A protocol's life after attacks...

Giampaolo Bella[1,2], Stefano Bistarelli[3,4], and Fabio Massacci[5]

[1] Computer Laboratory, University of Cambridge, UK
[2] Dip. di Matematica e Informatica, Università di Catania, Italy
giamp@dmi.unict.it
[3] Dip. di Scienze, Università "G. D'annunzio" di Chieti-Pescara, Italy
bista@sci.unich.it
[4] Istituto di Informatica e Telematica, CNR, Pisa, Italy
stefano.bistarelli@iit.cnr.it
[5] Dip. di Informatica e TLC, Università di Trento (Italy)
massacci@ing.unitn.it

Abstract. In the analysis of security protocols, it is customary to stop as soon as we find an attack. Tons of ink can be spilled on whether an "attack" is really an attack, but it goes without saying that there is no life after that, hence no interest in continuing the analysis. If the protocol is broken, then we ought to fix it.
Yet, fixing things is expensive and other measures may be more effective. In the physical world, most ATM safes would not resist heavy shelling with anti-tank bazookas, but banks don't worry about that. The attack will be noisy enough that cops will come within seconds from its start. To secure ourselves, we rely on a mixture of measures including the protection from attacks but also countermeasures after detection.
In the light of these considerations, the following question becomes of interest: *what can happen after an attack?* Does the villain leave enough traces that we can retaliate it on-the-fly? Or, if we can't or won't, does a subsequent forensic analysis allow us to discover who did it (and send the cops behind him)? If even this is impossible, can we discover that we have been hacked by looking at the logs?
To address these issues, we introduce the notions of *retaliation*, *detection*, and *suspicion*, which can be applied after an attack. These properties introduce more sophisticated formal relations between traces of actions, which go beyond the simple existentials that formal methods have made us used to.
These concepts should allow for a more comprehensive evaluation of security protocols. A protocol may well be vulnerable to an attack, but if we can retaliate afterwards, maybe fixing it isn't that necessary: the concrete possibilities of retaliation or detection may be enough to convince potential hackers to refrain from mounting the attack.

1 Introduction and Motivations

What is a security protocol, if we set technology aside? It is just a social behavior that principals of a distributed system must follow to obtain some important

collective benefits. For the good guys, we just set up clear, understandable, and acceptable rules describing this behaviour: execute the security protocol correctly, namely by the book. Because they are good guys, they will conform to the rules, and behave as we wanted. The bad guys, by definition, will not conform to the rules and execute the protocol incorrectly, namely arbitrarily.

Classical research in distributed systems and security starts from the need to counter the disruptive behavior of the nasty ones. In classical distributed algorithms, the main focus has been to design the protocol so that if the good guys outnumber the bad ones, the collective benefits will be achieved, no matter what the bad guys do (alternatively, prove that the good guys are doomed, no matter how many they are and how smart they are [4]). The security standpoint is to find a design such that, no matter what the bad guys do and no matter how many they are, they can prevent the good guys from achieving the desired collective benefits.

Each time an attack is published, the implied corollary is that the security experts have failed: the protocol has flaws, the good guys cannot achieve their ultimate goal, and we should go back to the drawing board. Yet, before abandoning the protocol, it is worth looking at what is left after the attack. This can lead us to more comprehensive evaluation of security protocols even if it requires continuation of a protocol analysis after an attack is found.

1.1 Our Contribution

It has never been considered whether it is at all possible to threaten the bad guys in case they execute the protocol incorrectly. In the real world, we impose a virtuous behavior on people by not letting them sin first (the classical security approach), and by making them repent of their sins ever since (sending them to jail). They would therefore weigh up the benefits of an incorrect execution with the consequent threats, and might choose to execute the protocol correctly if the threats were heavier.

Let's consider Lowe's example in his paper on Needham-Schroeder [6]. After the end of the attack, the bad guy asks for a transfer of money. Would he steal 1.000 Euro if the threat that 2.000 Euro could be stolen to him on the next day were significant? Would he transfer the money if the chances of being caught were significant?

In this paper, we propose a number of notions for the analysis of protocols beyond attacks: we introduce the notions of *retaliation*, *detection*, and *suspicion*. For instance, in a peer-2-peer environment or in network games, *direct retaliation* can be an effective threat to force the bad guys to play by the rules.

In the next section we show how Lowe's attack on Needham-Schroeder can be retaliated [1]. Then, we introduce the notion of retaliation, followed by those of detection and suspicion.

2 Retaliation in Needham-Schroeder

Let's start from a classical case: the (in)famous Needham-Schroeder public key protocol represented in Figure 1. We use the classical notation for security protocols [2]:

- keys are denoted by K, possibly extended with subscripts expressing the principals knowing them; the $^{-1}$ superscript expresses the inverse of a key;
- nonces are denoted by N;
- concatenation is denoted by a comma;
- encryption is denoted by a pair of curly braces with the key as a subscript; the type of the key determines the type of encryption.

1. $A \rightarrow B : \{\!|Na, A|\!\}_{Kb}$
2. $B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}$
3. $A \rightarrow B : \{\!|Nb|\!\}_{Kb}$

Fig. 1. The asymmetric Needham-Schroeder protocol

The goal of the protocol is *authentication*: at completion of a session initiated by A with B, A should get evidence to have communicated with B and, likewise, B should get evidence to have communicated with A. Assuming that encryption is perfect and that the nonces are truly random, authentication is achieved here by confidentiality of the nonces. Indeed, upon reception of Na inside message 2, A would conclude that she is interacting with B, the only principal who could retrieve Na from message 1. In the same fashion, when B receives Nb inside message 3, he would conclude that A was at the other end of the network because Nb must have been obtained from message 2, and no-one but A could perform this operation.

Lowe discovers [6] that the protocol suffers the "attack" described in Figure 2, whereby a malicious principal C masquerades as a principal A with a principal B, after A initiated a session with C. The attack, which sees C interleave two sessions, indicates failure of the authentication of A with B which follows from failure of the confidentiality of Nb.

Let's examine this protocol after the attack took place:

- B is the subject of the attack — we call him the *good* agent;
- C is the *bad* guy;
- A is just playing by the rules — we call him the *ugly* participant;
- the trace T of the protocol as in Figure 2 describes an attack

The predicate $Attack(T, \mathcal{G} := \{B\}, \mathcal{B} := \{C\}, \mathcal{U} := \{A\})$ can represent the previous fact with the meaning:

1. $A \rightarrow C : \{\!|Na, A|\!\}_{Kc}$
1′. $C \rightarrow B : \{\!|Na, A|\!\}_{Kb}$
2′. $B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}$
2. $C \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}$
3. $A \rightarrow C : \{\!|Nb|\!\}_{Kc}$
3′. $C \rightarrow B : \{\!|Nb|\!\}_{Kb}$

Fig. 2. Lowe's attack to the Needham-Schroeder Protocol

- B executes a run of the protocol apparently with A (in fact he receives/sends the following messages:$[\{\!|Na, A|\!\}_{Kb}, \{\!|Na, Nb|\!\}_{Ka}, \{\!|Nb|\!\}_{Kb}] \in T$);
- C knows the nonce Nb, which was meant to be known to A and B only by receiving the message $\{\!|Nb|\!\}_{Kc} \in T$ in step 3 (so he can complete the run with B and perform the attack).
- A plays according to the rules and is just a spectator in the battle between B and C.

By using the authentication attack if B is a bank for example, C can steal money from A's account as reported by Lowe [6]:

4. $C \rightarrow B : \{\!|Na, Nb,$ "Transfer £1000 from A's account to C's"$|\!\}_{Kb}$

The bank B would honour the request believing it came from the account holder A.

Notice however, that the same predicate *Attack* also holds if instantiated in a different way [1]. In fact, we have also $Attack(T, \mathcal{G} := \{A\}, \mathcal{B} := \{B\}, \mathcal{U} := \{C\})$:

- A executes a run of the protocol apparently with C (in fact he receives/sends the following messages:$[\{\!|Na, A|\!\}_{Kc}, \{\!|Na, Nb|\!\}_{Ka}, \{\!|Nb|\!\}_{Kc}] \in T$);
- B knows the nonce Na, which was meant to be known to A and C only, by receiving the message $\{\!|Na, A|\!\}_{Kb} \in \mathcal{T}$ in step 1′ (so he can send in step 2 the message $\{\!|Na, Nb|\!\}_{Ka}$ also without having C sending it to A).
- C is in this run just playing according to the rules.

In this case B is the bad guy. As C did previously, B can equally decide to illegally exploit his knowledge of Na. If also A is a bank, B can rob the robber as follows:

2″. $B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}$
4″ $B \rightarrow A : \{\!|Na, Nb,$ "Transfer £2000 from C's account to B's"$|\!\}_{Ka}$

The bank A would honour the request believing it came from the account holder C. After we found the attack $Attack(T, \mathcal{G} := \{B\}, \mathcal{B} := \{C\}, \mathcal{U} := \{A\})$ in the

trace T, we could have continued the analysis and find out that the bad guy could be easily punished, as in the example given above.

3 Vendetta...

To introduce the notion of retaliation we must identify the role played by each principal in the protocol execution. We are not interested in notions such as responder or initiator; rather, we intend to provide a behavioural characterization.

Good principals are denoted by the set $\mathcal{G}$. They currently obey the protocol, but may change their mind in the future when attacked.

Bad principals are denoted by the set $\mathcal{B}$. They also take part in the protocol, possibly communicating with friendly and neutral agents but, crucially, they can as well collaborate with the intruder who runs the network.

Ugly principals are denoted by the set $\mathcal{U}$. Usually they mind their own business and conform to the protocol. However they can also both collaborate with corrupt agents or help in future the friendly agent to retaliate.

Informally, we assume that

- *a trace* $\mathcal{T}$ is a protocol run, namely a sequence of sending, receiving and memorizing actions,
- *a protocol* P is the set of all traces of the protocol[1] (all possible executions),
- *a projection* $\mathcal{T}/\mathcal{A}$ of a trace $\mathcal{T}$with respect to a set of agents $\mathcal{A}$ is the subsequence of the actions in $\mathcal{T}$ that are performed[2] by an agent in $\mathcal{A}$,
- *an attack* is a trace $\mathcal{T}$ that satisfies some property $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$.

We take the following definition of vulnerable protocol:

Definition 1. A protocol P is vulnerable to an attack A for the principals in $\mathcal{G}$ *if there exists a trace* $\mathcal{T} \in P$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$.

The property as stated is suited for model checking, whereas people working with theorem provers will try to prove that there is no such trace.

Definition 2. A protocol P is immune to an attack A for the principals in $\mathcal{G}$ *if there exists no trace* $\mathcal{T} \in P$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$.

What is the essence of retaliation? Each time you try to cheat, we can hit you back. So, do you still want to cheat?

This makes sense if hitting you back is a meaningful property in the context of the protocol. If the corrupted principals $\mathcal{B}$ are protocol participants, then we can assume that they want to reap the benefits of the protocol (such as authentication), plus any additional benefits they may obtain by misbehaving. These latter benefits should be balanced with the threat of being hit back. So, we simply need to pay an attacker with the same coin.

[1] This can be defined in the formal model of choice: CSP [10], inductive definitions [8], strand spaces [12].

[2] Note that the action in which A sends a message to B is included in the projection over the set $\{A\}$, but not in the projection over the set $\{B\}$, as B might never receive that message.

Definition 3. A protocol P allows the retaliation of an attack A onto the principals in $\mathcal{B}$ *iff for every trace* $\mathcal{T} \in P$ *such that* $A(T, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$*, there exists a trace* $T_r \in P$ *extending* $\mathcal{T}$ *such that* $A(\mathcal{T}_r, \mathcal{B}', \mathcal{G}', \mathcal{U}', Spy)$*.*

In particular,

- when we have $\mathcal{B}' = \mathcal{G}$ we speak of *direct retaliation*;
- when $\mathcal{B}' \cap \mathcal{G} \neq \emptyset$ we have a *combined retaliation*;
- lastly, when $\mathcal{B}' \subset \mathcal{G} \cup \mathcal{U}$ we have an *arbitrary retaliation.*

This definition follows the intuition that each time there is an attack (a trace $\mathcal{T}$ such that $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$ some additional actions (hence $\mathcal{T}_r$ extending $\mathcal{T}$) can be done to retaliate. Who acted initially as friendly is now ready to act as corrupted (possibly against who was corrupted initially).

The property as stated is useful for theorem proving: this is a positive guarantee to make bad guys play by the rules. For model checking, and thus finding that not only are there attacks but you are even powerless against them, we need to turn the quantifiers.

Definition 4. A protocol P is vulnerable to an attack A without chance of retaliation onto $\mathcal{B}$ *if there exists a trace* $\mathcal{T} \in P$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$ *and no trace* $\mathcal{T}_r \in P$ *extending* $\mathcal{T}$ *is such that* $A(\mathcal{T}_r, \mathcal{B}', \mathcal{G}', \mathcal{U}', Spy)$ *where* $\mathcal{B}' \in \mathcal{G} \cup \mathcal{U}, \mathcal{B} \in \mathcal{G}' \cup \mathcal{U}'$*.*

The important caveat here is that we assume that the formerly friendly principals will jump side and collaborate with the *Spy*, owner of the network, to frame the formerly corrupted principals. The spy will act as a devil helping the current corrupt agent (whoever he/she is). This view may be subject to debate but the key observation is that we need in any case some collaboration with the network provider, otherwise after the attack she could simply start killing every message that is sent on the network.

4 Detection

The notion of *detection* is more complicated and requires the comparison of alternative traces. Let's see the intuition: you look at your logs and spot some fishy actions (the projection of the full trace with respect to yourself). These actions are enough to bring the bad guys to trial if you can show that there is no other full trace which would have left the same marks on your logs and not included the attack.

In contrast, detection would fail if the bad guys could show at Court one trace, beside yours with the attack, with the same marks on your logs, but without the attack. Indeed, it would have been enough for them to show a trace with an attack from somebody else. This would have been sufficient to get acquittal.

Definition 5. A protocol P allows the detection of an attack A from $\mathcal{B}$ *if for every trace* $\mathcal{T} \in P$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$*, for all traces* $\mathcal{T}_r \in P$ *such that* $\mathcal{T}_r/\mathcal{G} = \mathcal{T}/\mathcal{G}$*, it is* $A(\mathcal{T}_r, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$.

The intuition is that each time there is an attack on a trace $\mathcal{T}$, the friendly agents can check their own logs $\mathcal{T}/\mathcal{G}$ and pinpoint who acted corruptly, namely the principals in $\mathcal{B}$. The same logs could not have been created by some other set $\mathcal{B}'$ of corrupted principals.

The model checking version of the same property requires two pieces of evidence: the logs of the good guys and the "plausible" no-attack alternative concocted by the bad guy's counsel for the defence.

Definition 6. A protocol P is vulnerable to an attack A without possibility of detection of $\mathcal{B}$ *if there exists some trace* $\mathcal{T} \in P$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$ *and there exists a trace* $\mathcal{T}_r \in P$*,* $\mathcal{T}_r \neq \mathcal{T}$*, such that* $\mathcal{T}_r/\mathcal{G} = \mathcal{T}/\mathcal{G}$ *and it is not* $A(\mathcal{T}_r, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$.

5 Suspicion

Suspicion is a weaker version of detection: each time there is an attack (a trace $\mathcal{T}$ such that $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$) we can look at the logs (the projection $\mathcal{T}/\mathcal{G}$) and conclude that something fishy happened but we cannot pin-point the villain.

Also in this case, the model checking version of the same property, requires two pieces of evidence: the logs of the good guys and the "plausible" no-attack alternative concocted by the bad guy's counsel for the defence.

Definition 7. A protocol P allows the suspicion of an attack A *if for every trace* $\mathcal{T} \in P$ *and set* $\mathcal{B}$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$ *then for all traces* $\mathcal{T}_r \in P$ *such that* $\mathcal{T}_r/\mathcal{G} = \mathcal{T}/\mathcal{G}$ *there are sets* $\mathcal{U}'$ *and* $\mathcal{B}'$*, such that* $A(\mathcal{T}_r, \mathcal{G}, \mathcal{B}', \mathcal{U}', Spy)$.

In a nutshell: each time the log matches with your logs, the traces include some attack by somebody.

The model checking version is by far more complicated:

Definition 8. A protocol P is vulnerable to an attack A without possibility of suspicion *if there exists some trace* $\mathcal{T} \in P$ *and set* $\mathcal{B}$ *such that* $A(\mathcal{T}, \mathcal{G}, \mathcal{B}, \mathcal{U}, Spy)$ *and there exists a trace* $\mathcal{T}_r \in P$ *such that* $\mathcal{T}_r/\mathcal{G} = \mathcal{T}/\mathcal{G}$ *and for all sets* $\mathcal{U}'$ *and* $\mathcal{B}'$ *it is not* $A(\mathcal{T}_r, \mathcal{G}, \mathcal{B}', \mathcal{U}', Spy)$.

6 Implications for Formal Protocol Verification

We have been used to hair-splitting formal analysis of authentication. Can we formally analyze properties such as retaliation?

From a theoretical standpoint there is not a big difference. We have casted our properties as properties of traces because almost all research in tool-supported security verification is based on defining authentication or confidentiality as a property of traces [5–10] or fragments thereof [3, 11].

The key observation is that the emphasis in the traditional work on security verification was on finding attacks or showing that no attack existed. This was reflected on formal models by the nature of the checked properties, which were essentially of existential nature: is there a trace $\mathcal{T}$ in the protocol P such that $A(\mathcal{T})$ holds? Here, $\mathcal{T}$, P, and A could be complicated at wish. Indeed, A, as a formally defined property, could be awfully complicated (for instance including arithmetical constraints on the number of actions and arbitrarily many quantifiers). However, the trace as such, was only existentially qualified. Theorem proving fellows wished to prove that no such a trace existed, model checking guys longed for a witness of its existence.

Here, the properties are more complex: we run at least two quantifiers over traces. In some cases we have alternation of quantifiers (such as retaliation). Lifting the theory is simple. Lifting the automatic tool support will be the real challenge.

References

1. G. Bella and S. Bistarelli. Confidentiality levels and deliberate/indeliberate protocol attacks. *Springer, LNCS, Proc. Security Protocols 10th International Workshop, Cambridge, UK, April, 2002, Revised Papers*, to appear.
2. M. Burrows, M. Abadi, and R.M. Needham. A logic for authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.
3. Luigia Carlucci Aiello and Fabio Massacci. Verifying security protocols as planning in logic programming. *ACM Transactions on Computational Logic*, 2(4):542–580, 2001.
4. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.
5. R. Kemmerer, C. Meadows, and J. Millen. Three system for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.
6. G. Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, 1995.
7. J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murphi. In *Proceedings of the 16th IEEE Symposium on Security and Privacy*, pages 141–151. IEEE Computer Society Press, 1997.
8. Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.
9. Focardi R. and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.
10. Steve Schneider. Security properties and CSP. In *Proceedings of the 15th IEEE Symposium on Security and Privacy*, pages 174–187. IEEE Computer Society Press, 1996.
11. Dawn Song. Athena: An automatic checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 1999.
12. F. Thayer Fabrega, J. Herzog, and J. Guttman. Honest ideals on strand spaces. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 1998.