A Software-defined Networking-based Detection and Mitigation Approach against KRACK

Yi Li[1], Marcos Serrano[1], Tommy Chin[2], Kaiqi Xiong[1] and Jing Lin[1]

[1] *Intelligent Computer Networking and Security Lab, University of South Florida, Tampa, U.S.A.*

[2] *Department of Computing Security, Rochester Institute of Technology, Rochester, U.S.A.*

Keywords: KRACK, Software-defined Networking, WPA2, Network Security.

Abstract: Most modern Wi-Fi networks are secured by the Wi-Fi Protected Access II (WPA2) protocol that uses a 4-way handshake. Serious weaknesses have been discovered in this 4-way handshake that allows attackers to perform key reinstallation attacks (KRACKs) within the range of an Access Point (AP) to intercept personal information. In this paper, we study KRACK and present a software-defined networking (SDN)-based detection and mitigation framework to defend against KRACK. The proposed framework leverages the characteristic of an SDN controller, a global view of a network, to monitor and manage a Wi-Fi network traffic. It consists of two main components: detection and mitigation modules. Both of them are deployed on the SDN controller. The detection module will monitor network traffic and detect the duplicated message 3 of the 4-way handshake. Once KRACK has been detected, the mitigation module will update the flow table to redirect the attack traffic to a splash portal, which is a place to store attack traffic. Extensive experimental results demonstrate that the proposed framework can efficiently detect and mitigate KRACK. We achieve an average of 170.926 ms to detect KRACK and an average of 10.041 ms to mitigate KRACK in our experiments.

1 INTRODUCTION

WPA2 is a widely used protocol developed by the Wi-Fi Alliance to secure the majority of wireless networks. A WPA2 encrypted network can defend against intruders and identity thieves by providing unique encryption keys when a client is trying to connect to a network. However, researchers have discovered some vulnerabilities of WPA2 (Kumkar et al., 2012; Tsitroulis et al., 2014a). WPA2 uses two keys, Pairwise Transient Key (PTK) and Group Temporal Key (GTK), for encryption. A PTK is established through the 4-way handshake defined in the 802.11i, and it is unique for each client. A GTK is randomly generated by Access Point (AP), and it is shared among all clients. Such APs can be implemented as routers, modems, or gateways. Attackers can manipulate the GTK to launch attacks since it is used as an encryption key in the AP and a decryption key for clients. Attackers can inject forged data traffic through a GTK to all associated clients. Therefore, it can possibly result in several types of attacks, such as man-in-the-middle (MitM) attacks and denial of service (DoS) attacks (Teyou and Zhang, 2018).

In October 2017, researchers discovered a serious flaw in the WPA2 protocol that allows attackers within the range of an AP to perform KRACK (Vanhoef and Piessens, 2017). To be precise, attackers can intercept and steal a client's information, such as passwords, emails, and credit card numbers, that should be safely encrypted. In some scenarios, it is also possible for attackers to inject malicious contents or manipulate data on the website that a client is visiting. Correct implementation of WPA2 is likely to be affected since this weakness is in the Wi-Fi standard itself. Furthermore, the researchers have investigated devices that are still affected by KRACK and developed generalized attacks based on it (Vanhoef and Piessens, 2018). Moreover, they have recently discovered an easier way to attack unpatched devices and bypass the Wi-Fi's official countermeasure against KRACK.

In order to alleviate this issue, a framework called KrackCover was proposed to detect KRACK (Chin and Xiong, 2018). It will send an alert to the client once the attack has been detected. The experiment is conducted in the public network such as coffee shops and libraries. This approach can effectively warn clients for potential threats of KRACK. However, it did not provide any solution to mitigate the attack.

Li, Y., Serrano, M., Chin, T., Xiong, K. and Lin, J.
A Software-defined Networking-based Detection and Mitigation Approach against KRACK.
DOI: 10.5220/0007926202440251
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 244-251
ISBN: 978-989-758-378-0

After a careful investigation of KRACK, we propose a Software-Defined Networking based detection and mitigation framework to defend against KRACK. SDN is an emerging paradigm with the benefits of network visibility and network device programmability, and it is widely used in network security. With the separation of the control plane and the infrastructure plane, an SDN network is very easy to be managed by a centralized SDN controller since it has a global view of the whole network. Many researchers have adopted SDN in network management and network security (Scott-Hayward et al., 2013; Yan et al., 2016; Xie et al., 2018).

The proposed framework consists of the two main modules: detection and mitigation. Specifically, we apply SDN to a Wi-Fi AP where Wi-Fi's short-haul communications are converted into long-haul communications to the Internet using SDN. That is, the conversion of communication data are processed by an SDN switch under the management of an SDN controller. In the detection module of the proposed framework, the SDN controller inspects each incoming Wi-Fi network request where the duplicated message 3 of the 4-way handshake will be detected when an attacker launches KRACK. We configure an AP to be functioned as an OpenFlow vSwitch (OVS), which is monitored and managed by the SDN controller. The detection module is deployed on the SDN controller since the controller has a global view of the network. First, the SDN controller monitors all the traffic going through the network. Then, if an attacker launches KRACK to target a victim, the detection module can detect this attack by checking duplicated message 3 of the 4-way handshake. Next, once the attack has been detected, the mitigation module will update the entries of flow table in the OVS to re-direct the traffic flows to a splash portal, which is a place to store attack traffic. The proposed framework can effectively defend against KRACK with the SDN architecture.

The rest of the paper is organized as follows. Section 2 presents related work about Wi-Fi security. Section 3 gives the threat model. Section 4 introduces the design of the proposed framework. Next, Section 5 discusses the evaluation of the proposed framework, such as detection time. Lastly, Section 6 concludes our studies and presents future work.

2 RELATED WORK

Many studies have been done in the security of Wi-Fi networks (Yang and Huang, 2018; Alblwi and Shujaee, 2017; Noh et al., 2016). Tsitroulis et al. (Tsitroulis et al., 2014b) studied the WPA2 security protocol and presented several vulnerabilities of the WPA2 protocol. As the increasing number of devices are connected to Wi-Fi networks, it becomes very important to secure those wireless networks. Akram et al. (Akram et al., 2018) explored the security aspect of residential wireless local area network (WLAN) AP in a real-time scenario. The potential vulnerabilities in media access control (MAC) filtering, Hidden Service Set Identifier (SSID), and WPA2 have been investigated. They proposed a three-layer security mechanism to secure the WLAN AP.

Ghanem et al. (Ghanem and Ratnayake, 2016) explored potential attacks against WPA2 pre-shared key (WPA2-PSK), such as deauthentication attacks. This type of attacks will make Wi-Fi clients to re-authenticate an AP with the attacker's intent to capture the authentication keys during an 802.11 handshake. To prevent this type of attacks, they proposed a novel re-authentication protocol to improve the 4-way handshake without any hardware upgrade or cryptography algorithms. Their experimental results further showed that the proposed protocol effectively enhance the security of WPA2-PSK without compromising the performance of the network.

Since 2017 (Vanhoef and Piessens, 2017; Vanhoef and Piessens, 2018), researchers have been studied the effects of KRACK and detection methods to identify the attacks. Fehér et al. (Fehér and Sandor, 2018) presented the effects of KRACK and its corresponding user behavior analysis. A quantitative survey was created and filled out by 379 people. The survey results showed that many experienced users had updated the security patch to avoid KRACK. However, many users tended to use weak passwords that attackers can decrypt easily. While they argued that the best solution to countermeasure KRACK is to update each respective wireless device with its appropriate security patch, some devices cannot be patched for an update due to end-of-life support by vendors.

The security of the Internet of Things (IoT) is very important as there are many IoT devices in our daily life. Terkawi et al. (Terkawi and Innab, 2018) presented some major impacts of KRACK on IoT. They studied how attackers can perform KRACK on IoT and the potential damages that KRACK may pose on IoT. They showed that KRACK not only will compromise privacy but also will compromise the connected access control and may pose potential long-term damages in IoT.

Chin et al. (Chin and Xiong, 2018) proposed a framework called KrackCover, which is a wireless security framework for detecting KRACK. KrackCover is designed to assist in the detection of KRACK and to provide suggestions to help protect the privacy of clients. When KRACK is detected, the alert evokes the client. KrackCover consists of two virtual machines (VMs), a mobile phone, and multiple wireless antennas for monitoring and capturing all 802.11 messages. For the two VMs, one is used for launching the attack, and the other is used for detecting the attack. The experiment was carried out in real-world Wi-Fi environments, including coffee shops and public libraries.

Naitik et al. (Naitik et al.,) ran experiments of KRACK and presented a detection scheme to detect the threat. The detection was constructed with the following steps. First, the detection system extracted the Ethernet layer of network packets followed by the extraction of the IEEE802.1x header and then an extraction of the WPA key data. At last, the system checks for duplicated message 3. Victims of the attack were alerted if duplicate packets were detected.

To detect and mitigate KRACK, Securing Sam (Sam, 2019) has developed a detector and has given some mitigation suggestions (Securing Sam, 2019), such as using Advanced Encryption Standard (AES), disabling fast roaming, updating Wi-Fi software in a manual fashion, and actively monitoring for rogue APs.

SDN has become a popular paradigm due to its programmability and characteristic of the global view of a network which can easily be adapted to monitor and manage network traffic. Many research studies started to explore the use of SDN to improve network performance and security (Shin et al., 2016; Cheminod et al., 2017; Manzoor et al., 2018). In this paper, we utilize the benefits of SDN controller and proposed a detection and mitigation framework based on SDN to defend against KRACK.

3 THE THREAT MODEL

KRACK utilizes the flaws of the Wi-Fi standard protocol to reinstall an already-in-use key. It aimed at attacking the 4-way handshake of the WPA2 protocol by manipulating and replaying cryptographic handshake messages. When a client attempts to connect to a protected network through an AP, the handshake is performed to verify the credentials of both the AP and the client. In this paper, we focus on the Linux/Android vulnerability due to the large magnitude of devices that are affected by it. KRACK occurs in the following manners in details.

1. An attacker waits for a client/victim to connect or reconnect to a Wi-Fi network AP. For the latter situation, an attacker transmits a specially crafted packet to dissociate the client from the AP, also known as a deauthentication attack. Prior to a client connecting to the AP, the attacker leverages a separate rogue AP that falsely broadcasts the SSID of a legitimate Wi-Fi network. Client devices that connect and associate to the attacker's rogue AP present the ability for the attacker to execute a MitM attack.

2. The attacker first detects the client's connection attempt by capturing the wireless communication exchange for a WPA2 handshake packet. The attacker then proceeds to stage a rogue Wi-Fi network with a matching SSID to trick the victim's machine to connect to the attacker's AP. Commonly, wireless devices often connect to an AP with the strongest signal strength unless designated by a user-defined parameter such as an operating system setting or a driver configuration. A rogue AP for this attack case often distinguishes itself from other Wi-Fi APs through the signal strength metric to lure a victim machine to connect to the rogue Wi-Fi network. An attacker may accomplish a MitM once a victim connects to a rogue Wi-Fi network by means of the attacker forwarding traffic between a legitimate network and the victim.

3. As the new MitM, the attacker then listens for the WPA2 handshake. During this handshake, the third message of the handshake exchanges a key nonce that the attacker will withhold. The attacker will take these packets and replay them multiple times to the victim or a target machine. Because a protocol implementation bug in Linux/Android devices, the WPA2 key will be cleared at which point the attacker installs an all-zero key, or no encryption.

Effects of KRACK can be disastrous, as other forms of encryption such as SSL/TLS may be removed in combination with well-known methods. For this reason, an attacker may obtain any information, such as credit card numbers, personally identifiable information, or authentication credentials with minimal effort.

4 METHODOLOGY

We propose an SDN-based solution to detect and mitigate KRACK. The main modules in this framework are detection and mitigation. Besides these two main modules, we also implement an attack module in this framework. The standard Wi-Fi protocol uses the 4-way handshake to generate a session key, but it is vulnerable to key re-installation attacks. Figure 1 shows the architecture of the proposed framework. While the solid black lines represent the normal traffic going through the wireless network, the red dotted lines show network traffic generated by attackers, and the blue dashed lines indicate the wireless traffic with

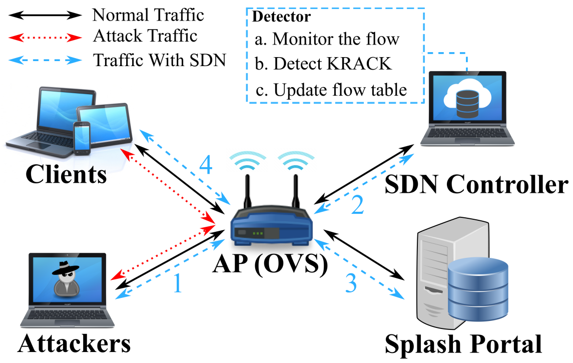

Figure 1: The Proposed Framework.

detection and mitigation. The SDN controller has a global view of this network so it can monitor and manage client authentication requests.

Initially, because of the vulnerability of the 4-way handshake, an attacker may attack the client easily through the AP once the client is connected to this network. When we adopt the SDN paradigm, the SDN controller will monitor the traffic going through the network. The AP functions as the OVS since all end devices are connected to it. The detection module and mitigation module are deployed on the SDN controller in order to take advantage of the controller. As shown in the blue dashed lines of Figure 1, labels 1, 2, 3, and 4 represent four different actions in this framework, respectively. They are discussed in detail as follows. Label 1 represents that an attacker launches the attack through AP. Label 2 indicates that the detector on the SDN controller will a) monitor the traffic, b) detect KRACK, and c) mitigate the attack traffic by updating the flow table. Label 3 means that after the SDN controller updates the entries of the flow table, the attack traffic flows will be redirected to a splash portal. Label 4 means if the traffic flow is not detected as KRACK, it will be sent to the clients. To be precise, we divide the framework into three stages: 1) attack stage, 2) detection stage and 3) mitigation stage.

4.1 Attack Stage

Most Wi-Fi networks are protected by some version of WPA or WPA2, which rely on the 4-way handshake defined in the 802.11i protocol. However, the design of the 4-way handshake has serious flaws that make it vulnerable to key reinstallation attacks. In the attack stage, we perform KRACK proposed in (Vanhoef and Piessens, 2017; Vanhoef and Piessens, 2018). A client/victim is tricked to reinstall an already-in-use key so that the attacker can steal the victim's personal information and even inject false data. KRACK is achieved by replaying and manipulating messages 3 of the 4-way handshake.

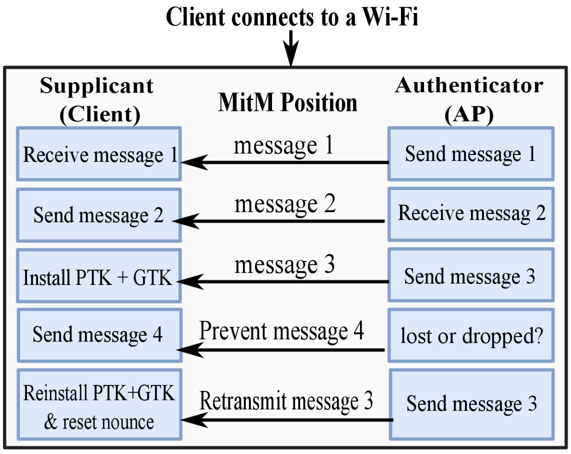

Figure 2: The Procedure of Attacking the 4-way Handshake.

The attack procedure is shown in Figure 2. When a client is trying to connect to a network, the 4-way handshake will be executed. The 4-way handshake provides mutual authentication based on the Pairwise Master Key (PMK) (a shared master secret) and negotiates a PTK. The PTK is derived from the PMK. The supplicant (client) receives message 1 and sends message 2 to the authenticator (AP). The supplicant will install the PTK and the GTK after receiving message 3 from the authenticator. Then, a data-confidentiality protocol will be used to encrypt normal data frames after the installation of the PTK and the GTK. If the supplicant does not send message 4 to the authenticator as the response of acknowledgment, the authenticator will think that message 3 may be dropped or lost. Therefore, the authenticator will retransmit message 3 to the supplicant. As a result, the supplicant will receive message 3 multiple times. Each time the supplicant receives message 3, the same PTK will be reinstalled and the associated parameters, such as receive packet number (i.e. replay counter) and incremental transmit packet number (i.e. nonce), are reset to their initial values.

In KRACK, an attacker will first establish a MitM position between the supplicant and the authenticator. The attacker uses this MitM position to prevent message 4 from arriving at the authenticator. Then, the attacker can utilize the retransmission manner of message 3 to reinstall an already-in-use PTK and force reset the value of nonce and replay counter. By reusing the nonce and the replay counter, the attacker can attack the data-confidentiality protocol, such as replay, as well as decrypt and forge the packets. The above can pose great threats to end user's privacy.

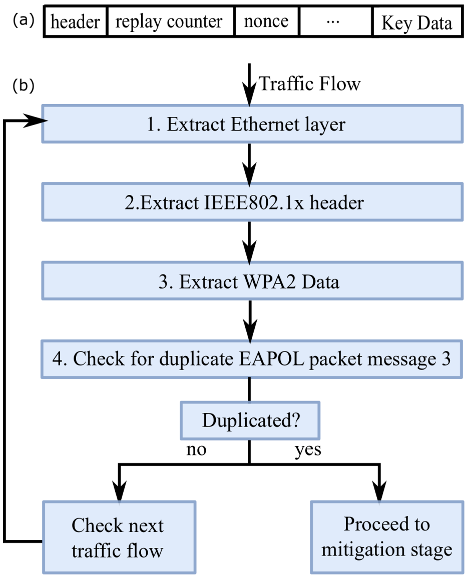
(a)
header
replay counter
nonce
···
Key Data
(b)
Traffic Flow
1. Extract Ethernet layer
2.Extract IEEE802.1x header
3. Extract WPA2 Data
4. Check for duplicate EAPOL packet message 3
Duplicated?
no
yes
Check next traffic flow
Proceed to mitigation stage

Figure 3: (a) Simplified Layout of an EAPOL Frame (b) Procedure of the Detection Stage.

4.2 Detection Stage

Since every message in the 4-way handshake is defined using EAPOL frames, it can be used to detect the duplicated packets. As shown in Figure 3 (a), the header defines the n-th message in the handshake. The replay counter field is used to count the replay frames. When the authenticator sends a message to the supplicant, the replay counter will be increased by one. If the supplicant receives the message and replies to the authenticator, the same replay counter is used. The nonce field can be used to detect the reuse of nonce since it transmits the fresh session key that derived from the random nonces generated by supplicant and authenticator. Therefore, the replay counter filed and nonce field are extremely helpful to detect the reinstallation of an already-in-use key. To be precise, if message 3 is retransmitted, a duplicated nonce will be transmitted and we just need to check if there is a reuse of nonce.

The SDN controller will monitor all the traffic going through the network by the wireless interface. In order to detect KRACK, the traffic packets need to be filtered and captured only EAPOL packets. The detection procedure is shown in Figure 3 (b). First, the Ethernet layer is extracted. Then, the IEEE802.1x header is extracted from the Ethernet layer. Next, WPA key data is extracted from IEEE802.1x header. At last, we analyze the WPA key data and check for the duplicate message 3 from EAPOL packets. If the reuse of nonce is detected, it will proceed to the mitigation stage. If there is no duplicated message 3, then, it will examine the next traffic flow.

4.3 Mitigation Stage

In an SDN network, the controller manages the traffic by adding rules in the flow table in the OVS. The controller communicates with the OVS through a secure channel, which is defined by the OpenFlow Protocol. An entry of the flow table consists of three important fields: a header field, an action field, and a statistic field. The header field is used to define the flow and the statistic field stores the network information of the flow, such as time-stamp of a packet that was last seen and the number of flows forwarded. The path of a flow can be directed in the action field. The flow can be handled by the following actions: forward to a destination port or ports, encapsulate and forward to the controller, or forward through the switch's normal processing pipeline.

In our study, we set up a splash portal to store attack traffic. We utilized the characteristic of the SDN network to update the flow rules in the flow table to prevent the attack traffic to be forwarded to the client. When KRACK is detected in the detection stage by checking the duplication of message 3, it will proceed to the mitigation stage. In this stage, we add a flow rule in the OVS to forward the attack traffic to the splash portal instead of forwarding it to the client. If there is no attack detected, the traffic will be forwarded through its normal path. In this way, the attack traffic will not be forwarded to the client. Meanwhile, we will send an alert to warn the clients.

4.4 Hardware Configuration

In this study, the experiment was set up in the lab environment. We constructed an SDN network to detect and mitigate KRACK. We used a Raspberry Pi 3 Model B, which has the specification of a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and 1GB RAM, as the SDN controller. We utilized a wireless antenna, an Alfa AWUS036NHA High Gain, to capture the nearby traffic. We setup Floodlight (Big Switch Networks, 2018) controller to monitor and update the flow rules. The detection and mitigation scripts were deployed on the SDN controller. A TP-Link AC 1750 Router was used as the AP and was configured as an OVS. A desktop with 16GB of memory and a quad-core processor was used to host two virtual machines: one was used as an attack machine and the other was used as a splash portal. The attack machine was running Kali Linux operating system with 2G of memory and a dual-core processor. To launch KRACK, we utilized two wireless antennas, a TP-Link WN722n v1 and an Alfa AWUS036NHA High Gain. The splash portal was running Ubuntu

Table 1: Times Recorded in the Experiment.

Script	Metrics	Description
Attack Script	$Atk1$	Time at which attack script begins
	$Atk2$	Time at which first message 3 was sent from authenticator
	$Atk3$	Time at which the duplicated message 3 was sent from authenticator
Detection Script	$Det1$	Time at which first message 3 was detected
	$Det2$	Time at which the duplicated message 3 was detected
	$Det3$	Time at which mitigation starts
Mitigation Script	Mtg	Time at which mitigation ends

with 1GB of memory and a single core processor. Lastly, we adopted a Samsung Galaxy S7 Edge to connect to the SDN wireless network with WPA2 encryption to test our experiment setup.

5 EVALUATION

To run the experiment, we first started the Floodlight in the background to monitor the traffic going through the network. Next, we configured the AP to enable OVS functions. We have adopted an online KRACK detector courtesy of Securing Sam's Github script (Sam, 2019) and modified it to meet the goal in our experiment. We have also written a mitigation script to update the flow table and integrated it with the modified detection script. The attack script is adopted from (Vanhoef and Piessens, 2017) with little modification. We have also modified both the attack script and the detection script to keep track of important time metrics as shown in Table 1. The detection script makes use of raw sockets to allow for detailed packet analysis. The analysis takes place byte by byte, ensuring the packet meets the criteria of the message 3 in the WPA2 handshake before continuing. Finally, once we capture the first message 3, the key nonce is stored for future comparison against other packet key nonces. Success at the detection stage led to the mitigation stage, and the controller will reroute the attack traffic to the splash portal by updating the flow table.

5.1 Experimental Results

The time metrics recorded in the attack script, detection script, and mitigation script and their description are showing in Table 1. There are three time metrics in the attack script, three time metrics in the detection script, and one time metric in the mitigation script. By analyzing the time metrics recorded in the experiment, we can obtain some important time measurements such as detection time and mitigation time. We extracted six time measurements to evaluate our detection and mitigation scheme. The detailed description of six time measurements are given as follows:

1. KRACK Detection Time: It measures the time of detecting KRACK. It is the time from the moment the authenticator sent the duplicated message 3 to the moment the duplicated message 3 is detected by the detection module. Therefore, it is calculated by $Det2 - Atk3$.

2. KRACK Mitigation Time: It is defined as the time from the moment the mitigation starts to the moment the mitigation ends. That is, KRACK Mitigation Time $= Mtg - Det3$. After the duplicated message 3 is successfully detected, the mitigation starts, the controller will update the flow table to redirect the traffic to the splash portal. On the splash portal, we use a socket to listen to the traffic and record the time when we receive the first packet redirected to it.

3. First Message 3 Detection Time: This is the time of detecting the first message 3. It is the time from the moment the authenticator sent the first message 3 to the moment the first message 3 is detected by the detection module. First message 3 detection time is $Det1 - Atk2$.

4. Overall Experiment Time: This measures the overall time taken to run the experiment, from the time at which attack starts to the time at which mitigation ends. It is calculated as $Mtg - Atk1$.

5. Attack Script Runtime: It is the time from the moment the attack script begins to the moment the authenticator sends the duplicated message 3. It can be calculated as $Atk3 - Atk1$.

6. Detection Script Runtime: This is the time between the first message 3 detected and the duplicated message 3 detected. It measures the runtime of the detection script. It can be calculated by $Det2 - Det1$.

Figure 4 shows the six time measurements over 500 successful KRACK. The times are recorded in milliseconds. We can see from the figure that the pattern of KRACK detection time, first message 3 detection time, and overall experiment time are similar. Since KRACK detection time is the time between du-

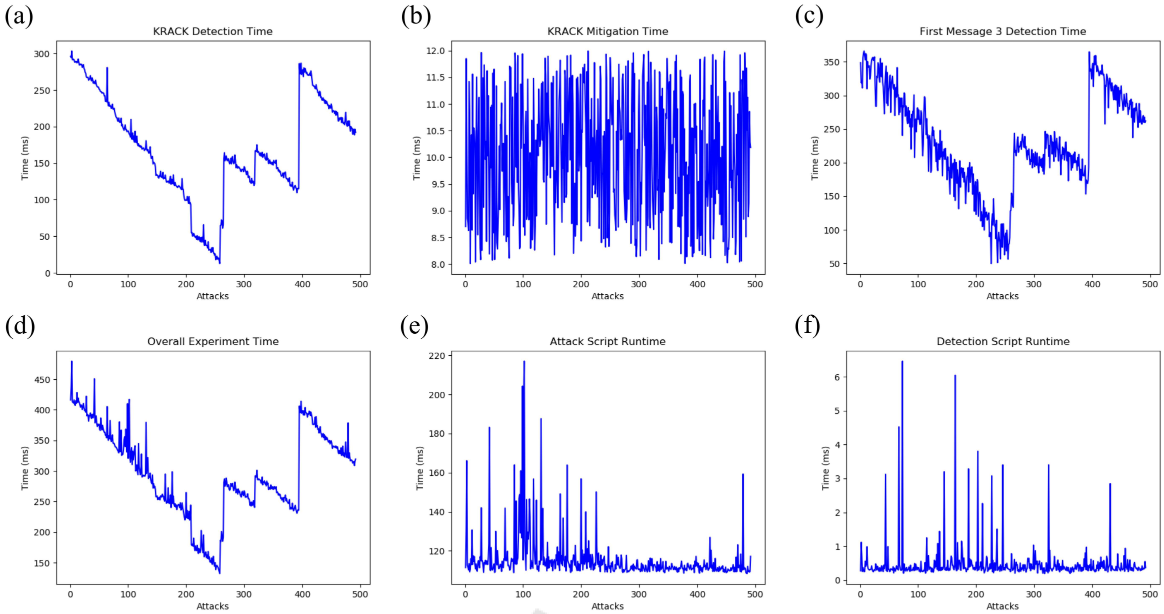

Figure 4: Time Measurements Recorded From 500 KRACK.

Table 2: Time Measurements (ms).

Time Name	Mean	Median	Mode	Max	Min	Std
KRACK Detection Time	170.926	163.527	164.819	303.315	12.738	70.410
KRACK Mitigation Time	10.041	10.107	8.351	11.992	8.01	1.176
First Message 3 Detection Time	233.832	227.832	49.839	366.092	49.839	72.805
Overall Experiment Time	296.467	268.851	336.879	479.595	132.003	71.930
Attack Script Runtime	115.456	112.279	109.22	217.079	108.312	12.085
Detection Script Runtime	0.475	0.341	0.318	6.464	0.192	71.930

plicated message 3 sent and detected and the first message 3 detection time is the time between first message 3 sent and detected, it demonstrates that our detection rate is pretty robust. Figure 4 (b) presents the mitigation time of KRACK, where we can see that the mitigation time is evenly distributed. The mitigation time includes the time controller responds to update the flow table and the communication to OVS to navigate the new path to the splash portal. Figure 4 (e) and (f) shows the running time of both attack script and detection script over 500 KRACKs. Most running times during the attacks are stable except for some attacks, they tend to have longer running time.

Table 2 shows the time analysis of these six time measurements. We analyzed the mean, median, mode, max, min and stand derivation of these time measurements. We can see from the table, a successful KRACK requires 115.456 ms on average while the fastest KRACK detection time is 12.738 ms and the fastest mitigation time is 8.01 ms. We can see that the standard derivation of mitigation time is 1.176 ms, which is very small. It shows that the mitigation time is usually stable in our experiments. The average time to detect the first message 3 packet is 233.832 ms. Detecting KRACK requires an average time of 170.926 ms, and we can see that the maximum KRACK detection time is much higher than the minimum KRACK detection time. There could be a propagation delay in the transmission due to the environmental constraints.

6 CONCLUSIONS AND FUTURE WORK

In this research, we have proposed an SDN-based detection and mitigation framework to defend against KRACK. Because of the global view of an SDN controller, we deployed our detection and mitigation module on the SDN controller to better monitor and manage the attack traffic. In the framework, we have three stages: attack stage, detection stage, and mitigation stage. In the attack stage, an attacker tried to deceive the authenticator to retransmit message 3 and trick the client to reinstall an already-in-use key. In the detection stage, we monitor all the traffic going through the SDN network and check for the duplicated message 3 transmission to detect KRACK. Finally, in the mitigation stage, the flow table was updated to navigate the attack traffic to a splash portal. The experiment was set up in a lab environment and important time metrics were recorded for evaluating the framework. Experimental results show that the detection and mitigation scheme in the proposed framework is very efficient to defend against KRACK.

In the future work, we will study the scalability of the proposed framework. We will conduct the experiments in a large scale real-world network. In the meantime, we will try to attack multiple clients by exploring different attacks within the umbrella of KRACK and investigate machine learning schemes to classify those attacks based on different attack types.

ACKNOWLEDGEMENT

We acknowledge NSF to partially sponsor the research work under grants #1633978, #1620871, #1636622, #1651280, #1620862, and #1620868, and BBN/GPO project #1936 through an NSF/CNS grant.

REFERENCES

Akram, Z., Saeed, M. A., and Daud, M. (2018). Real time exploitation of security mechanisms of residential WLAN access points. In *IEEE iCoMET*, pages 1–5.

Alblwi, S. and Shujaee, K. (2017). A survey on wireless security protocol wpa2. In *Int. Conf. security and management*, pages 12–17.

Cheminod, M., Durante, L., Seno, L., Valenza, F., Valenzano, A., and Zunino, C. (2017). Leveraging SDN to improve security in industrial networks. In *IEEE WFCS*, pages 1–7.

Chin, T. and Xiong, K. (2018). KrackCover: A wireless security framework for covering KRACK attacks. In *WASA*, pages 733–739.

Fehér, D. J. and Sandor, B. (2018). Effects of the WPA2 KRACK attack in real environment. In *IEEE SISY*, pages 239–242.

Ghanem, M. C. and Ratnayake, D. N. (2016). Enhancing WPA2-PSK four-way handshaking after re-authentication to deal with de-authentication followed by brute-force attack a novel re-authentication protocol. In *IEEE CyberSA*, pages 1–7.

Kumkar, V., Tiwari, A., Tiwari, P., Gupta, A., and Shrawne, S. (2012). Vulnerabilities of wireless security protocols (WEP and WPA2). *IJARCET*, 1(2):34–38.

Manzoor, S., Akber, S. M. A., Menhas, M. I., Imran, M., Sajid, M., Talal, H., and Samad, U. (2018). An SDN enhanced load balancing mechanism for a multi-controller wifi network. In *IEEE ICPESG*, pages 1–5.

Big Switch Networks ([Online]. Sept. 2018). Project Floodlight. Available: http://www.projectfloodlight.org/.

Naitik, S., Lobo, R., Vernekar, P. S., and Shetty, V. G. Mitigation of key reinstallation attack in WPA2 Wi-Fi networks by detection of nonce reuse. In *IRJET*, pages 1528–1531.

Noh, J., Kim, J., Kwon, G., and Cho, S. (2016). Secure key exchange scheme for WPA/WPA2-PSK using public key cryptography. In *IEEE ICCE-Asia*, pages 1–4.

Sam, S. ([Online]. 2019). KRACK detector. Available: https://github.com/securingsam/krackdetector.

Scott-Hayward, S., O'Callaghan, G., and Sezer, S. (2013). SDN security: A survey. In *IEEE SDN4FNS*, pages 1–7.

Shin, S., Xu, L., Hong, S., and Gu, G. (2016). Enhancing network security through software defined networking (SDN). In *IEEE ICCCN*, pages 1–9.

Terkawi, A. and Innab, N. (2018). Major impacts of key reinstallation attack on Internet of things system. In *IEEE NCC*, pages 1–6.

Teyou, C. C. T. and Zhang, P. (2018). Solving downgrade and DoS attack due to the four ways handshake vulnerabilities (WIFI). *IJEMR*, 8(4):1–10.

Tsitroulis, A., Lampoudis, D., and Tsekleves, E. (2014a). Exposing WPA2 security protocol vulnerabilities. *IJICS*, 6(1):93–107.

Tsitroulis, A., Lampoudis, D., and Tsekleves, E. (2014b). ExposingWPA2 security protocol vulnerabilities. *IJICS*, 6:93–107.

Vanhoef, M. and Piessens, F. (2017). Key reinstallation attacks: Forcing nonce reuse in WPA2. In *ACM CCS*, pages 1313–1328.

Securing Sam ([Online]. 2019). KRACK attacks white paper. Available: https://www.securingsam.com/index.php/2017/08/03/
krackattacks/.

Vanhoef, M. and Piessens, F. (2018). Release the Kraken: New KRACKs in the 802.11 standard. In *ACM CCS*, pages 299–314.

Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., and Liu, Y. (2018). A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. *IEEE Communications Surveys & Tutorials*.

Yan, Q., Yu, F. R., Gong, Q., and Li, J. (2016). Software-defined networking (SDN) and Distributed Denial of Service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys & Tutorials*, 18:602–622.

Yang, Q. and Huang, L. (2018). Overview of wireless security, attack and defense. In *Inside Radio: An Attack and Defense Guide*, pages 1–5.