CheckShake: Passively Detecting Anomaly in Wi-Fi Security Handshake using Gradient Boosting based Ensemble Learning

Anand Agrawal
anandgarg91@gmail.com
Birla Institute Technology and Science, Hyderabad

Urbi Chatterjee
urbic@cse.iitk.ac.in
Indian Institute of Technology, Kanpur

Rajib Ranjan Maiti
rajibrm@hyderabad.bits-pilani.ac.in
Birla Institute Technology and Science, Hyderabad

ABSTRACT

Recently, a number of attacks have been demonstrated (like key reinstallation attack, called KRACK) on WPA2 protocol suite in Wi-Fi WLAN. As the firmware of the WLAN devices in the context of IoT, industrial systems, and medical devices is often not patched, detecting and preventing such attacks is challenging. In this paper, we design and implement a system, called CheckShake, to passively detect anomalies in the handshake of Wi-Fi security protocols, in particular WPA2, between a client and an access point using COTS radios. Our proposed system works without decrypting any traffic. It passively monitors multiple wireless channels in parallel in the neighbourhood and uses a state machine model to characterize and detect the attacks. In particular, we develop a state machine model for grouping Wi-Fi handshake packets and then perform deep packet inspection to identify the symptoms of the anomaly in specific stages of a handshake session. Our implementation of CheckShake does not require any modification to the firmware of the client or the access point or the COTS devices, it only requires to be physically placed within the range of the access point and its clients. We use both the publicly available dataset and our own data set for performance analysis of CheckShake. Using gradient boosting-based supervised machine learning models, we show that an accuracy around 93.39% and a false positive rate of 5.08% can be achieved using CheckShake.

KEYWORDS

Wi-Fi, WPA2, KRACK, Ensemble model

ACM Reference Format:
Anand Agrawal, Urbi Chatterjee, and Rajib Ranjan Maiti. 2021. CheckShake: Passively Detecting Anomaly in Wi-Fi Security Handshake using Gradient Boosting based Ensemble Learning. In *Proceedings of (Cryptology eprint'21)*. eprint, IACR , 3 pages.

1 INTRODUCTION

Recently, a number of attacks has been reported that targets authentication, key negotiation, and encryption schemes in the WPA2 protocol suite, e.g., the WiFi key re-installation attack (KRACK) [9], and active and passive eave dropping attacks on Wi-Fi [2]. While some of those attacks could potentially be prevented by firmware updates of the involved devices, such updates are often not possible for a variety of reasons, including but not limited to, cost-benefit trade-offs to end users. A number of wireless intrusion detection systems (WIDS) have been proposed to address MAC-layer misbehaviour[1, 7], but those systems are focused on detecting specific individual malicious packets that indicate the misbehaviour of a set of devices/users in the neighborhood. Unfortunately, the attacks that manipulate the security handshake in wireless environment cannot be detected by the existing anomaly detection systems, we claim that the detection of such attacks requires a stateful observation of the packets being exchanged on (multiple) wireless channels.

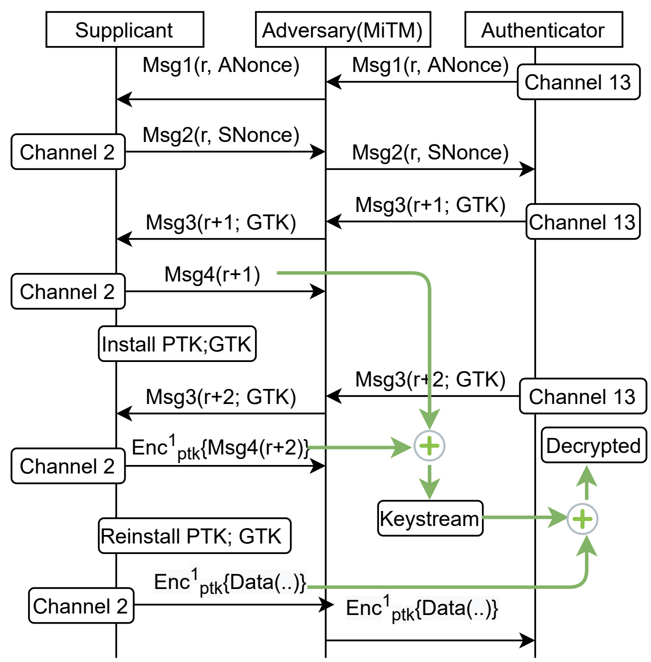

Figure 1: KRACK Attack

In this work, we aim to design and develop a tool that can passively capture Wi-Fi packets pertaining to connection establishment rather than data transfers and deep inspect the packets to look for anomalies in WPA2 handshakes. Secure connection in Wi-Fi goes through a well-defined sequence of probing, authenticating, associating, and handshaking (i.e., performing a 4-way handshake) that can be captured by a state machine which in turn can be used to detect anomalies in the establishment of security connection between a pair of an authenticator and a supplicant. We leverage this stateful property to secure connection establishment and check if the state machine executes through appropriate states in estimated time

Cryptology eprint'21, Archive,

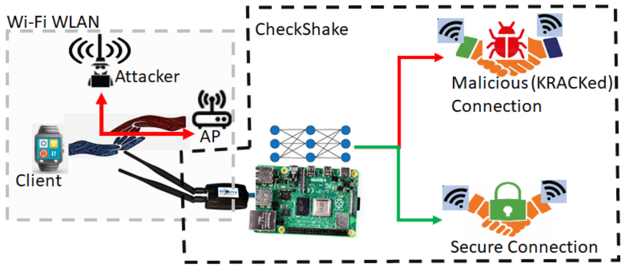

Figure 2: System model

frame, and any violation of the state machine potentially indicates an anomaly.

2 KRACK CHARACTERIZATION

KRACK attack, [9], exploits a weakness of 18-year-old WPA/WPA2 Wi-Fi security protocol. Recently, in [4], Wi-Fi traffic containing KRACK attack has been published for public use and further research. In this paper, for the first time, we inspect this trace in detail and identify the handshakes in the trace showing the symptoms of such an attack, and if not, then we specify the violated preconditions if any. Figure 1 shows the arrangement of KRACK attack, where the adversary is a channel-based MitM. The adversary impersonates the supplicant on channel 13 that the legitimate AP (Authenticator) is active, and it impersonates the authenticator on channel 2 on which the supplicant is already tricked to operate on.

3 DETECTING ANOMALY IN HANDSHAKE

In this section, system model depicts in Figure 2, and provide an overview of our newly proposed CheckShake framework.

3.1 Design of CheckShake

This section describes in detail the architecture of CheckShake that we propose in this paper for the first time. Passive detection of advanced attacks, like KRACK, involves efficient packet sniffing on more than one channel and deep inspection of the packets across different phases of Wi-Fi connection establishment, like authentication, association, 4-way handshake as well as beacon packets on multiple channels. We carefully decompose the system into subsystems and reduce the dependency on each others. CheckShake contains six different modules and each module performs a well-defined task. Figure 3 shows an architectural view of CheckShake

3.2 Session Grouping Module

The Grouping Module takes feature vectors sequentially as input and segregates the handshake sessions following the Mealy State Machine (MSM) model shown in Figure 4. Packets corresponding to one handshake session (i.e. packets from authentication to ideally *m4*) are put in a single group. The output g_1 (resp. g_2) in each transition in MSM indicates the current (resp. new) group of packets. The MSM starts with q_0 state when first *auth* request packet creates a handshake session indicated by g_1 as output of the transition. On successful authentication, client sends the *assoc* request to AP and the machine transits to q_2 state keeping the packet in the same

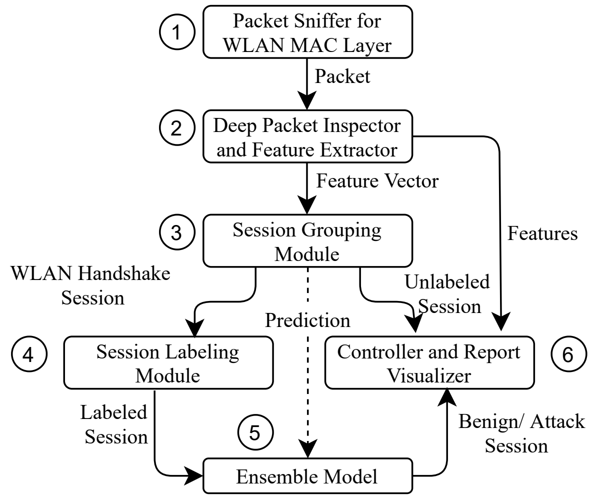

Figure 3: Different Modules of the CheckShake

group indicated by the transition output g_1. It waits in this state till repeated *assoc* (request or response) packets are seen. The machine moves to q_0 from q_1 if an *auth* request is observed, creating a new group indicated by the transition output g_2. The machine proceeds to q_2 from q_1 on seeing *m1* without creating any new group and remains in this state till *m1* is repeated. If an *auth* request/response is seen in q_2, then this packet is assigned to a new group and the machine moves to q_0 state. Alternatively, if any *assoc* packet is seen then the state changes to q_1 keeping the packets in the same group. In ideal case, the state changes to q_3 from q_2, if *m2* is observed. In state q_3, if an *auth* request is seen, then a new group is created and the state changes to q_0, however the state changes to q_1 or q_2 on seeing an *assoc* packet or *m1* respectively. In ideal case, the state changes to q_4 from q_3, if m_3 is seen. A new group is created only if an *auth* packet is seen in this state and the machine transits to q_0. Alternately, the state changes to q_1, q_2 or q_3, if *assoc* packet or m_1 or m_2 is seen respectively without creating any new group of packets. Finally, the state changes to q_5 from q_4, if m_4 is seen on fake channel and in ideal case changes to q_6, if m_4 i.e., $m'4$ (shown in state machine) is seen on legitimate channel and this indicates a successful and secure handshake.

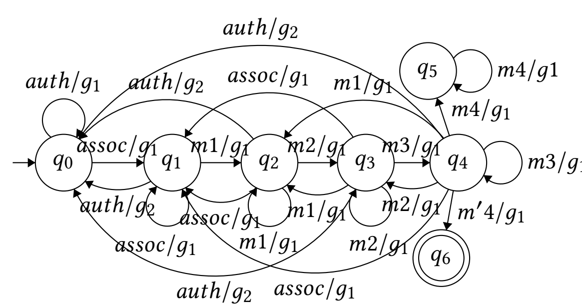

Figure 4: State Machine for Grouping Algorithm

Algorithm 1: Grouping Algorithm, uses Python syntax.

```
Input: f_list = List of Features extracted from EAPOL frame
Output: Grouping of different handshakes
group_dict = {count:[] for count in num_of_group}
for pkt in packet[f_list] do
    field_values = [], set_re=0, retransmit=0, group=[]
    count=0,layer_type = type(pkt[f_list].payload)
    if (isAuth(pkt)) then
        count = count+1, group.append(count)/* new group */
    if (isAssoc(pkt)) or (isM1(pkt)) or (isM2(pkt)) then
        group.append(count),
    if (isM3(pkt)) then
        if (set_re == 0) then
            group.append(count), set_re = 1
        if (set_re == 1) then
            group.append("remsg3"+str(count))
            retransmit = retransmit+1, set_re = 1
    if (isM4(pkt)) then
        group.append(count), count = count+1, set_re=0
    field_values.append(layer_type).fields[field]
    group_dict[count].append(pkt)
```

Table 1: Classification performance. 2C and 3C indicates binary and ternary classification and FPR is false positive rate.

Classifier	XGboost		LightGBM		Catboost	
Metric \ Label	2C	3C	2C	3C	2C	3C
Precision	87.76	82.52	87.68	82.33	91.04	82.25
Recall	87.12	82.57	87.12	82.57	90.15	82.57
F1 Score	87.15	82.37	87.15	82.18	90.17	82.08
Accuracy	87.12	82.57	87.12	82.57	90.15	82.57
FPR	8.47	10.16	8.47	8.47	5.08	5.08

Alternately, the state reaches to q_0 if *auth* packet is seen and a new group is created. Otherwise, the state transits to q_1, q_2, q_3, or q_4 state if *assoc* packet, *m1*, *m2* or *m3* is seen respectively without creating any new group. The state machine in Figure 4 is implemented using Algorithm 1 in Module 3. We use simple functions to check if a packet, pkt, is of desired type, e.g., `isAuth(pkt)` function returns *true* if pkt.type =0 and pkt.subtype = 11. Similarly, `isAssoc(pkt)`, `isM1(pkt)`, `isM2(pkt)`, `isM3(pkt)`, and `isM4(pkt)` functions return true if pkt is identified as *(re)assoc* request or response packet, *m1*, *m2*, *m3* or *m4* packet in 4-way handshake respectively.

3.3 Performance of CheckShake

Each column in Table 1 indicates the performance of both 2-class and 3-class classification. we observe that a maximum accuracy of 93.39% and weighted average accuracy of 90.15% for binary classification can be achieved with Catboost. The same in both LightGBM and XGboost is fixed at 87.12%. Though these initial results are exciting, in-depth feature engineering, better ML model selection and model validation are evident to reduce FPR, for instance, and we consider this as a part of our future exploration.

4 RELATED WORK

Cremers et al. in [5], have implemented WPA2 protocol suite in an automated security analysis tool, called *Tamarin prover*, to model the behavior of the protocols in presence of different attacks, in particular KRACK. The aim of their work is to show whether the proposed security patches can address the recent attacks like KRACK. In [6], Yi Li et al. proposed a software-defined network-based framework which replicates the behaviour of a client and an AP and requires that the AP transfers the packet of handshakes to the SDN controller. The controller is then responsible for detecting and mitigating the attacks like KRACK by using duplicated *m3*. In [3], Urbi et al. have proposed to create mutually authenticated APs and supplicants by using Physically Unclonable Functions (PUFs) and hence eliminate the possibility of creating a rogue AP which is one of the preconditions of launching KRACK attack. Authors in [8] have proposed a fuzzing mechanism to test whether a station (AP or client) is vulnerable to certain attacks like deauthentication.

5 CONCLUSION

The presented work is a first attempt to passively detect attacks like KRACK. Any client after joining the network starts exchanging the handshake message with AP, and their handshake message can indulge with other client messages, to avoid this scenario, separate state machine specific to MAC address of the client and AP is created. Our algorithm also have an identifying mechanism to distinguish between the genuine or fake retransmission of *m3* based on the channel it is originated from. The final goal of this line of work is to develop a fully automated KRACK attack detection tool.

REFERENCES

[1] Abdessadek Aaroud, Mohammed-Alamine El Houssaini, Ali El Hore, and Jalel Ben-Othman. 2017. Real-time detection of MAC layer misbehavior in mobile ad hoc networks. *In proc. of Applied Computing and Informatics* 13, 1 (2017), 1–9.

[2] Daniele Antonioli, Sandra Siby, and Nils Ole Tippenhauer. 2017. Practical Evaluation of Passive COTS Eavesdropping in 802.11b/n/ac WLAN. In *Proc. of Conference on Cryptology And Network Security (CANS)*.

[3] Urbi Chatterjee, Rajat Sadhukhan, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, Debashis Mahata, and Mukesh. M. Prabhu. 2020. Stupify: A Hardware Countermeasure of KRACKs in WPA2 Using Physically Unclonable Functions *(In proc. of WWW '20)*. Association for Computing Machinery, New York, NY, USA.

[4] E. Chatzoglou, G. Kambourakis, and C. Kolias. 2021. Empirical evaluation of attacks against IEEE 802.11 enterprise networks: The AWID3 dataset. *In IEEE Access* 9 (2021), 34188–34205.

[5] Cas Cremers, Benjamin Kiesl, and Niklas Medinger. 2020. A Formal Analysis of IEEE 802.11's WPA2: Countering the Kracks Caused by Cracking the Counters. In *29th USENIX Security Symposium*. USENIX Association, 1–17.

[6] Yi Li, Marcos Serrano, Tommy Chin, Kaiqi Xiong, and Jing Lin. 2019. A Software-defined Networking-based Detection and Mitigation Approach against KRACK. In *Proc. of the 16th International Joint Conference on e-Business and Telecommunications - Volume 2: SECRYPT,*. INSTICC, SciTePress, 244–251.

[7] Svetlana Radosavac, A. Cárdenas, J. Baras, and G. Moustakides. 2007. Detecting IEEE 802.11 MAC layer misbehavior in ad hoc networks: Robust strategies against individual and colluding attackers. *J. Comput. Secur.* 15 (2007), 103–128.

[8] Domien Schepers, Mathy Vanhoef, and Aanjhan Ranganathan. 2021. In *Proc. of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Abu Dhabi, United Arab Emirates) *(WiSec '21)*. Association for Computing Machinery, New York, NY, USA, 368–370. https://doi.org/10.1145/3448300.3468261

[9] Mathy Vanhoef and Frank Piessens. 2017. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In *Proc. of the 24th ACM Conference on Computer and Communications Security (CCS)*. ACM.