EvilTwinX: Rogue Access Point Attack

Team: Cyber Twins

Prepared by: Dev Sharma (179), Prateek (166)

Date: 14-09-2025

Classification: WiFi-Evil Twin — Educational / Authorized testing

Abstract

EvilTwinX is a controlled, educational proof-of-concept that demonstrates how an attacker can leverage an Evil Twin access point combined with a captive portal to deceive users into submitting Wi-Fi credentials. The tool captures WPA/WPA2 handshakes, provides a fake access point that mirrors the real SSID, and verifies submitted passphrases against the captured handshake. This report documents the architecture, operation flow, and results from a safe lab environment. The purpose is to highlight the risks of rogue access points, raise awareness of human-factor vulnerabilities, and support the development of defensive measures against wireless phishing techniques.

Table of Contents

1. Introduction

Wireless networks are widely used but inherently exposed: radio signals propagate beyond physical boundaries and management/authentication frames are often unauthenticated or weakly protected. An "Evil Twin" attack leverages this exposure by creating a rogue access point (AP) that impersonates a legitimate SSID. Combined with a captive portal — a web page presented to newly connected clients — an attacker can trick users into submitting credentials or other secrets into a page they believe to be trustworthy.

EvilTwinX is a lab-focused proof-of-concept that automates the common components of such an attack chain: scanning for candidate SSIDs, passively (and optionally actively) collecting WPA/WPA2 handshakes, standing up a fake AP that advertises the target SSID, presenting a captive portal to connected clients, and attempting to verify any submitted passphrases against the captured 4-way handshake. The tool intentionally mirrors real-world red-team techniques so defenders can reproduce, observe, and mitigate them in a controlled environment.

This introduction provides the technical and human context for the PoC. Technically, WPA/WPA2 PSK authentication relies on a pre-shared passphrase and a 4-way handshake that proves possession of that passphrase without sending it in plaintext. If an attacker captures a complete handshake, they can offline-verify candidate passphrases (or check a user-submitted passphrase) to see whether it matches. Human factors matter because users often expect captive portals (e.g., cafés, hotels), making them prone to entering credentials into a web form — and many devices will surface a captive-portal prompt automatically, which an attacker can mimic.

The intended audience for this report includes security practitioners, network defenders, and educators who need a repeatable lab exercise demonstrating how wireless phishing and Evil Twin techniques operate and how they can be detected and mitigated. This PoC is explicitly constrained to isolated, authorized lab environments and is not intended — nor is it legally or ethically permitted — for use against production networks or third-party users without written consent.

2. Scope & Objectives

Primary objectives

- Build a reproducible lab PoC demonstrating an Evil Twin with captive-portal credential harvest.
- Capture WPA/WPA2 handshake(s) and verify candidate PSKs against them.
- Provide logs and an automated cleanup to keep testbeds reliable.

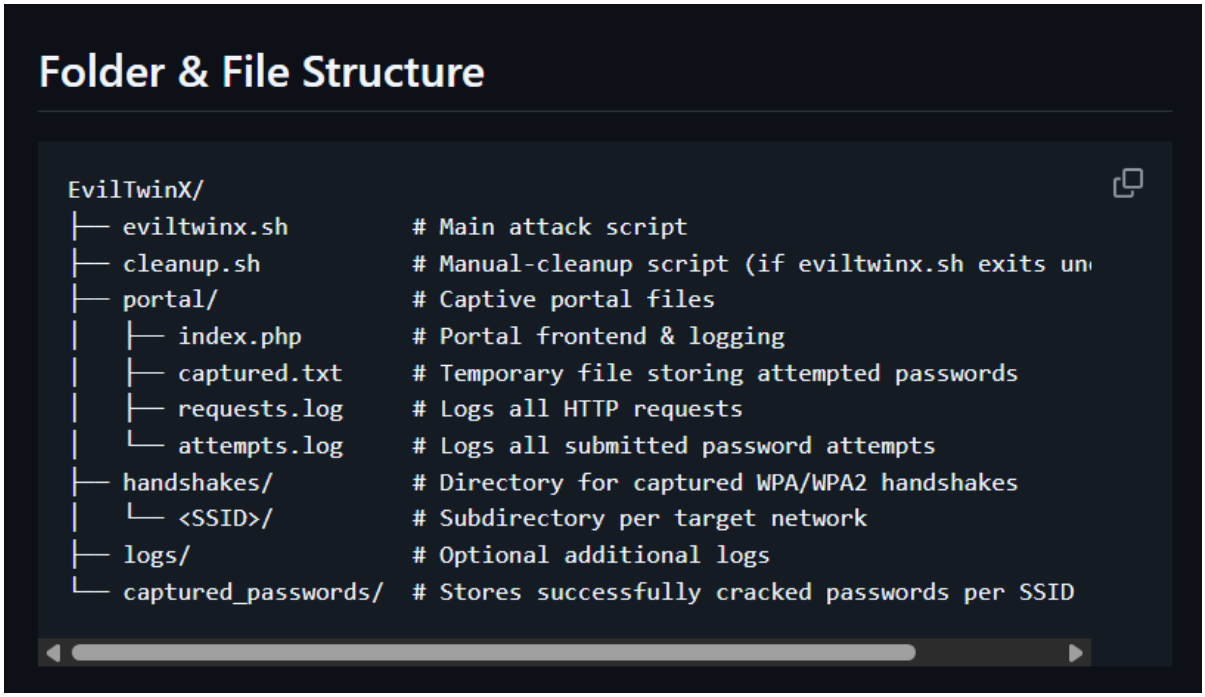
Out of scope

- Targeting production networks without written authorization.
- Integration with large scale cracking/cloud services.

3. Architecture & Components

Describe modules and responsibilities (conceptual):

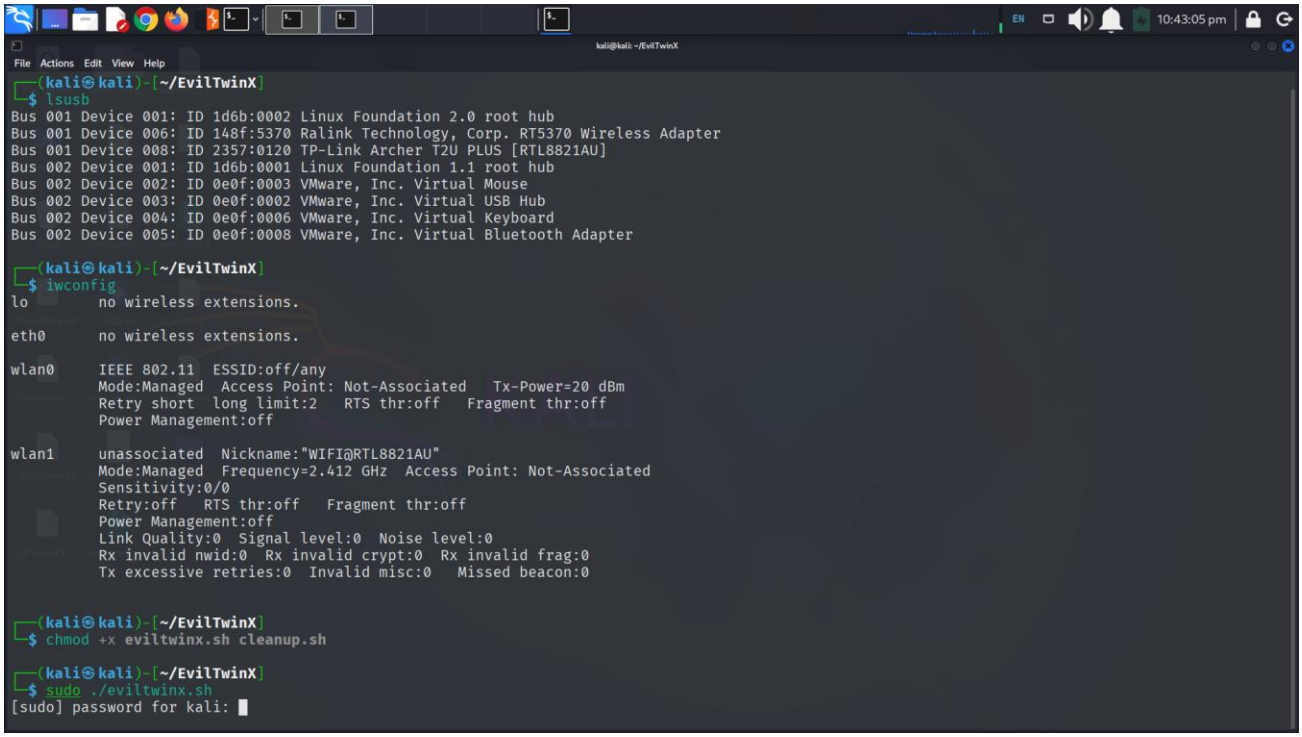
- **Scanner / Handshake capture** — monitor radio for 802.11 management frames and handshake capture.
- **Deauthentication module** — (conceptual) to provoke reconnects and increase handshake probability.
- **Fake AP (Evil Twin)** — broadcasting the chosen SSID on AP radio.
- **DHCP / DNS service** — assigns IPs and resolves portal hostname to local server.
- **Captive portal** — simple PHP or static page that collects submitted passphrases.
- **Verifier** — validates submitted passphrases against recorded handshake (requires complete handshake).
- **Logging & Cleanup** — structured logs, saved artifacts, and routines to restore interfaces.



4. Test Environment

List hardware, OS, network conditions, and isolation measures used during testing. Example details:

- Host OS: Debian/Kali Linux, root access.
- Wi-Fi adapters: two USB radios — one capable of monitor/injection, one capable of AP/master mode.
- Tools: hostapd (for AP), dnsmasq (DHCP/DNS), lightweight PHP server for portal (lab bundle).
- Isolation: RF-controlled room/low power and tests performed only on owned equipment.

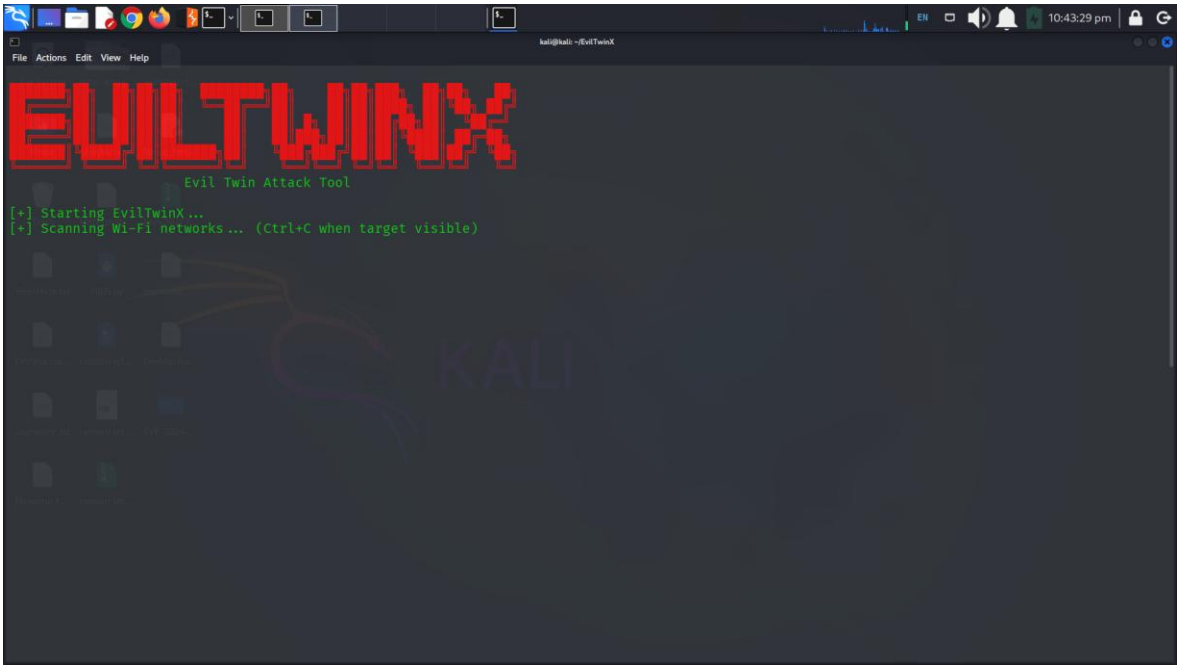


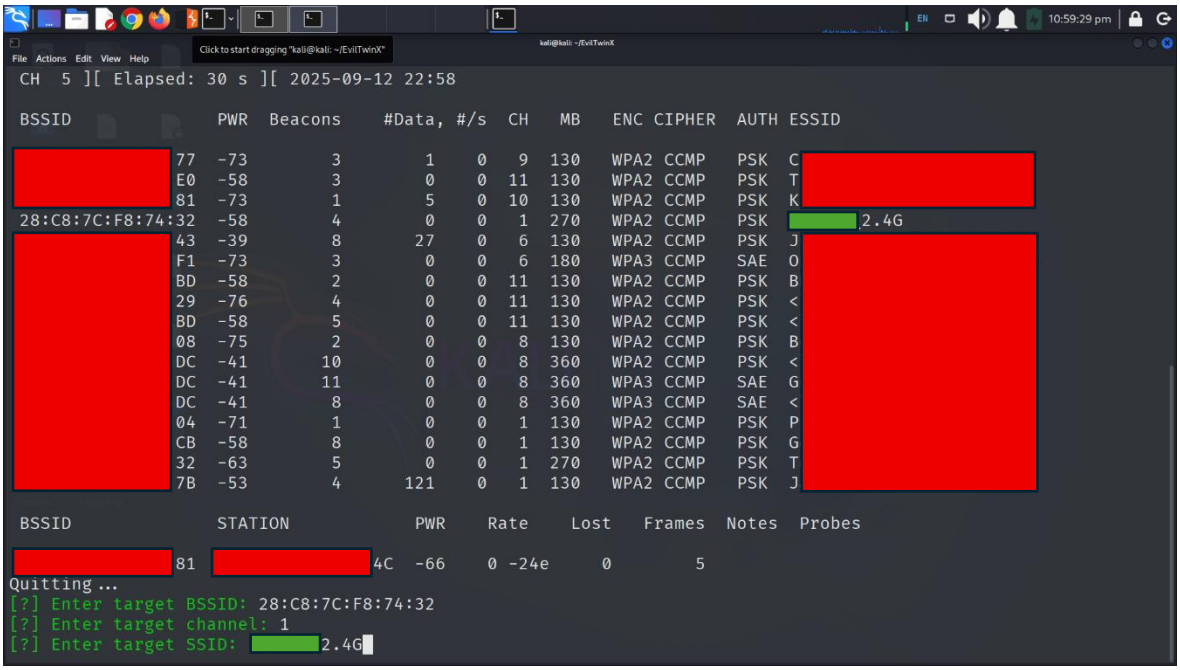
5. Operation Flow (Conceptual)

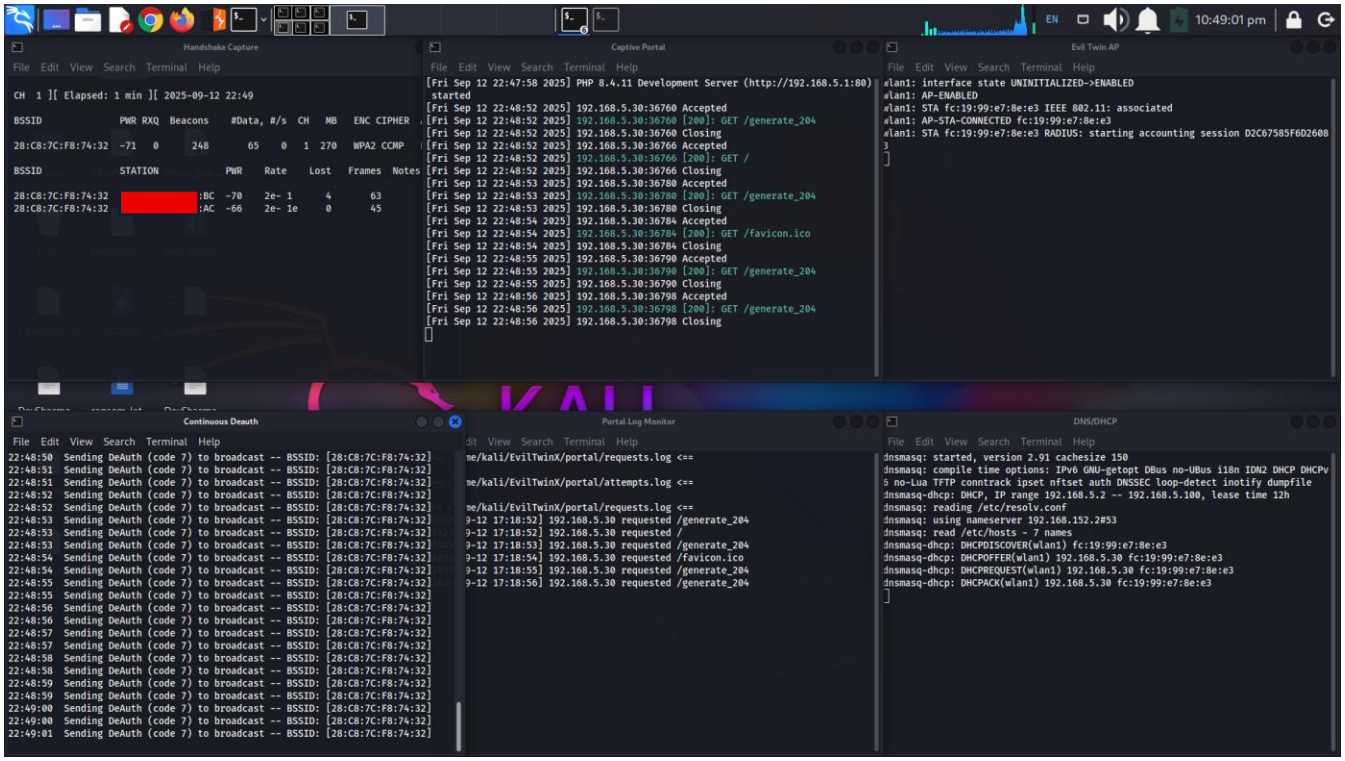
1. **Initialize environment:** Create workspace directories, reset or rotate previous artifacts, and put the monitor-capable radio into monitor mode.
2. **Scan & select target:** Use passive scanning to enumerate nearby SSIDs/BSSIDs/channels. The operator chooses a target SSID (and optionally a BSSID and channel).
3. **Capture handshake:** Start a capture session and optionally trigger deauthentication frames targeting connected clients to provoke reconnection and capture a complete 4-way handshake. Successful capture requires at least one active client to reauthenticate.
4. **Launch Evil Twin AP:** Configure the AP-capable radio with hostapd (or equivalent) to broadcast the target SSID on the chosen channel, start dnsmasq for DHCP/DNS, and launch the portal server (PHP or light server).
5. **Phishing interaction:** When a client associates to the fake AP and triggers captive-portal detection or HTTP traffic, the portal redirects/serves the login page. Submissions are logged into `portal/requests.log`, `portal/attempts.log`, and `portal/captured.txt`.
6. **Verify submitted passphrase:** The verifier takes the submitted passphrase and attempts to validate it against a stored handshake from `handshakes/`. If verification succeeds, the passphrase is saved under `captured_passwords/<SSID>.txt` and the PoC triggers cleanup. If verification fails, the attempt is logged and the system continues listening.

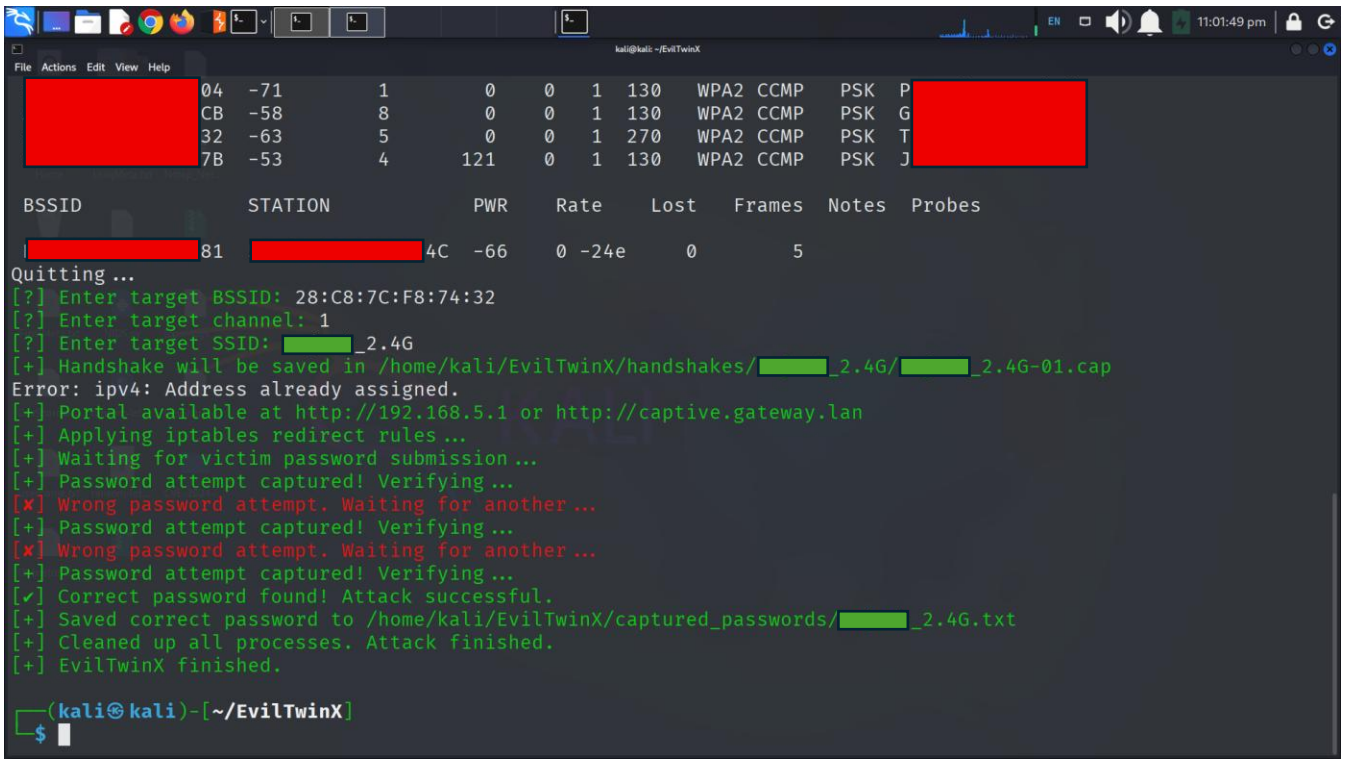
7. **Cleanup:** Stop hostapd/dnsmasq/portal processes, flush iptables rules used for redirection, restore interfaces to managed mode, and delete or rotate temporary files. `cleanup.sh` is provided to automate many of these tasks.

6. PoC Evidence (Screenshots & Photos)






Handshake Capture
CH 1][Elapsed: 1 min][2025-09-12 22:49
BSSID PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER
28:C8:7C:F8:74:32 -71 0 248 65 0 1 270 WPA2 CCMP
BSSID STATION PWR Rate Lost Frames Notes
28:C8:7C:F8:74:32 [redacted]:BC -70 2e- 1 4 63
28:C8:7C:F8:74:32 [redacted]:AC -66 2e- 1e 0 45
Captive Portal
[Fri Sep 12 22:47:58 2025] PHP 8.4.11 Development Server (http://192.168.5.1:80) started
[Fri Sep 12 22:48:52 2025] 192.168.5.30:36760 Accepted
[Fri Sep 12 22:48:52 2025] 192.168.5.30:36760 [200]: GET /generate_204
[Fri Sep 12 22:48:52 2025] 192.168.5.30:36760 Closing
[Fri Sep 12 22:48:52 2025] 192.168.5.30:36766 Accepted
[Fri Sep 12 22:48:52 2025] 192.168.5.30:36766 [200]: GET /
[Fri Sep 12 22:48:52 2025] 192.168.5.30:36766 Closing
[Fri Sep 12 22:48:53 2025] 192.168.5.30:36780 Accepted
[Fri Sep 12 22:48:53 2025] 192.168.5.30:36780 [200]: GET /generate_204
[Fri Sep 12 22:48:53 2025] 192.168.5.30:36780 Closing
[Fri Sep 12 22:48:54 2025] 192.168.5.30:36784 Accepted
[Fri Sep 12 22:48:54 2025] 192.168.5.30:36784 [200]: GET /favicon.ico
[Fri Sep 12 22:48:54 2025] 192.168.5.30:36784 Closing
[Fri Sep 12 22:48:55 2025] 192.168.5.30:36790 Accepted
[Fri Sep 12 22:48:55 2025] 192.168.5.30:36790 [200]: GET /generate_204
[Fri Sep 12 22:48:55 2025] 192.168.5.30:36790 Closing
[Fri Sep 12 22:48:56 2025] 192.168.5.30:36798 Accepted
[Fri Sep 12 22:48:56 2025] 192.168.5.30:36798 [200]: GET /generate_204
[Fri Sep 12 22:48:56 2025] 192.168.5.30:36798 Closing
Evil Twin AP
wlan1: interface state UNINITIALIZED->ENABLED
wlan1: AP-ENABLED
wlan1: STA fc:19:99:e7:8e:e3 IEEE 802.11: associated
wlan1: AP-STA-CONNECTED fc:19:99:e7:8e:e3
wlan1: STA fc:19:99:e7:8e:e3 RADIUS: starting accounting session D2C67585F6D2608
Continuous Deauth
22:48:50 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:51 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:51 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:52 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:52 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:53 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:53 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:53 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:54 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:54 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:55 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:55 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:56 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:56 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:57 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:57 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:58 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:58 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:59 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:48:59 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:49:00 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:49:00 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
22:49:01 Sending DeAuth (code 7) to broadcast -- BSSID: [28:C8:7C:F8:74:32]
Portal Log Monitor
me/kali/EvilTwinX/portal/requests.log <==
me/kali/EvilTwinX/portal/attempts.log <==
me/kali/EvilTwinX/portal/requests.log <==
9-12 17:18:52] 192.168.5.30 requested /generate_204
9-12 17:18:52] 192.168.5.30 requested /
9-12 17:18:53] 192.168.5.30 requested /generate_204
9-12 17:18:54] 192.168.5.30 requested /favicon.ico
9-12 17:18:55] 192.168.5.30 requested /generate_204
9-12 17:18:56] 192.168.5.30 requested /generate_204
DNS/DHCP
dnsmasq: started, version 2.91 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua TFTP conntrack ipset nftset auth DNSSEC loop-detect inotify dumpfile
dnsmasq-dhcp: DHCP, IP range 192.168.5.2 -- 192.168.5.100, lease time 12h
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 192.168.152.2#53
dnsmasq: read /etc/hosts - 7 names
dnsmasq-dhcp: DHCPDISCOVER(wlan1) fc:19:99:e7:8e:e3
dnsmasq-dhcp: DHCPOFFER(wlan1) 192.168.5.30 fc:19:99:e7:8e:e3
dnsmasq-dhcp: DHCPREQUEST(wlan1) 192.168.5.30 fc:19:99:e7:8e:e3
dnsmasq-dhcp: DHCPACK(wlan1) 192.168.5.30 fc:19:99:e7:8e:e3


kali@kali: ~/EvilTwinX
File Actions Edit View Help
[redacted] 04 -71 1 0 0 1 130 WPA2 CCMP PSK P[redacted]
CB -58 8 0 0 1 130 WPA2 CCMP PSK G
32 -63 5 0 0 1 270 WPA2 CCMP PSK T
7B -53 4 121 0 1 130 WPA2 CCMP PSK J[redacted]
BSSID STATION PWR Rate Lost Frames Notes Probes
[redacted]81 [redacted]4C -66 0 -24e 0 5
Quitting ...
[?] Enter target BSSID: 28:C8:7C:F8:74:32
[?] Enter target channel: 1
[?] Enter target SSID: [redacted]_2.4G
[+] Handshake will be saved in /home/kali/EvilTwinX/handshakes/[redacted]_2.4G/[redacted]_2.4G-01.cap
Error: ipv4: Address already assigned.
[+] Portal available at http://192.168.5.1 or http://captive.gateway.lan
[+] Applying iptables redirect rules ...
[+] Waiting for victim password submission ...
[+] Password attempt captured! Verifying ...
[x] Wrong password attempt. Waiting for another ...
[+] Password attempt captured! Verifying ...
[x] Wrong password attempt. Waiting for another ...
[+] Password attempt captured! Verifying ...
[✓] Correct password found! Attack successful.
[+] Saved correct password to /home/kali/EvilTwinX/captured_passwords/[redacted]_2.4G.txt
[+] Cleaned up all processes. Attack finished.
[+] EvilTwinX finished.
(kali㉿kali)-[~/EvilTwinX]
$

7. Observations & Results

Summarize measurable outcomes and qualitative observations:

- **Handshake capture success rate** — describe how deauth increases probability and conditions required (active client reconnect).
- **Portal interaction** — note user agent/captive portal detection behavior and HTTPS considerations.
- **Verification accuracy** — verification succeeds only with a complete handshake; list observed false negative causes.
- **Cleanup effectiveness** — what persisted and how to ensure full cleanup.

8. Limitations

- **HTTPS & captive portal detection:** Many modern clients use captive-portal detection and HTTPS-first behavior that can prevent or obscure captive portal prompts. This reduces the reliability of credential capture.
- **Handshake dependency:** Verification requires a valid and complete 4-way handshake for the specific client–AP pair. Partial captures often lead to failed verification.
- **Scope of PoC:** The tool is intended for controlled lab use only and is not hardened for stealth or mass deployment. It should not be used on unconsenting networks.
- **Dual-band & multi-AP complexity:** Simultaneous attacks across 2.4 GHz and 5 GHz require additional hardware and careful channel coordination

9. Legal, Ethical & Responsible Use

State the legal and ethical boundaries clearly:

- Only test on equipment/networks you own or with written permission.
- Destroy captured credentials after testing.
- Follow responsible disclosure if you discover third-party vulnerabilities.

10. Defensive Recommendations

Actionable, defender-oriented guidance (high level):

- Move to WPA3 or WPA2-Enterprise with certificate/EAP where feasible.
- Enforce management frame protection (802.11w) where clients/APs support it.
- Deploy WIDS/WIPS to detect rogue APs advertising known SSIDs.
- Train users to be suspicious of unexpected captive portals and discourage entering PSKs into web forms.
- Monitor for spikes in deauths and unusual DHCP leases.

11. Future Work

Suggested improvements to the PoC and research directions:

- Multi-adapter orchestration for simultaneous 2.4/5 GHz attacks in lab scenarios.
- Automated simulated clients to exercise captive portal flows reliably in CI.
- Better portal template management for realistic testing and red team exercises.
- Telemetry and reporting (timelines, success metrics, anonymized analytics).

12. References

Include canonical references for readers to learn more:

- 802.11 standards overview and management frame protection (802.11w).
- WPA/WPA2/WPA3 protocol descriptions.
- Wireless IDS/IPS vendor docs (for example only).
- Aircrack-ng Suite Documentation: https://www.aircrack-ng.org/doku.php
- Hostapd Documentation (Linux Wireless AP): https://w1.fi/hostapd/
- Dnsmasq Documentation: http://www.thekelleys.org.uk/dnsmasq/doc.html
- IEEE 802.11 Standard Overview: https://standards.ieee.org/standard/802_11-2020.html
- Wi-Fi Alliance Security Overview (WPA2/WPA3): https://www.wi-fi.org/discover-wi-fi/security
- Kali Linux Wireless Attacks Tools Documentation: https://www.kali.org/tools/#wireless-attacks

13. Appendices

Appendix A — Directory layout (example)

```
EvilTwinX/
├── eviltwinx.sh
├── cleanup.sh
├── portal/
│   ├── index.php
│   ├── capture.php
│   ├── captured.txt
│   └── requests.log
├── handshakes/
├── captured_passwords/
└── hostapd.conf, dnsmasq.conf (generated at runtime)
```

Appendix B — Example logs (redacted)

- `requests.log` — show anonymized request timestamps and non-secret metadata.
- `attempts.log` — list of password attempts (redact the passwords in the public report).

Appendix C — Checklist for safe lab testing

- Obtain written authorization.
- Use isolated RF environment.
- Set low transmit power.
- Delete/rotate captured secrets after testing.