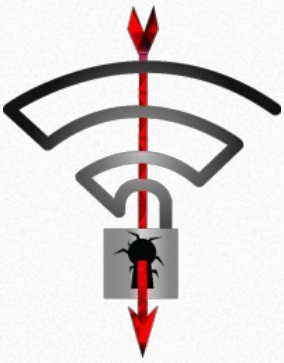


Key Reinstallation Attacks

Breaking WPA2 by forcing nonce reuse

Discovered by Mathy Vanhoef of imec-DistriNet, KU Leuven

INTRODUCTION

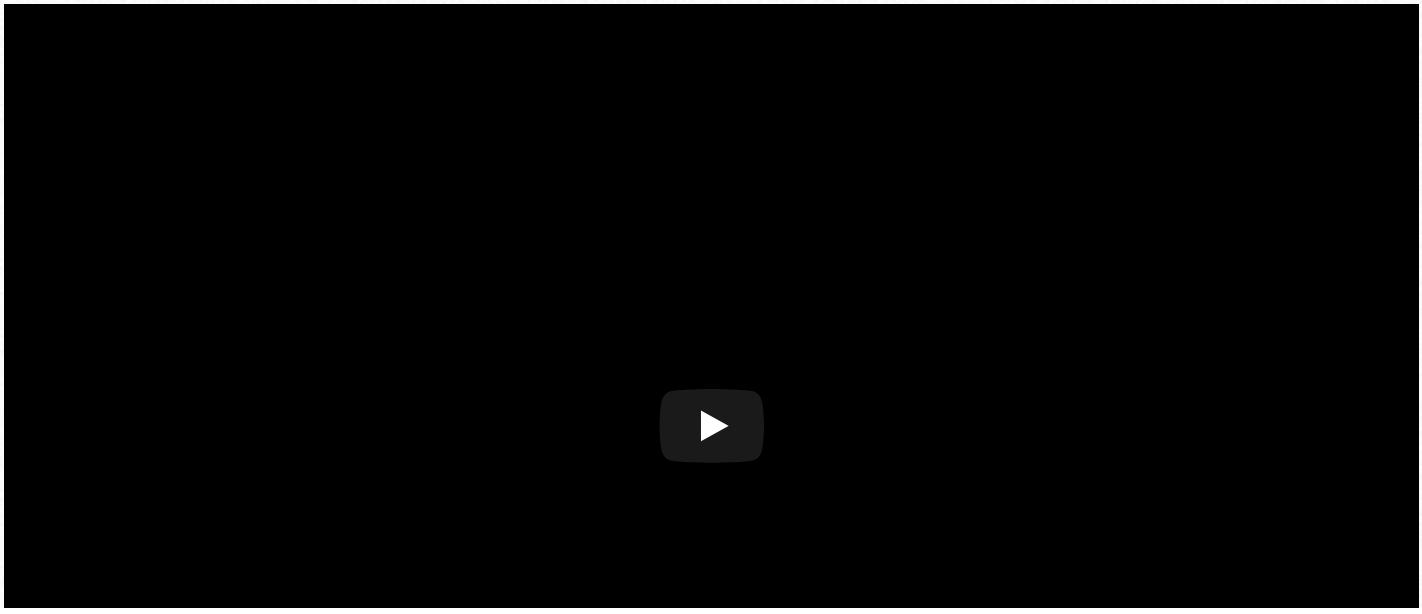
We discovered serious weaknesses in WPA2, a protocol that secures all modern protected Wi-Fi networks. An attacker within range of a victim can exploit these weaknesses using key reinstallation attacks (KRACKs). Concretely, attackers can use this novel attack technique to read information that was previously assumed to be safely encrypted. This can be abused to steal sensitive information such as credit card numbers, passwords, chat messages, emails, photos, and so on. **The attack works against all modern protected Wi-Fi networks**. Depending on the network configuration, it is also possible to inject and manipulate data. For example, an attacker might be able to inject ransomware or other malware into websites.

The weaknesses are in the Wi-Fi standard itself, and not in individual products or implementations. Therefore, any correct implementation of WPA2 is likely affected. To prevent the attack, users must update affected products as soon as security updates become available. Note that **if your device supports Wi-Fi, it is most likely affected**. During our initial research, we discovered ourselves that Android, Linux, Apple, Windows, OpenBSD, MediaTek, Linksys, and others, are all affected by some variant of the attacks. For more information about specific products, consult the database of CERT/CC, or contact your vendor.

The research behind the attack will be presented at the Computer and Communications Security (CCS) conference, and at the Black Hat Europe conference. Our detailed research paper can already be downloaded.

DEMONSTRATION

As a proof-of-concept we executed a key reinstallation attack against an Android smartphone. In this demonstration, the attacker is able to decrypt all data that the victim transmits. For an attacker this is easy to accomplish, because our key reinstallation attack is exceptionally devastating against Linux and Android 6.0 or higher. This is because **Android and Linux can be tricked into (re)installing an all-zero encryption key** (see below for more info). When attacking other devices, it is harder to decrypt all packets, although a large number of packets can nevertheless be decrypted. In any case, the following demonstration highlights the type of information that an attacker can obtain when performing key reinstallation attacks against protected Wi-Fi networks:



Our attack is not limited to recovering login credentials (i.e. e-mail addresses and passwords). In general, any data or information that the victim transmits can be decrypted. Additionally, depending on the device being used and the network setup, it is also possible to decrypt data sent towards the victim (e.g. the content of a website). Although websites or apps may use HTTPS as an additional layer of protection, we warn that this extra protection can (still) be bypassed in a worrying number of situations. For example, HTTPS was previously bypassed in non-browser software, in Apple's iOS and OS X, in Android apps, in Android apps again, in banking apps, and even in VPN apps.

DETAILS

Our main attack is against the 4-way handshake of the WPA2 protocol. This handshake is executed when a client wants to join a protected Wi-Fi network, and is used to confirm that both the client and access point possess the correct credentials (e.g. the pre-shared password of the network). At the same time, the 4-way handshake also negotiates a fresh encryption key that will be used to encrypt all subsequent traffic. Currently, all modern protected Wi-Fi networks use the 4-way handshake. This implies all these networks are affected by (some variant of) our attack. For instance, the attack works against personal and enterprise Wi-Fi networks, against the older WPA and the latest WPA2 standard, and even against networks that only use AES. **All our attacks against WPA2 use a novel technique called a key reinstallation attack (KRACK):**

Key reinstallation attacks: high level description

In a key reinstallation attack, the adversary tricks a victim into reinstalling an already-in-use key. This is **achieved by manipulating and replaying cryptographic handshake messages**. When the victim reinstalls the key, associated parameters such as the incremental transmit packet number (i.e. nonce) and receive packet number (i.e. replay counter) are reset to their initial value. Essentially, to guarantee security, a key should only be installed and used once. Unfortunately, we found this is not guaranteed by the WPA2 protocol. By manipulating cryptographic handshakes, we can abuse this weakness in practice.

Key reinstallation attacks: concrete example against the 4-way handshake

As described in the introduction of the research paper, the idea behind a key reinstallation attack can be summarized as follows. When a client joins a network, it executes the 4-way handshake to negotiate a fresh encryption key. It will install this key after receiving message 3 of the 4-way handshake. Once the key is installed, it will be used to encrypt normal data frames using an encryption protocol. However, because messages may be lost or dropped, the Access Point (AP) will retransmit message 3 if it did not receive an appropriate response as acknowledgment. As a result, the client may receive message 3 multiple times. Each time it receives this message, it will reinstall the same encryption key, and thereby reset the incremental transmit packet number (nonce) and receive replay counter used by the encryption protocol. We show that **an attacker can force these nonce resets by collecting and replaying retransmissions of message 3 of the 4-way handshake**. By forcing nonce reuse in this manner, the encryption protocol can be attacked, e.g., packets can be replayed, decrypted, and/or forged. The same technique can also be used to attack the group key, PeerKey, TDLS, and fast BSS transition handshake.

Practical impact

In our opinion, the most widespread and practically impactful attack is the key reinstallation attack against the 4-way handshake. We base this judgement on two observations. First, during our own research we found that most clients were affected by it. Second, adversaries can use this attack to decrypt packets sent by clients, allowing them to intercept sensitive information such as passwords or cookies. Decryption of packets is possible because a key reinstallation attack causes the transmit nonces (sometimes also called packet numbers or initialization vectors) to be reset to zero. As a result, **the same encryption key is used with nonce values that have already been used in the past**. In turn, this causes all encryption protocols of WPA2 to reuse keystream when encrypting packets. In case a message that reuses keystream has known content, it becomes trivial to derive the used keystream. This keystream can then be used to decrypt messages with the same nonce. When there is no known content, it is harder to decrypt packets, although still possible in several cases (e.g. English text can still be decrypted). In practice, finding packets with known content is not a problem, so it should be assumed that any packet can be decrypted.

The ability to decrypt packets can be used to decrypt TCP SYN packets. This allows an adversary to obtain the TCP sequence numbers of a connection, and hijack TCP connections. As a result, even though WPA2 is used, the adversary can now perform one of the most common attacks against open Wi-Fi networks: injecting malicious data into unencrypted HTTP connections. For example, an attacker can abuse this to inject ransomware or malware into

websites that the victim is visiting.

If the victim uses either the WPA-TKIP or GCMP encryption protocol, instead of AES-CCMP, the impact is especially catastrophic. **Against these encryption protocols, nonce reuse enables an adversary to not only decrypt, but also to forge and inject packets**. Moreover, because GCMP uses the same authentication key in both communication directions, and this key can be recovered if nonces are reused, it is especially affected. Note that support for GCMP is currently being rolled out under the name Wireless Gigabit (WiGig), and is expected to be adopted at a high rate over the next few years.

The direction in which packets can be decrypted (and possibly forged) depends on the handshake being attacked. Simplified, when attacking the 4-way handshake, we can decrypt (and forge) packets sent *by* the client. When attacking the Fast BSS Transition (FT) handshake, we can decrypt (and forge) packets sent *towards* the client. Finally, most of our attacks also allow the replay of unicast, broadcast, and multicast frames. For further details, see Section 6 of our research paper.

Note that our attacks **do not recover the password of the Wi-Fi network**. They also do not recover (any parts of) the fresh encryption key that is negotiated during the 4-way handshake.

Android and Linux

Our attack is especially catastrophic against version 2.4 and above of wpa_supplicant, a Wi-Fi client commonly used on Linux. Here, the client will install an all-zero encryption key instead of reinstalling the real key. This vulnerability appears to be caused by a remark in the Wi-Fi standard that suggests to clear the encryption key from memory once it has been installed for the first time. When the client now receives a retransmitted message 3 of the 4-way handshake, it will reinstall the now-cleared encryption key, effectively installing an all-zero key. Because Android uses wpa_supplicant, Android 6.0 and above also contains this vulnerability. This makes it **trivial to intercept and manipulate traffic sent by these Linux and Android devices**. Note that currently 50% of Android devices are vulnerable to this exceptionally devastating variant of our attack.

Assigned CVE identifiers

The following Common Vulnerabilities and Exposures (CVE) identifiers were assigned to track which products are affected by specific instantiations of our key reinstallation attack:

- CVE-2017-13077: Reinstallation of the pairwise encryption key (PTK-TK) in the 4-way handshake.
- CVE-2017-13078: Reinstallation of the group key (GTK) in the 4-way handshake.
- CVE-2017-13079: Reinstallation of the integrity group key (IGTK) in the 4-way handshake.
- CVE-2017-13080: Reinstallation of the group key (GTK) in the group key handshake.
- CVE-2017-13081: Reinstallation of the integrity group key (IGTK) in the group key handshake.
- CVE-2017-13082: Accepting a retransmitted Fast BSS Transition (FT) Reassociation Request and reinstalling the pairwise encryption key (PTK-TK) while processing it.
- CVE-2017-13084: Reinstallation of the STK key in the PeerKey handshake.
- CVE-2017-13086: reinstallation of the Tunneled Direct-Link Setup (TDLS) PeerKey (TPK) key in the TDLS handshake.
- CVE-2017-13087: reinstallation of the group key (GTK) when processing a Wireless Network Management (WNM) Sleep Mode Response frame.
- CVE-2017-13088: reinstallation of the integrity group key (IGTK) when processing a Wireless Network Management (WNM) Sleep Mode Response frame.

Note that each CVE identifier represents a specific instantiation of a key reinstallation attack. This means each CVE ID describes a specific protocol vulnerability, and therefore **many vendors are affected by each individual CVE ID**. You can also read vulnerability note VU#228519 of CERT/CC for additional details on which products are known to be affected.

PAPER

Our research paper behind the attack is titled Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2 and will be presented at the Computer and Communications Security (CCS) conference on Wednesday 1 November 2017.

Although this paper is made public now, it was already submitted for review on 19 May 2017. After this, only minor changes were made. As a result, the findings in the paper are already several months old. In the meantime, we have found easier techniques to carry out our key reinstallation attack against the 4-way handshake. With our novel attack technique, it is now trivial to exploit implementations that only accept encrypted retransmissions of message 3 of the 4-way handshake. In particular this means that **attacking macOS and OpenBSD is significantly easier than discussed in the paper**.

We would like to highlight the following addendums and errata:

Addendum: wpa_supplicant v2.6 and Android 6.0+

Linux's wpa_supplicant v2.6 is also vulnerable to the installation of an all-zero encryption key in the 4-way handshake. This was discovered by John A. Van Boxtel. As a result, all Android versions higher than 6.0 are also affected by the attack, and hence can be tricked into installing an all-zero encryption key. The new attack works by injecting a forged message 1, with the same ANonce as used in the original message 1, before forwarding the retransmitted message 3 to the victim.

Addendum: other vulnerable handshakes

After our initial research as reported in the paper, we discovered that the TDLS handshake and WNM Sleep Mode Response frame are also vulnerable to key reinstallation attacks.

Selected errata

- In Figure 9 at stage 3 of the attack, the frame transmitted from the adversary to the authenticator should say a ReassoReq instead of ReassoResp.

TOOLS

We have made scripts to detect whether an implementation of the 4-way handshake, group key handshake, or Fast BSS Transition (FT) handshake is vulnerable to key reinstallation attacks. These scripts will be released once we have had the time to clean up their usage instructions.

We also made a proof-of-concept script that exploits the all-zero key (re)installation present in certain Android and Linux devices. This script is the one that we used in the demonstration video. It will be released once everyone has had a reasonable chance to update their devices (and we have had a chance to prepare the code repository for release). We remark that the reliability of our proof-of-concept script may depend on how close the victim is to the real network. If the victim is very close to the real network, the script may fail because the victim will always directly communicate with the real network, even if the victim is (forced) onto a different Wi-Fi channel than this network.

Q&A

Do we now need WPA3?

No, luckily **implementations can be patched in a backwards-compatible manner**. This means a patched client can still communicate with an unpatched access point (AP), and vice versa. In other words, a patched client or access point sends exactly the same handshake messages as before, and at exactly the same moment in time. However, the security updates will assure a key is only installed once, preventing our attack. So again, update all your devices once security updates are available. Finally, although an unpatched client can still connect to a patched AP, and vice versa, *both* the client and AP must be patched to defend against all attacks!

Should I change my Wi-Fi password?

Changing the password of your Wi-Fi network does not prevent (or mitigate) the attack. So you do not have to update the password of your Wi-Fi network. Instead, you should make sure all your devices are updated, and you should also update the firmware of your router. Nevertheless, after updating both your client devices and your router, it's never a bad idea to change the Wi-Fi password.

I'm using WPA2 with only AES. That's also vulnerable?

Yes, that network configuration is also vulnerable. The attack works against both WPA1 and WPA2, against personal and enterprise networks, and against any cipher suite being used (WPA-TKIP, AES-CCMP, and GCMP). So everyone should update their devices to prevent the attack!

You use the word "we" in this website. Who is we?

I use the word "we" because that's what I'm used to writing in papers. In practice, all the work is done by me, with me being Mathy Vanhoef. My awesome supervisor is added under an honorary authorship to the research paper for his excellent general guidance. But all the real work was done on my own. So the author list of academic papers does not represent division of work :)

Is my device vulnerable?

Probably. Any device that uses Wi-Fi is likely vulnerable. Contact your vendor for more information.

What if there are no security updates for my router?

Our main attack is against the 4-way handshake, and does not exploit access points, but instead targets clients. So it might be that your router does not require security updates. We strongly advise you to contact your vendor for more details. In general though, you can try to mitigate attacks against routers and access points by disabling client functionality (which is for example used in repeater modes) and disabling 802.11r (fast roaming). For ordinary home users, your priority should be updating clients such as laptops and smartphones.

How did you discover these vulnerabilities?

When working on the final (i.e. camera-ready) version of another paper, I was double-checking some claims we made regarding OpenBSD's implementation of the 4-way handshake. In a sense I was slacking off, because I was supposed to be just finishing the paper, instead of staring at code. But there I was, inspecting some code I already read a hundred times, to avoid having to work on the next paragraph. It was at that time that a particular call to ic_set_key caught my attention. This function is called when processing message 3 of the 4-way handshake, and it installs the pairwise key to the driver. While staring at that line of code I thought *"Ha. I wonder what happens if that function is called twice"*. At the time I (correctly) guessed that calling it twice might reset the nonces associated to the key. And since message 3 can be retransmitted by the Access Point, in practice it might indeed be called twice. *"Better make a note of that. Other vendors might also call such a function twice. But let's first finish this paper..."*. A few weeks later, after finishing the paper and completing some other work, I investigated this new idea in more detail. And the rest is history.

The 4-way handshake was mathematically proven as secure. How is your attack possible?

The brief answer is that the formal proof does not assure a key is installed once. Instead, it only assures the negotiated key remains secret, and that handshake messages cannot be forged.

The longer answer is mentioned in the introduction of our research paper: our attacks do not violate the security properties proven in formal analysis of the 4-way handshake. In particular, these proofs state that the negotiated encryption key remains private, and that the identity of both the client and Access Point (AP) is confirmed. Our attacks do not leak the encryption key. Additionally, although normal data frames can be forged if TKIP or GCMP is used, an attacker cannot forge handshake messages and hence cannot impersonate the client or AP during handshakes. Therefore, the properties that were proven in formal analysis of the 4-way handshake remain true. However, the problem is that the proofs do not model key installation. Put differently, the formal models did not define when a negotiated key should be installed. In practice, this means the same key can be installed multiple times, thereby resetting nonces and replay counters used by the encryption protocol (e.g. by WPA-TKIP or AES-CCMP).

Some attacks in the paper seem hard

We have follow-up work making our attacks (against macOS and OpenBSD for example) significantly more general and easier to execute. So although we agree that some of the attack scenarios in the paper are rather impractical, do not let this fool you into believing key reinstallation attacks cannot be abused in practice.

If an attacker can do a man-in-the-middle attack, why can't he just decrypt all the data?

As mentioned in the demonstration, the attacker first obtains a man-in-the-middle (MitM) position between the victim and the real Wi-Fi network (called a channel-based MitM position). However, this MitM position does not enable the attacker to decrypt packets! This position only allows the attacker to reliably delay, block, or replay *encrypted* packets. So at this point in the attack, he or she cannot yet decrypt packets. Instead, the ability to reliably delay and block packets is used to execute a key reinstallation attack. After performing a key reinstallation attack, packets can be decrypted.

Are people exploiting this in the wild?

We are not in a position to determine if this vulnerability has been (or is being) actively exploited in the wild. That said, key reinstallations can actually occur spontaneously without an adversary being present! This may for example happen if the last message of a handshake is lost due to background noise, causing a retransmission of the previous message. When processing this retransmitted message, keys may be reinstalled, resulting in nonce reuse just like in a real attack.

Should I temporarily use WEP until my devices are patched?

NO! Keep using WPA2.

Will the Wi-Fi standard be updated to address this?

There seems to be an agreement that the Wi-Fi standard should be updated to explicitly prevent our attacks. These

updates likely will be backwards-compatible with older implementations of WPA2. Time will tell whether and how the standard will be updated.

Is the Wi-Fi Alliance also addressing these vulnerabilities?

For those unfamiliar with Wi-Fi, the Wi-Fi Alliance is an organization which certifies that Wi-Fi devices conform to certain standards of interoperability. Among other things, this assures that Wi-Fi products from different vendors work well together.

The Wi-Fi Alliance has a plan to help remedy the discovered vulnerabilities in WPA2. Summarized, they will:

- Require testing for this vulnerability within their global certification lab network.
- Provide a vulnerability detection tool for use by any Wi-Fi Alliance member (this tool is based on my own detection tool that determines if a device is vulnerable to some of the discovered key reinstallation attacks).
- Broadly communicate details on this vulnerability, including remedies, to device vendors. Additionally, vendors are encouraged to work with their solution providers to rapidly integrate any necessary patches.
- Communicate the importance for users to ensure they have installed the latest recommended security updates from device manufacturers.

Why did you use match.com as an example in the demonstration video?

Users share a lot of personal information on websites such as match.com. So this example highlights all the sensitive information an attacker can obtain, and hopefully with this example people also better realize the potential (personal) impact. We also hope this example makes people aware of all the information these dating websites may be collecting.

How can these types of bugs be prevented?

We need more rigorous inspections of protocol implementations. This requires help and additional research from the academic community! Together with other researchers, we hope to organize workshop(s) to improve and verify the correctness of security protocol implementations.

Why the domain name krackattacks.com?

First, I'm aware that KRACK attacks is a pleonasm, since KRACK stands for key reinstallation attack and hence already contains the word attack. But the domain name rhymes, so that's why it's used.

Did you get bug bounties for this?

I haven't applied for any bug bounties yet, nor have I received one already.

How does this attack compare to other attacks against WPA2?

This is the first attack against the WPA2 protocol that doesn't rely on password guessing. Indeed, other attacks against WPA2-enabled network are against surrounding technologies such as Wi-Fi Protected Setup (WPS), or are attacks against older standards such as WPA-TKIP. Put differently, none of the existing attacks were against the 4-way handshake or against cipher suites defined in the WPA2 protocol. In contrast, our key reinstallation attack against the 4-way handshake (and against other handshakes) highlights vulnerabilities in the WPA2 protocol itself.

Are other protocols also affected by key reinstallation attacks?

We expect that certain *implementations of other protocols* may be vulnerable to similar attacks. So it's a good idea to audit security protocol implementations with this attack in mind. However, we consider it unlikely that other *protocol standards* are affected by similar attacks (or at least so we hope). Nevertheless, it's still a good idea to audit other protocols!

Is there a higher resolution version of the logo?

Yes there is. And a big thank you goes to the person that made the logo!

When did you first notify vendors about the vulnerability?

We sent out notifications to vendors whose products we tested ourselves around 14 July 2017. After communicating with these vendors, we realized how widespread the weaknesses we discovered are (only then did I *truly* convince myself it was indeed a protocol weaknesses and not a set of implementation bugs). At that point, we decided to let CERT/CC help with the disclosure of the vulnerabilities. In turn, CERT/CC sent out a broad notification to vendors on 28 August 2017.

Why did OpenBSD silently release a patch before the embargo?

OpenBSD was notified of the vulnerability on 15 July 2017, before CERT/CC was involved in the coordination. Quite quickly, Theo de Raadt replied and critiqued the tentative disclosure deadline: *"In the open source world, if a person writes a diff and has to sit on it for a month, that is very discouraging"*. Note that I wrote and included a suggested diff for OpenBSD already, and that at the time the tentative disclosure deadline was around the end of August. As a compromise, I allowed them to silently patch the vulnerability. In hindsight this was a bad decision, since others might rediscover the vulnerability by inspecting their silent patch. To avoid this problem in the future, OpenBSD will now receive vulnerability notifications closer to the end of an embargo.

So you expect to find other Wi-Fi vulnerabilities?

"I think we're just getting started." — Master Chief, Halo 1