Replay Attack Vulnerabilities and Mitigation Strategies

A Study of a Unique Class of Attack and
Several Methods Attempting to Prevent Attacks of a Kind

Alex Goldschmidt

Abstract

Replay attacks are a unique class of network infiltration that have harmful effects both online and offline. With encryption becoming cryptographically stronger every year, a movement from attacks based on decryption to attacks based in replay of encrypted information is almost inevitable. In this project, replay attacks will be looked at more closely, along with several ways to protect against them. Replay attacks have been used to compromise SSL sessions, perform fraudulent transactions with chip-based credit cards, and even to break into cars and garage doors. While many methods have been put forward, including time-based encryption and origin/destination verification, there is no single solution to this problem, but instead one must incorporate several strategies to fully protect against such an attack.

Introduction

In the world of security, there are innumerable methods of compromising a network. Whether the attacker's goal is to shut down communication between parties, or to steal information from an unsuspecting victim, the security system in place must be able to defend against it. The best method of protecting information, which prevents a lot of these attacks from being feasible, is to encrypt all data that is transmitted from one place to another. However, what happens when the attacker doesn't need to know what the message actually says? This is the principle behind the Replay Attack.

Replay Attacks are listed as entry 294 in the Common Weakness Enumeration (CWE), where they are described as a "flaw [that] exists when the design of the software makes it possible for a malicious user to sniff network traffic and bypass authentication by replaying it to the server in question to the same effect as the original message" (Mitre, 2015). Bypassing authentication in this case could mean sniffing for hashed passwords and replaying them back at the server, still encrypted, as was the case with CVE-2005-3435 and CVE-2007-4961. In both of these cases, hashed passwords were simply played back to the server and entry was allowed.

Attacks of this variety aren't limited to the internet either. Chip-on-card systems have been compromised as well because of these attacks, incurring tens of thousands of dollars in fraudulent charges in some cases (Krebs, 2014). White hat hacker Samy Kamkar has also developed a device, dubbed RollJam, which employs a replay attack to trick car and garage doors into opening when a captured signal from a key fob is played back at a different time (Greenberg, 2015). These will both be examined in further detail, but for now it suffices to say that there is nothing a victim can do to prevent these from happening. The fault lies completely with the manufacturer. Even so, there are still a number of ways that developers can protect against replay attacks, and in the same way crackers are refining and improving their attack methods, so too are the "good guys" finding more clever ways to combat them.

To the Community

Every day, people are performing transactions on the network. From the moment a person wakes up, they are sending information across the airwaves and through routers, and most of the time without even realizing the risk they are putting themselves in. Nowadays, everyone is constantly checking their Facebook. In fact, it's probably the first thing a lot of people do in the morning, before even leaving bed. Logging in to the app requires a user's credentials be sent to a server and verified. Once authenticated, the user is given access to their Facebook page. What if someone else, with malicious intentions, was eavesdropping on that communication stream, and was able to pick up on the username and password. Even with the strongest encryption, they could simply relay that information to the server, who would do all the decryption and verification for the attacker. Before the victim has even left their bed, they've already given up anything that links to their Facebook account.

Somehow, though, the unaware civilian makes it out of bed and through their morning routine without an incident. They still need to get to work, though, and it's winter in New England. Nobody wants to fumble around with their keys before they can get into the warm interior, so they click their key

fob from inside the house to unlock the car. Now, not only can they get into their car quickly and with minimal contact with the cold December air, but so can the car thief who hijacked the signal.

Clearly, replay attacks pose a serious threat to everyday security. While there really is nothing this poor victim could have done differently (except maybe cutting themselves off from society all together), it is just as important that they are aware of where things can go wrong, and what measures, if any, they possibly could have taken to prevent attacks in the future. If more people know how serious of an issue this is, they can pressure manufacturers and developers to include even more safeguards in the future.

Defenses

Definition

Replay attack is a general term used to describe a plethora of attack methods. Because there are so many ways to perform such an attack, there is no single way to prevent them. Breaking them down into different classes of attack, though, make it much easier to identify how an attack method is being implemented, and defenses specific to that class of attack can be applied. Paul Syverson, in his paper "A Taxonomy of Replay Attacks", outlines a thorough classification structure for replay attacks. The key concept to recognize is that replay attacks prey on both parties in communication, so attack methods must first be separated into Origin and Destination.

The Origin of an attack can be either internal or external to the running process. An external replay attack occurs when a message from outside the current communication is used. Kamkar's RollJam is an example of a "run external" attack. The message used for the attack came from a previous transaction between key and car. That old message was then replayed in a separate transaction to carry out the attack. This is different from an internal attack, where the message being replayed is part of that same

stream of communication. Consider a Man in the Middle (MitM) attack. In a MitM attack, a malicious third party hijacks the communication channel between two parties. All traffic goes through them, and they can choose to send, modify, or discard any packets they want. If, for example, a person were trying to withdraw money from their bank account, an attacker could identify the critical message, change the bank account number, and replay the modified signal as many times as they wanted. Even if there are layers of encryption protecting the identity of the victim, the bank server will still recognize the transaction as valid because the attacker doesn't need to know who they are stealing from.

The Destination of an attack can either be deflected, or sent straight through. Both aforementioned attacks are examples of straight replays. The intended target was also the target for the attack. The adversary simply delayed, duplicated, or modified the original package to conform to their own purposes. Deflection covers any message that is redirected away from the intended source. There are two possibilities for who could receive the message. First, the signal could be replayed to a third party. If a message from a server is received en route to a user, an attacker could decode the message and possibly create fake messages to trick other users into thinking they are the server. If the message isn't being deflected to a third party, and it isn't being forwarded to the second party, then that only leaves reflection back to the sender. The full taxonomy can be put thusly:

1. Run external attacks
    a. Interleavings
        i. Deflections
            1. Reflections to sender
            2. Deflections to third party
        ii. Straight replays
    b. Classic Replays
        i. Deflections
            1. Reflections to sender
            2. Deflections to third party
        ii. Straight replays
2. Run internal
    a. Deflections
        i. Reflections to sender
        ii. Deflections to third party
    b. Straight replays

Interleaving and Classic Replays describe methods of attack that are either messages from one process being injected concurrently into another process (interleaving) and attacks that don't depend on the time of the session (classic). While delineating attack methods involving these classes is out of the scope of this paper, as they begin to employ much more complex authentication strategies, it is important to note their existence when observing the strengths and weaknesses of different defenses.

Online attacks and defenses

The most obvious attack vector for any replay attack is the Internet. It is massive, crowded, and intrinsically insecure. IEEE 802.11 is the protocol invented in 1997 to standardize network traffic over the internet. It operates at the physical and transport layer in the OSI model (Cisco, 2002). This means that any messages being sent from Alice to Bob goes through this last. It is at the bottom of the stack, so any vulnerability in it means the whole stack is compromised. So, obviously, there are several vulnerabilities in it. The purpose of this paper is not to discuss the faults with IEEE 802.11, but it is important to recognize that the catalyst behind a lot of existing security protocols is the fact that no security is offered at the Physical layer of the OSI model.

One such defense can be found in the Windows Communication Foundation (WCF). The WCF is "a framework for building service-oriented applications" (Microsoft, 2015). It allows clients the ability to send messages between two parties securely. While it includes the requisite encryption standards, such as SSL, it also has the ability to defend against replay attacks, to some capacity. While it is not enabled by default in WCF applications, the capability exists. They implement several common countermeasures for replay detection and avoidance. First is the use of a time-stamp, or some mark for when a message was sent. This is the best way to defend against "run external, classic replay" attacks. If the time the message was sent is outside the acceptable threshold, the message is rejected immediately. However, this defense is useless for interleaved processes, or for any class of internal attack, where the messages being replayed are arriving in the same time period as the original message. One attempt to mitigate this within WCF is

the **ReplayWindow** attribute, which describes how long a message can be valid after it has been sent. This is a check from the sender end as opposed to the timestamp which is validated by the receiver. In concurrent processes, or internal attacks, it's possible for the adversary to be holding on to messages within the system. However, it still does not truly prevent them from replaying signals immediately, in an attempt to overload the server in a DoS style attack. This is what the final defense measure implemented by WCF attempts to do. By keeping a Replay Cache updated, the server can limit the number of messages it will accept at a time. Therefore, if an attacker is spamming the server with the same message again and again, the server will stop them before they completely crash the system. While WCF has many safeguards in place against replay attacks, it is clear that preventing all attack vectors is virtually impossible.

In order for these higher level frameworks to function, though, the lower level protocols need to be airtight. The Secure Sockets Layer (SSL) Protocol is a Transport Layer protocol used to create secure connections between two parties. They use a many layered encryption scheme packed into a calculated Message Authentication Code (MAC) involving a MAC secret, sequence number, message length and contents, and two fixed character strings (Wagner, 1996). Using the MAC secret verifies the source of the message to the destination, which prevents external attacks from happening. Sequencing is another widely used strategy for preventing replay. Legitimate messages are numbered and the recipient will not accept packets out of order. This protects against internal delay attacks or duplication attacks. However, since there is nothing time sensitive about the security measure, a Man in the Middle attack combined with an internal straight replay attack can grant the attacker access to a seemingly secure connection.

One last low level defense is IP Protocol 51, the IP Authentication Header (IP AH). This header is attached to messages on the Transport Layer, and contains many of the techniques employed by SSL 3.0. The IP AH specifies the intended recipient to prevent deflection, sequence number to prevent delay, and length of message to prevent tampering. The IP AH also includes one extra measure, known as a nonce (AH, 2012). A cryptographic nonce is a one-time token used to verify the authenticity and uniqueness of a message. This prevents any sort of duplication attack, even if the sequence number is

tampered with. The only truly secure encryption is the one time key, and a nonce is the closest thing to a one time key in production today.

Offline attacks and defenses

While the taxonomy outlined previously was intended for analyzing replay attacks over the internet, there still exist systems offline that are equally, if not more, vulnerable to replay attacks. The internet is constantly changing and being improved, and changes are very easy to implement since it is all connected. Offline systems are harder to update, because they have no connection, mostly, to a main system. The only way to prevent replay attacks offline is to recognize the threat and prevent it before deploying the product, because once it leaves the factory it's too late to stop an attack.

One dire instance of an offline replay attack involves the "secure" microchip technology found in most credit cards today. The purpose of these chips was to prevent counterfeiting and add an extra layer of encryption to users' information. While the chips have done an incredible job in protecting user information, it appears that the developers were not concerned about replay attacks. There have been several documented cases of chip on card systems being exploited via replay attack to create fraudulent transactions, including two separate attacks originating from Brazil in 2014. The attackers were able to hijack a transaction terminal (ATM, cash register, etc.) and record the outgoing messages. By replaying those same messages back to the bank multiple times, the attackers were able to incur as much as $120,000 in fraudulent charges (Krebs, 2014). Theoretically, this sort of fraud is mitigated by checking the cryptograms, or counters, on transactions. However, the sequencing used in Europay-MasterCard-Visa (EMV) standard is difficult for many banks to implement effectively, meaning that the only safeguard in place against these attacks is often ignored. In the near future, when these microchips become the standard, it is likely that many places will simply disable, or at least lessen, the security measures in order to maintain their regular business. Often with these offline systems, sequencing on its own might not even be enough to prevent replay attacks.

Samy Kamkar's device, RollJam, implements a replay attack to break into cars and garage doors. RollJam is an updated version of a previous device he developed, called OpenSesame. They both exploit the vulnerabilities in fixed and rolling code systems, which are the protocols used in garage and car locks. In a fixed code system, a key is sent from the fob to the receiver with an instruction. The receiver recognizes the code and performs the desired action, like opening. These fixed code systems do not use any form of encryption, which is the first problem. Even so, they still wouldn't be safe from OpenSesame, which records the signal from the transmitter and stores it for later use. When the attacker chooses to return, they simply must play back the signal at the door and it will open.

Fixed code systems have been replaced by rolling code systems in most Remote Keyless Entry (RKE) devices. Rolling codes operate on the same principle as fixed codes, but instead of a single key, the transmitter sends the instruction with a different code every time, almost like a nonce. The garage door keeps a cache of acceptable values, and rotates through them with every valid code. In order to keep the transmitter and receiver synchronized, the receiver will accept several different codes as valid (normally 500). This prevents a person from being locked out of their car because they accidentally clicked the button out of range. This system might protect against OpenSesame, but it is not safe to RollJam. RollJam combines several attack strategies, but the crux of it is the replay attack. First, it causes an offline DoS attack by jamming the receiver. Therefore, when the victim pushes the button to unlock their car, the car never sees the message and doesn't do anything. The user will think there was a glitch, and press the button again. What really happens, though, is RollJam stole the original signal. When the user presses the button a second time, RollJam steals that one too, but plays back the first message it received. Since the car hadn't seen that first message's code, it accepts and allows the victim entry. Now, however, the attacker has a code with a valid sequence code, which can be used any time the attacker wants. This attack has already made victims of many people, including David Beckham, who has had two cars stolen from attackers using this device (Greenberg, 2015). Since these cars are already on the road, and the garage doors are already installed, there is no way to protect them against this attack. Newer models, however, have begun implementing time-based encryption schemes, such as Ultimate KeeLoq.

Conclusion

When it comes to securing devices, it is a fool's errand. No matter what defenses are put in place, it is very difficult to cover every corner case. As the taxonomy, beautifully organized by Syverson, clearly shows, there is no single way to categorize replay attacks. Therefore, there is also no single way to prevent them. There are many systems in place today that attempt to stop them from happening, but even the most secure systems can still be compromised. Tor, a popular open source platform for anonymous web browsing, is considered a highly secure means of traversing the internet, but even it is vulnerable to replay, albeit from very experienced crackers. By hijacking an entry and exit onion router, an attacker can duplicate a packet and send it along. When it gets to the compromised exit router, it will be rejected because it is duplicated, but that's all the attacker needs to confirm the destination and the source of the message, thereby compromising the anonymous connection (Pries, 2008).

Kamkar, and others like him, are critical to preventing more and more advanced attacks, such as the Tor attack described above. The scary truth is that in order to understand how a system needs to be protected, in most cases it first has to be broken. When the breaking is done by white hat hackers like Kamkar, the result is a heightened awareness of a serious issue or even a fix to the problem. Unfortunately, in most cases, these attacks are carried out maliciously, leaving innocent victims behind not knowing what they did wrong.

Wireless security can be thought of as the man trying to fix the holes in the dam. Every time a new hole appears, the man must go and fill it up to stop the water from coming through. Every time he fills it in the dam becomes stronger, but there will always be another place for the water to come in. Still, it is better that the holes get filled in than that they are just left open, even if it is futile to try to stop the water from coming through forever. The only thing one can hope for is that more people join in to help the man filling the holes, because the water in the dam will never go away.

References

AH, Authentication Header. (2012). Retrieved December 15, 2015, from http://www.networksorcery.com/enp/protocol/ah.htm

Greenberg, A. (2015, August 6). This Hacker's Tiny Device Unlocks Cars And Opens Garages. Retrieved December 15, 2015, from http://www.wired.com/2015/08/hackers-tiny-device-unlocks-cars-opens-garages/

Krebs. (2015, October 14). Krebs on Security. Retrieved December 15, 2015, from http://krebsonsecurity.com/2014/10/replay-attacks-spoof-chip-card-charges/

Microsoft. (2015, December 2). Replay Attacks. Retrieved December 15, 2015, from https://msdn.microsoft.com/en-us/library/aa738652(v=vs.110).aspx

Microsoft. (2015, December 2). What Is Windows Communication Foundation. Retrieved December 15, 2015, from https://msdn.microsoft.com/en-us/library/ms731082(v=vs.110).aspx

Mitre. (2015, December 8). Common Weakness Enumeration. Retrieved December 15, 2015, from https://cwe.mitre.org/data/definitions/294.html

Pries, R., Yu, W., Fu, X., & Zhao, W. (2008, May). A new replay attack against anonymous communication networks. In *Communications, 2008. ICC'08. IEEE International Conference on* (pp. 1578-1582). IEEE.

Syverson, P. (1994, June). A taxonomy of replay attacks [cryptographic protocols]. In *Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings* (pp. 187-191). IEEE.

Wagner, D., & Schneier, B. (1996, November). Analysis of the SSL 3.0 protocol. In *The Second USENIX Workshop on Electronic Commerce Proceedings* (pp. 29-40).