A Survey on WPA3

Kole W Swesey
Iowa State University
kwswesey@gmail.com

***Abstract*— This paper acts as a survey of 20 papers that are related to WPA3. WPA3 was announced by the Wi-Fi alliance in January of 2018 with the intention of being a replacement for WPA2, which has been the standard since 2004. While WPA2 is still considered secure and is industry standard, it still has a variety of weaknesses that can't be easily fixed without creating an entirely new specification. WPA3 offers a variety of new security features that make attacks on these WPA2 weaknesses obsolete when implemented. While WPA3 is still a new protocol, it has already had a few vulnerabilities discovered in it as well. While they have been fixed way before WPA3 routers became widespread, these vulnerabilities highlight the attack surface that WPA3 has and what we can learn from these vulnerabilities might help us find new ones quicker if they exist.**

Keywords—802.11, handshake, denial of service, nonce, side channel, man-in-the-middle, access point

I. Introduction

Since Wi-Fi was first introduced, a variety of security protocols have been introduced to encrypt the traffic. At the moment, the industry standard is using WPA2, but it is known for having some weaknesses. WPA3 was announced in January 2018 to act as a replacement for WPA2, and certification began in June 2018. Now, we are close to seeing routers starting to be sold that have WPA3 functionality out of the box. It is only a matter of time now before WPA3 finds its way into many home and corporate networks, so understanding the security features and improvements of WPA3 is important. The first section of this paper will cover the most critical issues in WPA2 that the wi-fi alliance made a point to fix or improve on with WPA3. The second section will explain the security enhancements and additional features that WPA3 has added. Finally, this paper will conclude with an overview of the new security issues that have been discovered in WPA3 and how they have been remediated.

II. issues with wpa2

KRACK (Key Reinstallation Attack) refers to a critical vulnerability discovered in the 3rd step of the WPA2 4 way handshake [1]. This handshake is done to negotiate a session key, with the client receiving and using the key after step 3. If an access point (AP) doesn't get the acknowledgement from the client that they received the key [step 4], then the AP will resend it. This means that the client can receive the third message multiple times if the acknowledgement never gets to the AP. When the client re-receives this message, it will reinstall the same session key, which leads to the vulnerability. This attack is performed by an attacker by acting as a man-in-the-middle between the client and the AP. From there they let the handshake carry out as normal until the client sends message 4 to the AP, which is dropped. This causes the AP to resend message 3, resulting in the client then using a key that is already in use (the same key as before) with the nonce and the replay counter now reset because the client treats it as a new key. From here, the attacker is able to replay, forge, and decrypt packets due to the nonce reuse. This technique can also be used to attack the peer key, group key, TDLS, and fast BSS handshakes that are also used in WPA2 networks. Of those listed, the one that has the most additional impact is the fast BSS handshake. This is because the BSS handshake, which is performed when a client roams from one AP to another, is the only one that is initiated by the client. This means that the KRACK attack can be performed in reverse and cause the AP to reuse the same key. Man-in-the-middle is also not required for the BSS handshake, an attacker can simply inject frames and eavesdrop to force a key reinstallation here. This attack does not make packet decryption trivial after the nonce is reused since the key is still unknown, but if an encrypted message contains known content, it can be done through a known plaintext attack to determine the key. It can also be done if the content is, for example, in plain english, in which case frequency analysis can be performed to decrypt the content. This attack can be more severe when the target client is a linux or android device. After the attacker resends the key, the linux or android client will not use the real key and will instead use a key of all zeroes, meaning an attacker would be completely able to decrypt all traffic it sees between the client and the AP since the key is now known. This appears to be the result of the fact that android/linux devices running wpa_supplicant 2.4 and 2.5 will clear the key from memory after it has been used. Mitigation for this attack involves the client verifying that the key they have received from the AP has not been reused, and if it has been, to not reset the nonce and replay counter. A client should also not reinstall a key if it sees step 3 of the handshake being resent. While patches have been deployed for both client and AP devices, both the client and the AP must be patched in order to completely defend against this attack. As a result, WPA2 handshakes can't be determined to be secure if one doesn't know if the other device is patched. Clients and APs that are using WPA3 however, are guaranteed to not be vulnerable to this attack.

WPA2 networks are also vulnerable to various attacks due to the fact that the management frames in WPA2 are not encrypted [2]. The first is MAC address spoofing, which involves an attacker modifying the management frame to disguise their MAC address as that of a different legitimate client that is not transmitting at the moment. From here, two different attacks are possible. The first one is forged deauthentication, in which an attacker spoofs the management frame of either a client or an AP when a client is authenticating to the network. These frames aren't encrypted, so an attacker can send spoofed deauthentication frames to cause the process of a client authenticating to an AP to stop. This can be repeatedly done to keep a client off of the network. A similar attack is forged disassociation, which is very similar to forged deauthentication except it uses dissociation frames instead of deauthentication frames. These frames are used in networks with multiple APs where a client might stop communication with one to use one that is faster. Along with keeping individual clients off of a network, there exists a denial of service attack that can overload the AP. This is accomplished by an attacker sending a variety of different management frames at once with different MAC addresses and a mix of deauthentications, disassiciatons, associations, authentications or bacon management frames until the WLAN is overloaded. Session hijacking is also possible by performing a disassociation or deauthentication attack while spoofing the MAC of the AP to disconnect a client. From there, the attacker can associate with the AP while spoofing the client's MAC address to steal their session. The solution to these attacks was the introduction of Protected Management Frames (PMF) that were introduced with 802.11w in 2009. PMF frames secure the frames that are responsible for deauthentication and disassociation through encryption, so an attacker would no longer be able to send spoofed management frames with a different MAC address. If they did, the AP would recognize them as being invalid. While this is an improvement, PMF is an optional part of the WPA2 protocol and as a result, a client connecting to an unknown WPA2 network does not know for certain that they will be protected against these attacks. WPA3 however, requires PMF as part of the protocol and thus ensures security for its clients.

Another issue with WPA2 is that the handshake allows for offline brute force attacks against the network password [3]. During WPA2's four way handshake, both the client and the server send nonces back and forth in order to generate keys. In step 1, the AP will send the client ANonce. Using that, the client is able to calculate the pairwise transient key (PTK). The client will respond in step 2 by sending the AP SNonce accompanied by a message integrity check (MIC) as well as the MAC address of the client. Now, the AP sends the client the group temporal key (GTK) in step 3 accompanied by the MAC address of the AP. The GTK is then installed by the client. If an attacker is able to capture the handshake, by using a wireless networking card that is set to either promiscuous or monitor mode, they can use the information in the handshake to perform an offline attack against the password. This is accomplished by first gathering the nonces, MAC addresses, and the MIC. The attacker uses this information to calculate the PTK, which requires the nonces and MAC addresses as well as the PMK to be calculated. The value for the PMK is whatever value the attacker has currently chosen to be the password in one round of brute forcing, whether it is an item in a dictionary or generated from a pattern. The key confirmation key (KCK) in the PTK is then used to calculate what the value of the MIC should be. If the calculated MIC is equal to the captured MIC, then the attacker has chosen the correct value for the PMK and therefore cracked the network password. Since this attack involves capturing the authentication handshake, the attacker does need to wait for a client to authenticate to the AP for this attack to be carried out. However, an attacker can potentially combine this attack with the deauthentication attack previously mentioned to force a target client to reauthenticate with the AP, so the attack doesn't need to wait. In order to address this issue in WPA3, the wifi organization added a different authentication handshake to the protocol that is intended to make offline password cracking impossible, even if the attacker is able to capture the entire handshake.

Along with the handshake having security issues due to it allowing for offline brute force attacks, the handshakes key exchange process does not provide perfect forward secrecy, which is found in most other popular encrypted communication methods [4]. perfect forward secrecy is a concept in cryptography in which the session keys of clients and servers would not be compromised in the event that the secrets used in the key exchange were. In the case of WPA2, the secret used in the WPA2 key exchange would be the pre shared key, which is the network password. This effectively means that anybody else that knows the pre shared key would be able to decrypt all of the traffic going through the network, but would not be able to fully decrypt packets that are using encrypted protocols such as HTTPS. An attacker could also capture traffic on a network that they do not know the password to and decrypt it at a later time if they are able to determine what the network password was at that point in time. This would only require a wireless networking card that is placed into promiscuous mode. WPA2 enterprise does provide perfect forward secrecy since it generates new keys for each session, but it is more complicated to set up and is often not used as a result. WPA3 does provide perfect forward secrecy with the additional handshake that it uses for key exchange

The Wi-Fi Protected Setup (WPS) feature that is common in WPA2 networks has proven to be a security risk [5]. WPS is a protocol that is designed to make it easier to connect clients to a wireless network and was enabled by default on all routers developed by major companies at the time this vulnerability was discovered. WPS involves an enrollee (the client attempting to join the network), a registrar (connected device assisting in on boarding the enrollee), and the AP. Push Button Connect (PBC) works by having the user push a button both the AP and the client device, leaving the PBC enabled until either 2 minutes pass or the client connects. The user also has to enter a PIN to the client, which is either generated by the AP or printed on the device. The first security issue in this design is that the PIN is the only authentication mechanism in this scheme and it is vulnerable to being brute forced. The other issue is related to the EAP-NACK message, which is what the AP sends to a client when it fails authentication. The process of authenticating the PIN involves splitting the PIN in half and authenticating each half during a separate step, with the AP replying with an authentication failure at a different time in the handshake depending on whether or not the first half is correct. Because of this, an attacker would be able to determine if the first half of the PIN is correct or not depending on when it receives the EAP-NACK message. This makes the required attempts to fully brute force the PIN to be equal to 20,000. Since the last digit in the PIN is actually a checksum, this further reduces the required attempts to 11,000. In testing, the researcher was able to brute force the PIN successfully approximately half the time and determine the network key. The primary recommended mitigation method is to just disable WPS on their routers, since this is an issue in the protocol and can't easily be patched. To handle this in WPA3, WPS has been removed from the standard and a different method for authenticating devices with an enrollee has replaced it.

One attacker technique that WPA2 does not offer protection against is a cloned base station, commonly referred to as an evil twin [6]. This attack involves setting up a malicious AP that is designed to spoof a target AP and waiting for a legitimate client to connect to the malicious AP unintentionally. After a client connects to the malicious AP, the malicious AP will forward the data received from the client to the target AP, allowing the malicious AP to act as a man-in-the-middle. The user will be able to send and receive data as normal, however the attacker now has the ability to decrypt all of the data that is sent in transit. The rogue AP will be configured to match the SSID and MAC address of the target AP for this to work, and it will also need to have the same network password, meaning the attacker would need to know the password for the AP for this attack to work. Spoofing the MAC address of the target AP is trivial since the MAC address is included in the AP's beacons. In order to increase the chances of a client connecting to the malicious AP, the attacker can send the deauth frames that were previously described to force clients to reconnect, and also increase the signal strength of their AP, since clients will likely choose the AP with the stronger signal. An attacker can also flip the attack around and target a specific client instead of a specific AP. A client might send out Probe Request frames that contain the SSID of its preferred AP(s), in which an attacker can set up a rogue AP that sets its SSID to whatever is in the probe request, which the client will then automatically associate to. This requires the network that the client is looking for to be unencrypted. The issue for the client is that client devices do not really differentiate different APs with the same SSID, and are often designed to automatically connect to an AP by its SSID alone. There is also no way for a client to verify that the AP is legitimate, or even if it is the same AP that they were just connected to. While the initial WPA3 specification did not offer a fix for this issue, later drafts included an optional mode that would offer protection against evil twin attacks.

III. Security improvements in wpa3

WPA3's handshake takes advantage of a new authentication mechanism called Simultaneous Authentication of Equals (SAE) [7]. SAE is primarily intended for mesh networks, and as a result, does not have fixed client or server roles. SAE requires a finite cyclic group where the discrete logarithm problem is computationally intractable, which in practice are prime modulus groups or, in the ideal case for WPA3, elliptic curve groups. Both groups have two operations defined for them, scalar and element operation. In the case of elliptic curve groups, the scalar operation is a point on the elliptic curve being multiplied by a scalar, while the element option is two points on the curve being added together. There is also an operation to take the inverse of a group element, which for elliptic curves is when the sum of two points on the curve is the point at infinity. When peers connect, they first select a "random" point on the elliptic curve using a shared random (to a 3rd party) one-way function. This acts as the password element. Next, each party chooses two random numbers called rand and mask, which are used to calculate the element E= inverse(mask•PW E) and scalar s = ((rand + mask)mod r) along with the password, where r is the group order. The parties then send each other their element and scalar values and are able to calculate a shared secret using one party's rand and the other party's element and scalar. Next, the parties compute an intermediate shared key k and an authentication confirmation token tok, which they send to each other. They parties validate the tok of the other party, and will then create the shared key if they are valid. The order in which the parties start the handshake has no impact on the final shared secret, which credits SAE as being a peer to peer protocol. This protocol is not trivially secure because it results in a shared secret. It is resistant to passive attacks (someone listening to the handshake) due to how the shared secret is calculated. Even if an attacker managed to see all the values sent back and forth between the two parties, they cannot verify the correctness of a password without attempting to authenticate, since the non-shared values are required in order to calculate a shared secret. The protocol is also resilient against active attacks, which involve a 3rd party modifying or forging messages between the two parties. Since an attacker does not know the rand or mask values used to generate the element or scalar values, they cannot construct a tok that one of the parties would verify and would just be making random guesses. An attacker also cannot perform any sort of dictionary attack on the protocol, the only way they could determine if a password is correct or not is to try to authenticate through the protocol, there are no offline calculations that they can run to gain an advantage on knowing if something is the correct password or not. Since the shared secret in use is based on random contributions and not previous SAE protocol runs, forward secrecy is provided. Since the function used to generate the shared secret is apparently random, as well as the inputs generated by both parties, obtaining any leaked intermediate keys from one run of the protocol wouldn't help compromise future runs of the protocol, so it is resistant against replay attacks. On the other hand, the author notes that neither party is capable of detecting man-in-the-middle attacks.

Another issue that WPA3 adds improvements to is the process of getting wireless IoT devices connected to a network [8]. IoT Devices are much more popular now than they were in 2004 and getting devices on the network is important. Because of this, the Wi-Fi alliance made sure to add features to WPA3 to allow for easy on boarding of IoT devices, primarily those without standard user interfaces. The new scheme requires a device already connected to the network to act as a configurator to get the IoT device, here called the enrollee, onboard. The first step in this scheme occurs when the enrollee is powered on and it advertises a configuration request to search for nearby configurators. The configurator can then see and accept the request, after which it will respond to the enrollee with information about the AP that it needs in order to connect. Finally, the enrollee will attempt to discover the network and then, if successful, use the information from the configurator to connect. The first step of the scheme (connecting the configurator and the enroller) can be implemented in multiple ways. One way involves using NFC, which would be highly secure due to the lower opportunities to sniff the sensitive information. However, there are drawbacks due to the cost of implementing NFC on IoT, surfaces potentially being scraped, and potential difficulty in pairing large IoT devices like refrigerators. A simpler, cheaper scheme would be to use QR codes. The enrollee would either display a QR code on screen or have it physically showing on a surface, with a configurator can then scan to receive information about the enrollee. From there it can directly connect to the enrollee and complete the scheme. This is a cheaper and simpler alternative, but the direct connection between the configurator and the enrollee is non-trivial to set up using 802.11 as opposed to NFC. Another alternative would be through the use of soft-AP based network onboarding, which involves using the enrollee as a temporary AP. The configurator would connect to the enrollee and send AP information to it, which the enrollee can then use to connect to the AP. This doesn't require proximity and minimizes extra costs, but it can take extra time to do. The configurator will be connecting to the enrollee for approximately 10 seconds and during that time won't be actually connected to the internet, which could cause issues for cloud IoT devices. Soft APs would be good for IoT devices that have wi-fi as a connectivity medium.

WPA3 also addresses security issues found in public Wi-Fi networks [9]. In WPA2 networks, if there is no password being used there is no encryption and anyone else on the network, or even just nearby, can sniff the traffic. WPA3's solution to this can be found in Opportunistic Wireless Encryption (OWE). The objective of OWE is to allow a client and an AP to calculate a shared secret without having a pre-shared password and also preventing people on the network from being able to sniff the traffic of others. This is done through diffie-hellman key exchange. Both parties have a private key that is a prime number in a public group. and perform a standard key exchange handshake to generate a shared key. While this does improve network security, these networks are still vulnerable to evil twin attacks. To prevent this, certificates can be used by the routers to prove the identity of the AP. However, this introduces a new issue of getting clients to verify that the certificate is legitimate. One approach to addressing this issue is trust on first use, in which clients would keep a saved list of network names and their public keys after connecting to them for the first time. This would protect clients as long as they connect to the correct network the first time, if they try to connect to the evil twin after that they will see that the public key has changed, meaning they are likely not connecting to the correct AP. This approach would be simple for the client, easy to implement (SSH already does this, but IoT devices might not have the memory capacity to do this), and requires little overhead. However, it is not completely effective since the first connection is still unverified. A different approach would be making network names unchangeable and having them just be the MAC address of the device. This would make it impossible for an evil twin to copy the name of a legitimate AP, however it would require more human activity to determine which network to connect to. It also is reliant on the network name to be truly impossible to change, but since MAC addresses can be spoofed already, this can't be guaranteed. If they can be spoofed, then this method would not be secure. As with the public key system, this method has little overhead. An alternative solution would be to make the network's public key the network name along with trust on first use. This would allow the client to verify that the key in the handshake matches the name of the network they are connecting to. Similar to the MAC address approach, this would not be user friendly, since a public key name doesn't help the user identify who or what the AP actually belongs to. This would be simple to set up from a technical point of view, only add a tiny bit more overhead, and should be more effective than trust on first use, but it wouldn't be in reality due to the greater chance of users accidentally connecting to the evil twin due to the confusing naming scheme. As a result, trust on first use is recommended as the best way of implementing OWE networks.

Approximately 2 years after WPA3 was initially announced, the Wi-Fi alliance announced an addendum to the protocol that defined SAE-PK. [10] This is meant to fix the issue where a shared password can be used by one of the people it is shared with to create an evil twin AP that uses the same password and perform man-in-the-middle attacks. This mode uses an additional public-private key system for authentication where the public key is used as a component in the password. All APs on a network would be configured to use the same key pair. When an AP sends an SAE confirm message to a station, an SAE-PK element will be sent containing the AP's public key, a modifier value, and a verifiable digital signature. The client is able to verify the public key using the password. If the client is unable to verify the key, the handshake is aborted. The password is generated by performing a series of mathematical operations, including a hashing algorithm, with mixins being the SSID, a random integer, and the public key of the AP(s). That value is then base32 encoded to make it case insensitive, and has separators added to make it more readable. Stations can assume that a password is for SAE-PK authentication if the password follows that pattern and auto enable SAE-PK when connecting. This security mechanism does not completely negate attacks from adversaries that know the network password, client filtering or isolation are still recommended to be used to protect against insider attacks. Stations could be configured to only use SAE-PK, and if so they would be resistant to downgrade attacks. When a client has SAE-PK enabled for a network, and is selecting between different discovered APs in that network that it thinks would be suitable to try and connect to, the client will attempt to authenticate with APs that are advertising support for SAE-PK, and if unsuccessful it will then try to connect to clients not advertising support for SAE-PK.

This update to WPA3 also included new features for WPA3 Enterprise networks that they would not otherwise have in WPA2 Enterprise [11]. When a client attempts to connect to a WPA2 network, it begins by verifying the identity of the Authentication Server (AS). This is done by the client receiving a certificate from the AS, which binds the name of the AS to a public key. Afterwards it sends the user credentials to the AS, followed by the AS demonstrating knowledge of the user credentials to the client. It's important to note that the name of the AS, not the SSID, is binded to the certificate. Because of this, the client must keep a network profile stored to associate the SSID with the AS. How a client does this is not defined by the WPA2 standard. Some clients may even have an option to skip the certificate validation altogether, which could allow for evil twin attacks. Clients might also begin the process of connecting to an evil twin but aborting when it learns the AS name, however by this point the evil twin already would have captured credential material. WPA3 offers a few solutions to mitigate these issues. One is that WPA3 forbids a client from skipping certificate validation or accepting any certificate. If the client can't verify the certificate, the user must accept the certificate explicitly using a User Override of Server Certificate (UOSC). The client can verify the certificate if the AS name matches the network profile, or it is the same as it was the last time the client tried to connect. Network Administrators can also constrain how clients learn the AS name. The client can either obtain the certificate on every connection, on the first connection only, or never and it has to be manually added to the client. The first option would be the closest to WPA2, the only difference being the UOSC requirement by the client if it notices an AS name change. The second would be Trust On First Use, resulting in the client never being in a position where they would connect to an evil twin if they connect to the correct network the first time. The final option would be the most secure, but would require the network administrators creating a system to distribute the network profile to clients for the certificate. Enterprise networks can also use a Trust Override Disable (TOD) policy, which is included in the certificate. The two TOD policies are TOD-STRICT and TOD-TOFU. TOD-STRICT means that clients can never use UOSC when connecting to a network and is used when certificates are distributed manually to clients. TOD-TOFU means USOC can only be used on the first connection attempt and is used when Trust On First Use is in place.

Methods of optimizing WPA3-SAE have been proposed for when identity pre-shared keys (iPSK) are used [12]. iPSKs are intended to be used in a scenario where the network password needs to be distributed to a large group of people, and those people might have their access revoked at some point. The example used would be in an apartment where every tennant needs access, but not after they move out. This is accomplished by having the Authentication, Authorization, and Accounting (AAA) server contain a list of passwords based on MAC address. As a result, the AAA server needs to be queried every time a client tries to connect to an AP for the password that matches their MAC address. Guest access might also be configured, which would cause latency when a guest tries to connect to the AAA server being queried for a MAC address that won't be there. In severe cases, this additional latency could lead to a client timeout. The proposed authentication process changes would decrease the time it takes to authenticate guest users and reject login attempts from non registered users. This is accomplished by having the AAA server send out RADIUS messages periodically, containing consolidated bit vector bloom filter details on every client that is registered in the AAA server. If a guest client connects and their MAC address is not in the bloom filter, then the AP should skip the process of querying the AAA server and immediately proceed with guest authentication. If a regular client connects to an AP, the AP will check the bloom filter and see that their MAC address is listed and continue the authentication process as normal. If a client that has been removed from the AAA server is trying to authenticate to an AP before the bloom refreshes, The AP will query the AAA server as if it was still in the database, but the AAA server will respond telling it to proceed with guest authentication.

IV. Security issues discovered regarding WPA3

Early cryptanalysis of the dragonfly protocol showed that offline dictionary attacks can be performed [13]. Recall that during the dragonfly handshake, Alice and Bob have a shared password P. Bob would calculate two scalars r_B and m_B within range of 1-q (q being a prime order) and add them together to get s_B, while Alice does the same to calculate r_A, m_A, and s_A. Both parties Then calculate E_A and E_B. If Bob was an attacker, he could instead calculate E_B to be equal to S_n, where S_n, which is the generator of the primes Z_p^* from 1-q. When E_b and s_b are sent to Alice, she computes a shared secret of ss= $(P^sB \cdot S_n)^{rA} = P^{sBrA} \cdot S^{rAn}$. This shared secret is the only value unknown to Bob in the calculated hash. Using a developed algorithm, they are able to compute $P^{sBrASr} S^{r\ A}_{n} = (P'^{sA} E_A)^{sB} \cdot R_x$, with the only unknown being R_x (a small subgroup element). Cross dividing the terms allows the equation to be rewritten as $P^{sBrASr} / (P'^{sA} E_A)^{sB} = R_x / S^{r\ A}_{n}$. The left term is an element in a subgroup of the prime order q, with the right term is an element in a small subgroup of n So the equation holds when both sides are elements in Z_p^*. Therefore, $(P'^{sA} E_A)^{s\mathbf{B}} = P^{rAsB}$, meaning $P' = P$ and the victim's password has been obtained. With the password, the attacker can send a valid response to Alice, so she has no idea that her password has been compromised. As a result, the attacker can derive the shared session key and connect to the network. In testing, they began by brute forcing prime factors of p-1 and found 5 values and calculated generators for each of the possible small subgroups and performed experiments to determine how long a dictionary attack would take if the correct password was the last element in the dictionary. They found that the attack will produce the password in a feasible amount of time, but it might take long enough that Alice would abort the handshake (but still produce the password). The fix for this vulnerability was to ensure that the received element E_B received by Alice or E_A received by Bob is in fact part of the group being used by the cryptographic scheme and aborting the handshake if it is not.

SAE has also been shown to be vulnerable to impersonation attacks that could be used to bypass authentication [14]. This is more specifically accomplished through the use of a random value attack. This attack would take place when parties Alice and Bob are exchanging keys, with an attacker Eve impersonating Bob. Recall that in the first phase of SAE, Alice generates scalars r_A and m_A from a random space 1-q, while Bob does the same to calculate r_B and m_B. They both then calculate s_A and s_B equal to $r_A + m_A$ mod p and $r_B + m_B$ mod p respectively, as well as $E_A = P^{-mA}$ and $E_{\mathbf{B}} = P^{-mB}$. If Eve is able to impersonate Bob, Eve can calculate the values R_B and m_B to be weak enough to get the password. If Alice starts a handshake with Bob but Eve is able to somehow intercept the message s_A and E_A and respond to Alice as Bob, she can reply with an r_B of 0 and an m_B of anything in the range of 1-q. Alice can't see that r_B is 0 since it is never sent. Eve does need to somehow determine E_B. Since Eve was able to receive E_a from Alice, she can calculate $E_B = E^{m\ B}_{A} = PE^{-mAmB}$ mod p. Now Eve can send s_B and E_B to Alice and calculate a shared secret of PE^{rBrA} modp, which would equal 1 modp since r_B is 0. This could result in Eve calculating the same shared secret that Alice calculates as long as Alice randomly picked $r_A = 1$. This is because the shared secret will be equal to $(PE^{SB} E_B)^{rA}$ mod p if r_A is equal to 1. Therefore, the chance of this attack being successful is 1 in p. The recommended prevention of this attack is to have clients verify that the calculated shared secret is not equal to 1 mod p, if that is what the shared secret ended up calculating to, it is likely because a random value attack is going on. If so, a client should abandon or restart the authentication process. They also mention that adding public key authentication would also greatly help protect against these types of attacks. Finally, it is recommended that the range of random values should instead be {2-(q-1)}, with q being a large prime, to not let the random value be weak.

Dragonblood is a series of vulnerabilities that affect the Dragonfly (SAE) protocol that is used as the first stage in the WPA3 handshake [15]. These attacks range from denial of service to leaking information about the network password using a series of side channel attacks, which can then be used to perform offline brute forcing attacks. The first of these is caused by weaknesses in the hash-to-curve function used to encode the password. The function contains a loop for finding an appropriate value along an elliptic curve function which, in some implementations, short circuits upon finding the right value. This early loop exit could leak an 'element' of the password's generation process, which could assist an attacker in performing an offline dictionary attack to figure out what passwords would perform that many loop iterations.The authors also Analyzed the downgrade protection features of WPA3 and finds them insufficient. WPA3 will downgrade to WPA2 if the client doesn't support WPA3, but can detect if this downgrade is being forced. The problem is, by the time the detection has kicked in, the attacker would have already captured a 4 way handshake and be able to perform an offline dictionary attack on the network password. This can be done by creating a rogue AP with the same SSID as the target and waiting for a client to connect while claiming to support WPA2 only. In some client implementations, this is possible even if the target network only supports WPA3. There are multiple groups of elliptic curve algorithm negotiations that can be chosen by the client, but this process can be interfered with by an attacker. There exist no mechanism to detect interference by a 3rd party, so a 3rd party can block the negotiation request and send a forged reply saying that the AP doesn't support the chosen cryptographic group, forcing a client to use a cryptographic group that might be the least secure one they support. This can be mitigated by having the client remember that WPA3 is supported by a network after connecting the first time and then never connecting to the AP with a weaker handshake than it used initially. Dragonfly's built in DOS prevention relies on stopping repeat communication based on MAC address, but by spoofing MAC addresses and sending 70 requests per second, a typical AP could quickly reach 100% CPU usage, while an attacking raspberry pi would only reach 14%. Possible countermeasures include not using MAC addresses in determining the password element, which would make this specific attack obsolete, or using more efficient hash to curve methods to determine the password, so the password generation process doesn't use such a large amount of the device's resources.Since the MAC address is used in determining the password element an attacker can also spoof their MAC address multiple times to see how that affects the timing, which is another side channel leak. This is fixed in the same way as the other timing attacks by using a constant time method instead. After performing one or more of these side channel attacks, an attacker can use a dictionary of passwords to simulate the hash-to-curve operation that the wireless network did and see if the side channel values match up. If they don't they can be discarded. If they do, they can be tested against the network. In testing, they found that if an attacker was able to take advantage of all of the discovered side channel attacks, they could recover a password located in the popular rockyou.txt dictionary with a 90% success rate. All of these issues have been fixed in wpa_supplicant 2.4.

Shortly after Dragonblood was disclosed, researchers discovered multiple denial of service vulnerabilities that revolve around the SAE mechanism WPA3 uses [16]. The first of these vulnerabilities affects WPA3's 4 way handshake downgrade protection mechanism. WPA3 networks send out a beacon containing, among other things, an RSNE (robust security network element), containing supported cipher suites. Clients use this to pick the strongest one that they also support to use for authentication. During the handshake, both parties make sure that the RSNE the other party sent hasn't been changed at any step, and if one party detects a change, it will abort the handshake. An attacker can set up a device spoofing a legitimate AP that broadcasts that they only support a weak cipher suite (such as WPA2-PSK), which the client will see as it authenticates with the legitimate AP. As the authentication process continues, the client will see the AP broadcasting it's actual cipher suite, which almost certainly will be different and have stronger ciphers available, and aborts. An attacker can repeatedly send beacons as the client tries to connect to keep a target off the network. The authors were able to successfully cause a client to abort its authentication process through this attack in testing. The second attack takes place during the commit phase where both parties will send a tuple to the other. If a party receives a tuple outside of a predefined range, the handshake will be aborted. An attacker can take advantage of this step by waiting for a client to send its first commit message and, by spoofing the AP, reply to the client with a commit message containing an "out of range" error message. This attack worked in testing as well, the client did receive the commit message from the legitimate AP shortly after, but the message was ignored because the client already aborted the handshake. The third and final attack occurs during the step in the handshake where the client sends the AP a commit message stating which elliptic curve or multiplicative group it wishes to use alongside the tuple. If the elliptic curve or multiplicative group isn't supported by the authenticator, The authenticator will reply back that it does not support it. Similar to the second attack, the attacker will act as a man in the middle and send a reply stating that whatever method was chosen isn't supported when a client tries to connect to an AP. The attack worked and was able to keep the client off of the network in 23 out of 23 tests. All these vulnerabilities have a common root cause, which is that the client will act on messages it thinks it is receiving from the authenticator immediately upon receiving them. The proposed solution is to have clients wait upon receiving messages from the authenticator during the handshake before acting upon them. If they did so, they would see not only the frames injected by the attacker, but also the replies from the legitimate AP very shortly after, which they could then use to complete the handshake and connect to the network.

Months later, the same researchers discovered an authentication vulnerability in WPA3 called Bad-Token, as well as a pair of denial or service vulnerabilities that carry over from WPA2 [17]. The vulnerability has to do with the confirm phase of WPA3 authentication where both sides compute values based on the Pairwise Master Key (PMK) and send them to each other to verify that they are correct. If one party notices that the other's value is invalid, it is supposed to abort the handshake and inform the other party. If one of these values was invalid, that should mean that the client is supplying an invalid password. However, if an attacker could inject an invalid value into the handshake, it could prevent a legitimate client from connecting. This would be done by sending a crafted commit message to party A before they are able to receive the commit message from party B or vice versa. They were able to successfully perform this attack in testing using a WPA3 AP and 2 WPA3 clients, acting as a legitimate user and an attacker. The attacker is spoofing the MAC address of the legitimate user and is using invalid network credentials. The legitimate client authenticated, connected, and disconnected to the network successfully; so it would auto connect on future attempts. The attacker then connects to the network at the same time as the legitimate client reconnects. Both clients then sent a commit message, which the AP received and replied back with a commit message. The clients then sent their confirm message to the AP. The confirm message that the attacker generated was received first by the AP, causing the AP to reply back to the legitimate client with an “Unspecified failure” error message. Once the legitimate client receives this message, it aborts the authentication process. The attackers were able to repeat this process, with the attacker relaunching this attack every time the legitimate client retries to authenticate with the AP and were able to keep the client from connecting for a full hour. The proposed countermeasures involve APs not immediately sending a commit message after a client connects, but instead waiting a few milliseconds to see if additional connection frames are received. The authors also note that two types of attacks that exist in WPA2 networks, self connection attacks and ghost attacks, can still be performed in WPA3 networks. In a self connection attack, an attacker spoofs an AP and then attempts to connect to it. Afterwards, the spoofed AP sends out disassociation frames, causing clients to disconnect from the legitimate AP. This can be easily fixed by having APs ignore connection attempts coming from a MAC address that matches their own, which is included in wpa_supplicant 2.7. In a ghost attack, Anattacker can inject a spoofed association response frame that causes a wireless client to abort the authentication process. In testing, the authors were able to successfully prevent an iphone from connecting to a known wifi network using a rogue AP to inject spoofed frames. The proposed solution to this type of attack is to have wifi clients not abort authentication immediately after receiving one association response and instead wait a bit to see if another arrives. If a second one did, the client could've seen the second association response and used that one to complete the handshake.

Over a year after dragonblood, researchers examined how well WPA3 protects against the described downgrade attacks as well as evil twin attacks in general [18]. The downgrade attack is based on WPA3's backwards compatibility. WPA3 support might not be available for all client devices due to the protocol's requirement of 802.11w Protected Management Frames (PMF), especially in IoT devices. As a result, WPA2 support in WPA3 APs is often necessary and is doable through WPA3 transition mode (WPA3-TM). WPA3-TM makes PMF optional and allows devices to connect using WPA2 if they aren't able to connect using WPA3. The problem with WPA3-TM is that it opens up the possibility of downgrade attacks when enabled, which would effectively undo all the security enhancements WPA3 provides if downgrade attacks are performed. Researchers have attempted to find ways to secure WPA3-TM such that downgrade attacks are infeasible. One proposed method from the Wi-Fi alliance is simply running separate WPA3 and WPA2 networks and disabling transition mode, but could confuse the user and result in difficulties getting devices to communicate across the two networks. In the dragonblood paper, the authors recommended that clients should remember if a station they previously connected to is running WPA3 and then refuse to connect to it later if it is running WPA2. The authors of this paper tested to see how WPA3 implementations implemented this mitigation, using a multitude of APs and client operating systems running the newest software as of June 2020. The downgrade attack is dependent on which AP the client will choose to authenticate with, the legitimate WPA3 one or the rogue WPA2 one. One observation that was made is that all stations refused to auto-connect to an AP that has downgraded its authentication method from WPA3 to WPA2 if WPA3-TM was not enabled. With WPA3-TM enabled however, the results were not as good. Android phones will auto connect to the fake WPA2 network if the legitimate AP was initially WPA2, then was upgraded to WPA3-TM because the phone will not save that the AP has a stronger security suite now. iPhones behaved the same as androids and the end user also could not see what security suite was being used by the AP. MacOS would actually show the user what security suite was being used and was resilient against the downgrade attack, even prompting the user if they tried to manually connect to the evil twin that the network they are trying to connect to should've been WPA3. Linux and Windows were also secure like macOS, but if a known WPA2 AP was upgraded to WPA3-TM,they would still reconnect to it using WPA2. When it comes to selecting a BSSID for the first time when an evil twin is present, Windows and Mac select the one with the better security suite, but iOS and Linux's NetworkManager appear to select one at random. The downgrade attack described in dragonblood that involves DoSing a legitimate AP with deauth frames so a client connects to the rogue AP worked on iOS and Android as well, but only when the network has been upgraded to WPA3-TM from WPA2 on Android. That attack wasn't successful at targeting mac devices, but they did find a window of opportunity after a mac laptop comes out of power saving mode and reconnects to the WiFi to perform the downgrade. Deauth worked on windows, but the downgrade did not due to the protections it has against downgrade attacks. Linux was also not vulnerable and could not even be DOS'd. The authors also discuss the concept of having SSID names be unique and discuss SSIDs being FQDNs with signed certificates, doing so would help prevent against evil twin attacks. Finally, the authors point out that the user experience needs to be considered when making big changes to WPA3. While security measures like having separate WPA2 and WPA3 networks with separate passwords, this could result in everyday users being confused as to what network and what password they are supposed to use.

Researchers have developed a framework for fuzzing the dragonfly handshake called dragonfuzz [19]. They took a white box model based approach to fuzzing the handshake, which is protocol and state aware. This results in the fuzzer being able to complete handshake steps normally and change what commands/fuzz it does based on the current step of the handshake. The components of this framework include a C program for fuzzing states of the SAE handshake, a python program for fuzzing the Auth-commit frame, and the fuzzing engine libFuzzer for fuzzing WPA3 functionality in hostapd and iwd. The C program runs on top of aircrack-ng and is capable of completing the entire SAE handshake and achieving complete code coverage. The python program was developed on boofuzz, a generic fuzzing framework designed here to catch simpler mistakes. The primary target of the researchers was hostapd 2.8, which contains fixes for all the vulnerabilities mentioned in the dragonblood paper. Using both the C and python programs, they weren't able to find any new vulnerabilities. The next target was hostapd v2.9-devel. The authors faced multiple difficulties here due to the fuzzer reaching code paths unrelated to SAE, a built in delay mechanism, and intrinsic memory management issues. However, manual code review of iwd did result in the discovery of a denial of service vulnerability in the handling of anti-clogging tokens. Anti-clogging refers to a DoS mitigation in WPA3-SAE for when an attacker sends a flood of forged commit frames to overload resources on a target. Upon receiving an auth-commit frame, an AP will reply with a new auth-commit frame containing an anti-clogging token. The client must then respond with an auth-commit frame with that token in order for it to be processed. The token is created using the MAC addresses of the client and the AP as well as a random secret, so only the AP can create valid tokens. This system allows an AP to throttle the process based on the attacker's identity. The line of code for creating this token contains a call to malloc() on the variable len minus 2 being provided. If len is 1 or 0, this will cause the program to crash since malloc is being called on a negative number. Sending a forged commit frame that claims to carry the commit token and is 34 or 35 bytes in length will cause the len variable to be a 1 or 0 and cause a crash. This vulnerability was patched in iwd 0.18. The main takeaway of this research project is that the model based fuzzing approach that the researchers took does not appear to be the best approach to fuzzing the handshake and instead recommend a fuzzing approach that is based in grey box or black box fuzzing. Due to the heavy parsing in the WPA3-SAE handshake, the authors claim that greybox and blackbox fuzzing would be more appropriate. They also state that iwd 802.11 supplicant software would be a good target for future fuzzing testing, since it has only been open sourced since 2016 and no known attempts at fuzzing it have been conducted.

While WPA3 does add a lot of protection that is not included in WPA2, there still remain parts of the protocol that are unprotected. Researchers have pointed out that Wi-Fi beacons are still vulnerable to forgery attacks [20]. Beacons are periodically sent out by APs to announce the presence of their networks to any clients that are nearby. These beacons use Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) to determine if the channel is clear to transmit by sensing if the medium is idle. Beacons are used to tell clients if they have buffered packets when they are in power saving mode. While Protected Management Frames are mandatory in WPA3, they do not cover beacons. The main type of attack that can be performed is beacon injection, sending forged beacons to clients. While beacons are broadcast and therefore can be sent to clients, they can be sent over unicast to increase the chances that a specific client receives the malicious beacon, due to auto retransmission. An attacker can also increase the chances of a target that is in sleep mode receiving the beacon by timing their malicious beacons to be sent immediately after a legitimate one is sent from an AP. Using this, attackers can silence a station using a power constraints attack. This attack takes advantage of the fact that beacons can be used to inform the clients of a maximum transmit power. By sending spoofed frames, an attacker can cause devices to decrease their transmit power, weakening their connection and lowering the effective range. In testing, this attack only worked on ipads, macbooks, and linux devices. Attackers can also send forged beacons that state that all clients have buffered frames waiting for them, causing them to exit sleep mode, causing the clients to waste energy. To prevent a client from receiving legitimate frames, an attacker can send beacons with incorrect timestamps that indicate when the next beacon will be sent. In Enhanced Distributed Channel Access (EDCA), APs can prioritize different types of traffic based on data frame type using Arbitrary IFS (AIFS), with different maximum and minimum backoff windows (CW_{max} and CW_{min}) . An attacker can forge beacons that set these values by maximising CW_{min}. Doing so against most devices was successful and slowed their communication. APs can send out Channel Switch Announcements (CSAs) to inform clients that it is switching wi-fi channels. An attacker can send out spoofed CSAs that falsely state they are moving to another channel to cause them to disconnect from the network. Repeatedly doing so can potentially keep clients off of the network for an extended period of time. In order to protect against these attacks, researchers have suggested adding an additional Information Element to the beacon. This can simply be done by reusing the Management Message integrity code Element (MME) of the found when using PMF. This allows a client connected to an AP to verify that the AP is the source of the beacon. If a client receives a beacon before connecting, they can verify it with the AP. If it is invalid, it can report the attacker to the AP with a notification frame.

V. Conclusion

While WPA2 is still considered generally secure, it has a number of issues that lead to the creation of WPA3. SAE makes use of elliptic curve cryptography to create a handshake that provides perfect forward secrecy and, when captured, should not be vulnerable to offline brute force attacks. Onboarding of IoT devices has been made easier in a secure way. Encryption for wi-fi networks that are not password protected is now possible with OWE. Multiple things, especially SAE-PK and improvements provided with WPA3-Enterprise, give networks protection against evil twin attacks. While these improvements are significant, vulnerabilities have been discovered in the protocol. The most significant of these is dragonblood, which demonstrates how timing attacks can make offline password cracking possible, how downgrade protection can be attacked to force WPA2 handshakes, and how denial of service attacks can still be achieved. Other vulnerabilities discovered demonstrate that these types of attacks are not simply one-off instances and might continue to be found as time goes on. While they have been patched, they act as a preview for how vulnerabilities might look when WPA3 is deployed in mass. These papers show that WPA3 is a strong step forward for wireless security, but as always, is not going to be the end of wi-fi security issues.

References

[1] Vanhoef, Mathy, and Frank Piessens. "Key reinstallation attacks: Forcing nonce reuse in WPA2." Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.

[2] Bertka, Benjamin. "802.11 w security: DoS attacks and vulnerability controls." Proc. of Infocom. 2012.

[3] Raju, K, Vallikumari, V, and Raju, K. (2011). Modeling and Analysis of IEEE802.11i WPA2-PSK Authentication Protocol

[4] Bednarczyk, Mariusz, and Zbigniew Piotrowski. "Will WPA3 really provide Wi-Fi security at a higher level?." XII Conference on Reconnaissance and Electronic Warfare Systems. Vol. 11055. International Society for Optics and Photonics, 2019.

[5] Viehböck, Stefan. "Brute forcing wi-fi protected setup." Wi-Fi Protected Setup 9 (2011).

[6] D. A. Dai Zovi and S. A. Macaulay, "Attacking automatic wireless network selection," Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, West Point, NY, USA, 2005, pp. 365-372, doi: 10.1109/IAW.2005.1495975.

[7] Harkins, Dan. "Simultaneous authentication of equals: A secure, password-based key exchange for mesh networks." 2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008). IEEE, 2008.

[8] Lee, Byung Moo, et al. "An Easy Network Onboarding Scheme for Internet of Things Networks." IEEE Access 7 (2018): 8763-8772.

[9] Van Ginkel, Richard, Erik Poll, and Thom Wiggers. "Security in public Wi-Fi networks." (2019).

[10] Wi-Fi Alliance. "WPA3 Specification Addendum for WPA3 R3" (2020).

[11] Bartoli, Alberto. "Understanding Server Authentication in WPA3 Enterprise."

[12] Dhammawat, Abhishek, et al. "OPTIMIZED SIMULTANEOUS AUTHENTICATION OF EQUALS (SAE) IDENTITY PRE-SHARED KEY (IPSK)." (2020).

[13] Clarke, Dylan, and Feng Hao. "Cryptanalysis of the dragonfly key exchange protocol." IET Information Security 8.6 (2014).

[14] Sun, Sheng. "A Chosen Random Value Attack on WPA3 SAE authentication protocol." IACR Cryptol. ePrint Arch. 2019 (2019): 801.

[15] Vanhoef, Mathy, and Eyal Ronen. "Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd." Proceedings of the 2020 IEEE Symposium on Security and Privacy-S&P 2020). IEEE, 2020.

[16] Lounis, Karim, and Mohammad Zulkernine. "WPA3 Connection Deprivation Attacks." International Conference on Risks and Security of Internet and Systems. Springer, Cham, 2019.

[17] Lounis, Karim, and Mohammad Zulkernine. "Bad-token: denial of service attacks on WPA3." Proceedings of the 12th International Conference on Security of Information and Networks. 2019.

[18] Dijksman, Raoul, et al. “Securing Home Wi-Fi with WPA3 Personal.” (2020).

[19] Scheuermann, Björn. "Model based fuzzing of the WPA3 Dragonfly handshake." 2019.

[20] Vanhoef, Mathy, Prasant Adhikari, and Christina Pöpper. "Protecting wi-fi beacons from outsider forgeries." Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks. 2020.