by Mathy Vanhoef

INTRODUCTION

11 May 2021 — This website presents FragAttacks (**fr**agmentation and **ag**gregation **attacks**) which is a collection of new security vulnerabilities that affect Wi-Fi devices. An adversary that is within range of a victim's Wi-Fi network can abuse these vulnerabilities to steal user information or attack devices. Three of the discovered vulnerabilities are design flaws in the Wi-Fi standard and therefore affect most devices. On top of this, several other vulnerabilities were discovered that are caused by widespread programming mistakes in Wi-Fi products. Experiments indicate that every Wi-Fi product is affected by at least one vulnerability and that most products are affected by several vulnerabilities.

The discovered vulnerabilities affect all modern security protocols of Wi-Fi, including the latest WPA3 specification. Even the original security protocol of Wi-Fi, called WEP, is affected. This means that several of the newly discovered design flaws have been part of Wi-Fi since its release in 1997! Fortunately, the **design flaws are hard to abuse** because doing so requires user interaction or is only possible when using uncommon network settings. As a result, in practice the **biggest concern are the programming mistakes in Wi-Fi products** since several of them are trivial to exploit.

The discovery of these vulnerabilities comes as a surprise, because the security of Wi-Fi has in fact significantly improved over the past years. For instance, previously we discovered the KRACK attacks, the defenses against KRACK were proven secure, and the latest WPA3 security specification has improved. Unfortunately, a feature that could have prevented one of the newly discovered design flaws was not adopted in practice, and the

other two design flaws are present in a feature of Wi-Fi that was previously not widely studied. This shows it stays important to analyze even the most well-known security protocols (if you want to help, we are hiring). Additionally, it shows that it's essential to regularly test Wi-Fi products for security vulnerabilities, which can for instance be done when certifying them.

To protect users, security updates were prepared during a 9-month-long coordinated disclosure that was supervised by the Wi-Fi Alliance and ICASI. If updates for your device are not yet available, you can mitigate some attacks (but not all) by assuring that websites use HTTPS and by assuring that your devices received all other available updates.

The research will be presented at the USENIX Security conference and a longer talk with more background will also be given at Black Hat USA this summer.

DEMO

The following video shows three examples of how an adversary can abuse the vulnerabilities. First, the aggregation design flaw is abused to intercept sensitive information (e.g. the victim's username and password). Second, it's shown how an adversary can exploit insecure internet-of-things devices by remotely turning on and off a smart power socket. Finally, it's demonstrated how the vulnerabilities can be abused as a stepping stone to launch advanced attacks. In particular, the video shows how an adversary can take over an outdated Windows 7 machine inside a local network.

FragAttacks: Demonstration of Flaws in WPA2/3
Mathy Vanhoef

Watch on

As the demo illustrates, the Wi-Fi flaws can be abused in two ways. First, under the right conditions they can be abused to steal sensitive data. Second, an adversary can abuse the Wi-Fi flaws to attack devices in someone's home network.

The biggest risk in practice is likely the ability to abuse the discovered flaws to attack devices in someone's home network. For instance, many smart home and internet-of-things devices are rarely updated, and Wi-Fi security is the last line of defense that prevents someone from attacking these devices. Unfortunately, due to the discover vulnerabilities, this last line of defense can now be bypassed. In the demo above, this is illustrated by remotely controlling a smart power plug and by taking over an outdated Windows 7 machine.

The Wi-Fi flaws can also be abused to exfiltrate transmitted data. The demo shows how this can be abused to learn the username and password of the victim when they use the NYU website. However, when a website is configured with HSTS to always use HTTPS as an extra layer of security, which nowadays close to 20% of websites are, the transmitted data cannot be stolen. Additionally, several browsers now warn the user when HTTPS is not being used. Finally, although not always perfect, recent mobile apps by default use HTTPS and therefore also use this extra protection.

DETAILS

Presentations

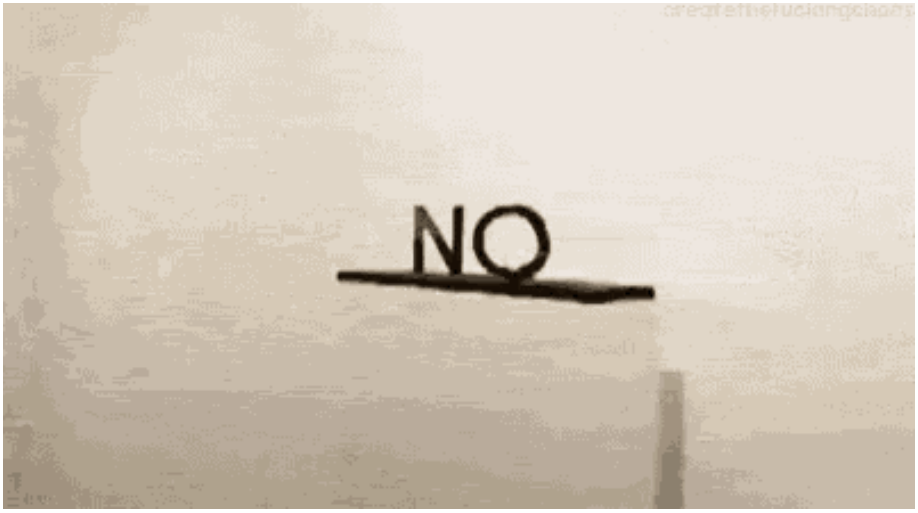
If you don't want to read the following text, you can also watch my USENIX Security or WAC4 presentation.

Plaintext injection vulnerabilities

Several implementation flaws can be abused to easily inject frames into a protected Wi-Fi network. In particular, an adversary can often inject an unencrypted Wi-Fi frame by carefully constructing this frame. This can for instance be abused to intercept a client's traffic by tricking the client into using a malicious DNS server as shown in the demo (the intercepted traffic may have another layer of protection though). Against routers this can also be abused to bypass the NAT/firewall, allowing the adversary to subsequently attack devices in the local Wi-Fi network (e.g. attacking an outdated Windows 7 machine as shown in the demo).

How can the adversary construct unencrypted Wi-Fi frames so they are accepted by a vulnerable device? First, certain Wi-Fi devices **accept any unencrypted frame even when connected to a protected Wi-Fi network**. This means the attacker doesn't have to do anything special! Two of out of four tested home routers were affected by this vulnerability, several internet-of-things devices were affected, and some smartphones were affected. Additionally, many Wi-Fi dongles on Windows will wrongly accept plaintext frames when they are split into several (plaintext) fragments.

Additionally, certain devices **accept plaintext aggregated frames that look like handshake messages**. An adversary can exploit this by sending an aggregated frame whose starts resembles a handshake message and whose second subframe contains the packet that the adversary wants to inject. A vulnerable device will first interpret this frame as a handshake message, but will subsequently process it as an aggregated frame. In a sense, one part of the code will think the frame is a handshake message and will accept it even though it's not encrypted. Another part of the code will instead see it as an aggregated frame and will process the packet that the adversary wants to inject.



A plaintext aggregated frame that also looks like a handshake message ☺

Finally, several devices process broadcasted fragments as normal unfragmented frames. More problematic, some devices **accept broadcast fragments even when sent unencrypted**. An attacker can abuse this to inject packets by encapsulating them in the second fragment of a plaintext broadcast frame.

Design flaw: aggregation attack

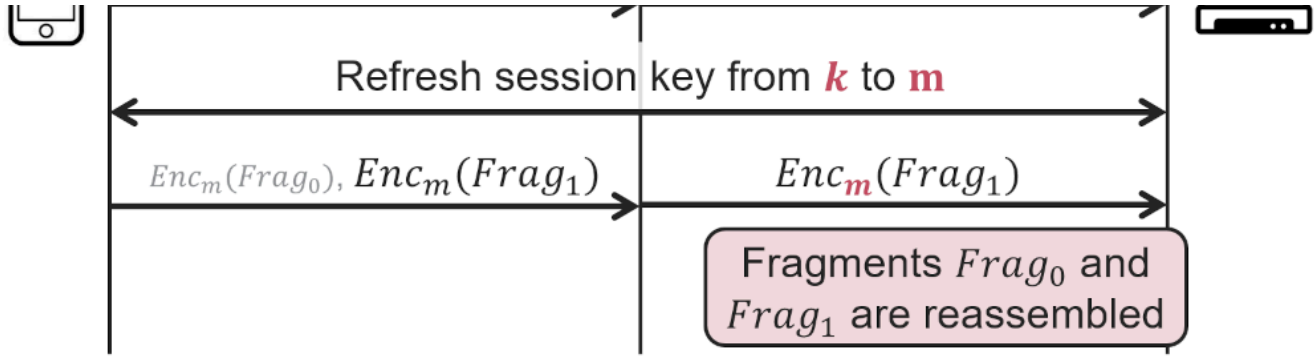
The first design flaw is in the frame aggregation feature of Wi-Fi. This feature increases the speed and throughput of a network by combining small frames into a larger aggregated frame. To implement this feature, the header of each frame contains a flag that indicates whether the (encrypted) transported data contains a single or aggregated frame. This is illustrated in the following figure:

Unfortunately, this **"is aggregated" flag is not authenticated** and can be modified by an adversary, meaning a victim can be tricked into processing the encrypted transported data in an unintended manner. An adversary can abuse this to inject arbitrary network packets by tricking the victim into connecting to their server and then setting the "is aggregated" flag of carefully selected packets. Practically all tested devices were vulnerable to this attack. The ability to inject packets can in turn be abused to intercept a victim's traffic by making it use a malicious DNS server (see the demo).

This design flaw can be fixed by authenticating the "is aggregated" flag. The Wi-Fi standard already contains a feature to authenticate this flag, namely requiring SPP A-MSDU frames, but this defense is not backwards-compatible and not supported in practice. Attacks can also be mitigated using an ad-hoc fix, though new attacks may remain possible.

Design flaw: mixed key attack

The second design flaw is in the frame fragmentation feature of Wi-Fi. This feature increases the reliability of a connection by splitting large frames into smaller fragments. When doing this, every fragment that belongs to the same frame is encrypted using the same key. However, receivers are not required to check this and will **reassemble fragments that were decrypted using different keys**. Under rare conditions this can be abused to exfiltrate data. This is accomplished by mixing fragments that are encrypted under different keys, as illustrated in the following figure:

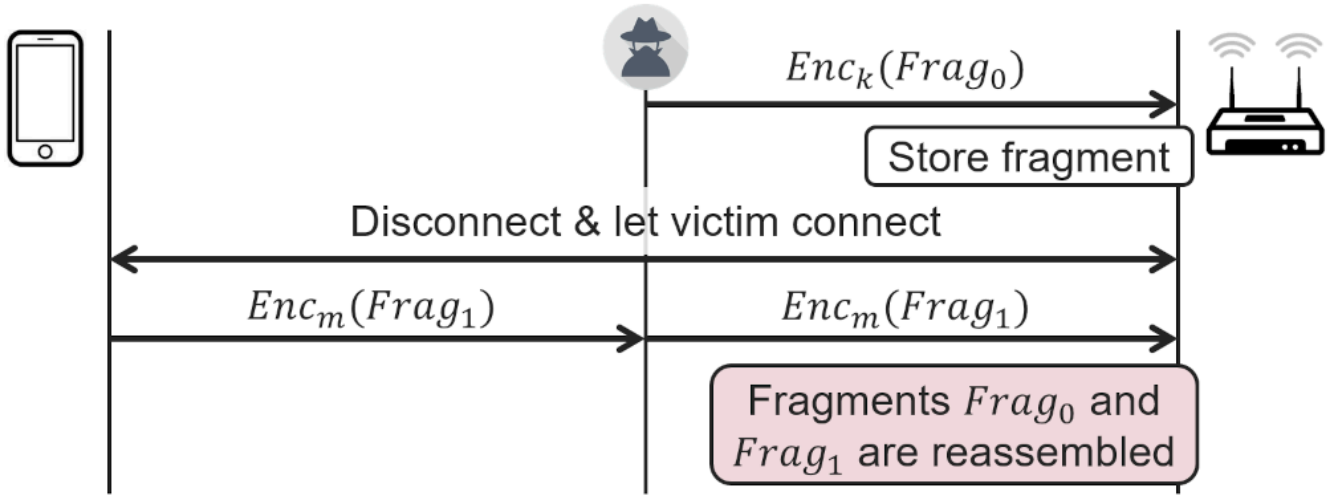


In the above figure, the first fragment received by the access point is decrypted using a different key than the second fragment. Nevertheless, the victim will reassemble both fragments. In practice this allows an adversary to exfiltrate selected client data.

This design flaw can be fixed in a backwards-compatible manner by only reassembling fragments that were decrypted using the same key. Because the attack is only possible under rare conditions **it is considered a theoretical attack**.

Design flaw: fragment cache attack

The third design flaw is also in Wi-Fi's frame fragmentation feature. The problem is that, when a client disconnects from the network, the Wi-Fi device is **not required to remove non-reassembled fragments from memory**. This can be abused against hotspot-like networks such as eduroam and govroam and against enterprise network where users distrust each other. In those cases, selected data sent by the victim can be exfiltrated. This is achieved by injecting a malicious fragment in the memory (i.e. fragment cache) of the access point. When the victim then connects to the access point and sends a fragmented frame, selected fragments will be combined (i.e. reassembled) with the injected fragment of the adversary. This is illustrated in the following figure:



In the above figure, the adversary injects the first fragment into the fragment cache of the access point. After the adversary disconnects the fragment stays in the fragment cache and

will be reassembled with a fragment of the victim. If the victim sends fragmented frames, which appears uncommon in practice, this can be abused to exfiltrate data.

This design flaw can be fixed in a backwards-compatible manner by removing fragments from memory whenever disconnecting or (re)connecting to a network.

Other implementation vulnerabilities

Some routers will forward handshake frames to another client even when the sender hasn't authenticated yet. This vulnerability allows an adversary to perform the aggregation attack, and inject arbitrary frames, without user interaction.

Another extremely common implementation flaw is that receivers do not check whether all fragments belong to the same frame, meaning an adversary can trivially forge frames by mixing the fragments of two different frames.

Additionally, against several implementations it is possible to mix encrypted and plaintext fragments.

Finally, some devices don't support fragmentation or aggregation, but are still vulnerable to attacks because they process fragmented frames as full frames. Under the right circumstances this can be abused to inject packets.

Assigned CVE identifiers

An overview of all assigned Common Vulnerabilities and Exposures (CVE) identifiers can be found on GitHub, and there is a list of known advisories from companies. Summarized, the design flaws were assigned the following CVEs:

- CVE-2020-24588: aggregation attack (accepting non-SPP A-MSDU frames).
- CVE-2020-24587: mixed key attack (reassembling fragments encrypted under different keys).
- CVE-2020-24586: fragment cache attack (not clearing fragments from memory when (re)connecting to a network).

Implementation vulnerabilities that allow the trivial injection of plaintext frames in a protected Wi-Fi network are assigned the following CVEs:

- CVE-2020-26145: Accepting plaintext broadcast fragments as full frames (in an encrypted network).
- CVE-2020-26144: Accepting plaintext A-MSDU frames that start with an RFC1042 header with EtherType EAPOL (in an encrypted network).
- CVE-2020-26140: Accepting plaintext data frames in a protected network.

- CVE-2020-26143: Accepting fragmented plaintext data frames in a protected network.

Other implementation flaws are assigned the following CVEs:

- CVE-2020-26139: Forwarding EAPOL frames even though the sender is not yet authenticated (should only affect APs).
- CVE-2020-26146: Reassembling encrypted fragments with non-consecutive packet numbers.
- CVE-2020-26147: Reassembling mixed encrypted/plaintext fragments.
- CVE-2020-26142: Processing fragmented frames as full frames.
- CVE-2020-26141: Not verifying the TKIP MIC of fragmented frames.

For each implementation vulnerability we listed the reference CVE identifier. Although each affected codebase normally receives a unique CVE, the agreement between affected vendors was that, in this specific case, using the same CVE across different codebases would make communication easier. For instance, by tying one CVE to each vulnerability, a customer can now ask a vendor whether their product is affected by a specific CVE. Please note that this deviates from normal MITRE guidelines, and that this decision was made by affected vendors independently of MITRE, and that this in no way reflects any changes in how MITRE assigns CVEs.

PAPER

Our paper behind the attack is titled Fragment and Forge: Breaking Wi-Fi Through Frame Aggregation and Fragmentation and was presented at USENIX Security. You can use the following bibtex entry to cite our paper:

```
@inproceedings{vanhoef-usenix2021-fragattacks,
  author = {Mathy Vanhoef},
  title = {Fragment and Forge: Breaking {Wi-Fi} Through Frame Aggregation and Fragment
  booktitle = {Proceedings of the 30th {USENIX} Security Symposium},
  year = {2021},
  month = {August},
  publisher = {{USENIX} Association}
}
```

Paper Clarifications

- In the paper in Section 6 regarding implementation vulnerabilities, when a tested device accepts plaintext frames, it will also accepted fragmented plaintext frames

(Table 1, 2, and 3).

USENIX Security Presentation

The pre-recorded presentation made for USENIX Security can already be viewed online. Note that the target audience of this presentation are academics and IT professionals:

Workshop on Attacks on Cryptography Presentation

A longer presentation that covers more details is the one that I gave at the WAC4 workshop:

The Untold Secrets Behind FragAttacks
Mathy Vanhoef

Watch on

Extra Documents

- An overview of all **attacks and their preconditions**. It also contains two extra examples on how an adversary can: (1) abuse packet injection vulnerabilities to make a victim use a malicious DNS; and (2) how packet injection can be abused to bypass the NAT/firewall of a router.
- Slides illustrating how the aggregation attack (CVE-2020-24588) works in practice. Performing this attack requires tricking the victim into connecting to the adversary's server. This can be done by making the victim download an image from the adversary's server. Note that JavaScript code execution on the victim is not required.
- Detailed slides giving an in-depth explanation of each discovered vulnerability.
- Overview slides illustrating only the root cause of each discovered vulnerability.

TOOLS

A tool was made that can test if clients or APs are affected by the discovered design and implementations flaws. It can test home networks and enterprise networks where

authentication is done using, e.g., PEAP-MSCHAPv2 or EAP-TLS. The **tool supports over 45 test cases** and requires modified drivers in order to reliable test for the discovered vulnerabilities. Without modified drivers, one may wrongly conclude that a device is not affected while in reality it is.

A **live USB image** is also available. This image contains pre-installed modified drivers, modified firmware for certain Atheros USB dongles, and a pre-configured Python environment for the tool. Using a live image is useful when you cannot install the modified drivers natively (and using a virtual machine can be unreliable for some network cards).

Apart from a tool to test if a device is vulnerable I also made proof-of-concepts to exploit weaknesses. Because not all devices currently have received updates these attacks scripts will be released at a later point if deemed useful.

Q & A

- How can I contact you?
- Are you looking for PhD students?
- Can I reuse the images on this website?
- Why did nobody notice the aggregation design flaw before?
- Why was the defense against the aggregation attack (CVE-2020-24588) not adopted?
- My device isn't patched yet, what can I do?
- Does using EAP-TLS increase the difficulty of attacks?
- Does using 802.11w help mitigate attacks?
- Why is Wi-Fi security important? We already have HTTPS.
- Will using a VPN prevent attacks?
- How did you discover this?
- How sure are you that all Wi-Fi devices are affected?
- Does this mean every Wi-Fi device is trivial to attack?
- How many networks use fragmentation?
- How many networks periodically refresh the pairwise session key?
- Isn't it irresponsible to release tools to perform the attacks?
- Where are all the attack tools?
- Do you have example network captures of the vulnerabilities?

- How long will you maintain the driver patches needed to run the test scripts?
- Why are so many implementations vulnerable to be non-consecutive PN attack?
- Why are so many implementations vulnerable to the mixed plaintext/encrypted fragment attack?
- Can an implementation be vulnerable to a cache attack without being vulnerable to a mixed key attack?
- Can the mixed-key attack be prevented in a backward-compatible manner?
- Is the old WPA-TKIP protocol also affected by the design flaws?
- Is the ancient WEP protocol also affected by the design flaws?
- Can fragmentation attacks be preventing by disallowing small delays between fragments?
- Are patches for Linux available?
- Did others also discover the plaintext injection issue (CVE-2020-26140)?
- Why do you use the same CVE for implementation issues in multiple different codebases?
- Why was the embargo so long?
- How did you monitor for leaks during the embargo?
- Are these vulnerabilities being exploited in practice?
- Why did Microsoft already fix certain vulnerabilities on March 9, 2021?
- Is the "Treating fragments as full frames" flaw (CVE-2020-26142) also applicable to APs?
- Can APs be vulnerable to attacks that send broadcast frames?
- Why are some of the tested devices so old?
- How did you make macOS switch to the malicious DNS server in the demonstration?
- Isn't nyu.edu using HSTS to prevent these kind of attacks?
- How do I reproduce the BlueKeep attack shown in the demonstration?

How can I contact you?

You can reach Mathy Vanhoef on twitter at @vanhoefm or by emailing Mathy Vanhoef.

Are you looking for PhD students?

See KU Leuven's DistriNet website for open positions. You can also directly contact us with spontaneous applications.

If you want to do network research at New York University Abu Dhabi in the Cyber Security & Privacy (CSP) team where the FragAttacks research was carried out, you can contact Christina Pöpper.

Can I reuse the images on this website?

Yes, you can use the logo, illustrations of the aggregation design flaw (mobile version), illustrations of the mixed key design flaw (mobile version), and illustrations of the fragment cache design flaw (mobile version).

Thanks goes to Darlee Urbiztondo for designing the logo.

Why did nobody notice the aggregation design flaw before?

When the 802.11n amendment was being written in 2007, which introduced supported for aggregated (A-MSDU) frames, several IEEE members noticed that the "is aggregated" flag was not authenticated. Unfortunately, many products already implemented a draft of the 802.11n amendment, meaning this problem had to be addressed in a backwards-compatible manner. The decision was made that devices would advertise whether they are capable of authenticating the "is aggregated" flag. Only when devices implement and advertise this capability is the "is aggregated" flag protected. Unfortunately, in 2020 not a single tested device supported this capability, likely because it was considered hard to exploit. To quote a remark made back in 2007: *"While it is hard to see how this can be exploited, it is clearly a flaw that is capable of being fixed."*

In other words, people did notice this vulnerability and a defense was standardized, but in practice the defense was never adopted. This is a good example that security defenses must be adopted before attacks become practical.

Why was the defense against the aggregation attack (CVE-2020-24588) not adopted?

Likely because it was only considered a theoretic vulnerability when the defense was created. To quote a remark made back in 2007: *"While it is hard to see how this can be exploited, it is clearly a flaw that is capable of being fixed."*

Additionally, the threat model that was used in the aggregation attack, were the victim is induced into connecting to the adversary's server, only become widely accepted in 2011 after the disclosure of the BEAST attack. In other words, the threat model was not yet widely known back in 2007 when the IEEE added the optional feature that would have prevented the attack. And even after this threat model became more common, the resulting attack isn't obvious.

My device isn't patched yet, what can I do?

First, it's always good to remember general security best practices: update your devices, don't reuse your passwords, make sure you have backups of important data, don't visit shady websites, and so on.

In regards to the discovered Wi-Fi vulnerabilities, you can mitigate attacks that exfiltrate sensitive data by double-checking that websites you are visiting use HTTPS. Even better, you can install the HTTPS Everywhere plugin. This plugin forces the usage of HTTPS on websites that are known to support it.

To mitigate attacks where your router's NAT/firewall is bypassed and devices are directly attacked, you must assure that all your devices are updated. Unfortunately, not all products regularly receive updates, in particular smart or internet-of-things devices, in which case it is difficult (if not impossible) to properly secure them.

More technically, the impact of attacks can also be reduced by manually configuring your DNS server so that it cannot be poisoned. Specific to your Wi-Fi configuration, you can mitigate attacks (but not fully prevent them) by disabling fragmentation, disabling pairwise rekeys, and disabling dynamic fragmentation in Wi-Fi 6 (802.11ax) devices.

Does using EAP-TLS increase the difficulty of attacks?

No. All attacks are possible no matter how the client authenticates the network. In other words, attacks are identical against home and enterprise networks. Moreover, it doesn't matter which EAP method you use in an enterprise network: all attacks remain possible and can be abused in exactly the same manner.

Important to remark is that exploiting the design flaws relies on a multi-channel machine-in-the-middle position (and abusing CVE-2020-26146 relies on this MitM as well). This MitM is not a traditional rogue AP. Instead, the attacker is copying all frames from the real AP to a different Wi-Fi channel. This can be done no matter which authentication method the network is using. You can read more about this MitM in my blog post.

Does using 802.11w help mitigate attacks?

No. Using 802.11w, also known as management frame protection, has no impact on any of the attacks. Even attacks that rely on the multi-channel machine-in-the-middle position (i.e. the design flaws and CVE-2020-26146) remain possible when using 802.11w. This is because this MitM isn't established by sending deauthentication or disassociation frames to first disconnect the client from the network. Instead, to establish this MitM, the attacker forces the client to switch to another channel. This is accomplished by spoofing beacon frames that contain malicious Channel Switch Announcements. These beacon frames can be spoofed even when 802.11w is enabled, meaning that enabling 802.11w won't make attacks harder.

You can read more about the multi-channel MitM in my blog post. This MitM can be established in precisely the same manner whether or not 802.11w is used.

Why is Wi-Fi security important? We already have HTTPS.

These days a lot of websites and apps use HTTPS to encrypt data. When using HTTPS, an adversary cannot see the data you are transmitting even when you are connected to an open Wi-Fi network. This also means that you can safely use open Wi-Fi hotspots as long as you keep your devices up-to-date and as long as you assure that websites are using HTTPS. Unfortunately, not all websites require the usage of HTTPS (i.e. they're not using HSTS), meaning they remain vulnerable to possible attacks.

At home, the security of your Wi-Fi network is also essential. An insecure network means that others might be able to connect to the internet through your home. Additionally, more and more devices are using Wi-Fi to transfer personal files in your local network without an extra layer of protection (e.g. when printing files, smart display screens, when sending files to a local backup storage, digital photo stands, and so on). More problematic, a lot of internet-of-things devices have tons of security vulnerabilities that can be exploited if an adversary can communicate with them. The main thing that prevents an adversary from exploiting these insecure internet-of-things devices is the security of your Wi-Fi network. It therefore remains essential to have strong encryption and authentication at the Wi-Fi layer.

At work, the security of Wi-Fi is also essential for the same reasons as mentioned above. Additionally, many companies will automatically allow access to sensitive services when a user (or adversary) is able to connect to the Wi-Fi network. Therefore strong Wi-Fi security is also essential in a work setting.

Will using a VPN prevent attacks?

Using a VPN can prevent attacks where an adversary is trying to exfiltrate data. It will not prevent an adversary from bypassing your router's NAT/firewall to directly attack devices.

How did you discover this?

The seeds of this research were already planted while I was investigating the KRACK attack. At that time, on 8 June 2017 to be precise, I wrote down some notes to further investigate (de)fragmentation support in Linux. In particular, I thought there might be an implementation vulnerability in Linux. However, a single unconfirmed implementation flaw isn't too spectacular research-wise, so after disclosing the KRACK attack I decided to work on other research instead. The idea of inspecting (de)fragmentation in Wi-Fi, and determining whether there really was a vulnerability or not, was always at the back of my mind though.

Fast-forward three years later, and after gaining some additional ideas to investigate, closer inspection confirmed some of my hunches and also revealed that these issues were more widespread than I initially assumed. And with some extra insights I also discovered all the other vulnerabilities. Interestingly, this also shows the advantage of fleshing out ideas before rushing to publish (though actually finishing the paper before submission was still a race against time..).

How sure are you that all Wi-Fi devices are affected?

In experiments on more than 75 devices, all of them were vulnerable to one or more of the discovered attacks. I'm curious myself whether *all* devices in the whole world are indeed affected though! To find this out, if you find a device that isn't affected by at least one of the discovered vulnerabilities, let me know.

Also, if your company provides Wi-Fi devices and you think that your product was not affected by any of the discovered vulnerabilities, you can send your product to me. Once I confirmed that it indeed was not affected by any vulnerabilities the name of your product and company will be put here! Note that I do need a method to assure that I'm indeed testing a version of the product that was available before the disclosure of the vulnerabilities (and that you didn't silently patch some vulnerabilities).

Does this mean every Wi-Fi device is trivial to attack?

The design issues are, on their own, tedious to exploit in practice. Unfortunately, some of the implementation vulnerabilities are common and trivial to exploit. Additionally, by combining the design issues with certain implementation issues, the resulting attacks become more serious. This means the impact of our findings depends on the specific target. Your vendor can inform you what the precise impact is for specific devices. In other words, for some devices the impact is minor, while for others it's disastrous.

How many networks use fragmentation?

By default devices don't send fragmented frames. This means that the mixed key attack and the fragment cache attack, on their own, will be hard to exploit in practice, unless Wi-Fi 6 is used. When using Wi-Fi 6, which is based on the 802.11ax standard, a device may dynamically fragment frames to fill up available airtime.

How many networks periodically refresh the pairwise session key?

By default access points don't renew the pairwise session key, even though some may periodically renew the group key. This means that the default mixed key attack as described in the paper is only possible against networks that deviate from this default setting.

Isn't it irresponsible to release tools to perform the attacks?

The test tool that we released can only be used to test whether a device is vulnerable. It cannot be used to perform attacks: an adversary would have to write their own tools for that. This approach enables network administrators to test if devices are affected while reducing the chance of someone abusing the released code.

Where are all the attack tools?

The code that has currently been released focusses on *detecting* vulnerable implementations. The proof-of-concepts scripts that perform actual attacks are not released to provide everyone with more time to implement and deploy patches. Once a large enough fraction of devices has been patched, and if deemed necessary and/or beneficial, the attack script will be publicly released as well.

Do you have example network captures of the vulnerabilities?

There are example network captures of the test tool that illustrate the root causes of several vulnerabilities.

How long will you maintain the driver patches needed to run the test scripts?

The modifications to certain drivers have been submitted upstream to Linux meaning they will be maintained by the Linux developers themselves. The patches to the Intel driver have not been submitted upstream because they're a bit hacky. Concretely, this means that drivers such as ath9k_htc will be supported out of the box, while for Intel devices you will have to use patched drivers and I'm not sure how much time I'll have to maintain those.

Why are so many implementations vulnerable to be non-consecutive PN attack?

That's a good question. I'm not sure why so many developers missed this. This widespread implementation vulnerability does highlight that leaving important cryptographic operations up to developers is not ideal. Put another way, it might have been better if the standard required an authenticity check over the reassembled frame instead. That would also better follow the principle of authenticated encryption.

Why are so many implementations vulnerable to the mixed plaintext/encrypted fragment attack?

The 802.11 standard states in section 10.6: "If security encapsulation has been applied to the fragment, it shall be deencapsulated and decrypted before the fragment is used for defragmentation of the MSDU or MMPDU". There is unfortunately no warning that unencrypted *fragments* should be dropped. And there are no recommend checks that should be performed when reassembling two (decrypted) fragments.

Can an implementation be vulnerable to a cache attack without being vulnerable to a mixed key attack?

Yes, although this is unlikely to occur in practice. More technically, let's assume that an implementation tries to prevent mixed key attacks by: (1) assigning an unique key ID to every fragment; (2) incrementing this key ID whenever the pairwise transient key (PTK) is

updated; and (3) assuring all fragments were decrypted under the same key ID. Unfortunately, in that case cache attacks may still be feasible. In particular, if under this defense key IDs are reused after (re)connecting to a network, for example because they are reset to zero, fragments that are decrypted using a different key may still be assigned the same key ID. As a result, cache attacks remain possible, because the fragments will still be reassembled as they have the same key ID.

Can the mixed-key attack be prevented in a backward-compatible manner?

Strictly speaking not, because the 802.11 standard does not explicitly require that a sender encrypts all fragments of a specific frame under the same key. Fortunately, all implementations that we tested did encrypt all fragments using the same key, at least under the normal circumstances that we tested, meaning in practice the mixed key attack can be prevented without introducing incompatibilities.

Is the old WPA-TKIP protocol also affected by the design flaws?

Strictly speaking not, though implementations can still be vulnerable. Note that TKIP should not be used because it is affected by other more serious security flaws. Additionally, TKIP has been deprecated by the Wi-Fi Alliance.

The TKIP protocol is not affected by the fragmentation-based design flaws (CVE-2020-24587 and CVE-2020-24586) because it verifies the authenticity of the full reassembled frame. This is in contrast to CCMP and GCMP, which only verify the authenticity of individual fragments, and rely on sequential packet numbers to securely reassemble the individual (decrypted) fragments.

Additionally, TKIP is not affected by the aggregation design flaw (CVE-2020-24588) because a receiver is supposed to drop A-MSDUs that are encrypted using TKIP. Indeed, in Section "12.9.2.8 Per-MSDU/Per-A-MSDU Rx pseudocode" of the 802.11-2016 standard it's specified that when using TKIP only normal MSDU frames are accepted.

Unfortunately, some implementations don't verify the authenticity of fragmented TKIP frames, and some accept aggregated frames (i.e. A-MSDUs) even when encrypted using TKIP. This unfortunately means that in practice TKIP implementations may still be vulnerable.

Is the ancient WEP protocol also affected by the design flaws?

Yes. The WEP protocol is so horrible that it doesn't even try to verify the authenticity of fragmented frames. This means an adversary can trivially perform aggregation-based attacks against WEP.

Similar to TKIP, the WEP protocol is not affected by the aggregation design flaw (CVE-2020-24588) because a receiver is supposed to drop A-MSDUs that are encrypted using WEP.

Nevertheless, in practice several WEP implementations do accept A-MSDUs and therefore are still vulnerable.

Finally, in case you've been living under a rock, stop using WEP, it's known to be a horrible security protocol.

Can fragmentation attacks be preventing by disallowing small delays between fragments?

This would make exploiting possible vulnerabilities harder and perhaps in some cases practically infeasible. Unfortunately this doesn't provide any guarantees though. I therefore recommend to fix the root cause instead.

Are patches for Linux available?

Yes! During the embargo I helped write some patches for the Linux kernel. This means an updated Linux kernel should (soon) be available for actively supported Linux distributions.

Did others also discover the plaintext injection issue (CVE-2020-26140)?

During the embargo I was made aware that Synopsys also discovered the plaintext injection vulnerability (CVE-2020-26140) in access points. They found that Mediatek, Realtek, and Qualcomm were affected, and to cover these three implementations the identifiers CVE-2019-18989, CVE-2019-18990, and CVE-2019-18991 were respectively assigned.

During the FragAttacks research I found that the same vulnerability was (still) present in other access points and that clients can be vulnerable to a similar attack. Additionally, and somewhat surprisingly, I also found that some devices reject normal (non-fragmented) plaintext frames but do accept fragmented plaintext frames (CVE-2020-26143).

Why do you use the same CVE for implementation issues in multiple different codebases?

Implementation-specific vulnerabilities usually get their own independent CVE identifier for each different codebase. However, because the same implementation issues seem to be present across multiple vendors it would make more sense to have a single CVE identifier for each common implementation issue. After all, the main purpose of CVE identifiers is to provide a single, common ID to be used across vendors to identify the same vulnerability. We therefore think it makes sense to assign only a single CVE identifier to each implementation issues. This enables vendors and customers to easily reference an implementation vulnerability and, for instance, check whether certain products are affected by one of the discovered vulnerabilities.

Why was the embargo so long?

The disclosure was delayed by two months in consensus with ICASI and the Wi-Fi Alliance. The decision on whether to disclose fast, or to provide more time to write and create patches, wasn't easy. At the time, the risk of leaks appeared low, and the advantage of delaying appeared high. Additionally, we were prepared to immediately disclose in case details would accidently leak publicly. Another aspect that influenced my decision was *the current situation*, meaning COVID-19, which among other things made it harder to safely get access to physical places/labs to test patches.

How did you monitor for leaks during the embargo?

During the last two months of the embargo, we were prepared to make the research public whenever information would seemed to be leaking. To detect leaks I personally searched for relevant keywords (CVE numbers, paper title, script names) on Google and social media such as Twitter. The Wi-Fi Alliance and ICASI were also monitoring for leaks (e.g. if questions came from people that shouldn't have known about it). This can detect innocent leaks. Detecting malicious leaks or usage of the vulnerabilities in stealthy attacks is a much harder problem (if even possible at all).

If you know about cases where some information was (accidently) leaked, it would be useful to know about that so that I can better estimate the impact of having long embargos. Any information you provide about this will remain confidential. This information will help me in future decision when weighing the option of a longer embargo versus disclosing research even when several vendors don't have patches ready (i.e. it won't be used to point fingers).

Are these vulnerabilities being exploited in practice?

Not that we are aware of. Because some of the design flaws took so long to discover my hunch is that those have not been previously exploited in the wild. But it is difficult to monitor whether one of the discovered vulnerabilities have been exploited in the past or are currently being exploited. So it is hard to give a definite answer to this question.

Why did Microsoft already fix certain vulnerabilities on March 9, 2021?

The original disclosure date was March 9, 2021. Roughly one week beforehand it was decided to delay the disclosure. At this time Microsoft had already committed to shipping certain patches on March 9. I agreed that already releasing certain patches without providing information about the vulnerabilities was, at that point, an acceptable risk. Put differently, the advantages of delaying the disclosure appeared to outweigh the risk that someone would reverse engineer the patches and rediscover certain attacks.

Is the "Treating fragments as full frames" flaw (CVE-2020-26142) also applicable to APs?

Yes, access points can also be vulnerable. In particular, during additional experiments that I recently performed, the vulnerability was also present in OpenBSD when it acted as an access point.

Can APs be vulnerable to attacks that send broadcast frames?

Yes, although they are less likely to be vulnerable compared to clients. This is because under normal circumstances clients never send a frame to the AP with a broadcast receiver address. Instead, clients first send broadcast/multicast network packets as unicast Wi-Fi frames to the AP, and the AP then broadcasts these packets to all connected clients. As a result, many APs will simply ignore Wi-Fi frames with a broadcast receiver address, because in normal networks those frames are only meant for clients.

Why are some of the tested devices so old?

I also tested some very old Wi-Fi devices and dongles to estimate how long the discovered vulnerabilities have been present in the wild. Note that some old devices may remain in use for a long time, for example, expensive medical or industrial equipment that is rarely replaced.

How did you make macOS switch to the malicious DNS server in the demonstration?

After injecting the ICMPv6 Router Advertisement with the malicious DNS server, macOS won't immediately use this DNS server. This is because macOS will only switch to the malicious DNS server if its current (primary) DNS server is no longer responding. To force this to happen, we briefly block all traffic towards the victim. This causes macOS to switch to the malicious DNS server.

Isn't nyu.edu using HSTS to prevent these kind of attacks?

Websites can use HSTS to force browsers to always use HTTPS encryption when visiting a website. This prevents the attack that was shown in our demo. Unfortunately, the website of NYU at the time did not properly configure HSTS. More technically, some subdomains such as globalhome.nyu.edu do instruct the browser to use HSTS by including the following header in responses:

```
strict-transport-security: max-age=31536000 ; includeSubDomains
```

Unfortunately, other subdomains such as shibboleth.nyu.edu remove HSTS by including the following header in responses:

```
Strict-Transport-Security: max-age=0
```

Combined with other configuration decisions, this meant that when a user would type nyu.edu in their browser, the initial request was sent in plaintext and therefore could be intercepted by an adversary.

Note that NYU has been informed of this issue and is investigating it.

How do I reproduce the BlueKeep attack shown in the demonstration?

First, when using the NAT punching technique, it is essential that you manually configure the CPORT parameter so that metasploit uses the correct client port. You can learn this port from the injected TCP SYN packet that arrives at the server. When using a different client port the router/NAT will not recognize the connection and will not forward it to the victim machine.

Second, you must set the AutoCheck parameter to zero. Otherwise metasploit will try to initiate multiple connections with the victim and that is problematic when manually specifying a client port through CPORT. This workaround of setting AutoCheck to zero can be avoided by punching multiple holes in the router/NAT and modifying the metasploit to use a different CPORT for each connection that will be initiated.

CREATIVE COMMONS ATTRIBUTION 4.0 INTERNATIONAL LICENSE | DESIGN INSPIRED BY TEMPLATED.