handshakes, that cleanly encrypt NAPS payloads that are sent through different existing protocols. This remedy/fix is general and can be scalably applied to all access points in the same bundle, without requiring extra establishment of trust anchors or device-to-device pairing.

Caveats and Benefits. If KDF or inter-access-point PAKE is directly applied on the original trust anchor to derive the new trust anchor, compromised access points will be able to "catch up" at any time and calculate the new trust anchor, which is undesirable. By requiring both KDF and PAKE during firmware updates, even compromised access points have to follow this key evolution in a timely way; otherwise, they will quickly lose the ability to perform NAPS. More importantly, even if an attacker eventually successfully reverse-engineers the KDF after a patch is applied, they will not be allowed to re-negotiate new trust anchors with updated access points.

8.3 Responsible Disclosure

We reported all vulnerabilities soon after their discovery. Table 3 records the timeline and patching status at the time of writing. We adhered to widely accepted responsible disclosure guidelines and gave vendors ample time (>90 days) to address the issues. We have not publicized any vulnerabilities. As of May 2024, we have received one CVE, and have submitted multiple CVE requests. The majority of vendors have been cooperative and have updated their firmware to patch the vulnerabilities.

9 RELATED WORK

Wireless Security. Since the introduction of 802.11 wireless standards, various types of attacks have been discovered that exploit vulnerabilities in 802.11 wireless protocols, including dictionary attacks that recover WPA passphrases [48, 76], DoS attacks [8, 26, 29, 35, 46, 51, 61, 76], side channel attacks [9, 14, 76], man-in-the-middle attacks [22, 71], key reinstallation attacks [73, 74], downgrade attacks [72], key recovery attacks [67], etc.

Table 3: Ethical Disclosure Timeline for Six Home Wireless Mesh Networks, and Wi-Fi Alliance

Vendor	Disclosure Time	Patched?
Netgear Orbi	03/2023	Yes
ASUS AiMesh	08/2023	Yes
TP-Link Deco	10/2023	Yes
Linksys	10/2023	Still fixing
Wyze	10/2023	Yes
AmpliFi	10/2023	Yes
Wi-Fi Alliance	10/2023	Still fixing

Another thread of research seeks to enhance and verify wireless protocol security with formal reasoning and proofs [16, 17, 30, 37, 64], automated testing [43], manual reviews [53, 54], improved designs [6, 69, 70], and real-world measurements [25, 60].

Our research, by studying most popular commercial products, improves the understanding of access control security in home wireless mesh networks, which are novel types of wireless networks.

Embedded Systems Security. During the past decade, researchers have built numerous tools to discover low-level vulnerabilities (e.g., buffer overflows, use-after-frees) as well as logic flaws in device firmware using static analysis, symbolic execution and fuzzing [23, 27, 58, 59, 62, 63, 83, 84]. Our approach, however, combines dynamic debugging with reverse engineering to take an initial step towards discovering NAPS protocol vulnerabilities in firmware binaries.

Protocol Security. There exist numerous previous studies on security/privacy of network protocols. They include (1) revealing low-level memory-safety vulnerabilities [19, 65], (2) discovering weaknesses in handling cryptographic keys [31, 32], and (3) unveiling time and space side-channels [5, 10, 11, 18, 41, 42, 45, 56, 57]. Our research, however, studies inter-access-point protocols for mesh networks and reveals their security vulnerabilities.

10 CONCLUSION

In this research, we analyzed access control mechanisms and protocols of six most popular home wireless mesh networks, and found both standardized and proprietary control plane protocols fundamentally broken, due to insufficient coordination across different protocol layers. This indicates the urgent need for (1) secure mechanisms/protocols that fit the mesh network scenario, (2) new mesh

network designs that balance security, interoperability, and network extensibility.

11 ACKNOWLEDGMENTS

This research was sponsored by the OUSD(R&E)/RT&L and was accomplished under Cooperative Agreement Number W911NF-20-2-0267. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ARL and OUSD(R&E)/RT&L or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes, notwithstanding any copyright notation herein.

REFERENCES

[1] 2009. IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)* (2009), 1–565. https://doi.org/10.1109/IEEESTD.2009.5307322

[2] 2023. Wireless Mesh Networking (WMN) Market 2023 is Booming Worldwide, Global Size-Share, Sales, Revenue, New Product Innovation,Competitor Ecosystem and Analysis till 2029. https://finance.yahoo.com/news/wireless-mesh-networking-wmn-market-092700635.html

[3] 2024. Wireless Mesh Network Market Size, Share, Trends and Industry Analysis. https://www.marketsandmarkets.com/Market-Reports/wireless-mesh-network-market-88410602.html

[4] ASUS AiMesh. 2024. What is AiMesh and AiMesh Router | ASUS. https://www.asus.com/microsite/aimesh/en/index.html

[5] Nadhem J Al Fardan and Kenneth G Paterson. 2013. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *2013 IEEE symposium on security and privacy*. IEEE, 526–540.

[6] Eman Alharbi, Noha Alsulami, Omar Batarfi, et al. 2015. An enhanced Dragonfly key exchange protocol against offline dictionary attack. *Journal of Information Security* 6, 02 (2015), 69.

[7] Wi-Fi Alliance. 2024. Wi-Fi EasyMesh Specification | Wi-Fi Alliance. https://www.wi-fi.org/file/wi-fi-easymesh-specification

[8] John Bellardo and Stefan Savage. 2003. 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. In *12th USENIX Security Symposium (USENIX Security 03)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/12th-usenix-security-symposium/80211-denial-service-attacks-real-vulnerabilities-and

[9] Giovanni Camurati, Sebastian Poeplau, Marius Muench, Tom Hayes, and Aurélien Francillon. 2018. Screaming channels: When electromagnetic side channels meet radio transceivers. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 163–177.

[10] Yue Cao, Zhiyun Qian, Zhongjie Wang, Tuan Dao, Srikanth V. Krishnamurthy, and Lisa M. Marvel. 2016. Off-Path TCP Exploits: Global Rate Limit Considered Dangerous. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 209–225. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cao

[11] Yue Cao, Zhongjie Wang, Zhiyun Qian, Chengyu Song, Srikanth V Krishnamurthy, and Paul Yu. 2019. Principled unearthing of TCP side channel vulnerabilities. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 211–224.

[12] Ranveer Chandra. 2006. *A virtualization architecture for wireless network cards*. Cornell University.

[13] Daming D Chen, Maverick Woo, David Brumley, and Manuel Egele. 2016. Towards automated dynamic analysis for linux-based embedded firmware.. In *NDSS*, Vol. 1. 1–1.

[14] Weiteng Chen and Zhiyun Qian. 2018. Off-Path TCP Exploit: How Wireless Routers Can Jeopardize Your Secrets. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 1581–1598. https://www.usenix.org/conference/usenixsecurity18/presentation/chen-weiteng

[15] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. 2016. On post-compromise security. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE, 164–178.

[16] Cas Cremers, Alexander Dax, Charlie Jacomme, and Mang Zhao. 2023. Automated Analysis of Protocols that use Authenticated Encryption: How Subtle AEAD Differences can impact Protocol Security. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 5935–5952. https://www.usenix.org/conference/usenixsecurity23/presentation/cremers-protocols

[17] Cas Cremers, Benjamin Kiesl, and Niklas Medinger. 2020. A Formal Analysis of IEEE 802.11's WPA2: Countering the Kracks Caused by Cracking the Counters. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 1–17. https://www.usenix.org/conference/usenixsecurity20/presentation/cremers

[18] Daniel de Almeida Braga, Pierre-Alain Fouque, and Mohamed Sabt. 2021. PARASITE: PAssword Recovery Attack against Srp Implementations in ThE wild. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2497–2512.

[19] Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, et al. 2014. The matter of heartbleed. In *Proceedings of the 2014 conference on internet measurement conference*. 475–488.

[20] John Engler, Chris Karlof, Elaine Shi, and Dawn Song. 2009. Is it too late for PAKE? *indicators* 5, 9 (2009), 17.

[21] John Engler, Chris Karlof, Elaine Shi, and Dawn Song. 2009. PAKE-based web authentication: The good the bad and the hurdles. In *Proc. IEEE Web 2.0 Security Privacy Workshop*. 1–9.

[22] Xuewei Feng, Qi Li, Kun Sun, Yuxiang Yang, and Ke Xu. 2023. Man-in-the-middle attacks without rogue AP: when WPAs meet ICMP redirects. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3162–3177.

[23] Xiaotao Feng, Ruoxi Sun, Xiaogang Zhu, Minhui Xue, Sheng Wen, Dongxi Liu, Surya Nepal, and Yang Xiang. 2021. Snipuzz: Black-box fuzzing of iot firmware via message snippet inference. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 337–350.

[24] Sadayuki Furuhashi. 2017. MessagePack: It's like JSON. but fast and small, 2014. (2017). http://msgpack.org

[25] Di Gao, Hao Lin, Zhenhua Li, Feng Qian, Qi Alfred Chen, Zhiyun Qian, Wei Liu, Liangyi Gong, and Yunhao Liu. 2021. A nationwide census on wifi security threats: prevalence, riskiness, and the economics. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 242–255.

[26] Stephen Glass and Vallipuram Muthukkumarasamy. 2007. A study of the TKIP cryptographic DoS attack. In *2007 15th IEEE International Conference on Networks*. IEEE, 59–65.

[27] Fabio Gritti, Fabio Pagani, Ilya Grishchenko, Lukas Dresel, Nilo Redini, Christopher Kruegel, and Giovanni Vigna. 2022. HEAPSTER: Analyzing the Security of Dynamic Allocators for Monolithic Firmware Images. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1082–1099.

[28] Dan Harkins and Warren Kumari. 2017. RFC 8110 - Opportunistic Wireless Encryption. https://www.rfc-editor.org/rfc/pdfrfc/rfc8110.txt.pdf

[29] Changhua He and John C Mitchell. 2004. Analysis of the 802.11 i 4-Way Handshake. In *Proceedings of the 3rd ACM Workshop on Wireless Security*. 43–50.

[30] Changhua He, Mukund Sundararajan, Anupam Datta, Ante Derek, and John C Mitchell. 2005. A modular correctness proof of IEEE 802.11 i and TLS. In *Proceedings of the 12th ACM conference on Computer and communications security*. 2–15.

[31] Sven Hebrok, Simon Nachtigall, Marcel Maehren, Nurullah Erinola, Robert Merget, Juraj Somorovsky, and Jörg Schwenk. 2023. We Really Need to Talk About Session Tickets: A Large-Scale Analysis of Cryptographic Dangers with TLS Session Tickets. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 4877–4894. https://www.usenix.org/conference/usenixsecurity23/presentation/hebrok

[32] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J Alex Halderman. 2012. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *21st USENIX Security Symposium (USENIX Security 12)*. 205–220.

[33] Guido R. Hiertz, Dee Denteneer, Sebastian Max, Rakesh Taori, Javier Cardona, Lars Berlemann, and Bernhard Walke. 2010. IEEE 802.11s: The WLAN Mesh Standard. *IEEE Wireless Communications* 17, 1 (2010), 104–111. https://doi.org/10.1109/MWC.2010.5416357

[34] Pascal Junod, Julien Rinaldini, Johan Wehrli, and Julie Michielin. 2015. Obfuscator-LLVM–software protection for the masses. In *2015 ieee/acm 1st international workshop on software protection*. IEEE, 3–9.

[35] Bastian Könings, Florian Schaub, Frank Kargl, and Stefan Dietzel. 2009. Channel switch and quiet attack: New DoS attacks exploiting the 802.11 standard. In *2009 IEEE 34th Conference on Local Computer Networks*. IEEE, 14–21.

[36] Cynthia Kuo, Jesse Walker, and Adrian Perrig. 2007. Low-cost manufacturing, usability, and security: An analysis of bluetooth simple pairing and wi-fi protected setup. In *Financial Cryptography and Data Security: 11th International Conference, FC 2007, and 1st International Workshop on Usable Security, USEC 2007, Scarborough, Trinidad and Tobago, February 12-16, 2007. Revised Selected Papers 11*. Springer, 325–340.

[37] Jean Lancrenon and Marjan Škrobot. 2015. On the provable security of the Dragonfly protocol. In *Information Security: 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings 18*. Springer, 244–261.

[38] Per Larsen, Andrei Homescu, Stefan Brunthaler, and Michael Franz. 2014. SoK: Automated Software Diversity. In *2014 IEEE Symposium on Security and Privacy*. 276–291. https://doi.org/10.1109/SP.2014.25

[39] Yong Li, Sven Schäge, Zheng Yang, Florian Kohlar, and Jörg Schwenk. 2014. On the security of the pre-shared key ciphersuites of TLS. In *Public-Key Cryptography–PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings 17*. Springer, 669–684.

[40] Andrés Marín López, Florina Almenáres Mendoza, Patricia Arias Cabarcos, and Daniel Díaz Sánchez. 2016. Wi-Fi Direct: Lessons learned. In *2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. IEEE, 1–8.

[41] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. 2020. Dns cache poisoning attack reloaded: Revolutions with side channels. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1337–1350.

[42] Keyu Man, Xin'an Zhou, and Zhiyun Qian. 2021. DNS cache poisoning attack: Resurrections with side channels. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 3400–3414.

[43] Chris McMahon Stone, Tom Chothia, and Joeri De Ruiter. 2018. Extending automated protocol state learning for the 802.11 4-way handshake. In *Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I 23*. Springer, 325–345.

[44] Anil Mengi, Abdesselem Kortebi, Helmut Lucht, Marcin Brzozowski, Michael Koch, Olivier Bouchet, and Oliver Maye. 2016. IEEE 1905.1 hybrid home networking standard and its implementation with PLC, Wi-Fi and Ethernet technologies. In *2016 International Symposium on Power Line Communications and its Applications (ISPLC)*. IEEE, 162–166.

[45] Robert Merget, Marcus Brinkmann, Nimrod Aviram, Juraj Somorovsky, Johannes Mittmann, and Jörg Schwenk. 2021. Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E). In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 213–230. https://www.usenix.org/conference/usenixsecurity21/presentation/merget

[46] CHJC Mitchell and Changhua He. 2005. Security Analysis and Improvements for IEEE 802.11 i. In *The 12th Annual Network and Distributed System Security Symposium (NDSS'05) Stanford University, Stanford*. 90–110.

[47] C Monico. 2005. GGNFS-A number field sieve implementation. (2005). http://www.math.ttu.edu/~cmonico/software/ggnfs/

[48] Robert Moskowitz. 2003. Weakness in passphrase choice in WPA interface. (2003). http://wifinetnews.com/archives/2003/11/weakness_in_passphrase_choice_in_wpa_interface.html

[49] OpenWrt. 2024. [OpenWrt Wiki] Unlocking the Netgear Telnet Console. https://openwrt.org/toh/netgear/telnet.console

[50] Jason Papadopoulos. 2016. Msieve. (2016). http://msieve.sourceforge.net

[51] Konstantinos Pelechrinis, Marios Iliofotou, and Srikanth V. Krishnamurthy. 2011. Denial of Service Attacks in Wireless Networks: The Case of Jammers. *IEEE Communications Surveys & Tutorials* 13, 2 (2011), 245–257. https://doi.org/10.1109/SURV.2011.041110.00022

[52] Colin Percival. 2009. Stronger key derivation via sequential memory-hard functions.

[53] Trevor Perrin. 2013. [TLS] Question regarding CFRG process. https://mailarchive.ietf.org/arch/msg/tls/M9Wrwd0iDEAk-PztgmrqIPEXvao/

[54] Trevor Perrin. 2013. [TLS] Review of Dragonfly PAKE. https://mailarchive.ietf.org/arch/msg/tls/A_SfHI4BsdAi4miklBs3TvUbu-Y/

[55] Niels Provos and David Mazieres. 1999. A future-adaptable password scheme.. In *USENIX Annual Technical Conference, FREENIX Track*, Vol. 1999. 81–91.

[56] Zhiyun Qian and Z Morley Mao. 2012. Off-path TCP sequence number inference attack-how firewall middleboxes reduce security. In *2012 IEEE Symposium on Security and Privacy*. IEEE, 347–361.

[57] Zhiyun Qian, Z Morley Mao, and Yinglian Xie. 2012. Collaborative TCP sequence number inference attack: how to crack sequence number under a second. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 593–604.

[58] Nilo Redini, Aravind Machiry, Ruoyu Wang, Chad Spensky, Andrea Continella, Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. 2020. Karonte: Detecting insecure multi-binary interactions in embedded firmware. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1544–1561.

[59] Tobias Scharnowski, Nils Bars, Moritz Schloegel, Eric Gustafson, Marius Muench, Giovanni Vigna, Christopher Kruegel, Thorsten Holz, and Ali Abbasi. 2022. Fuzzware: Using Precise MMIO Modeling for Effective Firmware Fuzzing. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 1239–1256. https://www.usenix.org/conference/usenixsecurity22/presentation/scharnowski

[60] Domien Schepers, Aanjhan Ranganathan, and Mathy Vanhoef. 2021. Let numbers tell the tale: measuring security trends in wi-fi networks and best practices. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 100–105.

[61] Matthias Schulz, Francesco Gringoli, Daniel Steinmetzer, Michael Koch, and Matthias Hollick. 2017. Massive reactive smartphone-based jamming using arbitrary waveforms and adaptive power control. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 111–121.

[62] Yan Shoshitaishvili, Ruoyu Wang, Christophe Hauser, Christopher Kruegel, and Giovanni Vigna. 2015. Firmalice-automatic detection of authentication bypass vulnerabilities in binary firmware.. In *NDSS*, Vol. 1. 1–1.

[63] Yan Shoshitaishvili, Ruoyu Wang, Christopher Salls, Nick Stephens, Mario Polino, Andrew Dutcher, John Grosen, Siji Feng, Christophe Hauser, Christopher Kruegel, et al. 2016. Sok:(state of) the art of war: Offensive techniques in binary analysis. In *2016 IEEE symposium on security and privacy (SP)*. IEEE, 138–157.

[64] Rajiv Ranjan Singh, José Moreira, Tom Chothia, and Mark D Ryan. 2020. Modelling of 802.11 4-way handshake attacks and analysis of security properties. In *Security and Trust Management: 16th International Workshop, STM 2020, Guildford, UK, September 17–18, 2020, Proceedings 16*. Springer, 3–21.

[65] George Arnold Sullivan, Jackson Sippe, Nadia Heninger, and Eric Wustrow. 2022. Open to a fault: On the passive compromise of TLS keys via transient errors. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 233–250. https://www.usenix.org/conference/usenixsecurity22/presentation/sullivan

[66] David Taylor, Thomas Wu, Nikos Mavrogiannopoulos, and Trevor Perrin. 2007. *Using the Secure Remote Password (SRP) protocol for TLS authentication.* Technical Report.

[67] Erik Tews and Martin Beck. 2009. Practical attacks against WEP and WPA. In *Proceedings of the second ACM conference on Wireless network security*. 79–86.

[68] TP-Link. 2024. Award-Winning Best Mesh WiFi Systems By TP-Link. https://www.tp-link.com/us/deco-mesh-wifi/

[69] Mathy Vanhoef, Prasant Adhikari, and Christina Pöpper. 2020. Protecting wi-fi beacons from outsider forgeries. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 155–160.

[70] Mathy Vanhoef, Nehru Bhandaru, Thomas Derham, Ido Ouzieli, and Frank Piessens. 2018. Operating channel validation: Preventing multi-channel man-in-the-middle attacks against protected Wi-Fi networks. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. 34–39.

[71] Mathy Vanhoef and Frank Piessens. 2014. Advanced Wi-Fi attacks using commodity hardware. In *Proceedings of the 30th Annual Computer Security Applications Conference*. 256–265.

[72] Mathy Vanhoef and Frank Piessens. 2016. Predicting, Decrypting, and Abusing WPA2/802.11 Group Keys. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 673–688. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/vanhoef

[73] Mathy Vanhoef and Frank Piessens. 2017. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1313–1328.

[74] Mathy Vanhoef and Frank Piessens. 2018. Release the Kraken: new KRACKs in the 802.11 Standard. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 299–314.

[75] Mathy Vanhoef and Eyal Ronen. 2020. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In *IEEE Symposium on Security & Privacy (SP)*. IEEE.

[76] Mathy Vanhoef and Eyal Ronen. 2020. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In *2020 IEEE Symposium on Security and Privacy (SP)*. 517–533. https://doi.org/10.1109/SP40000.2020.00031

[77] Stefan Viehböck. 2011. Brute forcing wi-fi protected setup. *Wi-Fi Protected Setup* 9 (2011).

[78] Jianliang Wu, Ruoyu Wu, Dongyan Xu, Dave Tian, and Antonio Bianchi. 2023. SoK: The Long Journey of Exploiting and Defending the Legacy of King Harald Bluetooth. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 23–23.

[79] Thomas D Wu et al. 1998. The Secure Remote Password Protocol.. In *NDSS*, Vol. 98. Citeseer, 97–111.

[80] Tatu Ylonen. 1996. SSH–secure login connections over the Internet. In *Proceedings of the 6th USENIX Security Symposium*, Vol. 37. 40–52.

[81] Bin Yuan, Zhanxiang Song, Yan Jia, Zhenyu Lu, Deqing Zou, Hai Jin, and Luyi Xing. 2023. MQTTactic: Security Analysis and Verification for Logic Flaws in MQTT Implementations. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 13–13.

[82] Yu Zhang, Wei Huo, Kunpeng Jian, Ji Shi, Haoliang Lu, Longquan Liu, Chen Wang, Dandan Sun, Chao Zhang, and Baoxu Liu. 2019. SRFuzzer: an automatic fuzzing framework for physical SOHO router devices to discover multi-type vulnerabilities. In *Proceedings of the 35th annual computer security applications conference*. 544–556.

[83] Yaowen Zheng, Ali Davanian, Heng Yin, Chengyu Song, Hongsong Zhu, and Limin Sun. 2019. FIRM-AFL: High-Throughput Greybox Fuzzing of IoT Firmware via Augmented Process Emulation. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1099–1114. https://www.usenix.org/conference/usenixsecurity19/presentation/zheng

[84] Yaowen Zheng, Yuekang Li, Cen Zhang, Hongsong Zhu, Yang Liu, and Limin Sun. 2022. Efficient greybox fuzzing of applications in Linux-based IoT devices via enhanced user-mode emulation. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 417–428.