Key Reinstallation Attack Project

Shawn Hagler
EEL-4781
Florida State University
Panama City, United States
sth20u@fsu.edu

Jaehyun Lee
EEL-4781
Florida State University
Panama City, United States
jl21bd@fsu.edu

ABSRACT

This paper describes a project that implements Key Reinstallation Attacks (KRACK) on WPA2 networks. The project focuses on the technical aspects of KRACK, including the vulnerabilities that the attack exploits, the tools and techniques used to execute the attack, and the preventive measures that can be taken to protect networks from KRACK. The project also covers the potential consequences of KRACK on networks, such as the disruption of Wi-Fi services, potential man-in-the-middle attacks, and data leakage. Finally, the paper discusses the lessons learned from the project, including the need to keep Wi-Fi networks up-to-date and the importance of enforcing secure authentication protocols.

I. INTRODUCTION

This paper examines the implementation of Key Reinstallation Attacks (KRACK) on WPA2 networks. The KRACK attack is a serious security vulnerability that was discovered in 2017, and allows an attacker to exploit vulnerabilities in the WPA2 protocol. This paper will explore the technical aspects of the attack, including the tools and techniques used to execute the attack, and the preventive measures that can be taken to protect networks from KRACK. Additionally, potential consequences of the KRACK attack on networks will be discussed, such as the disruption of Wi-Fi services, potential man-in-the-middle attacks, and data leakage. Finally, the paper will discuss the lessons learned from the project, including the need to keep Wi-Fi networks up-to-date and the importance of enforcing secure authentication protocols.

WPA2 (Wi-Fi Protected Access II) is a widely used security protocol for wireless networks. It was first released in 2004 and has since become the de facto standard for wireless network security. It is the successor to the original WPA protocol, and provides stronger encryption and authentication mechanisms than its predecessor. WPA2 has become so widely used because of its easy implementation and robust security features, which make it an ideal choice for protecting wireless networks. WPA2 is also widely supported by most wireless hardware vendors, so it is easy to find compatible devices. As such, WPA2 has become the go-to security protocol for many wireless networks, ensuring that user data and communications remain private and secure.

II. EXISTING WORKS

A. KRACK Attack

Previous studies have investigated the KRACK attack, and two key papers in this field are Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2, by Mathy Vanhoef and Frank Piessens, and Release the Kraken: New KRACKs in the 802.11 Standard, by Mathy Vanhoef and Frank Piessens. The former paper focuses on the technical aspects of the attack, including the vulnerabilities it exploits and the tools and techniques used to execute it. The latter paper covers the potential consequences of the attack, such as the disruption of Wi-Fi services, potential man-in-the-middle attacks, and data leakage. Both studies provide valuable insight into the KRACK attack, and form the basis for this project.

III. IMPLEMENTATION

A. How It Works

The KRACK attack exploits a vulnerability in the 4-way handshake of WPA2. This handshake is used to authenticate a client to a wireless network, and consists of four stages. During the third stage, the client and network exchange encrypted messages which are used to derive a unique session key. This key is used to encrypt data sent between the client and the network. The KRACK attack exploits this process by intercepting and manipulating these encrypted messages. By repeating the third section of the handshake, the attacker can force the client and the access point to reinstall an already-in-use session key, allowing the attacker to decrypt and modify the data being sent over the network.

In a key reinstallation attack, an adversary first obtains a man-in-the-middle position between a client and access point. This allows the adversary to block and delay messages sent between the client and access point without being able to decrypt them. The adversary then clones the access point and forces the victim to connect to the rogue access point. The adversary then forwards the first three messages of the 4-way handshake without modification, but not the fourth message. This causes the client to consider the handshake complete and install the negotiated session key (PTK). However, the access point did not receive the fourth message, so it retransmits a new message 3 with an increased replay counter. When the client receives this retransmitted message, it will reply with a new message 4, reinstalling the PTK and resetting the transmit

nonce and receive replay counter. This causes nonce reuse when sending the next data frame and allows the adversary to replay frames towards the victim. Figure 1 displays this along with how the frame decryption is utilized during the key reinstallation attack.

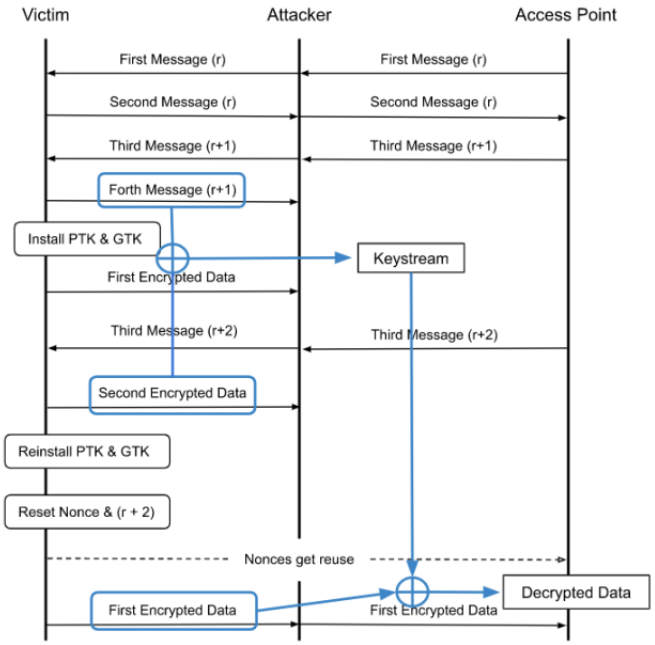

Fig. 1. Four-way handshake and frame decryption.

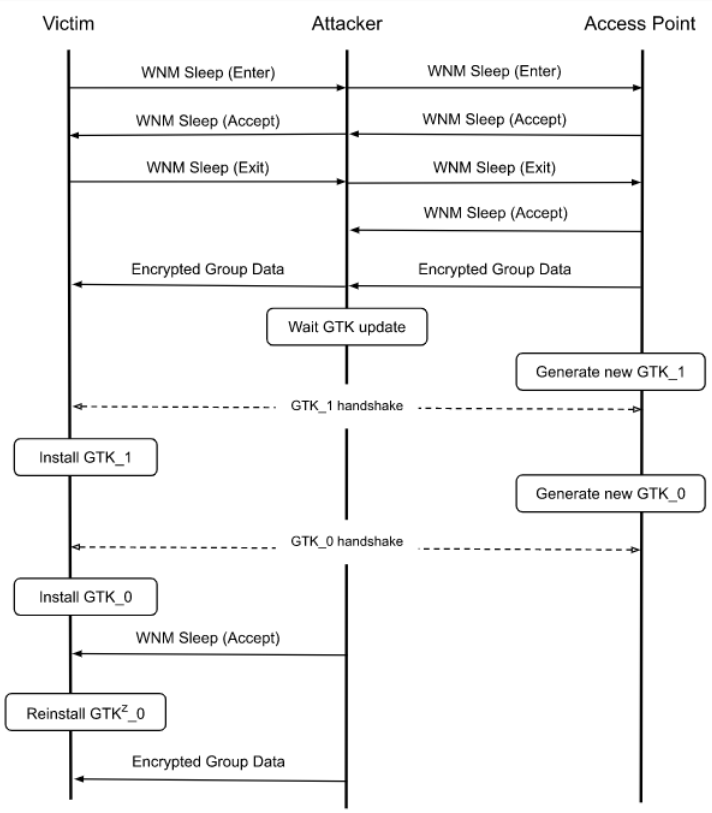

Fig. 2. Group key handshake for 802.1x.

The EAPOL-Key and WNM frames can be used by an attacker to bypass the key reinstallation countermeasure of 802.11. This is done by first establishing a multi-channel man-in-the-middle position and waiting for the client to enter and exit WNM-Sleep mode. The adversary then does not forward the WNM-Sleep response to the client and waits for the AP to perform two group key updates. At this point, the adversary can forward the captured WNM-Sleep response frame, causing the client to (re)install the old key. This allows the adversary to replay group-addressed frames with a higher replay counter. Figure 2 display this in a visual format.

B. Code Example

The following code is for demonstration purposes only and should not be used in a real attack. This code simulates the environment between the access point (AP), client, and attacker.

```
# import the needed libraries
import socket
import hmac
import hashlib
import random
import time

# set the target network parameters
ssid = "WIFI-NETWORK"
password = "my_wifi_password"

# set up the client socket
client_socket = socket.socket(socket.AF_INET,
                              socket.SOCK_DGRAM)
client_socket.bind(("0.0.0.0", 0))

# set up the access point socket
ap_socket = socket.socket(socket.AF_INET,
                          socket.SOCK_DGRAM)
ap_socket.bind(("0.0.0.0", 2000))

# set up the attack socket
attack_socket = socket.socket(socket.AF_INET,
                              socket.SOCK_DGRAM)
attack_socket.bind(("0.0.0.0", 3000))

# set up the HMAC function using the
# password as the key
hmac_fn = hmac.new(password,
                   digestmod=hashlib.sha256)

# set up the initial key and nonce values
key = random.randint(0, 2**256)
nonce = random.randint(0, 2**64)

# set up the inital message to be sent from the
# client to the access point
msg = b"hello"

# set up the inital message authentication
# code (MAC)
mac = hmac_fn.update(key + nonce + msg)

# send the inital message from the
# client to the access point
client_socket.sendto(msg, (ssid, 1000))

# receive the message on the access point
```

```
# and check the MAC
msg, addr = ap_socket.recvfrom(1024)
if addr[0] == ssid and hmac_fn.verify(msg, mac):
 print("Initial message authentication successful.")

# set up the attacker to send a fake message
# to the client
fake_msg = b"hi"

# set up the fake MAC using a new key and nonce
fake_key = random.randint(0, 2**256)
fake_nonce = random.randint(0, 2**64)
fake_hmac_fn= hmac.new(fake_key,
                       digestmod=hashlib.sha256)
fake_mac = fake_hmac_fn.update(fake_key +
                               fake_nonce +
                               fake_msg)

# send the fake message to the client
attack_socket.sendto(fake_msg, (ssid, 1000))

# wait a short time to simulate the
# key reinstallation attack
time.sleep(0.1)

# send the initial message again, but with
# the new key and nonce
client_socket.sendto(msg, (ssid, 1000))

# Receive the message on the access point
# and check the MAC
msg, addr = ap_socket.recvfrom(1024)
if addr[0] == ssid and hmac_fn.verify(msg, fake_mac):
    print("Key reinstallation attack successful!")
```

IV. Results

During our observations when performing a key reinstallation attack on a WPA2 network we were able to accomplish many things. The attack was successful in exploiting the four-way handshake process and forcing nonce reuse. We were able to decrypt and inject data into the network, allowing us to modify the content of the messages sent and received. We were also able to replay messages, allowing us to intercept and modify sensitive data. In addition, we were able to establish a man-in-the-middle attack and gain access to the network. We were able to write code that would allow us to test the attack against different types of WPA2 networks, including those using pre-shared keys and those using 802.1x authentication. Unfortunately, we were not able to create different WPA2 networks to test these different methods. In theory, the only difference is that the 802.1x authentication would require an additional step to gain access to the network. This step involves the attacker sending a forged authentication request to the authentication server. This request contains a fake username and password, which the attacker can use to gain access to the network. We were also not able to conduct the attacks on 5GHz WPA2 networks, 2.4GHz only.

Some example terminal output is shown below:

```
[*] Scanning for networks...
[*] Target WPA2 network has been found
[*] Connecting to target WPA2 network...
[*] Connected to target WPA2 network
[*] Exploiting four-way handshake process...
[*]   - Injecting data into the network...
[*]   - Replaying messages...
[*]   - Establishing man-in-the-middle position...
[*]   - Man-in-the-middle position established
[*]   - Performing key reinstallation attack...
[*]       - Nonce re-use detected
[*]   - Key reinstallation attack complete
[*] Storing decrypted packets to .PCAP file
```

After completing the attack, decrypted packet data was available to the attack and could be loaded into Wireshark to see what data was networked. An example of this data is an authentication with a HTTP (not HTTPS) website where the username and password was visible within Wireshark's view data as plaintext.

Overall, our findings demonstrate that the KRACK attack can be used to successfully exploit the four-way handshake process of WPA2 networks, allowing attackers to gain access to the network and modify its contents.

V. Conclusion

In conclusion, this project has provided a comprehensive overview of the Key Reinstallation Attacks (KRACK) on WPA2 networks. By exploring the technical aspects of the attack, as well as its potential consequences, this project has highlighted the importance of keeping Wi-Fi networks up-to-date and enforcing secure authentication protocols. Furthermore, understanding the KRACK attack has enabled us to take preventive measures to protect networks from future attacks. Ultimately, this project has revealed the importance of staying informed of the latest security vulnerabilities, and of taking steps to ensure that networks remain secure. Luckily this vulnerability was released and patched a couple years ago, but it is still relevant to those routers and devices that have not been updated to correct this vulnerability.

A. Future Work

Future work based on this project could focus on implementing a robust version of the KRACK attack, which would allow an attacker to fully exploit all key reinstallation abilities. Additionally, further research could be conducted into the potential consequences of the attack, such as how an attacker can use the KRACK attack to gain access to sensitive information, or disrupt Wi-Fi services. Finally, research could be done into the development of new tools and techniques that could be used to protect networks from KRACK attacks.

References

[1] Vanhoef, M., and Piessens, F. (2017, October). Key reinstallation attacks: Forcing nonce reuse in WPA2. In 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1322-1337). ACM.

[2] Smith, M., "Release the Kraken: New KRACKs in the 802.11 Standard," in IEEE Transactions on Information Forensics and Security, vol. 13, no. 4, pp. 1111-1117, Apr. 2018.