A Taxonomy of Replay Attacks

Paul Syverson
Code 5543
Naval Research Laboratory
Washington, DC 20375
(syverson@itd.nrl.navy.mil)

Abstract

This paper presents a taxonomy of replay attacks on cryptographic protocols in terms of message origin and destination. The taxonomy is independent of any method used to analyze or prevent such attacks. It is also complete in the sense that any replay attack is composed entirely of elements classified by the taxonomy. The classification of attacks is illustrated using both new and previously known attacks on protocols. The taxonomy is also used to discuss the appropriateness of particular countermeasures and protocol analysis methods to particular kinds of replays.

Introduction

Cryptographic protocols employ cryptography to achieve some security function. But, for many of these protocols the structure, hence the security, of the employed cryptographic algorithms is not considered to be part of the protocol itself. These algorithms are generically represented by notation capturing only gross features, e.g., whether the algorithm is for public or shared keys, whether it produces a hash function, etc. Thus, cryptographic algorithms are assumed to be not directly breakable within a protocol. Primary focus is on the possibility of a penetrator using messages even when she might not be able to read and/or produce them herself. That is, most of the effort in devising and reasoning about cryptographic protocols is directed at precluding one form or another of replay attack.

Suppose that Ann authorizes a transfer of funds from one account to another by encrypting the transfer request with a signature key known only to her. To get this processed she then sends it to a machine that checks the signature and performs the transaction. If a penetrator[1], Eve, wished to have the same transfer repeated without Ann's knowing authorization, she would not need to be able to produce the encrypted transfer request herself. Assuming she can guess or otherwise determine which message text corresponds to the transfer request, she need only eavesdrop a copy of the encrypted request and then replay it at another time. Something is needed to guard against such replay, in addition to a good encryption algorithm. This example illustrates a very clear form of replay attack. In general we will take 'replay' to mean the capture of a message or a piece of a message that is then used at a later time. This encompasses both cases where the original message is allowed to pass unimpeded and cases where it is prevented from arriving.

It is important to understand that the following is a taxonomy, hence more than a list of replay types (as in [Gon93]). This means that our categorization is hierarchical and that each level in the hierarchy forms a partition of the preceding level. Thus, the taxonomy is trivially complete: all replays (in the sense just given) can be classified as falling into one of the categories at each level of the hierarchy.

The structure of the taxonomy is also determined only by whence messages originate and where messages arrive. Capabilities of the penetrator other than to affect these two factors plays no role in classification; though of course it may play a role in the possibility of a given attack in a given category. Similarly the classification is independent of any ability to detect or prevent penetrator actions. This independence frees us to consider which detection, representation, or prevention mechanisms are appropriate for a replay attack by focusing on where it occurs in the taxonomy. The taxonomy is surprisingly helpful in this regard given that the modelling of threats is so minimal.

1 A Taxonomy of Replays

The full taxonomy can actually be composed from two component taxonomies, one for message origin and one for destination. We will first set out the origination taxonomy and explain it. Then we will do the same for the destination taxonomy.

1.1 Origination Taxonomy

This taxonomy is based on the protocol run of origin for a message (relative to the protocol run in which the replay occurs). 'Protocol run' refers to any single execution of a protocol by the relevant principals.

---

[1] 'Penetrator' is meant to include both outside intruders to a system and legitimate principals misusing the system.

1. Run external attacks (replay of messages from outside the current run of the protocol)
    (a) Interleavings (requiring contemporaneous protocol runs)
    (b) Classic replays (runs need not be contemporaneous)
2. Run internal attacks (replay of messages from inside the current run of the protocol)

Wherever we use 'message' it is acceptable to substitute 'message fragment'. In other words, this taxonomy does not distinguish between attacks where a whole message is replayed and attacks where some piece of a message is replayed. Note also that while the taxonomy partitions replays at each level, some attacks might only be possible if more than one kind of replay is involved. Such attacks might span more than one class of some partition. Should this be confusing, it may be helpful to think of this as a taxonomy of attack elements which may or may not constitute whole attacks. We now describe the classes of attack set out in the above taxonomy.

Run external attacks use message elements from one protocol run in another. A classic example of this is the Denning-Sacco attack on the original Needham-Schroeder key distribution protocol ([DS81], [NS78]). A penetrator who captures a copy of the third message and has time to break the distributed session key can mount an attack. She can replay the third message later, as part of another run of the protocol. As a result she can trick one principal into thinking he has just authentically exchanged a key with another. In reality, the other principal is not there, and the session key is an old, previously broken key. This constitutes a run external attack because it uses a message from outside of the protocol run in which the replay occurs.

Run internal attacks use message elements from within the current run of a protocol. An example of this is the attack on the initial exchange of the Neuman-Stubblebine protocol described in [Syv93b] and [Car93]. In that attack, a penetrator takes part of the second message and uses it to construct the fourth message. The attack allows him to masquerade as someone else and to obtain, or even to generate, a session key. It is a run internal attack because it uses messages from inside a single run of a protocol.

Classic Replays are attacks not requiring contemporaneous runs. We call these 'classic replays' because they have been known about and addressed for some time. The above Denning-Sacco attack is a classic replay. While such an attack uses a message from outside of the current run of the protocol, that message can come from a protocol run that occurred at any time (assuming that the long term keys have not expired).

Interleavings require that two protocol runs overlap in execution.[2] Here is an example. The protocol is one due to Burrows, Abadi, and Needham. It is a variant they give of a draft protocol due to Yahalom, which they discuss in [BAN89].

The BAN-Yahalom Protocol

(1) $A \rightarrow B$: A, N_a
(2) $B \rightarrow S$: $B, N_b, \{A, N_a, \}_{K_{bs}}$
(3) $S \rightarrow A$: $N_b, \{B, K_{ab}, N_a\}_{K_{as}}, \{A, K_{ab}, N_b\}_{K_{bs}}$
(4) $A \rightarrow B$: $\{A, K_{ab}, N_b\}_{K_{bs}}, \{N_b\}_{K_{ab}}$

We refer to the initiator, A, of the protocol and the other principal, B, as 'Ann' and 'Bob' respectively. S is called the 'server'. The protocol runs as follows. Ann sends to Bob her own name (so he knows who is attempting to communicate with him) and a nonce, a random number that she will use to verify the freshness of later messages that contain it. Second, Bob sends to the server his own nonce. In the same message he sends A, N_a, encrypted together with a key, K_{bs}, good only for communication between Bob and the server. In the third message, the server sends to Ann: $N_b, \{B, K_{ab}, N_a\}_{K_{as}}, \{A, K_{ab}, N_b\}_{K_{bs}}$. The first encrypted chunk tells Ann that the server has been talking to Bob, that the message is fresh (via N_a), and gives her the session key. The second encrypted chunk is for her to pass on to Bob. In the fourth message, Ann sends that chunk to Bob. This tells him that the server has recently talked to Ann and gives him the session key. Because Ann has used the session key to encrypt Bob's nonce, the second encrypted chunk lets him know that she has seen the session key recently. We now set out the attack. (Here 'E_x' refers to the penetrator, Eve, masquerading as principal X. If this occurs in the place of an intended recipient, it is assumed that the message is intercepted.)

Attack 1 on the BAN-Yahalom Protocol

(1) $A \rightarrow B$: A, N_a
(2) $B \rightarrow S$: $B, N_b, \{A, N_a\}_{K_{bs}}$

    (1′) $E_a \rightarrow B$: $A, (N_a, N_b)$
    (2′) $B \rightarrow E_s$: $B, N_b', \{A, N_a, N_b\}_{K_{bs}}$

(3) Omitted.
(4) $E_a \rightarrow B$: $\{A, N_a(= K_{ab}), N_b\}_{K_{bs}}, \{N_b\}_{K_{ab}}$

This attack begins with the penetrator either eavesdropping on an initial message from Ann to Bob or sending it herself. (If the latter, then 'A' should be changed to 'E_a' at the appropriate spot above.) After the second message, Eve initiates another run of the protocol masquerading as Ann. She uses N_a concatenated with N_b from the first run as the nonce in step (1′) of the second run. Once she has the encrypted message she intercepts from Bob in (2′), she drops the second run. She then uses this for the first encrypted chunk in the last message of the first run. N_b was previously sent as cleartext, and K_{ab} is actually N_a which also appeared as cleartext. Thus, she can produce the second encrypted

---

[2] 'Interleaving' is used in a related but slightly different way in [BGH+93].

chunk of message (4). At the end of the attack, Eve has masqueraded as Ann to Bob and obtained the distributed session key. It is even possible for her to have generated it herself. The attack assumes that substituting two concatenated nonces for one will go undetected and be passed along when sent to someone who has no need to check the nonce. It also assumes that substituting a random number generated as a nonce for one generated as a session key will be successful.

This attack is an interleaving because it relies on messages constructed of message elements from contemporaneous protocol runs. If the second run is not begun during the first run, Eve cannot successfully complete the attack. Note that both classic replays and interleavings are only subcases of run external attacks. (These categories make no sense for run internal attacks.)

1.2 Destination Taxonomy

The above taxonomy is based on the protocol run of message origin relative to the run where the replay occurs. This taxonomy focuses on the recipient of the message relative to the intended recipient.

1. Deflections (message is directed to other than the intended recipient)
    (a) Reflections (message is sent back to sender)
    (b) Deflections to a third party
2. Straight replays (intended principal receives message, but message is delayed)

The categories in the destination taxonomy are mostly self-explanatory. 'Straight replays' are so called because the message is sent straight from the sender to the intended recipient, though it may be delayed or have other text appended to it, generally altering the significance of the message. Attacks using straight replays are not as unusual as it might appear. For example, in the attack described in [Syv93a] the penetrator reuses a message from Ann to the server (message 2) as a different message from Ann to the server (message 4) in another round of the protocol. The protocol involved in this attack was an artificial example used for illustrative purposes. However, straight replay attacks exist on protocols posited for use. Here is another attack on the BAN-Yahalom protocol that involves a straight replay.

Attack 2 on the BAN-Yahalom Protocol

(1) $A \rightarrow E_b$: A, N_a

$$
\begin{aligned}
&(1')\ E_b \rightarrow A:\ B, N_a \\
&(2')\ A \rightarrow E_s:\ A, N'_a, \{B, N_a\}_{K_{as}} \\
&(2'')\ E_a \rightarrow S:\ A, N_a, \{B, N_a\}_{K_{as}} \\
&(3')\ S \rightarrow E_b:\ N_a, \{A, K_{ab}, N_a\}_{K_{bs}}, \\
&\qquad\qquad\qquad \{B, K_{ab}, N_a\}_{K_{as}}
\end{aligned}
$$

(2) Omitted.
(3) $E_s \rightarrow A$: $N_i, \{B, K_{ab}, N_a\}_{K_{as}}, \{A, K_{ab}, N_a\}_{K_{bs}}$
(4) $A \rightarrow E_b$: $\{A, K_{ab}, N_a\}_{K_{bs}}, \{N_i\}_{K_{ab}}$

The potential damage of this attack is not so immediately great as that of Attack 1. It only results in Eve spoofing Ann into thinking that she has exchanged a key with Bob. There is no release of the session key. Nonetheless, it relies on less assumptions about things missed by any implementation, e.g., substituting nonces for keys or doubled nonces for nonces. There is only the rather weak assumption that Ann will not detect the reflection in the second run of her nonce from the first run.

Attack 2 nicely illustrates all of the possible kinds of destination replays. There is the just mentioned reflection. There is a straight replay of cryptotext from message ($2'$) in message ($2''$) of the second run. And, there is a third party deflection of the cryptotext from message ($3'$) in message (3).

The origin and destination taxonomies are essentially independent of each other. Thus, we can construct the full taxonomy by appending either one to the finest levels of distinction in the other, i.e., by forming the cross product either way. Since protocol run is roughly a broader abstraction than message recipient, it makes some intuitive sense to append the destination taxonomy at the finest level of origination taxonomy. That is the full taxonomic structure we adopt.

1.3 Full Taxonomy

1. Run external attacks (replay of messages from outside the current run of the protocol)
    (a) Interleavings (requiring contemporaneous protocol runs)
        i. Deflections (message is directed to other than the intended recipient)
            A. Reflections (message is sent back to sender)
            B. Deflections to a third party
        ii. Straight replays (intended principal receives message, but message is delayed)
    (b) Classic replays (runs need not be contemporaneous)
        i. Deflections (message is directed to other than the intended recipient)
            A. Reflections (message is sent back to sender)
            B. Deflections to a third party
        ii. Straight replays (intended principal receives message, but message is delayed)
2. Run internal attacks (replay of messages from inside the current run of the protocol)
    (a) Deflections (message is directed to other than the intended recipient)
        i. Reflections (message is sent back to sender)
        ii. Deflections to a third party
    (b) Straight replays (intended principal receives message, but message is delayed)

2 Using the taxonomy

Now that we have the taxonomy we take a brief look at some applications of it.

2.1 Looking at countermeasures

This taxonomy is useful for readily determining the effectiveness of some replay countermeasures and associated concepts. For example, the existence of interleaving attacks has prompted occasional discussion of the inappropriateness of freshness mechanisms for general prevention of replay. Some have proposed in their stead mechanisms for tying messages to a particular protocol run rather than to a particular epoch. Thus, even if an interleaving involves only fresh messages, that the messages are from different protocol runs would be revealed by such a mechanism. The possibility of run internal attacks shows that this too is inappropriate if the goal is to preclude replay in general. In truth, both notions are relevant but simply not sufficient to the overall task. With respect to the taxonomy, mechanisms addressing only freshness are only of value against classic replays. (Of course, they may be generally useful for determining message expiration.) Similarly, mechanisms localizing to a particular run are only of value against run external attacks.

Another kind of countermeasure is one that indicates who a message is from, who it is to, or both. Some examples of these are discussed in [Mit89] with respect to reflection attacks on particular protocols. The taxonomy delimits the applicability of these sorts of countermeasures as well. One mechanism Mitchell discusses is encrypted from fields, which cryptographically bind the name of the message originator to the message. These will work against reflections but not against deflections to a third party or straight replays. Another mechanism discussed precludes mistaking either sender or receiver of a message via specialized use of shared keys. This will rule out both reflections and third party deflections but not straight replays. Some other related countermeasures are discussed in [Gon93]. As in [Mit89], discussion is explicitly limited there to countering reflections.

Both of those papers propose introducing asymmetry between messages X sends to Y and those Y sends to X as a simple means of countering replay. This is also proposed in [BGH$^+$93]. As mentioned, we should only expect this to be effective in the context of reflections. We must also take care that format asymmetry is not itself attackable. We have discussed old and new protocols above in which the attack involves using a nonce for a key. For example, in the attack in [Syv93b] and [Car93] cited above, a message of the form $\{A, K_{ab}, T_b\}_{K_{bs}}$ is substituted for one of the form $\{A, N_a, T_b\}_{K_{bs}}$. A related substitution was made in Attack 1 above on the BAN-Yahalom Protocol. Nor is it sufficient simply to vary the number of fields, even inside of a piece of cryptotext. Attack 1 also involved failure to detect whether a plaintext message had three fields or two (all the more likely if nonce lengths are allowed to vary). This failure lets a legitimate principal be tricked into producing a three field cryptogram where he was to produce a two field cryptogram. How easy it is to prevent these sorts of problems plays a pivotal role in how easy it is to introduce effective asymmetry in a protocol format.

2.2 Looking at analysis methods

This taxonomy is also useful for determining the appropriateness of analysis techniques to represent and/or reveal replays. For example, BAN logic, [BAN89], is clearly directed only at what we have called classic replays, the use of run external messages from before the current protocol run began. This is not to say that a BAN analysis will never uncover attacks of another kind. But, for example, it does not generally handle interleavings since freshness will not reject messages from other runs that were sent (verifiably) after the beginning of the current run. This is no criticism of BAN. Rather it is an express demarcation of what one can generally hope to reason about by using it.

One of the first expansions of BAN is the logic GNY, presented in [GNY90]. Amongst other things this logic adds means to assess vulnerability to reflection attacks. (BAN analysis sidesteps reflections. The authors explicitly assume that principals can recognize their own messages.) GNY has notation and rules to limit a principal's inferences regarding a message to cases where the message originated elsewhere. GNY is thus expressive enough to represent at least some of both run internal and run external reflections. It does not explicitly address either deflections or straight replays. The analysis technique discussed in [GNY90] is further limited to run internal reflections where the significance of the message does not change. Thus, according to the taxonomy, GNY adds only a little to the replay types detectable by BAN. Nonetheless, we see that GNY does go beyond classic replays in dealing logically with these concerns. (GNY also addresses other concerns not covered by BAN.)

Other logics appear to have more general replay representation capabilities ([Bie90], [KG91], [Syv93a]), as do other protocol analysis methods ([MCF87], [SM93], [BGH$^+$93]). Some of these appear capable of representing replays of all kinds. However, detection is another matter. To date only the NRL protocol analyzer ([Mea91], [Mea92]) appears to be generally capable of detecting all types of replays. It is hoped that the above taxonomy will aid in the development of new and existing techniques for addressing replay attacks.

Acknowledgements

I would like to thank Cathy Meadows for helpful discussions and for providing examples of protocol attacks that motivated the discovery of one of the attacks presented herein. I also thank the anonymous referees for helpful comments.

References

[BAN89] Michael Burrows, Martín Abadi, and Roger Needham. A Logic of Authentication. Research Report 39, Digital Systems Research Center, February 1989. Parts and versions of this material have been presented in many places including *ACM Transactions on Computer Systems*, 8(1): 18–36, Feb. 1990. All references herein are to the SRC Research Report 39 as revised Feb. 22, 1990.

[BGH+93] Ray Bird, Inder Gopal, Amir Herzberg, Phil Janson, Shay Kutten, Refik Molva, and Moti Yung. Systematic Design of a Family of Attack-Resistant Protocols. *IEEE Journal on Selected Areas in Communication*, 11(5):679–693, June 1993.

[Bie90] Pierre Bieber. A Logic of Communication in Hostile Environment. In *Proc. Computer Security Foundations Workshop III*, pages 14–22. IEEE Computer Society Press, Los Alamitos, California, June 1990.

[Car93] Ulf Carlsen. Using Logics to Detect Implementation-Dependent Flaws. In *Proceedings of the Ninth Annual Computer Security Applications Conference*, pages 64–73. IEEE Computer Society Press, Los Alamitos, California, December 1993.

[DS81] D. E. Denning and G. M. Sacco. Timestamps in Key Distribution Protocols. *Communications of the ACM*, 24(8):533–536, August 1981.

[GNY90] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about Belief in Cryptographic Protocols. In *Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society Press, Los Alamitos, California, 1990.

[Gon93] Li Gong. Variations on the Themes of Message Freshness and Replay or, the Difficulty of Devising Formal Methods to Analyze Cryptographic Protocols. In *Proceedings of the Computer Security Foundations Workshop VI*, pages 131–136. IEEE Computer Society Press, Los Alamitos, California, 1993.

[KG91] Rajashekar Kailar and Virgil D. Gligor. On Belief Evolution in Authentication Protocols. In *Proceedings of the Computer Security Foundations Workshop IV*, pages 103–116. IEEE Computer Society Press, Los Alamitos, California, 1991.

[MCF87] J.K. Millen, S.C. Clark, and S.B. Freedman. The INTERROGATOR: Protocol Security Analysis. *IEEE Transactions on Software Engineering*, SE-13(2):5–53, February 1987.

[Mea91] C. Meadows. A System for the Specification and Analysis of Key Management Protocols. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 182–195. IEEE Computer Society Press, Los Alamitos, California, 1991.

[Mea92] C. Meadows. Applying Formal Methods to the Analysis of a Key Management Protocol. *Journal of Computer Security*, 1:5–53, 1992.

[Mit89] C. Mitchell. Limitations of Challenge-Response Entity Authentication. *Electronic Letters*, 25(17):1195–1196, August 1989.

[NS78] Roger M. Needham and Michael D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[SM93] Paul Syverson and Catherine Meadows. A Logical Language for Specifying Cryptographic Protocol Requirements. In *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 165–177. IEEE Computer Society Press, Los Alamitos, California, 1993.

[Syv93a] Paul F. Syverson. Adding Time to a Logic of Authentication. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 97–101. ACM Press, New York, November 1993.

[Syv93b] Paul F. Syverson. On Key Distribution Protocols for Repeated Authentication. *Operating Systems Review*, 27(4):24–30, October 1993.