AirSnitch: Demystifying and Breaking Client Isolation in Wi-Fi Networks

Xin'an Zhou*, Juefei Pu*, Zhutian Liu*, Zhiyun Qian*, Zhaowei Tan*, Srikanth V. Krishnamurthy*, Mathy Vanhoef†
*University of California, Riverside †DistriNet, KU Leuven
{xzhou114,jpu007,zliu272}@ucr.edu, {zhiyunq,ztan,krish}@cs.ucr.edu
mathy.vanhoef@kuleuven.be

***Abstract*—To prevent malicious Wi-Fi clients from attacking other clients on the same network, vendors have introduced client isolation, a combination of mechanisms that block direct communication between clients. However, client isolation is not a standardized feature, making its security guarantees unclear.**

In this paper, we undertake a structured security analysis of Wi-Fi client isolation and uncover new classes of attacks that bypass this protection. We identify several root causes behind these weaknesses. First, Wi-Fi keys that protect broadcast frames are improperly managed and can be abused to bypass client isolation. Second, isolation is often only enforced at the MAC or IP layer, but not both. Third, weak synchronization of a client's identity across the network stack allows one to bypass Wi-Fi client isolation at the network layer instead, enabling the interception of uplink and downlink traffic of other clients as well as internal backend devices. Every tested router and network was vulnerable to at least one attack. More broadly, the lack of standardization leads to inconsistent, ad hoc, and often incomplete implementations of isolation across vendors.

Building on these insights, we design and evaluate end-to-end attacks that enable full machine-in-the-middle capabilities in modern Wi-Fi networks. Although client isolation effectively mitigates legacy attacks like ARP spoofing, which has long been considered the only universal method for achieving machine-in-the-middle positioning in local area networks, our attack introduces a general and practical alternative that restores this capability, even in the presence of client isolation.

I. Introduction

Client isolation [1], [2], [3], [4], [5] was introduced by vendors to mitigate insider attacks in Wi-Fi networks, such as ARP poisoning [6] and ICMP redirects [7] [8]. By disallowing clients to communicate with each other, it limits the attack surface and the risk of compromise. Although client isolation, also known as Access Point (AP) isolation, seemingly offers strong security benefits, it is not a properly standardized feature of the IEEE 802.11 standards [9]. Thus, the exact policy and enforcement mechanisms in real-world Wi-Fi networks are largely unexplored. In light of this, we ask: *"Does client isolation protect clients from attacking each other on Wi-Fi networks as intended, across different implementations?"*

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.241282
www.ndss-symposium.org

To answer the above question, we conduct a structured security analysis of client isolation across three relevant network layers: Wi-Fi encryption, packet switching, and packet routing. Moreover, we do so for small networks where a single physical AP broadcasts only a single visible network name, *and* for more complex Wi-Fi networks, where the victim and the attacker could be connected to different APs, and possibly even be connected to differently-named networks. Surprisingly, every router and network that we tested was affected by at least one of our attacks.

Our attacks affect both home and enterprise networks, enabling traffic injection towards victim clients, interception of traffic sent by the victim to the Internet, and a combination of both to perform Machine-in-the-Middle (MitM) attacks. Once a MitM is established, the attacker can intercept all link-layer traffic, facilitating higher-layer attacks. For instance, recent vulnerabilities in unpatched (D)TLS implementations can then be abused to decrypt HTTPS connections and compromise sensitive user information [10], [11], [12].

The attacks that we develop to bypass client isolation are guided by several key observations. *First,* at the Wi-Fi layer, we find that most Wi-Fi implementations use a shared Group Temporal Key (GTK) to protect broadcast or multicast communications. Often all clients have access to this key, even with client isolation enabled, meaning that this key can be abused by an insider to directly inject packets to victims, bypassing client isolation at the AP. Even the Passpoint standard [13], which has provisions similar to client isolation to mitigate insider attacks in protected hotspots, is flawed by design in this regard, as it also fails to securely manage group keys. Importantly, we point out that Passpoint only addresses the Wi-Fi encryption layer and does not extend protections to switching or routing, which are internal to APs (see below). The precise attack details depend on vendor implementations and the complexity of the targeted network, and collectively we refer to these as the *Abusing GTK* attack.

Second, we observe that many vendors only enforce client isolation at Layer-2 (MAC and link), but do not carry it over to Layer-3 (IP layer). Thus, we find that an attacker can *inject* packets to a victim, by using the AP's gateway MAC address as the layer 2 destination, but the victim's IP address as the layer 3 destination. These packets are typically accepted by the AP and forwarded to the gateway. If the gateway does not enforce client isolation at the IP layer, it will forward the datagram to its destination i.e., the victim client on the Wi-Fi network, allowing the attacker to reach the "layer-2 isolated" victim clients. We call this the *gateway bouncing* attack.

Third, we find that spoofing the victim's MAC address while connecting to the same network (but possibly a different AP) as the victim, enables the attacker to *intercept* downlink frames meant for the victim. Although it is known that Wi-Fi clients can use any MAC address to connect, it is surprising that this alone can be used to intercept another client's traffic, even with client isolation. We study under exactly which conditions our attack is possible, i.e., we investigate the impact of Wi-Fi features such as encryption, management frame protection, one physical AP broadcasting different network names, etc. When combined with the injection attacks to return intercepted traffic back to the victim, this allows the attacker to be a MitM on the downlink path.

Lastly, to obtain a full bi-directional MitM, we introduce techniques to intercept uplink traffic sent by clients as well. To achieve this, we found that it is possible to impersonate internal backend devices (e.g., the gateway) in the Wi-Fi network by spoofing their MAC addresses while connecting as a legitimate Wi-Fi client. By using this approach, an attacker can intercept uplink traffic sent by all other Wi-Fi clients. Surprisingly, even though this results in client-to-client traffic, it is often allowed by the network. Combined with our other techniques, this results in a full bi-directional MitM.

One root cause of most of our attacks is that existing Wi-Fi encryption protocols cannot securely synchronize the client's identity across network layers, devices, and network names, i.e., SSIDs. For instance, an adversary can establish a valid connection with an AP by spoofing a victim's MAC address on one SSID, while the victim is connected to a different AP or SSID. A second cause is that a device's Wi-Fi keys, MAC address, and IP address, are not strongly tied to each other. To overcome these limitations and improve Wi-Fi client isolation, we propose mitigation techniques, such as setting up multiple isolation domains, and discuss their feasibility.

We have tested 5 recent home routers (APs) from popular vendors as well as two widely-used open-source router distributions, and find that they were susceptible to at least some form of our attacks. In addition, we configure a local testbed to verify that a complete end-to-end MitM attack is viable in a realistically crafted enterprise setting. Importantly, we perform experiments in the wild, in two university networks, wherein we were able to fully realize a downlink attack from our attacker device on our own victim device (with no ethical violations on other users).

In summary, we make the following contributions.

- We introduce novel MitM primitives that break client isolation, which was commonly believed to protect Wi-Fi clients from one another, enabling MitM attacks relating to both uplink and downlink traffic.
- We show that Wi-Fi client isolation in single-BSSID networks (e.g., home networks) is fundamentally broken for all WPA versions.

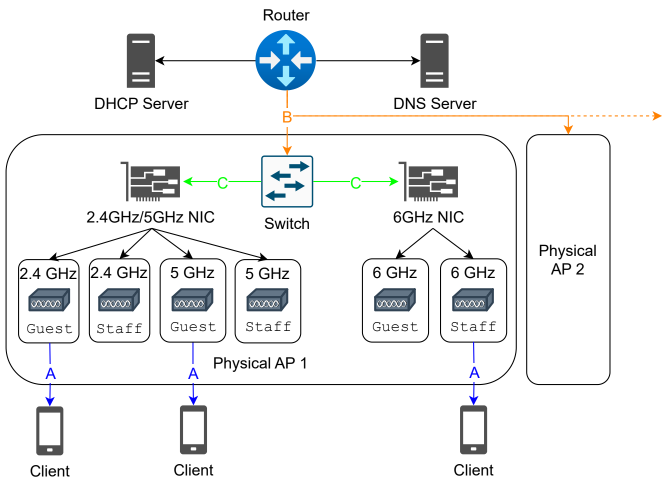

Fig. 1. Example Wi-Fi architecture where two SSIDs `Guest` and `Staff` are broadcast by two different wireless NICs within the same physical AP. We investigate client isolation at: (A) the Wi-Fi encryption layer; (B) the network layer; and (C) the link layer.

- We extend our bypasses to multi-AP and multi-SSID networks, common in enterprises, analyze their root cause, and sometimes even leak traffic in plaintext.
- We show that our attacks can also intercept traffic of internal wired devices, allowing higher-layer attacks previously thought impractical, which we illustrate by breaking weak back-end RADIUS passwords.
- We evaluate our attacks on various devices and networks including two real university networks, discuss defenses, and open-source our code for reproducibility of our tests.[1]

II. Wi-Fi Primer

We describe the architecture of modern Wi-Fi networks and the protection/encryption protocols used therein.

A. Wi-Fi Architecture

Figure 1 depicts the generalized architecture of Wi-Fi networks. At the core is a router or gateway. It connects physical Access Points (APs), which support Wi-Fi communication, to upstream networks (e.g., the Internet), and to services like DNS and DHCP servers. Functionally, the AP serves as a bridge, enabling translations between Ethernet frames and Wi-Fi frames to extend the wired network to the wireless domain. In enterprise networks, routers and physical APs are often implemented as separate physical devices to ensure scale and performance. Home environments typically use integrated Wi-Fi routers that combine these components. Each physical AP may host multiple wireless Network Interface Controllers (NICs), which are interconnected via internal switch(es) to bridge client traffic to the router and beyond.

A Wi-Fi network is identified by a user-friendly string called the Service Set Identifier (SSID), e.g., "eduroam", which allows clients to discover and join the network [14]. A single

[1]Our code is available at https://github.com/zhouxinan/airsnitch and is also mirrored at https://github.com/vanhoefm/airsnitch

physical AP can support SSIDs on different frequency channels, possibly using multiple NICs. Each SSID is broadcast by one or more Basic Service Sets (BSS), where a BSS consists of a central AP along with connected clients that all operate on the same frequency band and with the same parameters. A BSS is identified by a unique BSS Identifier (BSSID), which is typically the MAC address of one of the AP's NICs.

In addition, multiple Network Interface Controllers (NICs) with different BSSIDs can represent networks with the same SSID, enabling users to access a unified network referred to as an Extended Service Set (ESS). ESSID refers to the SSID used in an Extended Service Set (ESS). An ESS supports seamless roaming, allowing a Wi-Fi client to maintain a consistent logical identity while transitioning between APs within the ESS. ESS deployments are common in environments requiring mobility across large areas, such as hospitals or universities. To ensure secure roaming, all APs in an ESS broadcasting the same SSID are typically configured with a uniform authentication and encryption scheme (e.g., WPA2-Enterprise) with preset credentials. We also use the term "single-BSSID network" to refer to configurations where only one BSSID (typically from a single NIC) advertises the presence of a Wi-Fi network, as opposed to the ESSID setup.

As shown in Figure 1, virtualization to create virtual SSIDs [15] can further enable a single wireless NIC to broadcast multiple SSIDs/BSSIDs simultaneously, using adjacent MAC addresses. These BSSIDs can be configured to meet security or usability requirements for secure, isolated guest Wi-Fi networks. In Figure 1, two physical wireless NICs are configured to serve two separate ESSes: `Guest` and `Staff`, on 2.4 GHz, 5 GHz, and 6 GHz channels.

B. Wi-Fi Security

Wi-Fi Authentication and Key Management. Wi-Fi networks are commonly secured by mechanisms that conform to the WPA2 or WPA3 standards. For both, authentication and key management are handled via handshake procedures. To initiate the 4-way handshake, both the AP and the client generate a 256-bit Pairwise Master Key (PMK). WPA2 has two variants: WPA2-PSK (Pre-shared Key) and WPA2-Enterprise, and both generate PMK from a pre-shared credential. With respect to WPA2-PSK, the passphrase is identical for all clients, while WPA2-Enterprise uses credentials unique to each client (e.g., identified by username), and a distinct PMK is securely generated and assigned [16].

WPA3 is a marked improvement over WPA2, notably by introducing the Dragonfly handshake during the 802.11 authentication phase to convert the preshared Wi-Fi passphrase to a high-entropy PMK [17]. WPA3 is also available in two versions: WPA3-SAE (Simultaneous Authentication of Equals), which uses the same passphrase across clients, and WPA3-Enterprise, where each client must have its own credentials.

Once the PMK, as the top of the key hierarchy, is established, the subsequent 4-way handshake derives a Pairwise Transient Key (PTK) from PMK, MAC addresses, and nonces. The PTK is a session key that is later used to encrypt unicast communications. To protect broadcast and multicast data frames, the AP randomly generates a Group Master Key (GMK), and derives a Group Temporal Key (GTK) from the GMK. Additionally, the AP randomly generates an Integrity Group Temporal Key (IGTK) [9]. The GTK is used to encrypt and authenticate broadcast/multicast data traffic, while the IGTK is used to authenticate (but not encrypt) broadcast management frames. Both keys are securely transmitted, i.e., encrypted with PTK, from the AP to the client during the 4-way handshake [18], [19], [20].

After successfully completing the 4-way handshake, the AP converts incoming Ethernet frames from the Distribution System (DS), i.e., the wired network, into Wi-Fi frames, encrypting them with the PTK for unicast or the GTK for multicast/broadcast. This design choice is also motivated by the hidden terminal problem [21]: the AP is centrally positioned and can reliably reach all associated clients, making it the sole authorized sender of encrypted group-addressed traffic.

Passpoint. Passpoint was introduced to secure public Wi-Fi hotspots and includes provisions for client isolation at the Wi-Fi (but not network and link) layer using group key randomization [13]. Specifically, it recommends disabling Downstream Group-Addressed Forwarding (DGAF) and setting unique GTKs per client during the 4-way handshake to prevent broadcast abuse.

Wi-Fi Frames. A layer-2 Wi-Fi frame typically contains three MAC addresses. Address 1 and 2 always represent the receiver and the transmitter, respectively. The interpretation of Address 3 depends on the flags in the Frame Control (FC) field. If the ToDS flag is set and the FromDS flag is unset, Address 3 represents the logical destination of the frame. Conversely, if the ToDS flag is unset and the FromDS flag is set, Address 3 represents the logical source. The Frame Control (FC) field also includes the Power Management (PM) flag, which clients use to indicate sleep transitions. Setting the PM flag in a data frame signals the AP to buffer incoming traffic; clearing it notifies the AP that the client is awake. Though intended for energy efficiency, this mechanism introduces subtle security risks. As shown by [20], sleep transitions can be spoofed using management frames like WNM-Sleep Requests/Responses. A WNM-Sleep Response can carry a GTK.

III. Overview

In this section, we first introduce the concept of client isolation in Wi-Fi, followed by the threat model towards our attack for circumventing it. We then present an overview of our attack analysis methodology and summarize our results.

A. Client Isolation

While modern WPA2/WPA3 implementations secure Wi-Fi from external threats by mitigating attacks like KRACK [19], [20], Wi-Fi client isolation—usually the network's last line of defense against insider adversaries—remains a vendor-specific and sparsely documented mechanism, leaving its actual robustness and limitations largely unexplored in prior work.

We assume that the targeted Wi-Fi network enforces client isolation, a feature many vendors added to prevent clients from attacking each other on the same network [1], [2], [3], [4], [5], [22], [23]. Client isolation aims to ensure that Wi-Fi clients cannot intercept, transmit, or inject traffic from/to other clients within the same wireless local area network. This protection is crucial for mitigating client-to-client attacks in both personal and enterprise environments. This defense is also known as AP isolation, but we use *client isolation* throughout this paper. To specifically refer to client isolation between clients connected to the same network and to the same AP, i.e., client isolation within a BSSID, we use the term *Intra-BSSID isolation*.

Security Expectations. Client isolation must remain effective across all scenarios within a local area network, including clients connected to the same BSSID (e.g. in order to prevent compromised IoT devices from attacking other devices), different BSSIDs under the same (E)SSID, clients connected to different APs, or clients in other (E)SSIDs that are part of the same distribution system, i.e., wired network. To prevent all possible attacks, isolation should also be enforced between the Wi-Fi clients and the wired distribution system, ensuring that a Wi-Fi client cannot forge data frames to impersonate another device within the distribution system.

Real-world Client Isolation Mechanisms. By means of a careful manual analysis (of source code, documentation, and with simple experiments), we find that current implementations of client isolation typically rely on a combination of the following mechanisms: (1) Wi-Fi encryption protocols (e.g., WEP, TKIP, CCMP and GCMP), which aim to prevent clients from decrypting over-the-air frame payloads of other clients; (2) Intra-BSSID isolation, where the AP blocks direct communication between clients on the same BSSID by dropping such frames. (3) Inter-BSSID isolation, which blocks traffic between clients connected to different BSSIDs (but within a distribution system). (4) the use of guest network configurations, which isolate untrusted clients by assigning them to separate and restricted SSIDs.

B. Threat Model

Attacker's Goal. We consider an attacker that aims to bypass client isolation. It seeks to intercept a victim user's traffic and inject traffic towards other users and/or APs. This allows the attacker to thereby break the security expectations afforded by client isolation and potentially become a MitM. The attacker can then optionally use this ability to launch higher-layer attacks such as exploiting unpatched TLS vulnerabilities to intercept HTTPS traffic [10], [11], [12].

Attack Assumptions. In line with previous works [24], [25], [7], [26], [27], [28], [29], we consider an *in-network* attacker (a malicious insider) who has access to the Wi-Fi infrastructure by connecting to an open SSID or by using its own credentials to connect to an encrypted SSID, e.g., WPA2/3 Personal or Enterprise. The attacker is also capable of simultaneously transmitting and receiving Wi-Fi signals to and from both the victim and their associated AP. We assume that the attacker can achieve this on multiple channels simultaneously, by leveraging multiple wireless NICs that work on different frequency channels. In addition, we assume that the attacker can control a malicious server on the Internet, enabling it to coordinate with local in-network attackers and accept exfiltrated packets, among others.

C. Methodology

Our approach involves analyzing packet forwarding behaviors within typical wireless network setups in a structured way. We perform black-box and gray-box analysis of wireless routers, including reverse-engineering network configurations after rooting/jailbreaking commercial off-the-shelf (COTS) APs. This allows us to directly inspect how isolation mechanisms are enforced (or bypassed) across different layers of the network stack.

To structure our analysis, we focus on three critical boundaries where client isolation enforcement is expected: (A) the Wi-Fi encryption layer, where we examine the implications of shared key materials such as Group Temporal Keys (GTKs) and pre-shared passphrases; (B) the IP forwarding boundary, where we analyze how routing rules and gateway behavior may introduce unintended connectivity; and (C) the internal switching layer, focusing on how the AP's virtual interfaces and bridge configurations handle MAC-to-port mappings and frame forwarding.

At each boundary, we develop and test injection and interception techniques to evaluate whether traffic from one client can reach another, despite isolation policies being enabled. To benefit the research and open-source communities, we release our full measurement suite. This layered assessment allows us to pinpoint where isolation breakdowns occur and under what conditions, enabling the attacks described in later sections.

D. Summary of Results

Our evaluations show that Wi-Fi client isolation is flawed across encryption, routing, and switching layers. First, attackers can abuse shared GTKs to inject group-addressed frames, exploiting the symmetric WPA2/3 keys. Second, we introduce *gateway bouncing*, where traffic is routed between clients via the default gateway, bypassing client isolation via the IP layer. Third, we show that, surprisingly, APs can be tricked into rebinding session keys through MAC address spoofing, enabling attackers to intercept encrypted traffic. These attacks succeed across diverse AP models, proving that client isolation is neither cryptographically sound nor reliably enforced.

IV. Bypassing Client Isolation via Shared Keys

While Wi-Fi encryption is designed to prevent unauthorized access and protect client traffic, it also plays a critical role in enforcing client isolation—blocking direct communication between wireless devices on the same network. In this section, we demonstrate that this isolation is compromised when networks depend on shared cryptographic material, such as group keys or a common pre-shared passphrase, rather than employing distinct keys for individual clients.

We analyze how client isolation, even when advertised by vendors as a secure feature in WPA2/WPA3 networks, can be bypassed due to shared keys. We demonstrate two practical attack vectors: (1) traffic injection and interception using the network's shared passphrase; and (2) frame injection through abuse of the GTK. These attacks highlight a core tension in Wi-Fi security: while encryption thwarts outsiders, the use of shared keys leaves room for insiders to subvert in-network protections like client isolation.

A. Injection and MitM against Home WPA2/3

We found that many vendors advertise client isolation as a security feature, even for home WPA2/3 networks that use a shared password. For instance, the Omada documentation states that client isolation is *"used to protect the device against attacks from other devices in the same network"* [1]. Online security guides for home routers provide similar advice [2], [3], [4]. However, client isolation with a shared password is fundamentally flawed: an attacker can not only straightforwardly inject packets but also read a victim's packets, or even become a MitM:

Machine-on-the-Side Bypass. With respect to WPA2-PSK, it is well-known that a Machine-on-the-Side attacker who possesses the WPA2 passphrase (e.g., shared password in home Wi-Fi networks) can intercept handshake messages, calculate the victim's session key, to subsequently decrypt and inject WPA2 traffic over the air [30]. This trivially bypasses client isolation. If the victim is already connected to the network, they can be disconnected by spoofing de-authentication frames, resulting in a new 4-way handshake from which the adversary can derive the victim's session keys.

Rogue AP Bypass. With WPA3-SAE (used in WPA3-Personal), calculating the session key by monitoring a victim's traffic is not possible. However, an attacker can still clone the AP and communicate with victims once they connect, since WPA3 still uses a shared passphrase and client isolation is only enforced on the real AP. Additionally, even though WPA3 requires management frame protection, an attacker can still induce clients into connecting to the rogue AP, e.g., by spoofing (malformed) beacon frames with a channel switch announcement [31], [14].

WPA2/3-Enterprise. These attacks generally do not work against WPA2/3-Enterprise networks because common EAP methods use unique per-client credentials and may also use public-key cryptography to verify the identity of the network [16]. Note that most home routers support WPA2/3-Enterprise by configuring a (remote) RADIUS server. Alternatively, WPA3 Public Key can be used, where the shared passphrase is derived from a public key, meaning an adversary cannot create a rogue clone without knowing the corresponding private key [32], [27].

Summary. While some of the above subversions seem simple and trivial to accomplish, bypassing client isolation in enterprises is more challenging, especially for complex networks. However, as shown in the subsequent sections, the novel approaches we discover are still effective options for an attacker towards this objective.

B. Frame Injection Abusing GTK and IGTK

In the following subsections, we describe how the GTK and IGTK can be abused to perform frame injection in different settings.

1) Abusing GTK: We discover a new technique to bypass client isolation by exploiting the shared GTK (Group Temporal Key). Normally, a Wi-Fi client encrypts broadcast/multicast frames with its PTK, sending them to the AP for the latter to re-encrypt with the GTK (shared with all clients connected to the same BSSID). While the GTK is meant for the AP to control broadcast access, attackers can abuse it to wrap unicast IP traffic in a broadcast frame encrypted with the GTK, pretending that the frame comes from the AP by spoofing its MAC address. The victim accepts this frame, bypassing client isolation. The technique requires the attacker to temporarily connect to the victim's BSSID to obtain the per-BSSID GTK (which is shared during the handshake).

Our attack offers several advantages. First, since frames constructed using GTKs are delivered over the air (and are not forwarded/routed by the AP), they cannot be stopped by the AP-side restrictions. Second, this technique can be used to inject any layer-2 packets (e.g., ARP frames), and can even be used to inject unicast IP packets [18]. As will be shown later in § VII, we find that this works against all modern operating systems, i.e., they use the GTK to decrypt such frames with a broadcast/multicast MAC address, and deliver the embedded IP packet to the OS's IP layer. Third, abusing the GTK can allow an adversary to continue injecting frames even after their access has been revoked. This is because most APs do not update the GTK whenever a client leaves the network, but instead periodically update the GTK, e.g., every hour or day, or may even be configured to never update the GTK [19].

2) Escalating GTK Abuse using Passpoint Flaws: The Passpoint specification, which is the basis of protected hotspots, provides guidelines on how to isolate Wi-Fi clients with layer-2 encryption. Specifically, they recommend using Downstream Group-Addressed Forwarding (DGAF) Disable [13, §5.2], which among other things, tries to prevent abuse of shared group keys by requiring the AP to send a random group key to every client:

> *"Shall set to a unique random value the value of the GTK employed in the 4-Way Handshake"*

This effectively disables group key communication, where essential group-based traffic, such as ARP, is converted to a plurality of unicast frames by the AP. Note that the GTK might also be transported in other handshakes.

Non-randomized GTK. By further analyzing the Passpoint standard, we found that it does not specify a randomization of the group key when transported in the group key handshake, the Fast Initial Link Setup (FILS) handshake, the Fast BSS Transition (FT) handshake, or in WNM-Sleep Response frames [13]. This means that, although clients will receive a randomized group key when connecting to the network for the first time using the 4-way handshake, they will receive the real group key when one of the above handshakes is performed. For instance, since many networks periodically refresh the GTK [19] and use the group key handshake to send the new GTK to clients, an adversary can simply wait until this happens, and then abuse this GTK to inject frames and bypass client isolation. Alternatively, an adversary can spoof BSS Transition Requests to make a victim perform an FT handshake [19], avoiding the need to wait for the periodic group key handshake.

Non-randomized IGTK. In security-aware implementations, as mentioned, it is possible that GTK is randomized to maximize isolation; however, one can still abuse IGTK to inject frames to a victim. The IGTK is used to authenticate broadcast and multicast management frames. A second design flaw in Passpoint is that it does not require randomization of the IGTK. Although the IGTK cannot be directly abused to bypass client isolation, we found that the IGTK can be abused to trick a victim into using an attacker-controlled GTK, which subsequently can be abused to bypass client isolation.

To abuse the shared IGTK, an adversary can send a WNM-Sleep Response frame that is protected using the shared IGTK. Although these WNM-Sleep frames usually have a unicast received address, the standard does not prohibit broadcast addresses, meaning most clients will process this frame. Since we send this frame with a broadcast receiver address, all clients will receive this frame. However, only clients that are waiting for a WNM-Sleep Response frame will typically process this frame, other clients will ignore it. The adversary can now include a GTK in this WNM frame, causing the victim to install this GTK selected by the adversary. After this, the adversary can send broadcast or multicast data traffic that is protected using this GTK towards the victim(s), thereby bypassing client isolation.

Other shared keys. Apart from the GTK and IGTK, modern Wi-Fi networks also use a shared BIGTK and WIGTK to protect beacons and wake-up frames, respectively. However, we were unable to abuse them to bypass client isolation.

V. Attacking Switching and Routing

In today's Wi-Fi networks, it is increasingly common to have multiple BSSIDs. For instance, even in networks with a single AP, one can set up 2.4 GHz and 5 GHz channels, which correspond to two separate BSSIDs/SSIDs; a user can choose to connect to either. If client isolation is present in a single BSSID, one would expect that it would also exist across BSSIDs in the LAN to ensure security. More importantly, such a single-AP network can also be configured to create guest networks, with dedicated (virtual) SSIDs as covered in §II-A. From the security point of view, these guest networks should be isolated from the main network (they often require different Wi-Fi credentials). Complex networks with multiple APs are even more likely to use multiple BSSIDs.

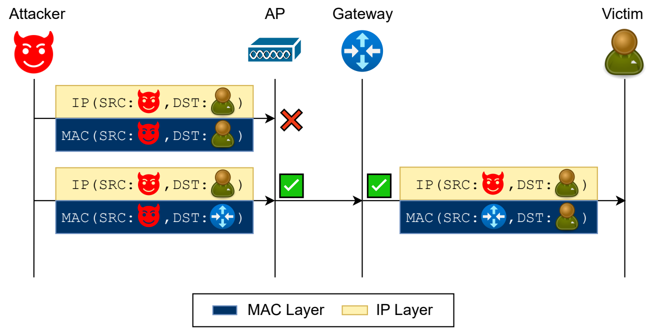

Fig. 2. Exploiting the gateway to bypass client isolation. While the attacker's direct layer 2 communication with the victim may be blocked, the gateway can be used to inject the packets at layer 3.

For example, enterprises, hotels, and university networks often create a separate SSID (and correspondingly multiple BSSIDs) for guests or visitors.

In this section, we first unearth the lack of IP layer isolation that allows breaking the client isolation barrier (B) in both basic and complex networks. Next, to break the barrier (C), we showcase a subtle interplay between Wi-Fi encryption and the internal layer-2 switching mechanism of APs, and thus discover a fundamental limitation of Wi-Fi encryption. We show that an attacker can not only inject packets but also intercept packets to and from a victim, thereby enabling the building blocks of MitM attacks.

A. *Injection Attacks at the Routing Layer*

Gateway Bouncing. A key observation we make is that even if layer 2 has client isolation, it should also be carried over to layer 3, i.e., the IP layer. In other words, clients within the same Wi-Fi network should not be allowed to *route* layer 3 packets to each other. We find that, unfortunately, this is not implemented in many Wi-Fi networks. Using the terminology in Figure 1, this means that while the AP can block direct Wi-Fi frame forwarding between two clients associated with the same AP, it still forwards packets that are sent to a gateway (a.k.a. router), which performs IP routing. Leveraging this oversight, we devise a straightforward and effective technique to bypass client isolation. Specifically, we find that an attacker can send data packets with the destination IP address being that of the victim and the destination MAC address being that of the network's gateway. Such a packet will be "bounced", i.e., routed, back to the victim via the gateway. As shown in Figure 2, this packet will be "accepted" by the network's gateway because its destination MAC address matches the gateway's MAC address. Next, since the packet destination IP address belongs to a different client, the routing infrastructure on the gateway will direct the packet to the victim client. Such a "bounced" packet will have its source MAC address being that of the network's gateway and the destination MAC address being that of the victim. The victim ultimately receives the packet, effectively allowing client-to-client injection despite layer-2 isolation.

B. Interception Attacks at the Switching Layer

Every BSSID on the same AP can be viewed as a virtualized hardware port. To understand our switching-based attacks, we first describe the switching and port architecture of modern Wi-Fi networks. Our explanation is based on the 802.11 standard and empirically confirmed by analyzing software from jailbroken APs spanning seven vendors (as will be covered in §VII). Most notably, we observe that every BSSID can be viewed as a virtualized hardware port. For example, when clients are connected to the same BSSID (e.g., 2.4 GHz frequency), it is conceptually analogous to the clients being connected to the same Ethernet port in a wired setting. Similarly, a guest BSSID has its own associated virtual port, distinct from the port for a trusted/main BSSID, and conceptually it is equivalent to a separate hardware Ethernet port. Indeed, each BSSID by design has a unique MAC address, just like in the case of traditional hardware switch ports. Upon receiving a Wi-Fi frame on a virtual port, the AP will translate it to an Ethernet frame, which gets switched/routed internally to other virtual ports, and potentially further through the distribution system.

Port stealing attacks exploiting multiple BSSIDs. Based on the above insight into the relationship between BSSIDs and virtual ports in Wi-Fi networks, we find that the classic port stealing attack [33], originally proposed for switches with physical ports, can be repurposed to break inter-BSSID isolation in the Wi-Fi setting.

Specifically, an attacker can send frames with the spoofed MAC address of a victim client from the attacker's own associated virtual port. This then triggers the layer-2 learning process on the AP with respect to its virtual ports, misleading the AP to incorrectly update its forwarding table to associate the victim's MAC address with the attacker's port. As a result, traffic originally destined for the victim is redirected to the attacker. This forms the basis of our MitM attacks described in the next section.

In addition, an attacker can also spoof the router's MAC address to redirect a victim's uplink traffic to the attacker's virtual port. This attack exploits the weak coupling between access points (APs) and the wired distribution system: since APs operate at layer 2 and are unaware of the router's actual location, whether on the wired or wireless side, they may mistakenly forward the victim's traffic to the attacker. Interestingly, this results in client-to-client traffic, since the victim's traffic now arrives at the attacker, who's also a client. Surprisingly, we nevertheless found that this attack often works, even when other client-to-client traffic is blocked. This highlights the ad-hoc nature of client isolation policies by vendors, the unpredictability of its security guarantees in practice, and further confirms the need for a standardized and agreed-upon definition of client isolation.

Overcoming Wi-Fi encryption to achieve port stealing. Unlike traditional Ethernet switches, where port stealing can be accomplished by simply injecting spoofed Ethernet frames, modern Wi-Fi networks require attackers to first authenticate with the network. Importantly, port stealing cannot be achieved by simply connecting to the Wi-Fi network with the attacker's own MAC address, and then spoofing frames with the victim's MAC address as source while using the attacker's Pairwise Transition Key (PTK) to encrypt the frame. This is because the AP uses the frame's transmitter and receiver MAC addresses to look up the corresponding PTK, i.e., the AP maps MAC addresses to PTK keys. As a result, the AP will use the victim's PTK to decrypt the frame, which will fail, meaning the spoofed frame is dropped and port stealing fails.

Our idea is to have the attacker authenticate using the victim's MAC address, but from a different virtual port, i.e., BSSID, than the victim. In other words, layer-2 learning for Wi-Fi virtual ports becomes possible only after the attacker completes an 802.11 association using the victim's MAC address. More specifically, our hypothesis is that the intercepted traffic will now be forwarded to the attacker's virtual port, using the attacker's negotiated PTK.

Indeed, we verified that the strategy works on most APs we have tested. After investigating the root causes, we find this behavior stems from how encryption credentials are managed in AP software such as `hostapd`. For each BSSID, i.e., virtual port, the AP maintains a mapping from a client MAC address to its PTK. By default, this mapping is scoped per-BSSID. Thus, if an attacker reuses the victim's MAC address to complete a handshake, the AP updates the security association for that MAC to point to the attacker's PTK. In effect, after port stealing, the AP will consult the new <MAC, PTK> mapping under the exploited BSSID instead of the mapping on the victim's associated BSSID. Thus, all packets destined for the victim's MAC address are encrypted using the attacker's keys, despite originally being associated with the victim station.

This reveals a decoupling issue between layer-2 forwarding, i.e., which virtual port the packet exits from, and the cryptographic association, i.e., which PTK is used for encryption. We illustrate this issue in Figure 3. According to IEEE 802.11i [34, §5.9.2.1], after a successful 4-way handshake, the AP opens the controlled port for general data traffic. However, what is less obvious, and exploited in this attack, is that the PTK used for encryption is not tightly bound to the original session or physical port but to the latest <MAC, PTK> binding.

In the most severe scenario, we experimentally verified that when the attacker connects to an open SSID without encryption, port stealing causes the victim's traffic to be forwarded in plaintext over the attacker-associated, unprotected BSSID. As a result, traffic normally protected by WPA2/3 encryption will be broadcast in plaintext, allowing trivial eavesdropping by any nearby observer. Therefore, port stealing undermines the efficacy of encryption regardless of the original security configuration of the victim's SSID.

C. Injection Attacks at the Switching Layer

We uncover a subtle injection method at the switching layer, which we term *Broadcast Reflection*. By crafting a Wi-Fi frame with `ToDS = 1` and Address 3 set to the broadcast

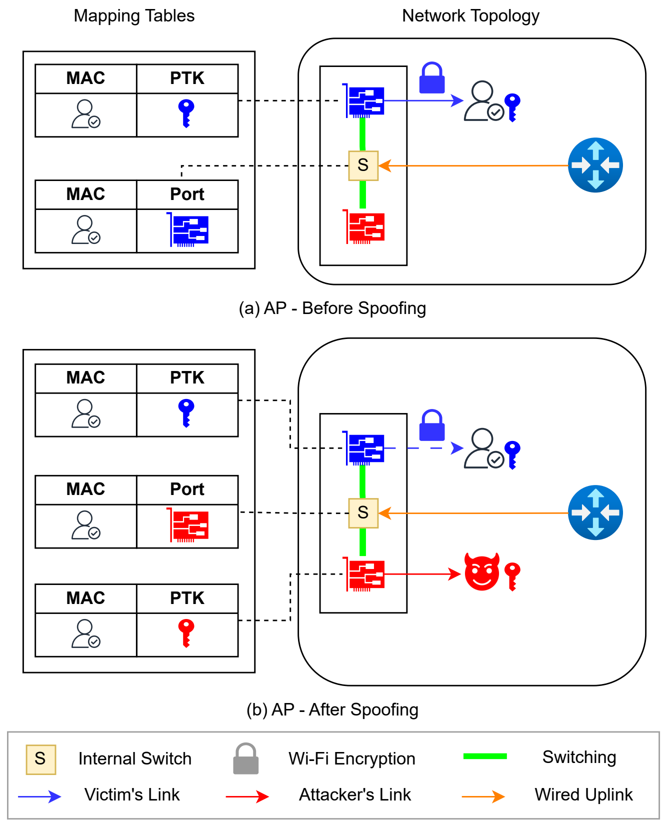

Fig. 3. The attacker spoofs the victim's MAC address on a different NIC, causing the internal switch to mistakenly associate the victim's address with the attacker's port/BSSID. As a result, frames intended for the victim are forwarded to the attacker and encrypted using the attacker's PTK.

MAC address `FF:FF:FF:FF:FF:FF`, the attacker causes the AP to treat the frame as group traffic and re-encrypt it using the GTK of the victim's BSSID. This results in the frame being delivered to all clients on that BSSID, including the victim.

Crucially, the embedded payload can contain unicast traffic targeted at the victim. As shown in §VII-C, most OSes accept such payloads when decrypted with the GTK. Unlike direct GTK abuse (§IV-B), this technique does not require the attacker to know or extract the GTK, making it effective from separate BSSIDs or even open networks.

Broadcast Reflection thus enables stealthy, GTK-free injection and can be chained with port stealing (§V-B) to reflect intercepted packets back to the victim, supporting MitM attacks even across BSSIDs.

VI. Gaining Full MitM in Enterprise Networks

In this section, we combine our previous interception and injection attacks to obtain a full bidirectional MitM in enterprise networks, bypassing client isolation.

A. MitM Attack on Victims on the same AP

Although combining port stealing and gateway bouncing appears to offer a straightforward path to a (bi-directional) MitM attack, executing it reliably requires addressing several key practical challenges, which we cover first:

Assumptions. For ease of exposition of such attacks, we assume that an AP supports simultaneous SSID operations across two frequency bands as is commonly the case: a 2.4 GHz channel and a 5 GHz channel. Each frequency channel also hosts one main SSID and one guest SSID. These in turn map onto four virtual ports to which a client might connect. The four virtual ports corresponding to these four BSSIDs are connected to the same software-enabled switch, which, in turn, is connected to an uplink port. This uplink port is connected to the gateway to reach the Internet.

We consider two attack scenarios: (1) an attacker has the credentials to associate with only untrusted/guest SSIDs and targets a client with a trusted SSID; (2) an attacker is connected to the same SSID (e.g., trusted) as the victim. In both cases we assume the network uses client isolation.

Intercepting the victim's downlink and uplink traffic. As discussed in §V-B, an attacker can intercept both uplink and downlink traffic using port stealing as long as the attacker and the victim are connected to different virtual ports, e.g., if the attacker is on guest/2.4GHz and victim is on main/2.4GHz, or if the attacker is on guest/2.4GHz and the victim is on guest/5GHz. It is important to note that to maintain a MitM attack and the stolen ports, the attacker has to send spoofed MAC frames continuously. This prevents the victim's legitimate uplink or downlink traffic from reclaiming the port via the learning process by which the switch updates its forwarding tables.

Returning intercepted downlink traffic back to the victim. To reliably maintain the MitM attack, it is necessary to return the intercepted frames back to the victim; otherwise, it will lead to a DoS attack that could be easily detected. We describe how to achieve this, using the two injection attacks mentioned earlier, and one assistive technique:

(1) Gateway bouncing. Gateway bouncing can be repurposed to inject intercepted packets back to the victim. However, it requires the AP to correctly forward the packet to a port associated with the victim's MAC address. This means that the attacker must relinquish control and allow the victim to reclaim the original port, e.g., by pausing port stealing and waiting for the victim to send an uplink frame. This restores the correct MAC-to-port mapping corresponding to the victim, ensuring that the packet is delivered as intended. The attacker can detect this uplink traffic by passively eavesdropping on the frequency channel used by the victim's MAC address and time the injection accordingly. Note that the attacker needs to periodically stop port stealing to keep the traffic flowing to the victim. Inevitably, it means that periodically, some downlink traffic would miss getting intercepted due to this pause. This technique also relies on IP spoofing to make the injected packet appear as if it is coming from the original server.

(2) Abusing GTK. An attacker can return the intercepted traffic using the per-BSSID GTK, which does not require the above port restoration, since GTK abuses do not depend on the AP to forward packets. It is worth noting that this technique enables IP spoofing that APs cannot stop: while an AP can

detect abuse by decrypting injected frames, it may not even overhear them due to hidden terminals [21] or signal coverage limitations. Furthermore, the Passpoint flaws (§IV-B2) make GTK abuse more potent and persistent. However, the downside of this approach is that the attacker must be able to connect to the victim's BSSID, and a direct link must be viable between the attacker and the victim; this is also subject to hidden terminal issues or signal strength limitations.

(3) Client-triggered port restoration. To eliminate the delay inherent in gateway bouncing, where the attacker must wait for the victim to send an uplink frame and reclaim its MAC-to-port mapping, an attacker can instead trigger this restoration proactively. By crafting GTK-encrypted unicast ICMP Echo Requests, the attacker induces the victim to generate ICMP Echo Replies, which cause the AP to rebind the victim's MAC address to its original port. This enables the timely reinjection of intercepted packets via gateway bouncing, without relying on natural uplink traffic.

Returning intercepted uplink traffic back to the gateway with server-triggered port restoration. To relay the intercepted *uplink* traffic to the benign server on the Internet (thereby completing the MitM role on the uplink), we need the gateway's MAC address to be correctly mapped to AP's uplink port. However, the attacker has stolen the gateway's MAC address for uplink interception, which disrupts the mapping; this requires the attacker to release it temporarily in order to restore the correct forwarding path.

Unfortunately, this restoration relies on the AP passively learning the correct port again, which only happens when it sees traffic from the gateway. To address this, the attacker can either wait passively for such traffic (e.g., from the Internet), or proactively trigger the same. Toward accomplishing the latter, we develop a novel proactive technique which we call "Server-triggered Port Restoration", which briefly restores the correct forwarding path to ensure reliable delivery of outbound victim traffic while minimizing loss of attacker control. The attacker first establishes a connection (e.g., TCP, UDP, or ICMP) with an external server and agrees on a fixed transmission schedule, such as once every 100 ms; with this periodicity, the attacker triggers the server to send a burst of packets back. This preemptively restores the gateway's MAC-to-port mapping by triggering the AP's Layer-2 learning mechanism, effectively maintaining hardware port control. During this brief window, the attacker forwards queued or intercepted packets upstream via the real gateway to the proper server. Afterwards, the attacker alternates between "Server-triggered Port Restoration" and port stealing, to maintain control of the uplink path.

B. Cross-AP MitM Attacks are Practical

The previous attacks assumed that the attacker and the victim share the same AP. We also discover and demonstrate that cross-AP MitM attacks are not only possible but practical in enterprise and campus networks where multiple APs share a wired distribution system (Figure 1).

Although port stealing was originally devised for hosts on the same switch [33], we show that attackers can hijack MAC-to-port mappings at a higher layer, i.e., at the level of the distribution switch [35]—to intercept traffic to victims associated with different APs. This escalates the attack beyond its traditional limits, breaking the assumption that separate APs provide effective isolation.

This discovery exposes a blind spot in client isolation: even physically separated APs, broadcasting different SSIDs, offer ineffective isolation if connected to a common distribution system. By redirecting traffic at the distribution switch, attackers can intercept and manipulate victim traffic across AP boundaries, expanding the threat model for modern Wi-Fi networks. We further detail and demonstrate this attack in a practical setting in §VII-G.

TABLE I
FIRMWARE VERSIONS AND AP DAEMONS OF TESTED APS/ROUTERS

Device Model	Firmware Version	AP Daemon
Netgear Nighthawk X6 R8000	V1.0.4.84_10.1.84	nas
Tenda RX2 Pro	V16.03.30.14_multi	hostapd
D-Link DIR-3040	1.13	apsond
TP-Link Archer AXE75	1.1.8 Build 20230718	hostapd
ASUS RT-AX57	3.0.0.4.386_52332	hostapd
DD-WRT v3.0-r44715	v3.0-r44715	nas
OpenWrt 24.10	24.10.0 r28427	hostapd
Ubiquiti AmpliFi Alien Router	v4.0.8, g0c028c5c	hostapd†
Ubiquiti AmpliFi Router HD	v4.0.3, g0bc740d76d	hostapd
LANCOM LX-6500	6.00.0085	lancom daemon
Cisco Catalyst 9130	IOS XE 17.2.1.11	unknown

† This device also uses hap-wifirouter for device management.

VII. EVALUATION AND MEASUREMENT

In this section, we conduct a comprehensive evaluation of various techniques across home routers, a local enterprise network testbed, and a university network.

A. Experiment Setups

We selected 5 recent home routers from different popular vendors that advertised a form of guest or client isolation. In addition, we test two widely-used open-source router distributions DD-WRT and OpenWRT (see Table I for firmware versions and AP daemons, and Table II for results). During our initial inspection, we observed that secure configurations were inconsistently applied and dispersed across devices. For the purposes of testing, we enabled AP mode on each router to simulate a typical home environment, given that an external modem was already functioning as the network gateway. Whenever a router supported a "guest network" feature, we activated it so we could test its isolation properties. Additionally, if a client isolation option was available in the wireless settings, we selected it to further segregate client-to-client communication.

B. Measuring Inter-BSSID Isolation Policies

Inconsistent and ad-hoc isolation policies. In addition to intra-BSSID isolation mechanisms, we observed inconsistencies in inter-BSSID isolation mechanisms across various vendors and AP devices, enabling different attack variants (see Table II). Inter-BSSID isolation mechanisms, a critical component for segregating traffic between virtual BSSIDs, are frequently missing or misconfigured.

Specifically, we find that many APs fail to enforce strict separation between these virtual BSSIDs' associated ports. We forge layer-2 frames targeting other clients under the same AP, and found that all tested APs allow some degree of unintended switching that violates client isolation between these virtual BSSIDs. As shown in Table II, different AP devices manifested inconsistent behaviors with regard to whether a source client in a guest/main BSSID can use layer-2 frames to directly reach a client in another guest/main BSSID under the same AP device. We emphasize that the use of `hostapd` is neither a sufficient nor a necessary condition for an AP to be vulnerable to port stealing, indicating only a partial correlation.

C. Measuring GTK Abuse Acceptance by OSes

Table IV shows whether OSes accept higher-layer unicast traffic inside GTK-encrypted broadcast frames. Specifically, we tested for the acceptance of unicast IPv4 ping requests (group-ping), unicast IPv6 ping requests (group-ping6), and unicast ARP requests (group-arp-unicast). These tests simulate GTK-encrypted injections that bypass client isolation.

Most OSes, including macOS, iOS, and Android, reply to all three tests, confirming acceptance of the GTK-encrypted traffic. Windows 11 and Ubuntu 22.04 stand out: when enabling `drop_unicast_in_l2_multicast` on Linux, an option to drop unicast IP packets received in link-layer multicast/broadcast frames [36], Linux drops the IPv4 ping request but still replies to IPv6 and ARP-based probes. These results confirm that most OSes accept unicast IP datagrams inside GTK-encrypted broadcast frames.

D. Experiments Against Home Routers

Single-BSSID Attacks. We tested our abusing GTK, machine-on-the-side, and rogue AP attacks against seven home Wi-Fi routers, which represent single-BSSID networks, and found that these techniques are effective in bypassing client isolation in all cases. Furthermore, we verified that the "abusing GTK attack" is general and independent of the encryption methods, including WPA2 and WPA3. For WPA2-PSK, we verified that machine-on-the-side bypass works for all seven APs: the PTK can be successfully derived by the attacker to launch decryption and injection of packets over the air, bypassing client isolation. For the rogue AP attack against both WPA2 and WPA3, we verified that the attack works with all seven Wi-Fi routers, including dissociating victim clients and forcing them to connect to our rogue AP.

Inter-BSSID Attacks. To understand the generality of the inter-BSSID isolation bypass techniques, we carried out experiments on all seven APs. Table II presents the results. Five tested single-AP routers were found to permit the circumvention of isolation mechanisms between guest and main networks, allowing frame/packet injection. Also, in Table III, six devices allowed uplink/downlink port stealing, and four of them allowed breaking barriers between guest and main networks. Interestingly, Tenda RX2 Pro, DDWRT v3.0-r44715, and OpenWrt 24.10 successfully blocked port stealing attempts when configured with recommended guest network settings [37], [38]. Through reverse engineering, we discovered that isolating the guest and main networks using separate internal switches, i.e., bridges, effectively blocked the attacks for DDWRT and OpenWrt. Tenda RX2 Pro had some mechanisms to prevent the same client MAC address from being used on two BSSIDs concurrently, which also helped block port stealing. Nevertheless, these results highlight the prevalence of weak and chaotic inter-BSSID isolation mechanisms (i.e., unintended switching and routing) across diverse models and vendors.

End-to-End Attack against Netgear R8000. In this local experiment, we configure guest and trusted SSIDs on both 2.4 GHz and 5 GHz bands, yielding four BSSIDs. To access the Internet, the Netgear AP is connected to a gateway router via a wired link. Restricted to the guest network, the attacker aims to act as a MitM, intercepting all uplink and downlink traffic from a victim on the trusted SSID. The attacker begins by connecting to the guest SSID with the victim's MAC address, but on a different frequency to avoid disconnecting the victim.

After association, the attacker rapidly transmits a burst of 802.11 data frames with the ToDS flag set to 1 and the address 3 field set to the address of the bridge `br0`. In our experiments, these frames are ICMP Echo Reply packets which are used to intentionally avoid soliciting any responses that can trigger unwanted layer-2 learning (other packets that do not solicit responses can also be used). Since these packets are sent with the victim's MAC address and are correctly encrypted, they trigger the layer-2 learning and cause the AP to redirect the victim's downlink traffic to the guest SSID. The attacker then returns the intercepted traffic back to the victim using the gateway bouncing technique. Similarly, an attacker performs the uplink traffic interception by spoofing the MAC address of the AP's uplink node (i.e., gateway router) and returns this intercepted traffic back to the victim's server. The full attack takes about 2 seconds to complete. Throughout the attack, the victim is watching a streamed Youtube video and does not experience significant lag.

E. Experiments Against Enterprise-Grade Devices

We also tested enterprise-grade devices. As these are more expensive, we limited our tests to two Ubiquiti routers, a Cisco Catalyst 9130, and a LANCOM LX-6500. Both Ubiquiti devices support guest networks and client isolation, and were vulnerable to gateway bouncing, abusing GTK, broadcast reflection, and port stealing (see Table II and III).

To test the Cisco and LANCOM devices, we configured them to advertise both a main and guest SSID. To enable client isolation on the Cisco device we set "P2P Blocking Action" to drop, and on LANCOM we disallowed "Communication between end devices on this SSID". In this setup, both Cisco and LANCOM were affected by abusing GTK and downlink port stealing. The Cisco device was also affected by broadcast reflection, and the LANCOM device by uplink port stealing. During the downlink port stealing attack, the victim's uplink

TABLE II
MEASUREMENT OF THE FEASIBILITY OF INJECTING TRAFFIC FOR SELECTED SINGLE APS. ALL TESTED WITH CLIENT ISOLATION ENABLED. TOP DEVICES ARE HOME ROUTERS, BOTTOM DEVICES ARE ALL-IN-ONE ENTERPRISE ROUTERS.

Device Model	**Direct L2 Forwarding**				**Abusing GTK**		**Gateway Bouncing**			
	G→M	M→M	G→G	M→G	M→M	G→G	G→M	M→M	G→G	M→G
Netgear Nighthawk X6 R8000	×	✓	×	✓	✓	✓	✓	✓	✓	✓
Tenda RX2 Pro	✓	✓	✓	✓	✓	✓	×	✓	✓	×
D-Link DIR-3040	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TP-Link Archer AXE75	×	✓	×	×	✓	✓	✓	✓	✓	✓
ASUS RT-AX57	✓	×	✓	×	✓	✓	×	✓	×	✓
DD-WRT v3.0-r44715	×	✓	×	×	✓	✓	×	✓	×	×
OpenWrt 24.10	×	×	✓	×	✓	✓	×	✓	×	×
Ubiquiti AmpliFi Alien Router	×	✓	✓	×	✓	✓	×	✓	✓	✓
Ubiquiti AmpliFi Router HD	×	✓	✓	×	✓	✓	×	✓	✓	✓
Cisco Catalyst 9130	×	×	×	×	✓	✓	✓	✓	✓	✓
LANCOM LX-6500	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

M: Main network, G: Guest network; $X \rightarrow Y$: whether a client in network X can inject a packet towards another client in network Y.

TABLE III
MEASUREMENT OF THE FEASIBILITY OF INTERCEPTING TRAFFIC FOR SELECTED SINGLE APS. ALL TESTED WITH CLIENT ISOLATION ENABLED.

Device Model	**Downlink Port Stealing**				**Uplink Port Stealing**			
	G←M	M←M	G←G	M←G	G←M	M←M	G←G	M←G
Netgear Nighthawk X6 R8000	✓	✓	✓	✓	✓	✓	×	×
Tenda RX2 Pro	×	×	×	×	N/A★	✓	N/A★	N/A★
D-Link DIR-3040	✓	✓	✓	✓	✓	✓	✓	✓
TP-Link Archer AXE75	✓	×	✓	✓	✓	✓	×	✓
ASUS RT-AX57	×	✓	×	×	✓	✓	✓	✓
DD-WRT v3.0-r44715	×	×	×	×	×	✓	×	×
OpenWrt 24.10	×	×	×	×	×	×	×	×
Ubiquiti AmpliFi Alien Router	×	✓	✓	×	×	✓	×	×
Ubiquiti AmpliFi Router HD	×	✓	✓	×	×	✓	×	×
Cisco Catalyst 9130	✓	✓	✓	✓	×	×	×	×
LANCOM LX-6500	✓	✓	✓	✓	✓	✓	✓	✓

★ Tenda RX2 Pro does not support guest SSIDs under AP mode. M: Main network, G: Guest network.
$X \leftarrow Y$: whether a client in network X can intercept traffic of another client in network Y.

traffic was no longer forwarded by the AP. We believe this is because these devices have difficulty handling two client with the same MAC address over different BSSIDs. We consider it interesting future work to test uplink port stealing attacks when the targeted network is using different devices to broadcast BSSIDs, where the adversary can then connect to a different physical AP than the victim to perform the attack. More generally, enterprise devices often require tedious manual configuration, which may influence our tests, and we therefore test network deployments in the wild next.

F. Experiments in Networks in the Wild

Testing University Networks. It is widely believed that WPA2/3-Enterprise prevents traffic interception due to the usage of per-client credentials. In these experiments, our aim is to show how our attacks break this assumption.

We test two university networks (in different countries) and find that our attacks are viable in the real world. Both universities follow a similar setup, and we describe one of them here for space and brevity. The university network has three SSIDs: one open SSID with captive portals that optimistically collect guest information without deploying Wi-Fi encryption, one staff/employee WPA2-Enterprise SSID accessible only with valid 802.1X university credentials, and one `eduroam` SSID [39] with WPA2-Enterprise encryption accessible with valid university credentials registered with `eduroam`. Every physical AP managed by this university can concurrently serve these three SSIDs on multiple channels. In this tested network, there is also intra-BSSID isolation for every BSSID.

In both our experiments, the attacker is connected to the open guest network, and the goal is to intercept the downlink traffic of another victim client (our own for ethical reasons) associated with the trusted SSID. To intercept the victim's downlink WPA2-Enterprise traffic, the attacker first scans all channels to identify the victim's MAC address and associated BSSID. Our attacker device uses the victim's MAC address to connect to the same AP's open SSID and finishes captive portal authentication. It then floods many frames to that of the shared gateway/router and with ToDS set to 1. The gateway/router then redirects the victim's downlink packets to the attacker-chosen AP (i.e., port stealing). These packets are finally leaked in plaintext through the open SSID. To return intercepted plaintext traffic back to the victim, the attacker uses gateway bouncing or broadcast reflection.

We conducted all university tests during weekend nights, at times when no other clients were observed on the network. To further ensure that our experiments did not impact unintended

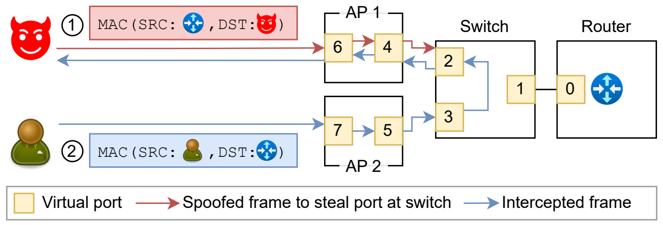

Fig. 4. Illustration of port stealing. The adversary spoofs the router's MAC address and sends uplink frames with ToDS=1 (step ①), causing the switch to map the router's MAC address to port 2 and redirect the victim's uplink traffic from AP2 to AP1 (step ②). Blue arrows show the redirected traffic path.

parties, we restricted injected frames to use the victim's IP as the destination and applied Port Stealing and Gateway Bouncing only to MAC/IP addresses under our control.

G. *Attacks in Enterprise Settings on a Local Testbed*

We first performed the multi-AP attacks (§VI-B) in a local testbed (TP-Link EAP613), with the topology similar to Figure 4. We verified that the attacker is able to intercept both the downlink and uplink traffic of a victim associated with a different AP, as well as returning the traffic back to the victim. For simplicity, we will focus on how to intercept the uplink traffic. Using the guest Wi-Fi credentials, the attacker first uses two NICs to complete handshakes and establishes two connections to the guest SSID on `AP1`, i.e., port 6 in Figure 4, with the MAC address of the gateway (NIC1) and a random MAC address (NIC2) respectively. Once associated, the attacker exploits the layer-2 learning process of the `Switch`. Specifically, the attacker uses NIC1 to transmit a series of ICMP Echo Reply packets with the layer-2 header being `ToDS = 1` and `address 3 = attacker's random MAC address`. These data frames cause the `Switch` to map the router's MAC address to the attacker's port, i.e., `port 2`. To return intercepted uplink data frames back to the real gateway, the attacker uses the random MAC address to perform "Server-triggered Port Restoration" as discussed in §VI-A. When the gateway MAC address is mapped again to `port 1`, the attacker relays all intercepted traffic internally through NIC2, and forwards such traffic to the real gateway/router. For ease of exposition, we call this technique "Inter-NIC Relaying".

Case study on how our attacks can facilitate a higher-layer compromise. We further investigate whether the impact of the attack can be amplified to accomplish a higher-layer compromise. In WPA2-Enterprise, a Wi-Fi client first finishes 802.11 authentication and association with an AP. Then, the client negotiates a PMK with a remote authentication server (e.g., RADIUS). The authentication server then returns this PMK to the AP, for it to start a four-way handshake with the Wi-Fi client to negotiate a PTK and a GTK. Under the hood, enterprise APs facilitate PMK negotiation between clients and remote authentication servers using the RADIUS protocol [40]. This protocol acts as an intermediary, securely wrapping authentication/authorization requests to ensure proper validation of client credentials (e.g., enterprise employee credentials). For robust security, the communication channel between any enterprise AP and its remote authentication server must be encrypted to prevent interception or tampering. In practice, RADIUS remains the industry-standard authentication method, supported by nearly all enterprise APs, and serves as the backbone for onboarding enterprise Wi-Fi clients.

Using the same local enterprise testbed, we discover that by spoofing a gateway MAC address and connecting to an AP, an attacker can steal uplink RADIUS packets. In the RADIUS protocol, the client is the AP and the server is the remote authentication server, and they pre-share a passphrase. This passphrase is used to encrypt and authenticate RADIUS packet fields, such as to encrypt PMK in transit and derive the Message Authenticator, a hash for integrity-protection. We verified that an attacker, having intercepted the first RADIUS packet sent from the enterprise AP, can brute-force the Message Authenticator and learn the AP passphrase. This allows the attacker to set up a rogue RADIUS server and associated rogue WPA2/3 access point, which allows any legitimate client to connect, thereby intercepting their traffic and credentials.

H. *Measuring Port Stealing and GTK Abuse Performance*

Attack Setup. We showcase the downlink interception performance of our MitM framework, built by combining port stealing with GTK Abuse, with a realistic Wi-Fi setup (Table V). The attacker's program adopts a multi-process architecture: a main controller process performs port stealing, while three auxiliary subprocesses respectively handle frame capture, re-encryption using the GTK, and frame injection. The target AP under measurement is a Netgear R8000, connected to an upstream router, which is further connected to a MacBook running macOS 15.7.2 hosting the iPerf3 server used for performance measurements. Both the victim and the attacker are PCs equipped with an Intel Core 5 120U CPU, running Ubuntu 22.04 with Linux kernel 6.8.0. The injection NIC is Alfa AWUS036ACM. The attacker and the victim are on separate BSSIDs and the attacker abuses the GTK of the victim's BSSID with the goal of measuring if the attack degrades performance. Each test case instructs the iPerf3 server to send fixed 10 Mbps UDP traffic, runs for one minute, and is repeated five times.

Results and Factors Affecting Attack Success. Our system succeeds in all downlink MitM tests. We note a short initial disruption when the attacker associates with a different BSSID, using the victim's MAC address: some downlink frames are lost during this time since the attacker has yet to establish access to the network. After this period, the attacker can successfully intercept and inject packets, although with occasional losses. Below, we highlight key factors that affect the performance of the attack:

1) Environmental and Distance Factors: Increasing the physical distance between the attacker and AP, or introducing obstacles such as a closed door (test case barrier in Table V), causes additional loss as expected, due to weaker channel con-

TABLE IV
MEASURING OPERATING SYSTEM ACCEPTANCE OF GTK-ENCRYPTED WI-FI FRAMES

OS	group-ping	group-ping6	group-arp-unicast
Windows 11			
Firewall On	×	✓	✓
Firewall Off	✓	✓	✓
Others			
macOS 15.4	✓	✓	✓
iOS 18.3.2	✓	✓	✓
Android 14	✓	✓	✓
Ubuntu 22.04	×*	✓	✓

✓: Test case passed (OS replied to the probe).
×: Test case failed (OS did not reply).
*: When `drop_unicast_in_l2_multicast` is enabled, the group-ping test case does not result in a reply.

TABLE V
MEASUREMENT OF PORT STEALING AND GTK ABUSE PERFORMANCE UNDER REALISTIC CONDITIONS.[†]

Exp.	Setup	Performance			
	Attacker	Loss	Throughput	Jitter	Success
Base	Near AP	1.7%	8.89 Mbps	1.23ms	5/5
Distance	Far	3.1%	8.93 Mbps	0.61ms	5/5
Barrier	Wall-sep.	7.0%	8.59 Mbps	2.08ms	5/5

[†] Loss = loss rate ignoring initial disruption; Success = end-to-end success rate.

ditions. Importantly, the attack continues to function with high success, demonstrating robustness under channel degradation.

2) Bit rate limitation: During the attack, our NIC conservatively uses the lowest modulation and coding scheme to inject traffic. This is because the injection uses the group key and multicasts, which do not expect ACKs (most of the experienced losses occur during the injection). To increase the reliability of transmissions, it uses the lowest transmission rate of 10 Mbps in our case, and thus is unable to realize a higher injection speed. If the server sends at a higher rate, this can lead to buffering and packet drops at the attacker. We note that our experiments use iPerf3 which do not have rate control; in practice however, we expect senders to control the sending rate (e.g., with backoffs and retransmissions in TCP) which will help recover from these losses, and the server's sending rate will adapt to the rate at which packets are returned to the victim; except for a slow down of delivery the victim should not perceive any losses in that case. Finally, this limitation can be overcome by more capable radios, such as higher-power transmitters or even software-defined radios (SDRs [41]), which could potentially alleviate or bypass this limitation by enabling more reliable, higher-rate frame injection.

VIII. DISCUSSION AND DEFENSE

In this section, we discuss the practical applications of our attacks, how different attack techniques can be combined, and propose defenses to improve Wi-Fi security.

A. Combining Multiple Attack Techniques

As summarized in Table VI, individual techniques are capable of achieving potent attacks themselves, but one can combine them to achieve even stronger attacks such as MitM. For example, Inter-NIC Relaying and Abusing GTK can act as enablers for more sophisticated attack chains to help return intercepted packets to the victim, achieving stable MitM attacks. Interesting future work is exploring additional attack building blocks to enable more compositions.

Our attacks can also be combined with known flaws, e.g., the GTK is predictable on some routers [18], meaning it can be abused to return intercepted traffic even if the adversary cannot connect to the victim's BSSID.

B. Facilitating Higher-Layer Attacks

As manipulating low-level port states can serve as building blocks for powerful attacks targeting higher network layers, we discuss possible consequences of intercepting both uplink and downlink traffic of a victim. We showcased how our attacks can facilitate spoofing a RADIUS server on our local testbed in § VII-G. Here we discuss other possibilities.

Traffic Analysis. Our attacks give the attacker access to the IP packets at the network layer. This facilitates various attacks, e.g., even today 6% and 20% of pages loaded on Windows and Linux, respectively, do not use HTTPS [42]. This allows an attacker to steal the victim's cookies, despite the growing adoption of HTTPS [43]. Our attacks also enable the interception of local intranet websites or services, which are more likely to use plaintext connections. Moreover, even when the victim does use higher-layer encryption such as HTTPS, the used IP addresses are still revealed, which is often enough to know which website is being visited [44]. Finally, traffic analysis techniques can even be used to learn the exact webpage a victim is visiting [45], [46].

DNS and DHCP poisoning. An attacker can intercept and attack any plaintext traffic of the victim. For example, the attacker can intercept a victim's DNS traffic and poison the DNS cache in the victim OS [47], [48]. Alternatively, the attacker can modify the DHCP record and change the gateway address and DNS server that will be used by the victim. These attacks can have a long-lasting impact on the victim, even after the attacker stops being a MitM.

C. Defense

Improving Network Isolation. To improve isolation mechanisms on single APs, untrusted BSSIDs (e.g., guest networks) can be put in isolation groups, i.e., VLANs. VLANs logically separate network segments, meaning an attacker on one VLAN cannot send packets to or snoop on another VLAN. This prevents a client in the untrusted BSSID from launching the port stealing attack to redirect traffic destined to the victim in a trusted BSSID. In addition, one could even put each user within (the equivalent of) a unique VLAN, which would securely synchronize the client's authenticated identity across the network stack; however, the implementation challenges of such a method will need to be investigated. For example, doing so will likely require software updates on most home routers and access points since we find that only a few of the

TABLE VI
PROPERTIES OF TECHNIQUES: DETAILED EFFECT EVALUATION ACROSS ATTACKS.

Technique / Layer	**Decryption**	**Modification**	**Injection**	**Section(s)**	**Wi-Fi Standard Affected★**			
					WPA2-P	**WPA2-E**	**WPA3-P**	**WPA3-E**
Gateway Bouncing (3)	No	Yes	Yes	V-A	✓	✓	✓	✓
Rogue AP (2)	Yes	Yes	Yes	IV-A	✓	×	✓	×
Machine-on-the-side (2)	Yes	Yes	Yes	IV-A	✓	×	×	×
Abusing GTK (2)	MC/BC*	Yes†	Yes†	IV-B, VI-A	✓	✓	✓	✓
Port Stealing (2)	Yes	No	No	V-B	✓	✓	✓	✓
Broadcast Reflection (2)	No	Yes	Yes	V-C	✓	✓	✓	✓
Server-triggered Port Restoration (2)	No	No	Yes (Enabling)	VI-A	✓	✓	✓	✓
Client-triggered Port Restoration (2)	No	No	Yes (Enabling)	VI-A	✓	✓	✓	✓
Inter-NIC Relaying (2)	No	Yes	Yes (Dependent)	VII-G	✓	✓	✓	✓

* MC/BC denotes multicast/broadcast.
† By delivering unicast IP packets encrypted with GTK directly to the victim.
★ P denotes Personal mode, and E denotes Enterprise mode.

AP devices we analyzed seem to support VLANs out of the box. We have verified via experiments that TP-Link EAP613 can put guest SSIDs into separate VLANs, which effectively nullifies the exploitation techniques listed in Table VI.

Spoofing Prevention. While Layer 2 spoofing defenses like port security [35] exist in wired networks, they are rarely applied in Wi-Fi due to virtual ports tied to BSSIDs. An absent safeguard is rejecting spoofed gateway MACs on the wireless side. Another effective policy would be disconnecting clients whose MAC appears on multiple BSSIDs (as seen in Tenda RX2 Pro), preventing cross-BSSID spoofing.

Similarly, IP spoofing prevention [49] can be an effective defense. For example, it can stop the gateway bouncing from returning an intercepted packet where the source IP address may belong to an external server (for relaying intercepted downlink traffic of the victim).

Group Key Security. APs can also stop the shared GTK abuse by using per-client randomized GTKs, which is what Passpoint attempts to do. That is, Passpoint contains a feature called Downstream Group-Addressed Forwarding (DGAF) Disable [13, §5.2]. Under this option, each client is given a random group key, effectively disabling group-addressed traffic, and essential traffic is instead translated into unicast frames sent individually to each client.

Using Device-to-device Encryption to Protect Wi-Fi Traffic. The attacks in this paper undermine the confidentiality, integrity, and authenticity of Wi-Fi traffic. Encrypting layer-2 frame payloads *end-to-end* with unique pairwise keys can effectively thwart these exploits. One standardized solution is MACsec (IEEE 802.1AE) [50], which establishes secure, device-to-device encryption at the link layer. By combining encryption, integrity verification, and device authentication, MACsec can thwart our attacks. Via real-world experiments, we verified that MACsec can be integrated with WPA2/3. By installing cryptographic keys on wired and/or Wi-Fi interfaces, two devices within the same Wi-Fi infrastructure can securely communicate: attackers cannot read traffic encrypted by MACsec and can at most cause DoS.

Standardizing client isolation. Lastly, we recommend to standardize the security guarantees that client isolation should deliver. Crucial future work is then to formally model client isolation and to develop techniques to efficiently enforce and verify its security guarantees.

D. Responsible Disclosure

Subsequent to completing our exploration of the Wi-Fi client isolation attack surface, we promptly notified the appropriate vendors and followed widely accepted responsible disclosure practices. Each vendor was given more than 90 days to develop fixes, and no vulnerability was publicly disclosed prior to mitigation. Most vendors have responded and sought cooperation, and several have already issued software updates.

Because our attacks exploit multiple protocols, standards, and their cross-layer interactions, it is difficult for a single vendor to recognize the full security impact in isolation. As a result, effective long-term mitigation requires ecosystem-level coordination across standards bodies, device manufacturers, and network operators. The Wi-Fi Alliance has addressed the missing randomization of the IGTK in version v3.4 of Passpoint. LANCOM has confirmed our findings and has meanwhile added an option to randomize group keys. Ubiquiti is currently determining appropriate mitigations. D-Link proactively provided us with two routers to help validate attacks and defenses. At the time of writing, other vendors are still evaluating their products, a process that is inherently time-consuming given the cross-layer nature of the vulnerabilities.

IX. RELATED WORK

In this section, we briefly overview attacks and defenses related to the 802.11 standard, revisit port stealing attack for the Ethernet and distinguish our work.

Wi-Fi Security. There has been a notable history of attacking and defending Wi-Fi. After researchers concluded that WEP is insecure [51], [52], [53], TKIP and CCMP were devised, featuring the four-way handshake and the group key handshake that negotiate cryptographic keys to protect Wi-Fi traffic. The four-way handshake was later formally analyzed and slightly improved [54], [55]. However, TKIP as a short-term security solution was quickly defeated [56], [57], [58], [59]. During 2016, even CCMP was found to allow downgrade attacks [18], and in 2017 key reinstallation attacks were discovered against

WPA2 [19], [20], showing that an attacker could abuse these handshakes to intrude into WPA2-protected Wi-Fi networks.

The WPA3 protocol introduces the Dragonfly handshake to mitigate security issues in WPA2, preventing the offline dictionary attack while bringing stronger security properties. However, downgrade attacks, side-channel attacks, and denial-of-service attacks were also discovered in WPA3 [25], [17], [26]. These issues were addressed in an updated version of WPA3 [60]. Beyond traditional Wi-Fi networks, modern home wireless mesh networks were also found to introduce new attack surfaces, allowing attackers to root them remotely [61].

The Hole 196 attack abuses the observation that group keys are shared between different clients [62], similar to our GTK Abuse attack, but does not study the abuse of group keys under client isolation. Similarly, Vanhoef and Piessens noted that broadcast Wi-Fi frames may contain unicast IP packets [18], but did not study this in the context of client isolation. Lastly, Knight demonstrated that client isolation in open and WPA2 networks is insecure [63], since a malicious insider can trivially inject packets directly to a client, but did not consider WPA3 or Enterprise networks.

Although existing Wi-Fi encryption protocols can protect traffic that is transmitted over the air, they do not protect how traffic is routed by internal switches or access points. Additionally, although enterprise authentication can verify a client's 802.1X identity such as its username, the identities at other layers of the stack, such as the MAC addresses, are not protected. It is precisely these two aspects that our attacks abuse, thereby bypassing all existing Wi-Fi cryptography.

Port Stealing Attack for Ethernet. Port stealing is a layer-2 attack against Ethernet switches [33], where the attacker sends frames with the victim's source MAC address and their own destination MAC address to exploit the switch's learning and filtering behavior. Once packets are intercepted, the attacker sends a broadcast ARP request to trigger the victim's ARP reply, restoring the victim's MAC-to-port binding. This allows the attacker to forward intercepted packets back to the victim. We demonstrated how novel variants of port stealing can be used to break client isolation in protected Wi-Fi networks, how our adaptations even work cross-BSSID and cross-AP in more complex networks, and how in some cases can even cause secret traffic to be broadcast in plaintext.

One of the attacks in [24] intercepted other clients' traffic in Enterprise Wi-Fi networks, but did not further study client isolation. Instead, its focus was on security context overriding attacks. Additionally, although they intercepted uplink traffic of clients, they did not consider attacks across BSSIDs or SSIDs, and could not return traffic back to the victim.

The original port stealing technique's use of ARP has led to its misclassification as ARP spoofing [64], prompting flawed defenses like Dynamic ARP Inspection (DAI) [65]. While DAI can block ARP-based port stealing, we exploit other network protocols and manipulate Wi-Fi port states to bypass client isolation. This is possible because manipulating Wi-Fi port states works at a lower layer than DAI and is more general.

X. Conclusions

In this paper, we developed attacks that bypass Wi-Fi client isolation. Via comprehensive experiments, including against two real-world Enterprise networks, we have shown that every tested network was vulnerable to at least one attack. We believe that a root cause of these vulnerabilities is the missing standardization of client isolation: this defense was added by vendors without proper public review. However, we have shown that client isolation is surprisingly tedious to get right in modern Wi-Fi networks due to their complexity. Moreover, we have shown that client isolation in home networks, which is often a configuration option in routers, is fundamentally flawed. We hope our work motivates standardization groups to more rigorously specify the requirements of client isolation and that Wi-Fi vendors will implement the same more securely.

Acknowledgments

This work is partially funded by the Research Fund KU Leuven, and by the Flemish Research Programme Cybersecurity. This research is also 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.

Ethics Considerations

For ethical reasons, all experiments were conducted exclusively using our own client devices. We did not target any real users, and the victim in each scenario was our own device. Moreover, we refrained from intercepting uplink traffic in university networks, as doing so could impact all clients associated with the targeted AP; we explicitly avoided capturing any data from other users. When testing the real-world university networks, we first obtained authorization to do so. We have disclosed the vulnerabilities to affected vendors, as well as the Wi-Fi Alliance. The Wi-Fi Alliance has acknowledged our findings, and we are awaiting their further action.

References

[1] TP-Link, "Brief introduction of ap isolation," https://www.tp-link.com/ch/support/faq/2089/, 2025, accessed 14 April 2025.

[2] M. Horowitz, "Router security checklist," https://www.routersecurity.org/checklist.php, 2025, accessed 14 April 2025.

[3] Micro Center, "Understanding AP isolation: A comprehensive guide to enhancing your Wi-Fi network security," https://www.microcenter.com/tech_center/article/6778/what-does-the-ap-isolation-setting-on-routers-do, 2025, accessed 14 April 2025.

[4] dd-wrt, "Advanced wireless settings - DD-WRT wiki," https://wiki.dd-wrt.com/wiki/index.php/Advanced_wireless_settings, 2025, accessed 14 April 2025.

[5] Cisco Meraki, "Wireless client isolation," https://documentation.meraki.com/MR/Firewall_and_Traffic_Shaping/Wireless_Client_Isolation, 2025, accessed 3 June 2025.

[6] B. Fleck and J. Dimov, “Wireless access points and ARP poisoning,” *Online document*, 2001.

[7] X. Feng, Q. Li, K. Sun, Y. Yang, and K. Xu, “Man-in-the-middle attacks without rogue AP: when WPAs meet ICMP redirects,” in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 3162–3177.

[8] F. Fietkau, “Re: [PATCH 1/2] cfg80211: add ap isolation support - Felix Fietkau,” https://lore.kernel.org/linux-wireless/4BD621FA.1000405@openwrt.org/.

[9] IEEE Std 802.11, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2024.

[10] S. Hebrok, S. Nachtigall, M. Maehren, N. Erinola, R. Merget, J. Somorovsky, and J. Schwenk, “We really need to talk about session tickets: A Large-Scale analysis of cryptographic dangers with TLS session tickets,” in *USENIX Security*. USENIX Association, Aug. 2023.

[11] N. Erinola, M. Maehren, R. Merget, J. Somorovsky, and J. Schwenk, “Exploring the unknown DTLS universe: Analysis of the DTLS server ecosystem on the internet,” in *USENIX Security*, 2023.

[12] P. Fiterau-Brostean, B. Jonsson, K. Sagonas, and F. Tåquist, “Automata-based automated detection of state machine bugs in protocol implementations.” in *NDSS*, 2023.

[13] W.-F. Alliance, *Passpoint Specification Ver. 3.3*, 2020.

[14] M. Vanhoef, P. Adhikari, and C. Pöpper, “Protecting Wi-Fi beacons from outsider forgeries,” in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 155–160.

[15] B. Aboba, “Virtual access points,” *IEEE 802.11-03/154r1*, 2003.

[16] M. H. Hue, J. Debnath, K. M. Leung, L. Li, M. Minaei, M. H. Mazhar, K. Xian, E. Hoque, O. Chowdhury, and S. Y. Chau, “All your credentials are belong to us: On insecure wpa2-enterprise configurations,” in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1100–1117. [Online]. Available: https://doi.org/10.1145/3460120.3484569

[17] M. Vanhoef and E. Ronen, “Dragonblood: Analyzing the dragonfly handshake of WPA3 and EAP-pwd,” in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 517–533.

[18] M. Vanhoef and F. Piessens, “Predicting, decrypting, and abusing WPA2/802.11 group keys,” in *USENIX security*, 2016.

[19] ——, “Key reinstallation attacks: Forcing nonce reuse in WPA2,” in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1313–1328.

[20] ——, “Release the Kraken: new KRACKs in the 802.11 standard,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 299–314.

[21] F. Tobagi and L. Kleinrock, “Packet switching in radio channels: Part ii - the hidden terminal problem in carrier sense multiple-access and the busy-tone solution,” *IEEE Transactions on Communications*, vol. 23, no. 12, pp. 1417–1433, 1975.

[22] TP-Link, “How to set up Device Isolation on a TP-Link Archer Router or Deco Mesh System,” 2025. [Online]. Available: https://www.tp-link.com/us/support/faq/3968/

[23] ASUS, “[Wireless Router] How to Set up AP Isolated feature?” 2025. [Online]. Available: https://www.asus.com/us/support/faq/1044821/

[24] D. Schepers, A. Ranganathan, and M. Vanhoef, “Framing frames: Bypassing Wi-Fi encryption by manipulating transmit queues,” in *USENIX Security*, 2023.

[25] M. Vanhoef, “Fragment and forge: breaking Wi-Fi through frame aggregation and fragmentation,” in *USENIX security*, 2021.

[26] Z. Wang, X. Feng, Q. Li, K. Sun, Y. Yang, M. Li, G. Du, K. Xu, and J. Wu, “Off-path TCP hijacking in wi-fi networks: A packet-size side channel attack,” in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/off-path-tcp-hijacking-in-wi-fi-networks-a-packet-size-side-channel-attack/

[27] M. Vanhoef and J. Robben, “A security analysis of WPA3-PK: Implementation and precomputation attacks,” in *International Conference on Applied Cryptography and Network Security*, 2024.

[28] Y. Yang, X. Feng, Q. Li, K. Sun, Z. Wang, and K. Xu, “Exploiting sequence number leakage: Tcp hijacking in nat-enabled wi-fi networks,” *arXiv preprint arXiv:2404.04601*, 2024.

[29] S. D. Nguyen, M. Mimura, and H. Tanaka, “Slow-port-exhaustion dos attack on virtual network using port address translation,” in *2018 Sixth International Symposium on Computing and Networking (CANDAR)*. IEEE, 2018, pp. 126–132.

[30] R. Moskowitz, “Weakness in Passphrase Choice in WPA Interface - Wi-Fi Networking News,” 2003. [Online]. Available: https://wifinetnews.com/archives/2003/11/weakness_in_passphrase_choice_in_wpa_interface.html

[31] D. Schepers, A. Ranganathan, and M. Vanhoef, “On the robustness of Wi-Fi deauthentication countermeasures,” in *Proceedings of the 15th ACM conference on security and privacy in wireless and mobile networks*, 2022, pp. 245–256.

[32] Wi-Fi Alliance, “WPA3 specification version 3.5,” https://www.wi-fi.org/file/wpa3-specification, Dec. 2025, accessed 14 April 2025.

[33] A. Ornaghi and M. Valleri, “Man in the middle attacks,” in *Blackhat Conference Europe*, vol. 1045, 2003.

[34] “Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements-part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications: Amendment 6: Medium access control (mac) security enhancements,” *IEEE Std 802.11i-2004*, pp. 1–190, 2004.

[35] A. Buhr, D. Lindskog, P. Zavarsky, and R. Ruhl, “Media access control address spoofing attacks against port security,” in *5th USENIX Workshop on Offensive Technologies (WOOT 11)*, 2011.

[36] S. Explorer, “drop_unicast_in_l2_multicast | sysctl-explorer.net.” [Online]. Available: https://sysctl-explorer.net/net/ipv4/drop_unicast_in_l2_multicast/

[37] O. Wiki, “[OpenWrt Wiki] Guest Wi-Fi using LuCI,” 2025. [Online]. Available: https://openwrt.org/docs/guide-user/network/wifi/guestwifi/configuration_webinterface

[38] D.-W. Wiki, “Guest Network - DD-WRT Wiki,” 2025. [Online]. Available: https://wiki.dd-wrt.com/wiki/index.php/Guest_Network

[39] L. Florio and K. Wierenga, “Eduroam, providing mobility for roaming users,” in *Proceedings of the EUNIS 2005 Conference, Manchester*, 2005.

[40] S. Goldberg, M. Haller, N. Heninger, M. Milano, D. Shumow, M. Stevens, and A. Suhl, “RADIUS/UDP considered harmful,” in *USENIX Security*, 2024.

[41] X. Jiao, W. Liu, M. Mehari, M. Aslam, and I. Moerman, “openwifi: a free and open-source ieee802.11 sdr implementation on soc,” in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020.

[42] Google, “Google transparency report: HTTPS encryption on the web,” https://transparencyreport.google.com/https/overview, accessed on August 5, 2025.

[43] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz, “Measuring HTTPS adoption on the web,” in *USENIX Security*. Vancouver, BC: USENIX Association, Aug. 2017. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/felt

[44] S. Patil and N. Borisov, “What can you learn from an ip?” in *Proceedings of the 2019 Applied Networking Research Workshop*, ser. ANRW ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 45–51. [Online]. Available: https://doi.org/10.1145/3340301.3341133

[45] B. Miller, L. Huang, A. D. Joseph, and J. D. Tygar, “I know why you went to the clinic: Risks and realization of HTTPS traffic analysis,” in *Privacy Enhancing Technologies: 14th International Symposium, PETS 2014, Amsterdam, The Netherlands, July 16-18, 2014. Proceedings 14*. Springer, 2014, pp. 143–163.

[46] E. Papadogiannaki and S. Ioannidis, “A survey on encrypted network traffic analysis applications, techniques, and countermeasures,” *ACM Comput. Surv.*, vol. 54, no. 6, Jul. 2021. [Online]. Available: https://doi.org/10.1145/3457904

[47] F. Alharbi, J. Chang, Y. Zhou, F. Qian, Z. Qian, and N. Abu-Ghazaleh, “Collaborative client-side DNS cache poisoning attack,” in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019.

[48] F. Alharbi, Y. Zhou, F. Qian, Z. Qian, and N. Abu-Ghazaleh, “DNS poisoning of operating system caches: Attacks and mitigations,” *IEEE Transactions on Dependable and Secure Computing*, 2022.

[49] Aruba Networks, “What does prohibit IP spoofing do and how do i enable it?” 2025, accessed on June 6, 2025. [Online]. Available: https://community.arubanetworks.com/community-home/librarydocuments/viewdocument?DocumentKey=a0d2aa96-24e8-400a-888d-6fa73a5feac0

[50] "IEEE standard for local and metropolitan area networks-media access control (MAC) security," *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, pp. 1–239, 2018.

[51] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile communications: The insecurity of 802.11," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, 2001, pp. 180–189.

[52] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16–17, 2001 Revised Papers 8*. Springer, 2001, pp. 1–24.

[53] A. Stubblefield, J. Ioannidis, A. D. Rubin *et al.*, "Using the fluhrer, mantin, and shamir attack to break WEP." in *NDSS*, 2002.

[54] C. He and J. C. Mitchell, "Analysis of the 802.11 i 4-way handshake," in *Proceedings of the 3rd ACM Workshop on Wireless Security*, 2004.

[55] C. He, M. Sundararajan, A. Datta, A. Derek, and J. C. Mitchell, "A modular correctness proof of IEEE 802.11i and TLS," in *Proceedings of the 12th ACM conference on Computer and communications security*, 2005, pp. 2–15.

[56] K. G. Paterson, B. Poettering, and J. C. Schuldt, "Plaintext recovery attacks against WPA/TKIP," in *Fast Software Encryption: 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers 21*. Springer, 2015, pp. 325–349.

[57] E. Tews and M. Beck, "Practical attacks against WEP and WPA," in *Proceedings of the second ACM conference on Wireless network security*, 2009, pp. 79–86.

[58] M. Vanhoef and F. Piessens, "All your biases belong to us: Breaking RC4 in WPA-TKIP and TLS," in *USENIX Security*, 2015, pp. 97–112.

[59] ——, "Practical verification of WPA-TKIP vulnerabilities," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, 2013, pp. 427–436.

[60] Wi-Fi Alliance, "WPA3 specification version 3.4," https://www.wi-fi.org/file/wpa3-specification, 2024, accessed on January 10, 2025.

[61] X. Zhou, Q. Deng, J. Pu, K. Man, Z. Qian, and S. V. Krishnamurthy, "Untangling the knot: Breaking access control in home wireless mesh networks," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 2072–2086.

[62] M. S. Ahmad, "Wpa too!" in *DEF CON*, 2010.

[63] B. Knight, "Bypassing wifi client isolation," Retrieved 15 December 2025 from https://pulsesecurity.co.nz/articles/bypassing-wifi-client-isolation, 2025.

[64] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE communications surveys & tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[65] A. Vázquez-Ingelmo, Á. Moreno-Montero, and F. J. García-Peñalvo, "Threats behind default configurations of network devices: wired local network attacks and their countermeasures," *Handbook of Computer Networks and Cyber Security: Principles and Paradigms*, 2020.

Appendix A
Artifact Appendix

This appendix describes how to evaluate the functionality of our scripts that were used to test devices for the following three vulnerabilities: (1) Gateway bouncing; (2) Port stealing and its attack variants; (3) Abusing GTK. We include scripts that set up a virtualised Wi-Fi environment, making it easier to verify the functionality of our code.

A. Description & Requirements

1) How to access: The code tested during artifact evaluation is available at https://doi.org/10.5281/zenodo.17905486 It contains scripts and explanations helpful to reproducing some Wi-Fi exploitation techniques in the paper. We recommend following the README.md in this linked archive, and also repeat all essential commands in this appendix.

2) Hardware dependencies: For evaluation, an x86-64 processor is recommended and we recommend at least 2 CPU cores and 4GB of RAM. No physical wireless NIC is required, as we use linux `mac80211_hwsim` to simulate wireless NICs. Our code was tested on Ubuntu 22.04.5 LTS.

3) Software dependencies: Please use a clean/new Ubuntu 22.04.5 installation for evaluation. `sudo` access is required. Software dependencies will be automatically installed.

4) Benchmarks: No data is needed to run the tests.

B. Artifact Installation & Configuration

1) Do only once: After a clean/new Ubuntu 22.04.5 installation, please run the following two commands with Internet on to install software dependencies.

```
$ sudo apt update
$ sudo apt install libnl-3-dev \
    libnl-genl-3-dev libnl-route-3-dev \
    libssl-dev libdbus-1-dev pkg-config \
    build-essential git python3-venv \
    aircrack-ng rfkill net-tools dnsmasq \
    tcpreplay macchanger
```

Select "No" for the question "Change MAC automatically?" during `macchanger` installation. After these, launch a terminal window *A* and clone the repository. Then enter the project's directory, and with an Internet on, run:

```
./setup.sh
```

With Internet on, launch another terminal window *B*, and also `cd` into the project folder, and run:

```
$ cd macstealer/research
$ ./build.sh && ./pysetup.sh
```

2) Repeatable: Do these for every claim in the paper.

In terminal window *B*, confirm you are always in the folder `macstealer/research`, and run:

```
$ sudo su
$ source venv/bin/activate
```

Go back to terminal window *A* and execute:

```
$ cd setup
$ sudo su
$ source venv/bin/activate
```

C. Experiment Workflow

For three major claims below, we respectively: (1) create a simulated networking environment; and (2) run the test case and inspect the screen output to prove the effectiveness of exploitation techniques.

D. Major Claims

The main claims of our paper are:

- (C1): Gateway Bouncing can inject Wi-Fi traffic in vulnerable setups regardless of encryption protocols used, including WEP, TKIP, CCMP and GCMP.
- (C2): Port Stealing can intercept Wi-Fi traffic in vulnerable setups regardless of encryption protocols used, including WEP, TKIP, CCMP and GCMP.

- (C3): Abusing GTK can inject Wi-Fi traffic in vulnerable setups regardless of encryption protocols used. As GTK is a WPA feature, TKIP, CCMP and GCMP are vulnerable to Abusing GTK.

E. Evaluation

1) Experiment (E1): [Gateway Bouncing] [30 human-minutes + 0 compute-hour]: This experiment simulates 2 Wi-Fi APs `wlan0` (using WPA3-Personal), `wlan1` (using WPA2-Personal), and lets a victim client connect to `wlan0` AP, and lets the attacker connect to `wlan1` AP. Both APs have `ap_isolate=1` on. An attacker then uses Gateway Bouncing technique in `macstealer.py` to inject a UDP packet to the victim, bypassing client isolation.

[How to] We will use window *A* to launch vulnerable networking setups and use window *B* to connect victim and attacker to `wlan0` and `wlan1` respectively and perform Gateway Bouncing.

[Preparation] In terminal *A*, perform "Repeatable" actions in Section C, and run:

```
$ ./setup-br0-gwbounce.sh
```

In terminal *B*, perform "Repeatable" actions in Section C.

[Execution] In terminal *B*, run a **one-liner**:

```
$ python3 macstealer.py wlan2 --c2c-ip \
 wlan3 --other-bss --no-ssid-check \
 --config \
 client-simulated-AE-gatewaybouncing.conf
```

[Results] If you see at least one line of the following log, this test case passes:

```
>>> Client to client traffic at IP layer
    is allowed (PSK{passphrase_atkr}
    to SAE{passphrase_victim}).
```

Press Ctrl-C in terminal *B* to end this test case.

2) Experiment (E2): [Port Stealing] [30 human-minutes + 0 compute-hour]: This experiment simulates 2 Wi-Fi APs `wlan0` (using WPA3-Personal), `wlan1` (using WPA2-Personal), and lets a victim client connect to `wlan0` AP, and lets the attacker connect to `wlan1` AP. Both APs have `ap_isolate=1` on. An attacker then uses Port Stealing technique in `macstealer.py` to intercept downstream packets sent from the gateway/router to the victim, bypassing client isolation.

[How to] We will use window *A* to launch vulnerable networking setups and use window *B* to connect victim and attacker to `wlan0` and `wlan1` respectively and perform Port Stealing.

[Preparation] In terminal *A*, press Ctrl-C to exit any experiment that is possibly still running. Only perform "Repeatable" actions in Section C again if you accidentally shut terminal *A* off, and then run:

```
$ ./setup-br0-portsteal.sh
```

In terminal *B*, remember to perform "Repeatable" actions in Section C if you have not done so.

[Execution] In terminal *B*, run a **one-liner**:

```
$ python3 macstealer.py wlan2 \
   --c2c-port-steal wlan3 \
   --other-bss --no-ssid-check --config \
   client-simulated-AE-portsteal.conf \
   --server 192.168.100.1
```

[Results] If you see at least one line of the following log, this test case passes:

```
>>> Downlink port stealing is successful.
```

Press Ctrl-C in terminal *B* to end this test case.

3) Experiment (E3): [Abusing GTK] [30 human-minutes + 0 compute-hour]: This experiment simulates 2 Wi-Fi APs `wlan0` (using WPA3-Personal), `wlan1` (using WPA2-Personal), and lets a victim client connect to `wlan0` AP, and lets the attacker connect to the same `wlan0` AP. Both APs `wlan0` and `wlan1` have `ap_isolate=1` on. An attacker then uses Abusing GTK technique in `macstealer.py` to inject frames directly to the victim, bypassing client isolation. After that, we let a victim client connect to `wlan1` AP, and lets the attacker connect to the same `wlan1` AP and perform Abusing GTK again, to prove the claim C3.

[How to] We use window *A* to launch vulnerable networking setups and use window *B* to connect victim and attacker to `wlan0` and `wlan1` in turn and perform Abusing GTK.

[Preparation] In terminal *A*, perform "Repeatable" actions in Section C, and run:

```
$ press./setup-br0-gtkabuse.sh
```

In terminal *B*, perform "Repeatable" actions in Section C.

[Execution] In terminal *B*, run:

```
$ python3 macstealer.py wlan2 \
   --c2c-gtk-inject wlan3 --other-bss \
   --no-ssid-check --config \
   client-simulated-AE-gtkabuse.conf \
   --no-id-check --c2m-mon-channel 6
```

[Results] If you see at least one line of the following log, this sub-test-case passes:

```
>>> GTK wrapping ICMP ping is allowed
    (SAE{passphrase_victim} to
    SAE{passphrase_victim}).
```

Press Ctrl-C in terminal *B* to end this sub-test-case.

[Execution] In terminal *B*, run:

```
$ python3 macstealer.py wlan2 \
   --c2c-gtk-inject wlan3 --other-bss \
   --no-ssid-check --config \
   client-simulated-AE-gtkabuse2.conf \
   --no-id-check --c2m-mon-channel 1
```

[Results] If you see at least one line of the following log, this sub-test-case passes:

```
>>> GTK wrapping ICMP ping is allowed
    (PSK{passphrase_atkr} to
    PSK{passphrase_atkr}).
```

Press Ctrl-C in terminal *B* to end this test case.