Replay attack

Connected to: Packet (information technology) Internet Protocol

Spoofing attack

From Wikipedia, the free encyclopedia

This article needs additional citations for verification. Please help improve this articl...

A **replay attack** (also known as **playback attack**) is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and re-transmits it, possibly as part of a masquerade attack by IP packet substitution.

Another way of describing such an attack is: "an attack on a security protocol using replay of messages from a different context into the intended (or original and expected) context, thereby fooling the honest participant(s) into thinking they have successfully completed the protocol run."[1]

How a replay attack works

Suppose Alice wants to prove her identity to Bob. Bob requests her password as proof of identity, which Alice dutifully provides (possibly after some transformation like a hash function); meanwhile, Eve is eavesdropping on the conversation and keeps the password (or the hash). After the interchange is over, Eve (posing as Alice) connects to Bob; when asked for a proof of identity, Eve sends Alice's password (or hash) read from the last session which Bob accepts, thus granting Eve access.[1]

Prevention and countermeasures

General countermeasure for all replay attacks

Replay attacks can be prevented by tagging each encrypted component with a sessionid and a component number.[1] Using this combination of solutions does not use anything that is interdependent on one another. Because there is no interdependency there are fewer vulnerabilities. This works because a unique, random session id is created for each run of the program thus run to run becomes more difficult to replicate. In this case an attacker would be unable to perform the replay because on a new run the session id would have changed.[1]

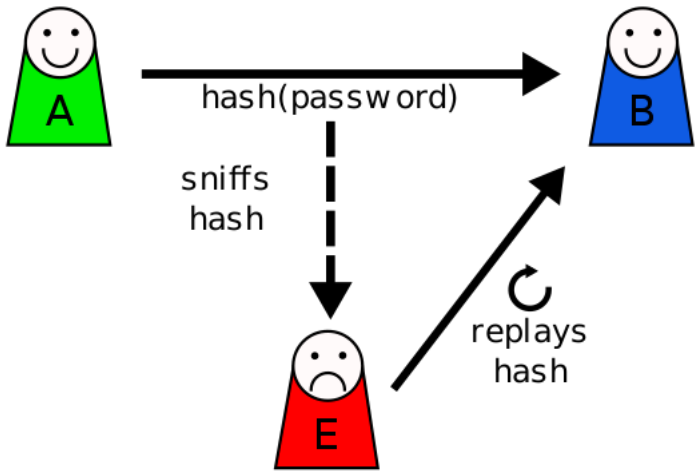

Illustration of a replay attack. Alice (A) sends her hashed password to Bob (B). Eve (E) sniffs the hash and replays it.

More on session identifiers

Session ids, also known as session tokens, are one mechanism that can be used to help avoid replay attacks. The way generating a session id works is as follows.

1. Bob sends a one-time token to Alice, which Alice uses to transform the password and send the result to Bob. For example, she would use the token to compute a hash function of the session token and append it to the password to be used.
2. On his side Bob performs the same computation with the session token.
3. If and only if both Alice's and Bob's values match, the login is successful.
4. Now suppose an attacker Eve has captured this value and tries to use it on another session. Bob would send a different session token, and when Eve replies with her captured value it will be different from Bob's computation so he will know it is not Alice.

Session tokens should be chosen by a random process (usually, pseudorandom processes are used). Otherwise Eve may be able to pose as

Bob, presenting some predicted future token, and convince Alice to use that token in her transformation. Eve can then replay her reply at a later time (when the previously predicted token is actually presented by Bob), and Bob will accept the authentication.

One-time passwords

One-time passwords are similar to session tokens in that the password expires after it has been used or after a very short amount of time. They can be used to authenticate individual transactions in addition to sessions. These can also be used during the authentication process to help establish trust between the two parties that are communicating with each other.

Nonces and MAC

Bob can also send nonces but should then include a message authentication code (MAC), which Alice should check.

Timestamps

Timestamping is another way of preventing a replay attack. Synchronization should be achieved using a secure protocol. For example, Bob periodically broadcasts the time on his clock together with a MAC. When Alice wants to send Bob a message, she includes her best estimate of the time on his clock in her message, which is also authenticated. Bob only accepts messages for which the timestamp is within a reasonable tolerance. The advantage of this scheme is that Bob does not need to generate (pseudo-) random numbers, with the trade-off being that replay attacks, if they are performed quickly enough i.e. within that 'reasonable' limit, could succeed.

Countermeasures in specific scenarios

Kerberos protocol prevention

The Kerberos authentication protocol includes some countermeasures. In the classical case of a replay attack, a message is captured by an adversary and then replayed at a later date in order to produce an effect. For

example, if a banking scheme were to be vulnerable to this attack, a message which results in the transfer of funds could be replayed over and over to transfer more funds than originally intended. However, the Kerberos protocol, as implemented in many versions of LDAP and Microsoft Windows Active Directory, includes the use of a scheme involving time stamps to severely limit the effectiveness of replay attacks. Messages which are past the "time to live (TTL)" are considered old and are discarded.[2]

There have been improvements proposed, including the using a triple password scheme. These three passwords are used with the authentication server, ticket granting server, and TGS. These servers use the passwords to encrypt messages with secret keys between the different servers. The encryption that is provided by these three keys help aid in preventing replay attacks.[3]

Secure routing in ad hoc networks

Wireless ad hoc networks are also susceptible to replay attacks. In this case the authentication system can be improved and made stronger by extending the AODV protocol. This method of improving the security of Ad Hoc networks increases the security of the network with a small amount of overhead.[4] If there were to be extensive overhead then the network would run the risk of becoming slower and its performance would decrease. So by keeping a relatively low overhead the network can maintain better performance while still improving the security.

Real world examples of replay attack susceptibility

There are several real world examples of how replay attacks have been used and how the issues were detected and fixed in order to prevent further attacks.

Remote keyless-entry system for vehicles

Many vehicles on the road use a remote keyless system, or key fob, for the convenience of the user. Modern systems are hardened against simple replay attacks, but are vulnerable to buffered replay attacks. This attack is performed by placing a device that can receive and transmit radio waves

within range of the target vehicle. The transmitter will attempt to jam any RF vehicle unlock signal sent to it, while placing it in a buffer for later use. Upon further attempts to unlock the vehicle, the transmitter will jam the new signal, cache it, and play back the old one, creating a rolling buffer that is one step ahead of the vehicle. At a later time, the attacker may use this buffered code to unlock the vehicle.[5][6]

Game controller measurements

The sensor measurement and movements of game controllers are susceptible to replay attacks. A game controller has the capability to do many maneuvers that create several different measurements that are recorded by the system in order to determine how the player should move, what actions they do, what settings should be used, and more. Replay attacks can be used to record delayed sensor measurements of a controller. By recording this an attacker can then replay those measurements to the controller and cause deterioration in performance and possibly even unstable behavior. This can be done without having to know anything about the system or the controller design.

To prevent such an attack from occurring a signal can be added to the controller that increases the detection rate against replay attacks. This signal is a Gaussian signal that is added to the controller input. It essentially goes through the input and does not allow anything that is finds to be suspicious through. If it detects something that is odd and is suspected to be from an attacker, the detection signal will trigger an alarm that will briefly stop system execution to account for the malicious behavior. This approach seems to work decently however, there is a trade-off between controller performance and attack detection. The added attack detection slows down and hinders the performance of the controller.[7]

Text-dependent speaker verification

Various devices use speaker recognition, to verify the identity of a speaker. In text-dependent systems, an attacker can record the target individual's speech that was correctly verified by the system, then play the recording again to be verified by the system. A counter-measure was devised using spectral bitmaps from the stored speech of verified users. Replayed speech has a different pattern in this scenario and will then be rejected by the system.[8]