Capturing Antique Browsers in Modern Devices: A Security Analysis of Captive Portal Mini-Browsers

Ping-Lun Wang[1], Kai-Hsiang Chou[2], Shou-Ching Hsiao[2], Ann Tene Low[2], Tiffany Hyun-Jin Kim[3], and Hsu-Chun Hsiao[2,4]

[1] Carnegie Mellon University, Pittsburgh PA, USA**
pinglunw@andrew.cmu.edu
[2] National Taiwan University, Taipei, Taiwan
{b07705022,d10922007,r09944020}@ntu.edu.tw, hchsiao@csie.ntu.edu.tw
[3] HRL Laboratories, Malibu CA, USA
hjkim@hrl.com
[4] Academia Sinica, Taipei, Taiwan

Abstract. Granting access to public Wi-Fi networks heavily relies on captive portals that are accessible using dedicated browsers. This paper highlights that such browsers are crucial to captive portals' security, yet have not been emphasized in prior research. To evaluate the security of *captive portal mini-browsers*, we built an assessment tool called *Wi-Fi Chameleon* and evaluated them on 15 popular devices. Our evaluation revealed that they all lacked the essential security mechanisms provided by modern browsers. Indeed, many provided no warnings even when using HTTP or encountering invalid TLS certificates, and some did not isolate sessions, enabling attackers to silently steal users' sensitive information (e.g., social networking accounts and credit card numbers) typed in captive portals and stored in their browsing histories. Moreover, even if a captive portal mini-browser is equipped with all security protections that modern browsers provide, users are still susceptible to existing captive portal attacks. We discuss the best practice of a secure captive portal mini-browser and two possible approaches to mitigate the vulnerabilities. For end-users, we proposed a browser extension for immediate deployability. For access points and captive portal mini-browser vendors, we proposed a comprehensive solution compatible with RFC 8952, the standard of captive portals.

Keywords: captive portal mini-browser, Wi-Fi, security assessment

1 Introduction

A *captive portal* is a network that initially blocks a user from accessing the Internet until s/he fulfills a customized access condition [24], such as submitting a login credential, agreeing to terms of service, providing payment information,

** This work was done while the first author was at National Taiwan University.

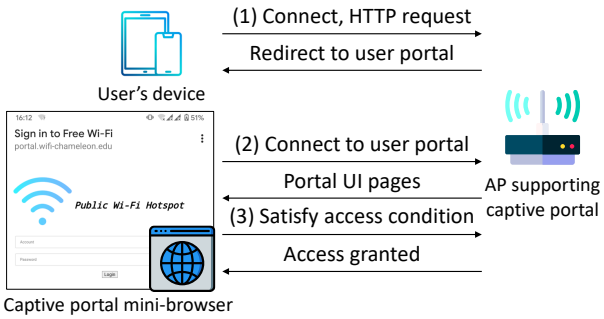


Fig. 1: Standard interaction flows among a user, a captive portal mini-browser, and an Access Point (AP) supporting captive portal. Upon satisfying all the interaction requirements, the user can access the Internet using the Wi-Fi.

or watching a video. A captive portal usually comes with a *user portal*: a web server that instructs the user to fulfill the access condition. In addition, many operating systems (OS) have built-in miniature browsers dedicated to handle user portals, which we refer to as *captive portal mini-browsers* or *user portal browsers* in this paper. Figure 1 provides an example captive portal mini-browser and its interactions with a captive portal.

Because of their customizable access conditions and user-friendly web interfaces, captive portals are widely used in public Wi-Fi networks, allowing people to connect to the Internet even without prior knowledge of the public Wi-Fi, such as a pre-shared key. However, captive portals introduce additional attack vectors to the Wi-Fi ecosystem. For instance, prior research has shown that malicious users can escape flawed captive portals via DNS tunneling or ARP spoofing [13, 33]. Furthermore, malicious Wi-Fi operators can exploit erroneous captive portal mini-browser implementations to control the device [12] or trick users into connecting to rogue captive portals to steal their sensitive information [16, 19, 23, 28].

This paper highlights that the captive portal mini-browsers are crucial to the security of captive portals: they are the last line of defense for users who may accidentally connect to a malicious captive portal, and such browsers' weaknesses may lead to adversaries evading detection. Compared to regular browsers (e.g., Chrome and Safari) that are commonly used to surf the Internet, however, captive portal mini-browsers are under-studied [3].

Our goal of this paper is threefold: (1) Assess the overall security strength of captive portal mini-browsers, each of which is hosted on a different platform/device; (2) Assess their security strengths against two well-known attacks (i.e., evil-twin and history-stealing attacks); and (3) Discuss two possible approaches to protect users from the attacks. For (1), we developed a purpose-built assessment tool called Wi-Fi Chameleon that creates a testing captive portal to perform several security checks on a captive portal mini-browser, and used Wi-Fi Chameleon to investigate the security of captive portal mini-browsers

on 15 types of popular laptops and mobile devices. Our investigation revealed that many captive portal mini-browsers lacked essential security mechanisms provided by modern browsers and did not follow best practices for displaying security warnings. Specifically, 14 out of the 15 sampled devices did not provide any security warnings for HTTP, and none of them issued a security warning for deprecated TLS protocols (e.g., TLS 1.0 and 1.1). Thus, attackers can take advantage of weak captive portal mini-browser implementations to silently downgrade or disable security protection. We also revealed a severe security vulnerability of Xiaomi devices: their captive portal mini-browsers did not validate SSL/TLS certificates, allowing an attacker to create a plausible user portal with a self-signed certificate.[5]

For (2), we evaluated captive portal mini-browsers against two captive portal attacks. First, evil-twin attacks [19, 23, 28] create a phishing user portal to steal users' login credentials and personal information. Second, history-stealing attacks [16] steal users' browsing histories by capturing the browsing data stored in a captive portal mini-browser, thereby inferring the captive portals connected by a user and the user's location history. Our evaluation results show that all 15 types of popular laptops and mobile devices are vulnerable to both evil-twin and history-stealing attacks due to the captive portal mini-browser's weaknesses.

For (3), we discussed possible approaches to the defense. Although evil-twin and history-stealing attacks are well-known attacks against any Wi-Fi access points (AP), they are more difficult to mitigate when applied to captive portals because of captive portals' unique characteristics as follows:

- **Limited Internet access before satisfying the access condition.** A device in a captive state has limited Internet access; for example, it may be able to access the Wi-Fi operator's official website only. When a user requests a user portal Universal Resource Identifier (URI), s/he needs to validate its authenticity without relying on online security checks. Thus, defense schemes may not work as expected if they simply adopt common approaches for authenticating web URIs, such as digital certificates or safe browsing check [18], both of which require network access.
- **Diverse captive portal implementations.** Users choose a Service Set IDentifier (SSID) and interact with a user portal to connect to a Wi-Fi AP. While the SSID mechanism is defined in IEEE 802.11, no standard exists for captive portals until November 2020 [24], resulting in diverse captive portal implementations. For example, some Wi-Fi operators use HTTP as their user portals' connection protocol, and some, HTTPS, while others may even use a self-signed TLS certificate for HTTPS communication to reduce the cost of purchasing a valid certificate. Unfortunately, this allows the attacker to downgrade or modify the connection protocol while maintaining the Wi-Fi AP's validity. Designing a secure captive portal mini-browser is thus

[5] We reported this issue to Xiaomi. They initially misunderstood it as a Wi-Fi router's issue. We provided additional information about this security vulnerability, but we have not received any response yet.

more challenging as it needs to prevent downgrade attacks while maintaining backward compatibility at the same time.

We proposed two possible approaches. The first approach is to secure access to user portals even when Internet access is limited using a browser extension, which is easier to deploy but only serves as a temporary fix. The second approach is verifying the AP's identity, which is more comprehensive but requires changes on the APs and the captive portal mini-browsers.

This paper makes the following contributions:

- To evaluate the security of captive portal mini-browsers, we developed an assessment tool called Wi-Fi Chameleon, which exposed several security flaws affecting 15 popular laptops and mobile devices that we assessed.
- We evaluated captive portal mini-browsers against well-known evil-twin and history-stealing attacks, and showed that the browsers do not defend against all attacks regardless of security protections on them.
- To mitigate these attacks, we proposed a temporary fix using a browser extension that transforms a regular browser into a secure captive portal mini-browser, and a more comprehensive fix by verifying the AP's identity.

2 Problem Definition

Unlike regular browsers, captive portal mini-browsers serve the unique purpose of granting access to public Wi-Fi networks. Consequently, captive portal mini-browsers require security mechanisms that may be unique to them. For example, a captive portal mini-browser needs to detect phishing attacks and downgrade attacks when online security checks are unavailable and backward compatibility is maintained. Before we lay out the desired security properties and the attacker models for captive portal mini-browsers, we briefly introduce captive portals and captive portal mini-browsers.

2.1 Background

Captive portals. Captive portals usually block users from accessing the Internet until certain conditions are fulfilled, and a user must perform the following three procedures to use a captive portal, which we also illustrate in Figure 1.

1. **Detect captive state:** When a user's device connects to a Wi-Fi AP, it first sends an HTTP request. If a captive portal exists, it sends a redirect response containing the user portal URI to the user's device.
2. **Connect to user portal:** Next, the user's device launches its captive portal mini-browser and navigates to the user portal, which is a web server providing a User Interface (UI) to assist users for granting access.
3. **Satisfy access condition:** The user then interacts with the user portal and gains the Internet access if s/he fulfills the conditions.

Captive Portal Mini-browsers. Most portable devices provide a captive portal mini-browser, a minimal web browser designed solely to facilitate their interaction with user portals. When a device detects the existence of a captive portal, it pops up the captive portal mini-browser and navigates to the user portal. Some Android and Windows devices do not provide captive portal mini-browsers; instead, they use their default browsers to navigate to user portals.

2.2 Attacker Model

Evil-twin and history-stealing attacks [10, 16, 23, 29] are two well-known attacks on captive portal mini-browsers. We briefly explain how these attacks work, with an emphasis on how they can exploit the weaknesses of captive portal mini-browsers.

Evil-twin attacks. Evil-twin attacks lure users to connect to a fake Wi-Fi AP whose SSID is the same as that of an existing Wi-Fi AP. Once users connect to the fake AP, the attacker can perform social-engineering attacks or Man-in-the-Middle (MitM) attacks.

When mimicking an AP that uses a captive portal, existing evil-twin attack tools usually create a user portal that is cloned from a real captive portal to steal users' login credentials [21,31,37]. While the content of the user portal web page can be cloned easily, the evil twin cannot use the same URI if it is protected by TLS. However, attackers can perform the following attacks.

- **SSL stripping to downgrade connection:** An attacker can perform an SSL stripping attack, changing the connection from HTTPS to HTTP, in order to create an evil twin with the same hostname as the real AP's user portal.
- **Redirect to a similar URI:** An attacker can create an evil twin's URI using a malicious domain whose name is similar to the real user portal's.
- **Fake certificate for the same URI:** An attacker can use a fake certificate so that the evil twin uses the same URI as the real user portal. The fake certificate can be either a self-signed certificate or a revoked certificate of the real user portal.

History-stealing attacks. A history-stealing attack tries to steal a user's browsing history by collecting browsing data such as HSTS records and cookies [1]. Dabrowski *et al.* [16] have shown that a malicious Wi-Fi operator can utilize a captive portal to perform a history-stealing attack. In particular, two attack entry points exist for history-stealing attacks using malicious captive portals: (1) captive portal mini-browser, and (2) regular browser.

- **Captive portal mini-browser:** An attacker can construct a malicious captive portal to steal browsing history inside captive portal mini-browsers. Although captive portal mini-browsers are used only for captive portals, the

attacker can retrieve the list of previously-connected captive portals, and thus infer the locations, from the captive portal mini-browser's browsing history.
- **Regular browser:** An attacker can construct a malicious captive portal that first performs the method proposed by Dabrowski *et al.* [16] to prevent the designated captive portal mini-browser from popping up and lures the user to open the malicious user portal using a browser s/he normally uses. The attacker can then steal the browser history from this browser.

2.3 Desired Properties

A secure captive portal mini-browser should provide the following security properties. Properties (1)–(4) are commonly provided by regular browsers, and Properties (4)–(5) ensure that history-stealing and evil-twin attacks are protected.

1. **Connection protocol needs to be secure:** When an insecure protocol is used, such as HTTP, warnings should be provided, and connections should be blocked when downgrade attacks are detected.
2. **Certificates need to be validated:** Using TLS certificates can help eliminate MitM attacks, based on the assumption that browsers validate these certificates, and warn users or refuse to connect given invalid certificates.
3. **Sensitive information needs to be protected:** Common use cases for captive portals include requesting login credentials and credit card numbers. Captive portal mini-browsers should warn about the risk of eavesdropping when private information is transmitted using insecure protocols.
4. **User data needs to be protected and isolated:** Cookies and local storage allow a website to save the user data in the client browser. Hence, improperly-implemented cookies and local storage can cause serious privacy issues. Captive portal mini-browsers should provide protections and use an isolated environment to defend against history-stealing attacks.
5. **User portal URI needs to be authenticated:** Captive portal mini-browsers should be able to determine the correctness of a user portal URI even when they are offline so that evil-twin attacks can be protected.

3 Assessment Tool: Wi-Fi Chameleon

To assess the security protections captive portal mini-browsers provide, we developed a tool called Wi-Fi Chameleon that can be considered a universal captive portal. Unlike the regular captive portals that request login credentials, Wi-Fi Chameleon provides capabilities to record, filter, or modify the network traffic between the captive portal mini-browser and the testing website by (1) launching attacks as mentioned in Section 2.2 to assess the vulnerabilities and (2) verifying security properties as delineated in Section 2.3.

Wi-Fi Chameleon allows users to test whether their captive portal mini-browsers satisfy testing criteria, as shown in Figure 2. To evaluate a security

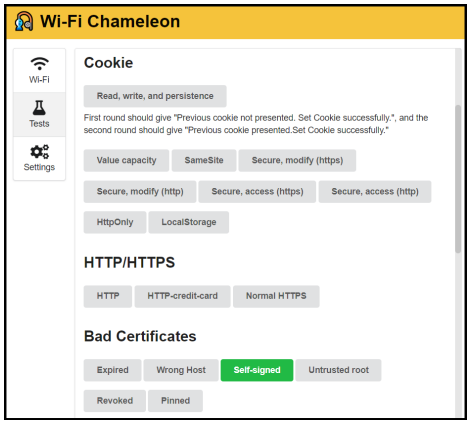


Fig. 2: Wi-Fi Chameleon's user interface (UI) for selecting a testing criterion, and it will perform the indicated security check using a testing site.

metric associated with the testing criteria, such as the protection against self-signed certificates, Wi-Fi Chameleon selects the corresponding testing website as the user portal. Wi-Fi Chameleon relies on several testing websites to perform security checks, and some of them are borrowed from existing browser testing websites, including the web-platform-tests Project by W3C [38], BadSSL [8], SSL Client Test by Qualys SSL Labs [22], and Mixed Content Examples by Northwoods [30].

However, Wi-Fi Chameleon cannot test all criteria simultaneously, as some of the tests are not compatible to each other, and some require users to visually confirm. Also, since we want to check what security warnings are provided for each security vulnerability, such as the warning messages for invalid certificates and insecure cipher suites, these tests cannot be performed simultaneously.

When a device connects to Wi-Fi Chameleon, which serves as an AP, it performs the first procedure in Figure 1 to connect to a captive portal, but now the user portal is replaced by a testing website, and Wi-Fi Chameleon acts as a relay between them. The device then deploys its captive portal mini-browser to open the testing website[6], which examines the captive portal mini-browser and displays the result on the web page, as illustrated in Figure 3.

Implementation. We implemented Wi-Fi Chameleon on Raspberry Pi 4, a low-cost device with low power consumption. We use the built-in wireless interface to create a captive portal and use an additional USB Wi-Fi dongle for Internet connection. Wi-Fi Chameleon can be installed on any hardware that is running the Linux OS with a wireless interface in AP mode and an additional network

[6] Devices without a designated captive portal mini-browser use their default browsers.

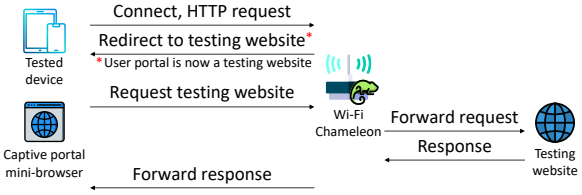


Fig. 3: The testing process of a captive portal mini-browser. Instead of satisfying the access conditions, the captive portal mini-browser now performs security checks provided by the testing website.

interface (e.g., Ethernet) for Internet connection. Note that most commodity laptops satisfy this requirement.[7]

4 Security Strength Evaluation

Compared to regular browsers, captive portal mini-browsers only need to provide minimal functionality to allow a user to access the user portal. However, the user may connect to an untrusted captive portal, in which case the captive portal mini-browser should be robust enough to defend against potential attacks.

In this section, we describe our evaluation methodology using Wi-Fi Chameleon. Since different device platforms host different captive portal mini-browsers, we selected 18 widely-adapted devices for our evaluation, and evaluated them using security metrics that represent the desired properties as described in Section 2.3.

4.1 Tested Devices

The 18 portable devices to be tested with Wi-Fi Chameleon included smartphones, tablets, laptops, and a game console as shown in Table 3 in Appendix A.1. We chose portable devices since they are more likely to be used to connect to captive portals, which are usually seen in public places, and multiple devices from the same manufacturer (e.g., Samsung) since different OS versions may support different captive portal mini-browsers. We ensured that the selected devices take more than 85% of market shares or have been sold for more than 89 million units [35, 36, 39].

Our evaluation revealed that some devices exhibited almost identical behaviors in response to captive portals. Hence, we categorized them into groups, mostly according to the OS. Although Xiaomi devices are Android-based, they have a customized captive portal mini-browser that behaves quite differently from other Android devices'. Grouping by manufacturers and browser behaviors, we categorized 18 devices into 5 groups: 1) Apple devices, 2) Android devices, 3)

[7] The Wi-Fi Chameleon code is available here (https://github.com/csienslab/Wi-Fi-Chameleon).

Nintendo Switch, 4) Xiaomi devices, and 5) devices that lack captive portal mini-browsers, which instead use their default browsers for the same purpose. Since the devices from the fifth group do not contain captive portal mini-browsers, they were not evaluated in our experiments.

4.2 Evaluation Metrics

To understand the security gap between captive portal mini-browsers and regular browsers, we adopt several metrics commonly used by regular browsers. Next, we introduce these metrics, highlight their relevance with captive portals, and compare their differences from regular browsers.

Connection protocol indicator. Captive portal mini-browser should clearly indicate the connection protocol, and provide warnings before any insecure interaction commences. Both the use of insecure cipher suites and unsafe protocols can be potential source of harm. We tested the former using a 1024-bit Diffie-Hellman key exchange, and the latter with SSL 2 and 3, and TLS 1.0 and 1.1. Note that most regular browsers would warn their users about weak encryption or refuse to connect.

Even if the initial connection is encrypted via HTTPS, resources within the web page may be requested using HTTP. Hence, browsers should block these insecure requests, especially if the content is JavaScript or an iFrame page. Wi-Fi Chameleon checks whether such mixed content is blocked or warned.

Another technique that protects users from using unsafe connections is HSTS. The HSTS protocol informs the browser that a certain website should only be accessed using HTTPS, and the browsers should redirect their users to HTTPS if a website uses HTTP. While regular browsers usually preserve HSTS records, they may leak browsing history to a malicious user portal. Therefore, we include a metric that requires HSTS to be enabled during the same session, but its records should be discarded once the session ends. Although discarding HSTS records increases the risk of SSL stripping attacks, we believe this is a reasonable trade-off between users' security and privacy because further measures can be taken to prevent SSL stripping attacks, such as the secure captive portal mini-browser extension we propose in Section 7.

Certificate validation. We check whether the captive portal mini-browsers warn the user, or refuse to continue when they encounter TLS certificates with any of the following issues: (1) Certificate is expired; (2) Certificate has a mismatched common name; (3) Certificate is self-signed; (4) Certificate is signed by an untrusted root certificate authority (CA); (5) Certificate is revoked.

Sensitive information. Previous work [1] indicates that some captive portals require users to provide their social networking accounts (e.g., Google and Facebook), which can be highly valuable to attackers. We therefore check whether warnings are provided when sensitive information is transmitted via HTTP.

Cookies and local storage. Two common ways to protect cookies from being stolen are Secure and HttpOnly flags. The former means that cookies should only be sent when HTTPS is being used, while the latter prohibits JavaScript from accessing them. We assess if these two functions are properly implemented.

Another goal of testing cookies and local storage is to determine whether the user portal is visited in a sandbox environment, i.e., a completely new session. Although regular browsers preserve cookies and local storage data, there is a risk of leaking users' browsing history. Wi-Fi Chameleon browses the testing website twice and checks if cookies or local storage persists between the two visits, in which case the session is not isolated.

URI authentication. Strip and redirect attackers modify the user portal URI to create an evil twin. Therefore, Wi-Fi Chameleon checks if the evaluated browser detects this modification.

4.3 Evaluation Results

Table 1 summarizes our findings. We also included a group of regular browsers for comparison.

Connection protocol indicator. We found that no captive portal mini-browsers warned their users about insecure HTTP connections in advance, and most did not clearly show the string HTTP or any other obvious indicators that would have allowed their users to understand the risk. Even worse, the captive portal mini-browsers provided by Xiaomi devices do not display the URI at all. Figure 4 shows how user portal URI is displayed by an Apple device (4a and 4b) and an Android device (4c and 4d). As we can see, neither of them indicate the use of HTTP protocol when the captive portal mini-browser connects to a user portal using HTTP. They should instead provide more indication (such as showing"http" in the URI) when an insecure protocol is used.

(a) MacBook Air, HTTP (b) MacBook Air, HTTPS (c) Pixel 3a, HTTP (d) Pixel 3a, HTTPS

Fig. 4: User portal URIs displayed by MacBook Air and Pixel 3a

The only captive portal mini-browser that indicated a security risk was provided by Nintendo Switch. However, its warning message did not display automatically; the user must press the '+' button to check the warning. Figure 6 in Appendix A.2 shows the warning message provided by Nintendo Switch. While showing warning messages to the user provides situational awareness, users are unlikely to click a button to learn about the security warning. We suggest to

Table 1: Inspection result of captive portal mini-browsers. Regular refers to regular browsers. Unfortunately, no inspected captive portal mini-browsers satisfy all metrics.

		Apple	**Android**	**Nintendo**	**Xiaomi**	**Regular***
Connection Protocol Indicator	HTTP warning	✘	✘	▲[a]	✘	✓
	HTTP indication	✘	✘	▲[a]	✘	✓
	HTTPS indication	✓	✓	▲[a]	✘	✓
	HSTS enabled [8]	✓	✓	✘	✘	✓
	DH1024 disabled [8]	✓	✓	✓	✓	✓
	SSL 2 disabled [22]	✓	✓	✓	✓	✓
	SSL 3 disabled [22]	✓	✓	✓	✓	✓
	TLS 1.0 warning [8]	✘	✘	✘	✘	✓
	TLS 1.1 warning [8]	✘	✘	✘	✘	✓
	Mixed content (JavaScript & iFrame) is blocked [30]	✓	✓	✓	✓	✓
Certificate Validation	Expired certificate warning [8]	✓	✓	✓	✘	✓
	Common name mismatch warning [8]	✓	✓	✓	✘	✓
	Self-signed certificate warning [8]	✓	✓	✓	✘	✓
	Untrusted root CA warning [8]	✓	✓	✓	✘	✓
	Revoked certificate warning [8]	✘	✘	✘	✘	▲[b]
Sensitive Information	Password input warning in HTTP [8]	✘	✘	✘	✘	✓
	Credit card input warning in HTTP [8]	✘	✘	✘	✘	✓
Cookie and Local Storage	Secure cookie enabled [38]	✓	✓	✓	✓	✓
	HttpOnly enabled [38]	✓	✓	✓	✓	✓
	Cookie is not persistent	✓	✘	✓	✘	✓
	Local storage is not persistent	✓	✘	✓	✘	✓
URI Auth.	Detect evil-twin attacks	✘	✘	✘	✘	✘

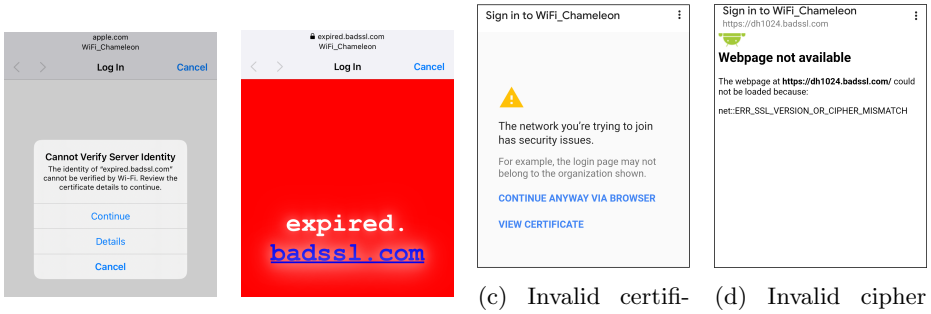
* Chrome browser, desktop version in incognito mode. Safari browser on iOS and Chrome browser on Android also have similar behaviors.

[a] Users can press the '+' button to check the URI and the security warning.

[b] Chrome browser may use a cached certificate-revocation list to check whether a particular certificate has been revoked, but such a list may be outdated.

proactively warn users and help them understand the risks before any interactions begin.

Regarding deprecated protocols, we found that no captive portal mini-browsers issued any warning about the usage of TLS version 1.0 or 1.1. Unfortunately, such a lack of a warning could lead users to erroneously believe that their current connections are secure. With regard to weak cipher suites, all the tested captive portal mini-browsers refused to connect when 1024-bit prime numbers are used for Diffie-Hellman key-exchange. Most of them showed error messages (e.g., "cipher mismatch") and refused to render the web page, while Apple devices refused to deploy their captive portal mini-browsers. Figure 5d illustrates that when Android devices encountered weak cipher suite for TLS, they do not render the web page and the user portal cannot be connected.



(a) Invalid certificate warning, iOS

(b) Invalid certificate warning, iOS

(c) Invalid certificate warning, Android 11

(d) Invalid cipher suite warning, Android 11

Fig. 5: The warning messages provided by captive portal mini-browsers

As for HSTS, the captive portal mini-browsers provided by Nintendo Switch and Xiaomi devices do not enable HSTS, while others enable HSTS during the same session, and the HSTS records are discarded when the session ends.

Certificate validation. Almost all captive portal mini-browsers (except those on Xiaomi) warn their users when certificates are invalid. The most common technique is to pop up a warning and let the user decide whether or not to continue. If the user decides to ignore the warning, most regular browsers indicate that the certificate is erroneous. However, Apple devices still display a lock sign even when warnings are ignored. As shown in Figure 5a and 5b, the captive portal mini-browser provided by an Apple device still displays that the user portal is protected by HTTPS, which is indicated by the lock icon in front of the URI. As this may confuse the user, Apple devices' captive portal mini-browsers should instead indicate that the user portal is not secure when an invalid certificate is encountered.

The captive portal mini-browser provided by Xiaomi devices neither validates certificates nor provides any warning messages about invalid certificates.

We carefully inspected their captive portal mini-browser and discovered that it ignores TLS errors on purpose. Unfortunately, such an action has an undesirable consequence of degrading security drastically.

Figure 5c shows the warning messages provided by an Android device for an invalid certificate. We found that captive portal mini-browsers provided by Android devices provide the option to use a regular browser to connect to the user portal given an invalid certificate, possibly because a regular browser provides more details about such certificates. However, this compromises the independence of separate connection sessions and could lead to leakage of users' private information, as explained in Section 4.2.

None of the tested captive portal mini-browsers informed their users of revoked certificates, while the regular browser uses a revocation list and provides a warning.

Sensitive information. None of the captive portal mini-browsers on the tested devices displayed a warning when the visited web page was using HTTP and contained a credit card or password input field. Such a lack of warning could lead to leaking credit card numbers and passwords.

Cookies and local storage. All the examined captive portal mini-browsers implemented HttpOnly and secure cookies correctly, but we found that Android and Xiaomi devices did not execute their captive portal mini-browsers in a sandboxed environment. These captive portal mini-browsers kept cookies and local storage data across different sessions, which can risk users' privacy.

URI authentication. Unfortunately, none of the examined captive portal mini-browsers could detect URI modification.

5 Attack Vulnerability Analysis

Our evaluation results in Section 4 revealed that most captive portal mini-browsers lack security mechanisms provided by regular browsers. In this section, we further evaluate them with the five attacker models as defined in Section 2.2.

We assume that a user does not notice any differences when an attacker slightly modifies the URI of the authentic user portal, which is difficult to identify (e.g., *wifi.chameleon.edu* vs. *wifi.chame1eon.edu*). Also, the URI of the authentic user portal is unknown for captive portals if they never connected to it before. In addition, we conservatively assume that the user cautiously stops using a captive portal if the captive portal mini-browser indicates any warning messages.

We discovered that all captive portal mini-browsers are vulnerable to some of the attackers. Moreover, even if users access a user portal using regular browsers, they are still susceptible to some of the above mentioned attacks. Table 2 summarizes the susceptibility results of the tested devices.

Table 2: Status of tested devices against attacker models. We use ▲ to denote possible false positives or vulnerabilities under some attack scenarios. Note that none of them can defend against all attacks.

	Evil-twin			**History-stealing**	
	Strip	Redirect	Fake	User portal	Regular
Regular[1]	▲	✘	✓	✓	N/A
Apple	✘	✘	▲	✓	▲
Android	✘	✘	▲	✘	✓
Nintendo	▲	✘	▲	✓	N/A
Xiaomi	✘	✘	✘	✘	✓

[1] Chrome, desktop version in incognito mode.

SSL stripping. Since the regular browser and the captive portal mini-browser provided by Nintendo Switch provide warning messages for HTTP connections, they can defend against SSL stripping attacks. Using HTTP warnings can defend against this attack, but it may create false positives because some captive portals still use HTTP to access their user portals. Since there is a risk of affecting the usability in practice, we mark them as ▲.

The captive portal mini-browsers on the Apple and Android devices do not warn HTTP connections. Hence, users cannot detect SSL stripping attacks. Xiaomi devices' captive portal mini-browsers do not show the URI; hence, users cannot detect the attack.

Redirect. All the devices are vulnerable because the evil twin triggers no warning messages and the user does not know the correct URI for the user portal.

Fake certificate. When this attack uses a self-signed certificate for the evil twin's user portal, only Xiaomi devices are vulnerable. However, if the attacker possesses a revoked certificate of the real user portal, none of the captive portal mini-browsers can defend against this attack. On the other hand, the regular browser can detect this attack given a cached and up-to-date certificate revocation list.

History-stealing attack on captive portal mini-browsers. Our evaluation results show that Android devices and Xiaomi devices keep the cookie and local storage data in their captive portal mini-browsers across different sessions, and thus they are vulnerable to an attacker that steals the browsing history. Other devices provide a clean environment for each session.

History-stealing attack on regular browsers. We examined three regular browsers with the highest market shares (i.e., Chrome, Safari, and Firefox) on the 15 tested devices. We found that only Firefox on MacBook Air could be used

as a captive portal mini-browser, while other browsers did not connect to the user portal when the device was connected to a captive portal.

Because Firefox and the MacBook Air's captive portal mini-browser use different URIs to detect whether the device is in a captive state[8], an attacker can bypass the captive portal mini-browser by redirecting only Firefox's detection URI to the user portal. Additionally, Firefox uses normal browsing mode instead of private browsing mode to connect to the user portal. We used Wi-Fi Chameleon to test whether a history-stealing attack works and verified that Firefox on MacBook Air is vulnerable.

This attack does not apply to Nintendo Switch because it contains only the captive portal mini-browser. It does not apply to devices that use the regular browser either, because no captive portal mini-browser exists.

6 Discussion

6.1 Case Study: Browser Perspective

A question that naturally arises is why these insecure browsers might exist in the first place. While we are unaware of any official statement on this question from the vendors of our testing devices, we make the best inference from the information we have.

On Apple's OSes (iOS and macOS), the captive portal mini-browser is named *Captive Network Assistant*, or *CNA*. To the best of our knowledge, Apple does not release a manual on the implementation or behavior of CNA, except for hinting at the method used to detect captive portals and supports for RFC 8910 [7]. Judging from the discussion on the developer forums [6] and Wireless Broadband Alliance's report [3], we infer that CNA is the stripped and restricted version of Safari bundled with the OS. Developers reported several restrictions, such as limited JS features and disallowing open native apps, including native Safari. The best inference we can make is that these restrictions are posed due to security concerns and user experience.

Similar situations apply to Android. The official documents only stated how Android detects the existence of the captive portal and the supports for RFC 8910 [5], but not why the captive portal mini-browsers are implemented as such. The Wireless Broadband Alliance's report [3] mentioned captive portal mini-browsers without details.

Lack of transparency on the design decisions of the captive portal mini-browsers makes it hard to conclude the reason behind the insecure designs of the captive portal mini-browsers. Our best speculation is that these designs are intended to enhance the user experience. Since many existing captive portals use HTTP or HTTPS with invalid certificates, suppressing the warning might be an understandable approach. However, it is worth questioning the goodness of this trade-off.

[8] MacBook Air's captive portal mini-browser uses http://captive.apple.com/hotspot-detect.html to detect a captive portal, while Firefox uses http://detectportal.firefox.com/canonical.html.

6.2 Case Study: Access Point Perspective

If the above speculation is valid, we need to question why the captive portal providers might not use the public domain names with valid certificates and provide HTTPS connections, forcing the vendors into shipping captive portal mini-browsers with insecure designs. A large-scale survey would be out of the scope of this work, so we instead resort to case studies on the commercially available APs.

We chose several commercially available APs among several popular brands, such as Cisco, ASUS, Netgear, and D-Link. Given that certificates are bounded to fully-qualified domain names or non-reserving IP addresses, user portals with HTTPS support tend to use public domain names or public IP addresses. Therefore, these user portals are external web services independent of the APs. To our best knowledge, only Cisco's APs [14], MikroTik's RouterOS [27], and TP-Link's EAP series [25] explicitly support external web service as the user portals. Other vendors, such as ASUS's devices [9] and D-Link's DBA series [15], do not mention the certificates of the user portal or the public domain names in the manuals[9], while providing built-in captive portals.

Our findings suggest that some commercially-available APs supporting captive portals do not explicitly allow or encourage network administrators to add certificates and use HTTPS for user portals, indicating that these APs are not designed to be secure by default.

7 Defense Scheme

In this section, we discuss two possible approaches to protect users from the attacks discussed in Section 5.

Ideally, captive portal mini-browsers should follow the best practices mentioned in Section 2.3, and the Wi-Fi AP should always serve the user portal using HTTPS with valid certificates. However, as we discussed in Section 6, these may be unrealistic in a short term. Therefore, we proposed a deployable approach for the end-users, serving as a temporary fix before the captive portal mini-browsers and Wi-Fi APs follow the best practices. This approach uses a browser extension on Chrome and Firefox to secure the captive portal mini-browser for the end-users.

The second approach is a more comprehensive fix and solves some of the limitations of the first approach. This approach may require changes to the captive portal mini-browsers and the firmware of the APs. Notably, this approach can be easily integrated with RFC 8952.

7.1 Approach 1. Secure Captive Portal Browser Extension

As shown in Table 2, regular browsers are secure against all attacks except SSL stripping and redirection to another URI. To also mitigate these two attacks,

[9] This does not indicate that ASUS and D-Link devices do not support HTTPS for the user portals, but they may require special configurations or workarounds.

we propose a solution which uses the *HTTPS list*, a list containing trusted user portals that are using HTTPS.

Using the trust-on-first-use assumption, when a new captive portal is connected, its user portal URI is recorded to the HTTPS list and being trusted as the correct user portal URI. When a known captive portal is connected, its user portal URI is compared with those stored in the HTTPS list to check whether the URI has been modified. This approach detects (1) the SSL stripping attack when a trusted user portal changes to HTTP, and (2) the redirection attack when the host is different. This solution is similar to HSTS, but it does not require the server-side's participation, and thus it is simple to deploy.

To prevent exposing users' connection history to an attacker, instead of blocking the connection to the evil twin's user portal, our extension sends a request to the user portal in the background and prevents the user from having further interactions with it. This makes sure that our extension behaves the same when an attack is detected.

Security Analysis Our extension is secure against all the attacks in Section 2.2. For the SSL stripping and redirection attacks, the HTTPS list prevents the modification of the user portal URI. Since a regular browser in private browsing mode can already defend against fake certificates and history-stealing attacks, we only need to ensure that the user portal is browsed in private browsing mode.

Discussion This approach has several improvements compared to existing solutions, which we describe in more details in Section 8. For example, it is a purely client-side solution with little modification to the client's device and supports a broad range of devices. Furthermore, it does not rely on any prior knowledge of the captive portal or security assumptions other than the trust-on-first-use assumption. Considering the deployability, the required security assumptions, and the effectiveness of defense, we believe that this is a practical solution to improve the security of captive portal mini-browsers.

Functionalities Because Chrome and Firefox in private browsing mode already satisfy the majority of the security requirements, the extension only needs to provide the following functionalities.

1. Detect captive state and record the user portal URI to the HTTPS list if the URI is using HTTPS and is trusted.
2. Block the connection if a host in the HTTPS list changes the connection protocol to HTTP.
3. Block the connection if an unrecorded URI is provided by a known AP.
4. Navigate to the user portal in private browsing mode.
5. Block connections to the user portal if they are not in private browsing mode.

The fourth and the fifth functionalities guarantee that the user portal is browsed in private browsing mode, even when a device uses its default browser as the captive portal mini-browser. These devices usually navigate users to the

user portal with private browsing disabled and without the user's consent, while our extension can block these connections and instructs the user to securely connect to a user portal.

Implementation We developed a browser extension to support secure captive portal mini-browsers on Chrome and Firefox.[10] When users launch our extension in their browsers, it detects the captive portal and provides instructions to securely access the user portal.

Limitations One downside of this approach is that it generates false-positives when a captive portal changes its user portal URI. In this situation, the user needs to verify the correctness of the new URI, possibly through contacting the Wi-Fi provider, and updates the HTTPS list manually.

Another limitation is that to check whether an AP had been connected before, one would require knowing the Wi-Fi AP's SSID or other identifiers. However, the Wi-Fi AP's information is unavailable to a browser extension. To bypass this limitation, our extension instructs the user not to trust a new URI that is provided by a previously-connected Wi-Fi AP. Consequently, our extension only needs to store the user portal's hostname in the HTTPS list and does not need to know the Wi-Fi AP's information.

Moreover, the extension relies on the trust-on-first-use assumption and an up-to-date certificate-revocation list. Nevertheless, our design can be integrated with other defense schemes to relax these trust assumptions, such as using known fingerprints or network behavior to detect an evil twin. We discuss such schemes in Section 8.

Note that even a secure captive portal mini-browser cannot protect a careless user who ignores the security warnings. In addition, such a browser cannot guarantee security when connecting to a misconfigured captive portal that uses HTTP for the communication protocol or uses invalid certificates for HTTPS.

7.2 Approach 2. Identity Verification Strategies

A more comprehensive way to mitigate the attack is to verify the identity of the AP, since identity verification achieves high security but makes strong trust assumptions about CAs and/or out-of-band channels. This section presents two defense strategies to relax such trust assumptions: SSID distinction and trusted certificate pinning.

When a user connects to a Wi-Fi AP, (*SSID distinction*) the user equipment first checks whether the SSID prefix matches the common name (CN) of the user portal's certificate. This procedure aims to prevent a rogue AP from setting the same SSID as the real AP. If the SSID prefix matches the CN, the captive

[10] Our extension for Firefox is available at https://mzl.la/3iBVtD8, and our code is available at https://tinyurl.com/2baakxt2. To install our extension on Android, please use Firefox Nightly and add a custom add-on collection with user ID "16929574" and collection name "Wifi-Chameleon".

portal mini-browser, with the desired properties mentioned in Section 2.3, then performs security checks and displays proactive warnings if there are any security issues. For the first-time connection, (*trusted certificate pinning*) the user equipment pins the user portal certificate provided through a trusted channel. This procedure can be skipped for future connections.

SSID Distinction While evil-twin attacks clone the SSID of the real AP to trick the users into connecting, this can be prevented through SSID distinction, which binds the SSID with the user portal certificate.

To bind an SSID with a certificate, we require the real AP to prefix its SSID with the CN of its user portal's certificate. Since the attacker cannot possess another valid certificate with the same CN as the real user portal's certificate, the attacker must use a different SSID for the rogue AP.

Trusted-certificate Pinning A certificate is pinned only after we confirm it is valid and belongs to the real AP. It is recommended that Wi-Fi operators provide information through out-of-band channels to aid first-time users' confirmation that a certificate is trusted. Pinning a user portal certificate simplifies the validation process for future connections.

Integration with RFC 8952 Our defense strategy is compatible with the captive portal architecture proposed in RFC 8952, and its integration only requires slight changes to component of user equipment. Our defense strategy requires the captive portal mini-browser to check SSID distinctions, validate the user portal's certificates, display proactive security warnings, and pin the trusted certificates. Since RFC 8952 has no naming restriction on the SSID, adding prefix to it does not interfere with the standard.

8 Related Work

Our work investigated the security of captive portal mini-browsers and the protections against rogue APs. Accordingly, we review related work on captive portals' privacy issues, client-side defenses against rogue APs, and captive portal standardization efforts.

Captive Portals' Privacy Issues Ali *et al.* [1] developed CPInspector to evaluate captive portals' privacy risks around personal-identification information collection, cookies, and user tracking. They found that user portals may expose a user's privacy and track users before they consent. This attack falls under the history-stealing attacks in our categorization, and our evaluation suggests that certain devices are susceptible to these attacker models. Dabrowski *et al.* [16] implemented a history-stealing attack by modifying the user portal to let the browser fetch external references of images and use cookies or HSTS records to reveal users' browsing history. These studies motivated us to check whether captive portal mini-browsers isolate each session, and whether they can be bypassed.

Client-Side Defense for Rogue APs To detect rogue APs, several schemes passively collect identifying information ("fingerprints") pertaining to APs, either by monitoring network traffic or by measuring radio signals, and use the fingerprints to detect an evil twin. Alotaibi and Elleithy [4], and Chae *et al.* [11] utilized the Beacon Frame, a broadcast message sent from an AP, as the fingerprint, while Bauer *et al.* [10] and Gonzales *et al.* [17] collect the fingerprints from the environment. All these schemes assume that the real AP's fingerprint or the correct environment is known in advance, while our defense scheme does not require any prior knowledge.

Network probing is to actively probe the network to detect anomalies. If a user connects to a rogue AP, its network route [20,28] or connection latency [19, 28,34] may deviate from those it experienced before. Han *et al.* [19] established that Round Trip Time (RTT) differs for rogue APs, and Mustafa and Xu [28] additionally used internet service provider information and global IP addresses to detect rogue APs. Song *et al.* [34] measured inter-packet arrival time to classify rogue APs. Since network probing relies on a network's baseline behavior, it is more suitable for networks without dynamic nature. Our defense scheme, on the other hand, is not affected by the network environment.

Captive Portal Standardization Efforts Passpoint technology [2], based on the IEEE 802.11u standard, uses a standardized process for users to sign up for Internet access. To defend against evil-twin attacks, a server certificate must contain the server's "friendly names" and the hash value of its logo so that this information cannot be copied. Since certificates for HTTPS usually do not require these fields, issuing a certificate for Passpoint requires more validations. Passpoint also enforces Online Certificate Status Protocol (OCSP) stapling [32] so that the revocation state of a certificate can be verified even without Internet access. While these requirements protect the users from evil-twin attacks, both Wi-Fi providers and users need to change their hardware or software to deploy this solution.

EAP-SH [26], an EAP-compliant protocol based on EAP-TLS, allows the usage of the captive portal within the 802.11 authentication framework. In the current paradigm, the access to the APs using captive portals as access control is not protected at the link layer, so it can be eavesdropped on or perform MitM attacks. EAP-SH, designed to solve these problems, works as the following. When performing the authentication, the TLS session is built via EAP-TLS. Then, the Authentication Server and the Supplicant serve as a tunnel, encoding the HTTP traffic between the Captive Portal and the browser into the EAP messages. Therefore, all communications over the air are encrypted by the TLS session established via EAP-TLS. However, because the browser communicates with the Authentication Server, not the user portal, users can no longer use the URIs and the certificates of the user portals to tell if they are using the authentic AP.

RFC 8952 [24] proposes a standardized design of captive portals. While existing captive portals usually forge an HTTP response to redirect users to the user portal, RFC 8952 states that captive portals should not break any protocols, and the user portal URI should instead be provided by a provisioning service. Other

RFC 8952 security requirements include that such a URI must use HTTPS and that its TLS certificate must be validated. However, even if these requirements are fulfilled, a user is still susceptible to a redirection attack as the attacker can also construct an evil twin that complies with these requirements.

9 Conclusion

This paper identifies security flaws in captive portal mini-browsers. We developed a security assessment tool called Wi-Fi Chameleon to show that (1) captive portal mini-browsers only provide minimal security protections and (2) they are vulnerable to evil-twin and history-stealing attacks. To protect users' sensitive information and privacy, we highlight the desired security properties for a secure captive portal mini-browser. In addition, we propose two different approaches as mitigation. The first approach, focusing on deployability, is a browser extension providing a capability to maintain the end-user security before most devices and Wi-Fi APs provide a safer way for public Internet access. Our second approach attempts to add identity verification as well as the client-side protections, complementing the standardization effort (e.g., RFC 8952 [24] and Passpoint [2].)

Acknowledgement

This research was supported in part by the Ministry of Science and Technology of Taiwan under grants MOST 110-2628-E-002-002 and 111-2628-E-002-012.

A Appendix

A.1 Tested Devices

A.2 Screenshots of Test Devices

In this section, we provide the screenshots of our test result.

Figure 6 shows the warning message provided by Nintendo Switch when the user portal is connected using HTTP. As shown in the figure, the user can press the '+' button to see more information about the user portal. The page information then shows that this user portal is using HTTP and warns the user that this connection is not encrypted. While showing warning messages to the user provides situational awareness, users are unlikely to click a button to learn about the security warning. We suggest proactively warning users and helping them understand the risks before interactions begin.

References

1. Ali, S., Osman, T., Mannan, M., Youssef, A.: On privacy risks of public wifi captive portals. In: Data Privacy Management, Cryptocurrencies and Blockchain Technology, pp. 80–98. Springer (2019)

Table 3: Tested devices, including 18 types of popular laptops and mobile devices. The bottom three have no captive portal mini-browsers and thus were excluded from our subsequent evaluation.

Group	Device	Operating System
Apple devices	MacBook Air	Big Sur 11.4
	iPhone 6s	iOS 14.0
	iPad Air	iPadOS 14.0
Android devices	Pixel 3a	Android 11
	Samsung S21+	Android 11
	Samsung A42 5G	Android 10
	ASUS ROG Phone 3	Android 10
	HTC U20	Android 10
	LG Velvet 5G	Android 10
	Sony Xperia 1 II	Android 11
	Oppo Reno5 5G	Android 11
	vivo X50 Pro	Android 11
Nintendo Switch	Nintendo Switch	12.0.3
Xiaomi devices	Mi 11 Lite 5G	Android 11
	Mi 10T	Android 10
No user portal browser	realme X7 Pro 5G	Android 10
	Huawei Mate 20 Pro	Android 9
	HP Envy x360	Windows 10

2. Alliance, W.F.: Passpoint (2019), https://www.wi-fi.org/discover-wi-fi/passpoint
3. Alliance, W.B.: Captive network portal behavior (2019), https://captivebehavior.wballiance.com/
4. Alotaibi, B., Elleithy, K.: An empirical fingerprint framework to detect rogue access points. In: 2015 Long Island Systems, Applications and Technology. pp. 1–7. IEEE (2015)
5. Android: Captive portal api support — android developers (2022), https://developer.android.com/about/versions/11/features/captive-portal
6. Apple: Just how limited is the captive network assistant? - apple coummunity (2013), https://discussions.apple.com/thread/5258403?tstart=0
7. Apple: How to modernize your captive network - discover - apple developer (2020), https://developer.apple.com/news/?id=q78sq5rv
8. April King, Lucas Garron, C.T.: Badssl (2015), https://badssl.com/
9. ASUS: [guest network] how to set up captive portal? — official support — asus global (2021), https://www.asus.com/support/FAQ/1034977/
10. Bauer, K., Gonzales, H., McCoy, D.: Mitigating evil twin attacks in 802.11. In: 2008 IEEE International Performance, Computing and Communications Conference. pp. 513–516. IEEE (2008)
11. Chae, S., Jung, H., Bae, I., Jeong, K.: A scheme of detection and prevention rogue ap using comparison security condition of ap. In: Advances in Computer Science and Electronics Engineering, 2012 Universal Association of Computer and Electronics Engineers International Conference on. pp. 302–306 (2012)

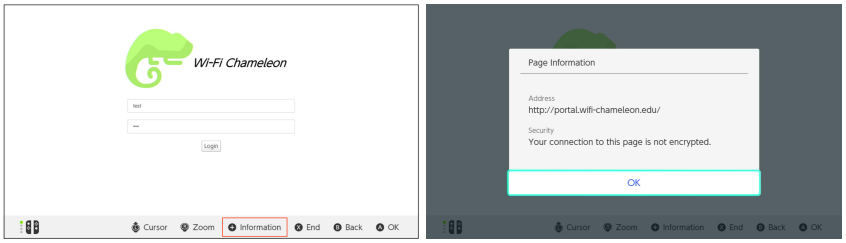


Fig. 6: The warning message for HTTP connection provided by Nintendo Switch

12. Chen, L., Grassi, M.: Exploiting user-land vulnerabilities to get rogue app installed remotely on ios 11 (2018), https://recon.cx/2018/montreal/schedule/events/113.html
13. Chen, W.L., Wu, Q.: A proof of mitm vulnerability in public wlans guarded by captive portal. Proceedings of the Asia-Pacific Advanced Network **30**, 66 (12 2010). https://doi.org/10.7125/APAN.30.10
14. Cisco: Configuring captive portal (2022), https://www.cisco.com/assets/sol/sb/isa500_-emulator/help/guide/ad1982733.html
15. D-Link: Nuclias cloud documentation (2020), https://media.dlink.eu/support/products/dba/dba-2820p/documentation/dba-2820p_man_reva1_1-10_eu_multi_20201202.pdf
16. Dabrowski, A., Merzdovnik, G., Kommenda, N., Weippl, E.: Browser history stealing with captive wi-fi portals. In: 2016 IEEE Security and Privacy Workshops (SPW). pp. 234–240. IEEE (2016)
17. Gonzales, H., Bauer, K., Lindqvist, J., McCoy, D., Sicker, D.: Practical defenses for evil twin attacks in 802.11. In: 2010 IEEE Global Telecommunications Conference GLOBECOM 2010. pp. 1–6 (2010)
18. Google: Google safe browsing (2021), https://safebrowsing.google.com/
19. Han, H., Sheng, B., Tan, C.C., Li, Q., Lu, S.: A timing-based scheme for rogue ap detection. IEEE Transactions on parallel and distributed Systems **22**(11), 1912–1925 (2011)
20. Hsu, F.H., Hsu, Y.L., Wang, C.S.: A solution to detect the existence of a malicious rogue ap. Computer Communications **142**, 62–68 (2019)
21. kleo: Evil portals (2016), https://github.com/kleo/evilportals
22. Labs, Q.S.: Ssl client test (2021), https://clienttest.ssllabs.com:8443/ssltest/viewMyClient.html
23. Lanze, F., Panchenko, A., Ponce-Alcaide, I., Engel, T.: Undesired relatives: Protection mechanisms against the evil twin attack in ieee 802.11. In: Proceedings of the 10th ACM Symposium on QoS and Security for Wireless and Mobile Networks. p. 87–94. Q2SWinet '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2642687.2642691, https://doi.org/10.1145/2642687.2642691
24. Larose, K., Dolson, D., Liu, H.: Captive Portal Architecture. RFC 8952 (Nov 2020). https://doi.org/10.17487/RFC8952, https://rfc-editor.org/rfc/rfc8952.txt
25. tp link: Omada sdn controller user guide (2022), https://static.tp-link.com/upload/software/2022/202203/20220331/1910013160-Omada%20SDN%20Controller%20User%20Guide.pdf
26. Marques, N., Zúquete, A., Barraca, J.P.: Eap-sh: an eap authentication protocol to integrate captive portals in the 802.1 x security architecture. Wireless Personal Communications **113**(4), 1891–1915 (2020)

27. MikroTik: Hotspot customisation - routeros - mikrotik documentation (2022), https://help.mikrotik.com/docs/display/ROS/Hotspot+customisation
28. Mustafa, H., Xu, W.: Cetad: Detecting evil twin access point attacks in wireless hotspots. In: 2014 IEEE Conference on Communications and Network Security. pp. 238–246. IEEE (2014)
29. Mónica, D., Ribeiro, C.: Wifihop - mitigating the evil twin attack through multi-hop detection. vol. 6879, pp. 21–39 (09 2011). https://doi.org/10.1007/978-3-642-23822-2_2
30. Northwoods: Mixed content examples (2021), https://www.mixedcontentexamples.com/
31. P0cL4bs: wifipumpkin3 (2018), https://github.com/P0cL4bs/wifipumpkin3
32. Pettersen, Y.N.: The Transport Layer Security (TLS) Multiple Certificate Status Request Extension. RFC 6961 (Jun 2013). https://doi.org/10.17487/RFC6961, https://rfc-editor.org/rfc/rfc6961.txt
33. Schmoe, J.: Tunneling through captive portals with dns (2017), https://0x00sec.org/t/tunneling-through-captive-portals-with-dns/1465
34. Song, Y., Yang, C., Gu, G.: Who is peeping at your passwords at starbucks?—to catch an evil twin access point. In: 2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN). pp. 323–332. IEEE (2010)
35. StatCounter: Desktop operating system market share worldwide (2021), https://gs.statcounter.com/os-market-share/desktop/worldwide
36. StatCounter: Mobile vendor market share worldwide (2021), https://gs.statcounter.com/vendor-market-share/mobile/
37. sud0nick: Portalauth (2016), https://github.com/sud0nick/PortalAuth
38. W3C: web-platform-tests (2019), https://web-platform-tests.org/
39. Wikipedia: List of best-selling game consoles (2021), https://en.wikipedia.org/wiki/List_of_best-selling_game_consoles#cite_note-:0-35