Derailing Attacks

Saša Radomirović and Mohammad Torabi Dashti

Institute of Information Security
Dept. of Computer Science, ETH Zurich
{rsasa,torabidm}@inf.ethz.ch

Abstract. We introduce derailing attacks, a class of blocking attacks on security protocols. As opposed to blunt, low-level attacks such as persistent jamming, derailing only requires a minimal, application-level intervention from the attacker. We give a simple definition of derailing attacks in an abstract formal model, and demonstrate that derailing attacks are viable in practice through examples from two application domains, namely radio-frequency identification and fair exchange protocols.

Keywords: Formal Models · Availability · RFID · Fair Exchange

1 Introduction

Alice and Bob play correspondence chess by postal mail. They are both absorbed in and obsessed with their game, which has been ongoing for several months. Charlie, Alice's boyfriend, is not happy with the situation: he feels he has lost Alice to chess. If only he could stop their game! Charlie thinks of intercepting the posts coming from Bob, and destroying them. This is however an expensive and risky solution. Charlie knows that Alice would get suspicious if no mail from Bob arrived for a while, and she might even get worried about Bob, which Charlie would not like. She would perhaps start calling up Bob or even consider to travel to meet Bob. Charlie's interception would therefore work only if he can cut off Alice and Bob's communication through *all* means (e.g. telephone, emails, and homing pigeons) and for an indefinite length of time. This, of course, is infeasible.

Charlie realizes that there is a much simpler and cheaper solution: he just needs to make Alice believe that Bob is playing wrong moves. Then Alice would stop playing with Bob, because she would think that he is either trying to cheat or he has lost his total devotion to the game and is making mistakes. To reach his goal, Charlie only needs to make their chess boards inconsistent. For example, Charlie could modify a post from Alice to Bob so that Bob would think Alice makes a certain move, while Alice makes another. This would change Bob's chess board in one direction, while Alice's would change in another. Once they lose the consistency of their chess boards, Charlie can sit back and observe Alice's growing frustration. She will eventually give up on playing with Bob, and Charlie will win back Alice.

This correspondence chess scenario illustrates blocking attacks, where the adversary's goal is to prevent, or prematurely terminate, communication between two parties. A trivial means of blocking is simply to disable the communication media altogether. This is a blunt approach because it does not allow the attacker to selectively block communication. It could moreover be unsustainable if the attacker incurs the prohibitive costs of, say, jamming a signal indefinitely. A more refined approach is to exploit the features of the communication protocol for terminating the exchange of information. Examples include the Chinese firewall's use of TCP packets for resetting TCP sessions, and using IEEE 802.11 standard "deauthentication" packets for ending a wireless session. In this paper, we introduce *derailing attacks*, a class of more elaborate blocking attacks on communication protocols. A derailing attack on a protocol disrupts the communication between the protocol's participants by corrupting their internal states. Charlie's clever trick is a simple instance of these: Charlie injects a message that adversely affects Alice and Bob's states, namely it makes their chess boards inconsistent.

In Section 2, we describe two instances of derailing attacks on published protocols, and in Section 3, we discuss how derailing attacks can be thwarted. There we also formally define derailing attacks. We conclude the paper in Section 4.

2 Derailing Attacks in Practice

Below, we give two examples of derailing attacks. These attacks have been studied in the literature, but they have not been recognized as belonging to the same class of attacks.

Our first example comes from the domain of radio frequency identification (RFID) protocols. RFID tags are inexpensive devices that communicate wirelessly with RFID readers. A large class of RFID protocols store correlated "state information" on the tag and on the reader's side. RFID reader and tag are said to be *synchronized* if the tag's state information allows the reader to uniquely identify the tag. For privacy concerns, this information is updated after each transaction between a tag and a reader. The update cannot be carried out atomically due to concurrency. An adversary can exploit a flawed update protocol to force the tag and the reader into a desynchronized state. Once they are there, they can no longer communicate. Desynchronization resistance has been formally defined in [2] and a number of protocols vulnerable to desynchronization attacks have been investigated in [3]. Desynchronization is an instance of derailing attacks.

Note that, similarly to Charlie's hesitation to destroy the posts, the desynchronizing adversary would prefer a derailing attack over perpetually blocking the communication between the tag and the reader. The latter is not only unsustainable and practically infeasible, it is also intrusive and overt, thus more likely to raise an alarm. For example, the owner of the RFID tag might become concerned about the stalker who shows up around all RFID readers. We remark that Charlie's trick and the RFID desynchronization attacks do not take effects immediately. This is in contrast to the Chinese firewall scenario where

the attacker can tear down a session immediately by sending a TCP FIN or RST packet.

Our second example pertains to fair non-repudiation protocols. Zhou, Deng and Bao [6] proposed a protocol where P sends a message to Q containing a certain key k encrypted with the public key of a trusted third party T. The encrypted key is meant to enable Q to achieve fairness with the help of T. That is, Q and T should be able to generate a non-repudiation token whenever P and T are able to generate such a token; we do not need to further specify the token for our purpose. Now, since Q cannot check if the cipher text he receives is in fact k encrypted with the public key of T, a malicious P can derail the protocol by sending a fake key to Q after having received a non-repudiation token from Q. Consequently Q cannot achieve fairness; indeed, Q and T cannot successfully communicate. Boyd and Kearney [1] discovered the flaw, and proposed a fix for it. However, their fixed protocol suffers from a similar derailing attack. Then, Zhou [5] proposed another way to mitigate the flaw; his mitigation is not effective either. The story of the protocol is summarized by Gürgens, Rudolph and Vogt [4].

Note that in the example of fair non-repudiation protocols a derailing attack undermines a security requirement of the protocol, namely its fairness. In the RFID example an availability requirement is violated. However, the RFID authentication requirement is not necessarily affected by the attack. Suppose that an RFID protocol is deployed in a building to keep out unauthorized people. The reader would open a protected door only if it can authenticate a person by identifying the tag the person carries. Now, derailing attacks do not undermine the "security" of the building: an unauthorized person is kept out. So are authorized ones if the attack succeeds.

In the next section, we discuss how derailing attacks can be mitigated.

3 Thwarting Derailing Attacks

Authentication and integrity protection are seemingly sufficient for thwarting derailing attacks. For example, if Alice and Bob were able to recognize the tampered messages, then Charlie's trick would be fruitless. Similarly, the aforementioned RFID desynchronization attacks are mitigated by ensuring that the protocol satisfies Lowe's agreement property. This would however run contrary to the resource constraints of low-cost RFID tags. A detailed description of this point falls outside the scope of this note.

Note that, in the aforementioned fair non-repudiation example, even if all the messages exchanged between P and Q were authenticated, the attack would still be possible. This is because T cannot determine whether P or Q was the origin of the fake encryption of k. By modifying the protocol such that P must send to Q not only the encrypted key k but also its signed hash, the derailing attack is mitigated. This is intuitively because then T is able to identify the origin of a malformed token: if P were the origin, then T would issue an affidavit to Q, as a replacement for the non-repudiation token; cf. [4].

In short, an evidence of tampering in the correspondence chess problem and the RFID example allows the participants to detect, and thus thwart, derailing attacks. Similarly, in the fair non-repudiation example, requiring P's signature would provide Q with an evidence of tampering, and would enable T to identify the culprit. This discussion illustrates that protection against derailing cannot be readily reduced to well-understood security requirements such as message authentication. Below, we give a simple definition of derailing attacks in an abstract formal model.

A Formal Model

A formal definition of derailing attacks can be given through a notion of safe and unsafe configurations. We start by defining a simple protocol model.

A participant executing a protocol is a state machine that exchanges messages over an asynchronous communication medium and changes state upon reception of a message. We assume the presence of an additional state machine, the *outsider*. We distinguish between two types of messages. The green messages, which are sent by a protocol participant, and the red messages, which are sent by the outsider. State transitions that occur due to reception of a green message are called green transitions and similarly for red transitions.

The set of configurations $\mathcal{C}$ (global states) of a protocol is the Cartesian product of the sets of states of the protocol's participants. The state transition of participants induces a natural transition relation on the set of configurations. A protocol's security requirements define a *safe* set of configurations $\mathcal{S} \subset \mathcal{C}$. Intuitively, these are the configurations in which all security requirements are met. The safe configurations are not closed under the transition relation due to asynchrony of communication. During the execution, the protocol may therefore leave the set of safe configurations. We define the set of *transitional configurations* $\mathcal{T} \subset \mathcal{C}$ as configurations from which the protocol can reach a safe configuration without taking any red transitions. The transitional configurations trivially contain the set of safe configurations. The set of *unsafe configurations* $\mathcal{U}$ is the complement of the set of transitional configurations. Figure 1 illustrates these three sets.

A protocol is susceptible to *derailing attacks* if there exists a sequence of transitions (green or red) that takes the protocol from a safe configuration to an unsafe one. We assume that by making use of secure communication channels, we can eliminate the red transitions from the set of transitions induced by a protocol execution. We illustrate these notions on our two protocol examples.

Example 1. Consider the RFID protocol. The security requirement is that the RFID tag can authenticate itself to the reader. The set of safe configurations consists of pairs of states, where the tag and reader are synchronized. For simplicity we assume that they are synchronized when they store the same key. The outsider is an attacker with the standard Dolev-Yao capabilities except for the ability to indefinitely block communication. To update their keys the protocol

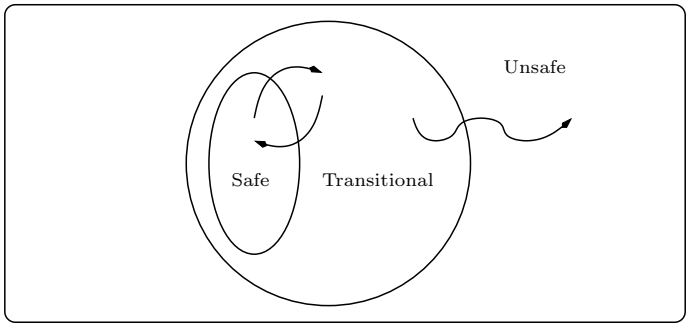

Fig. 1. Safe, transitional and unsafe configurations.

participants must leave the set of safe configurations due to asynchrony. The outsider can interfere with the protocol, by injecting a red message, so that the tag and reader update their keys differently. This prevents further communication and leaves the protocol in an unsafe configuration.

If the communication channels were secure, the red message would not be accepted by the protocol participants. That is, the set of unsafe states would not be reachable for a well-engineered protocol. An unrestricted Dolev-Yao attacker could, of course, block the communication between reader and tag and keep them in a transitional configuration. As discussed before, this is unsustainable.

Example 2. Consider the aforementioned fair exchange protocol. The security requirement to be satisfied is that Q can obtain a non-repudiation token whenever P can and vice-versa. Therefore the set of safe configurations consists of pairs of states in which either both P and Q have a non-repudiation token or neither of them has.

Note that if only P, but not Q, receives a non-repudiation token, restarting the protocol by instructing Q to "forget" about the current exchange does not result in a safe configuration. Moreover, the attack described in Section 2 still succeeds even if all communication channels are secure, i.e., only green transitions are possible. The attacker in this example is not an outsider.

Derailing attacks can be characterized as the reachability of a configuration from which all paths to safe configurations include a red transition. That is, in its abstract form derailing poses a *nested reachability* problem: Can the system reach a configuration from which no safe configuration is reachable (only using green transitions). Let us write $\rightarrow_g \subseteq \mathcal{C} \times \mathcal{C}$ for the set of green transitions, and $\rightarrow_r$ for the set of red ones. Let $\rightarrow = \rightarrow_g \cup \rightarrow_r$, and write $\mathcal{C}_0$ for the set of initial configurations. Then, the system is susceptible to derailing if and only if

$$\exists C_0 \in \mathcal{C}_0, C \in \mathcal{C}. \quad \big(C_0 \rightarrow^* C \ \wedge \ \neg \exists C_S \in \mathcal{S}.\ C \rightarrow_g^* C_S\big).$$

Here, $\rightarrow_{\cdot}^*$ is the transitive closure of the relation $\rightarrow_{\cdot}$.

The formalization of this property in a security protocol verification tool is not straightforward, due to the aforementioned nested reachability problem. Moreover, in such a tool, the outsider is often identified with the communication network. This makes it difficult to automatically distinguish between red and green messages. Investigating this point is left for future work.

4 Conclusion

We have introduced derailing attacks on availability, and have demonstrated their viability in two application domains. Derailing demands minimal effort from the attacker, and it is covert. These characteristics set derailing apart from blunt, overt blocking attacks, such as persistent jamming. In contrast to common denial of service attacks on availability, derailing cannot be fully addressed through replication. Formal verification techniques can mitigate the problem by assisting protocol designers to find the flaws that enable derailing. While the primary purpose of this note is to bring attention to this class of attacks, we have also given a lightweight formalization of derailing. Further investigations in this direction are left for future work.

Acknowledgments. We thank Jan Cederquist and Sjouke Mauw for insightful discussions.

References

1. Boyd, C., Kearney, P.: Exploring fair exchange protocols using specification animation. In: ISW '00. LNCS, vol. 1975, pp. 209–223. Springer (2000)
2. van Deursen, T., Mauw, S., Radomirović, S., Vullers, P.: Secure ownership and ownership transfer in RFID systems. In: Proc. 14th European Symposium On Research In Computer Security (ESORICS'09). Lecture Notes in Computer Science, vol. 5789, pp. 637–654. Springer (2009)
3. van Deursen, T., Radomirović, S.: Attacks on RFID protocols (version 1.1). Cryptology ePrint Archive, Report 2008/310 (August 2009), http://eprint.iacr.org/2008/310
4. Gürgens, S., Rudolph, C., Vogt, H.: On the security of fair non-repudiation protocols. Int. J. Inf. Sec. 4(4), 253–262 (2005)
5. Zhou, J.: Achieving fair non-repudiation in electronic transactions. Journal of Organizational Computing and Electronic Commerce 11(4), 253–267 (2001)
6. Zhou, J., Deng, R., Bao, F.: Evolution of fair non-repudiation with TTP. In: ACISP '99. LNCS, vol. 1587, pp. 258–269. Springer (1999)