Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp310-315)

On Formal Verification Methods for Password-based Protocols: CSP/FDR and AVISPA

ABDELILAH TABET, SEONGHAN SHIN, KAZUKUNI KOBARA, and HIDEKI IMAI
Institute of Industrial Science
University of Tokyo
4-6-1 Komaba Meguro-ku Tokyo 153-8505
JAPAN

http://imailab.iis.u-tokyo.ac.jp

Abstract: Formal verification methods have proved a high talent in finding potential attacks automatically in several security protocols. So far, many formal methods have been proposed in the literature. In this paper we checked the abilities of two well-known checking tools, CSP/FDR and AVISPA, in detecting off-line attacks that may exist in password-based authentication protocols. For this, we apply these two formal methods to several variants of password-based protocols, vulnerable to off-line attack, so that we analyze the results and then show the weaknesses of each method.

Key-Words: Passwords**,** Off-line Attacks, Authentication Protocols, Formal Methods, Casper, CSP, FDR, AVISPA

1 Introduction

With the rapid spread of use of Internet and the sharp growth of network-based services, the communication security has become more relevant than any other time before. Sometimes, malicious attackers with little of efforts get authenticated to servers and get access to important documents or services illegally. For these reasons security protocols received a lot of attention and loomed on the horizon as a potential solution. Literature has a huge number of security protocols aiming to establish a secure communication channel where secrecy and authentication are assured. However, literature also showed that many of them failed to reach what they were designed for.

The most likely to happen is off-line attacks that seem difficult to predict in many security protocols by just hand-analysis. In order to check security holes automatically, formal verification methods have been proposed so far. Even though specifying a protocol and its verification via formal methods are a hard task, it is very important to do before any implementation.

Formal methods with different approaches were introduced to fulfill an exact specification and analysis of the security protocols. Literature has showed that model checking has been proven to be a very successful approach for analyzing security protocols. Their basic approach is to compare a model of the protocol along with a model of the intruder. Therefore the checking tool can explore all the state space of the abstraction model and extract counterexamples states in which security protocol failed to fulfill its security goals. Moreover it also gives a possible scenario of the attack which an intruder can mount. However, model checking is limited to a finite system containing a small state space; otherwise it will be confronted with the state explosion problem.

Murphi (by Mitchell et al. [1]), Brutus (by Marrero et al.[2]), ESTELLE [7], NRL Protocol Analyzer [8,9,10], FDR [5], and AVISPA by AVISPA-project group [12] are examples of model checking tools. In particular, model checking using CSP/FDR has been widely used in formal methods [4,11] and frequently cited in the security literature, after finding man-in-the-middle attack of Needham-Schroeder public key protocol [4,5]. AVISPA also received much interest by declaring that it can find almost all the known and some unknown attacks in several security protocols [13].

This paper is constructed as follows: Section 2 introduces a password-based security protocol for authentication and off-line attacks. Section 3 gives a brief introduction to formal methods to be used in analysis, and the results of these methods on several

password-based authentication protocols. In Section 4, we discuss about why the formal methods get the results with the conclusion that they are not enough for detecting off-line attacks in security protocols. In addition, we show a verification result of AVISPA on an example that is a modified protocol from one (cited in the library of protocols checked by AVISPA). Finally, we summarize this work.

2 Password-based Protocols

Owing to the usability and convenience of passwords, password-based authentication protocols have been extensively investigated for a long time where two parties (e.g., Alice and Bob) share a short password in advance. However, designing a secure authentication protocol is not trivial since there are existing two major attacks on passwords: on-line and off-line attacks. The on-line attack is a series of exhaustive search for a secret performed on-line, so that an attacker can sieve out possible secret candidates one by one communicating with the target party. In contrast, the off-line attack is performed off-line massively in parallel where an attacker exhaustively enumerates all possible secret candidates, in an attempt to determine the correct one, by simply guessing a secret and verifying the guessed secret with recorded transcripts of a protocol. While on-line attacks are applicable to all of the password-based protocols equally, they can be prevented by letting a server take appropriate intervals between invalid trials. But, we cannot avoid off-line attacks by such policies, mainly because the attacks can be performed off-line and independently of the parties.

2.1 Off-line Attacks

As one trivial example of off-line attacks, consider a simple unilateral authentication protocol (like CHAP [16]) in which one party sends a random challenge c and the other party replies with $r=H(pw,c)$, where pw represents the shared password and H is a cryptographic hash function. While this protocol can be proven secure (for an appropriate choice of H) when pw has high entropy, it is completely insecure when the entropy of pw is small. Indeed, in the latter case a passive attacker who obtains a single transcript (c,r) can run an off-line attack, trying all values of pw' until one satisfying $r=H(pw',c)$ is found. Consequently, ensuring immunity to off-line attacks in password-based protocols has been a very critical issue in research fields.

However, many of these protocols failed to fulfill its security purpose in the case where users chose short passwords. Such situation can be very dangerous since it would be easy for an attacker to mount a dictionary attack on the password and therefore break the secrecy of the authentication which the protocol was designed to establish. In this paper, we focus on off-line attacks in password-based authentication protocols.

3 Casper/FDR and AVISPA

This section introduces two well-known formal methods for verifying security properties (i.e., secrecy of password and authentication) of password-based protocols. The two methods to be used in this paper are CSP/FDR [3,5] and AVISPA verification tool [12].

3.1 CSP/FDR

The FDR (Failure Divergence Refinement) is one of the widely-used verification methods especially after it exposed the man-in-the-middle attack of Needham-Schroeder public key protocol [4]. The checker FDR is a model-checking tool for concurrent and reactive systems modeled in CSP (Communication Sequential Processes) [3]. Since generating CSP code is a time consuming and error-prone task, Casper was produced by Gavin Lower in 1997 [14] to overcome CSP code writing difficulties. Casper compiles an easy input script to a CSP code. Therefore it simplifies the task for non-experts and non familiars with CSP code to produce a CSP code without having much knowledge about its notations. In security protocols, FDR is restricted to a small type of systems where, for example, nonces and keys types are finite. The input of Casper must contain the protocol definitions, the type of system to be checked, the initial knowledge of the agents and the specification of protocol's goals; it defines the data types to be used and the intruder's abilities.

3.2 AVISPA

AVISPA stands for Automated Validation of Internet Security Protocols and Applications; it has been realized by the AVISPA project group. The AVISPA verification tool is publicly available since June 2005: it consists of independently developed, but interconnected, modules. A protocol designer specifies the protocol along with the security properties which the protocol is supposed to achieve by the High-Level Protocol Specification Language

HLPSL [15]. The HLPSL is a role-based formal language that allows specifying data structures, alternative intruder models, complex security properties, different cryptographic primitives and their algebraic properties.

Once a protocol is specified by the HLPSL, it is translated into equivalent IF specifications by the HLPSL2IF translator. This IF specification is input to the back-ends of the AVISPA tool, which implements different analysis techniques. The current version of the tool, to be used in this paper, integrates four back-ends: On-the-fly Model-Checker (OFMC), Constraint-Logic-based Attack Searcher (CL-AtSe), SAT-based Model Checker (SATMC), Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP).

4 Analyzing Security Protocols using CSP/FDR and AVISPA

In this section, we treat eight password-based authentication (and key exchange) protocols: a generalization of password-based authentication and key exchange protocol (Protocol I), IPsec PSK aggressive and main modes (Protocol II and III), two examples of Diffie-Hellman based encrypted key exchange (Protocol IV and V), three examples of RSA-based encrypted key exchange (Protocol VI, VII and VIII). In fact, most of the protocols we prepare here are vulnerable to off-line attacks. That's the reason why we want to see whether CSP/FDR and AVISPA can really find out security holes in the protocols.

For simplicity, we denote two parties as Alice and Bob with each ID, *A* and *B*, respectively. In addition, "*h*" means a cryptographic hash function and "||" means concatenation.

4.1 Generalization of Password-based Authentication and Key Exchange (Protocol I)

In Protocol I, Alice and Bob share a short password *pw* and want to establish a common session key through a challenge-response protocol. Alice first sends her ID *A* along with a nonce *r1*. Based on these values, Bob calculates *v2=h(pw||SID||"v2")* where *SID=h(r1||r2||A||B)*, *r2* is a nonce chosen by Bob and "*v2*" is a pre-shared word between Alice and Bob, then sends *v2* along with its ID *B* and *r2*. Responding to this, Alice replies with *v1=h(pw||SID||"v1")* where *SID=h(r1||r2||A||B)*. In the last step of Protocol I, Alice and Bob check the validity of *v1* and *v2*, respectively, and then the shared secret key is computed as *sk=h(pw||SID||"sk")* where "*sk*" is a pre-shared word that Alice and Bob will use for the construction of the common key.

For analysis of this protocol, we first use CSP/FDR by modeling its specification and security properties with Casper. Since the password is shared between Alice and Bob, we checked the secrecy of the password and the authentication of the protocol. As a result, we find that the intruder is able to mount a man-in-the-middle attack between the legal users in order to obtain *v2* which serves as a verifier for his guess. The only unknown value to the attacker is *pw*, therefore mounting an off-line dictionary attack on the password and computing the value *v2* the attacker easily verifies its guess. This attack is listed below:

1. Alice → I_Bob: R1, Alice
1. I_Alice → Bob: R1, Alice
2. Bob → I_Alice: Bob, R2, h(pw,R1,R2,A,B)
2. I_Bob → Alice: Bob, R2, h(pw,R1,R2,A,B)

Then we analyze the same protocol by the AVISPA tool in order to verify the previous result we got from CSP/FDR. We described the protocol by using HLPSL and checked the secrecy and the authentication of the protocol. However, the AVISPA tool said that the protocol is "Safe": which means that the password is always not guessable for an attacker even when it is poorly-chosen one.

4.2 IPsec PSK Aggressive and Main Modes (Protocol II and III)

In Protocol II (IPsec PSK aggressive mode), Alice and Bob share a short password *pw* and want to construct a secret key *sk=h(SEC||SID||"sk")* where *SEC=h(pw||r1r2G)* and *SID=h(r1G||r2G||A||B)*. First Alice sends her ID *A* and a Diffie-Hellman public value *r1G*. Then Bob replies with *B,r2G,v2* where *B* is his ID, *r2G* is a Diffie-Hellman public value and *v2=h(SEC||SID||"v2")*. Bob computes the Diffie-Hellman key *r1r2G* by using *r1G* received from Alice. Similarly Alice computes *v1=h(SEC||SID||"v1")* and sends it to Bob. Now both Alice and Bob are able to compute the secret key *sk* if *v1* and *v2* are correct.

In the analysis of this protocol, we described the protocol using Casper notations. However, Casper doesn't allow describing the algebraic properties of the Diffie-Hellman protocol. Note that Casper instead allows its characterization by a public and a secret key when it is the case of asymmetric and, a symmetric key

when it is the case of symmetric encryption. For Protocol II, we define a new datatype *F = G | Exp(F,Nonce)*, unwinding 2 as well as defining commutativity and associativity properties of the exponent, in order to give the legal users and the attacker the possibility to calculate the Diffie-Hellman key based on the Diffie-Hellman public value of the other. We checked the secrecy and the authentication of the protocol using FDR. As a result, FDR was not able to converge and therefore not able to say whether Protocol II is safe or not.

By AVISPA, we described the protocol using the algebraic properties of exponent provided by HLPSL language. As a result, the backbone CL-AtSe responsible for checking protocols failed to detect any attack.

Similarly we checked IPsec PSK main mode (Protocol III) which looks mostly like the aggressive mode except that Alice and Bob use the Diffie-Hellman key as a session key to encrypt their IDs and *v1* and *v2*. In fact, Alice first sends *r1G*. Bob generates a session key *sk=h(r1r2G)*, use it to encrypt his ID *{B}_sk* and send it along with his Diffie-Hellman public value *r2G*. Alice replies by *{A,v1}_sk*, which is calculated the same way as in the previous protocol and Bob also sends *v2* encrypted by *sk*. As the previous protocol, we modeled the protocol by using the *Exp* datatype mentioned above, gave to the intruder the possibility to masquerade as Alice, share the session key with Bob and then mount a dictionary attack on the password. However, FDR cannot decide if it is secure or not. AVISPA's answer on this protocol is "Safe".

4.3 Diffie-Hellman based Encrypted Key Exchange Protocols (Protocol IV and V)

We prepare two examples of EKE protocols where one is secure and the other is insecure. Both are based on the Diffie-Hellman protocol over some algebraic operations like multiplication. In the secure protocol (Protocol IV), Alice and Bob share a password *pw* and two different generators *G,Q* of a finite group. Alice first sends her ID *A* along with *y1=r1G+pwQ* where *r1* is a nonce. Bob replies by his ID *B* along with *y2=r2G+pwQ* and *v2=h(pw||r1r2G||y1||y2||A||B||"v2")*. On receiving these messages from Bob, Alice sends her *v1=h(pw||r1r2G||y1||y2||A||B||"v1").* After a check of the values *v1* and *v2*, Alice and Bob compute the secret key sk=h(*pw||r1r2G|| y1|| y2|| A||B||"sk"*).

For the analysis of this protocol, we regard the multiplication as encryption since Casper doesn't allow many algebraic operations. Therefore, we model *y1* as *{r1G}_pwQ*. Besides that, we define *Exp* datatype, and its associativity and commutativity properties as in the previous section. As a result, we get a "!" meaning that the program is able to converge to an exact result. However, AVISPA found no attack on the protocol.

By the same way, we analyzed Protocol V which looks exactly like the former except that this protocol utilizes only one generator *G* of the group. By replacing *Q* in the former with *G*, we get the new protocol that is not secure again off-line attacks. However, the results we got for the former are the same for this one.

4.4 RSA-based Encrypted Key Exchange Protocols (Protocol VI, VII and VIII)

We verify the security of three examples of RSA-based EKE protocols. In the first example (Protocol VI), Alice and Bob share a password *pw*. First Alice generates a nonce *r1* and sends it with her ID *A* to Bob who generates a RSA public key *(e,n)*, and a nonce *r2*, and sends them along with his ID *B*. Alice generates a random value *t* in order to construct a secret key *SEC=h(pw||t)* and sends *t* to Bob under the from *{t*h(pw)}_(e,n)*. Since only Bob has the RSA private key *(d, n)*, he is the only one who is able to decrypt this message and sends *v2=h(SEC||SID||"v2")* where *SID=h(r1||r2||A||B)*. Alice replies by *v1=h(SEC|| SID||"v1")* after verifying *v2*. If Bob verifies *v1*, they can compute the secret key *sk=h(SEC||SID||"sk").*

We modeled the protocol using Casper and effectuating some modifications, such as minimizing the number of variables at the input of the hash functions without affecting the security properties of the protocol, in order to avoid the state space explosion. FDR detects an attack described as follows:

1. Alice → I_Alice: *Alice, R1*
1. I_Alice → Bob: *Alice, R1*
2. Bob → I_Alice: *Bob, R2, PK*
2. I_Alice → Alice: *Bob,R2, PKm*
3. Alice → I_Alice: *{{T}{Pw}}{PKm}*
3. I_Alice → Bob: *{{T}{Pw}}{PK}*
4. Bob → I_Alice: *h(Pw,T,R1,R2)*

Here I_X means that the Intruder is masquerading as the legal user X in this protocol and *PKm* is the intruder's public key. The intruder will mount a man-middle attack to be able to guess the value of the password in the end. Therefore, Alice believes *Pw* is a secret shared with Alice, however, the intruder knows *Pw*. But, AVISPA says that this protocol is "Safe".

The second example (Protocol VII) is similar to the former one. First Alice sends *r1* and *A*, a nonce and her ID respectively to Bob who will respond by *r2, B*, and *(e,n)*. Using Bob's RSA public key, Alice selects a random value *t* and sends it, which is encrypted by RSA and later masked with the password (*{{t}_(e,n)}_pw*), along with *v1*. If *v1* is correct, Bob sends *v2*. If *v2* is correct, Alice and Bob can compute the secret key *sk* as in the first example.

In the analysis, we found an attack by FDR as follows:

1. Alice → I_Bob: *Alice, R1*
2. I_Bob → Alice: *Bob, R1, PKm*
3. Alice → I_Bob: *{{T}{PKm}}{Pw},h(Pw,T,R1,R2)*

In this attack, the intruder guesses the value *Pw* and verifies his guess by verifier *h(Pw,T,R1,R2)*. Hence, Alice believes *Pw* is a secret shared with Bob, however it is known to the intruder. With AVISPA, we couldn't find any attack.

The third example (Protocol VIII) is the same as Protocol VII except that Alice sends v1 (sent to Bob in the third message flow in Protocol VII) in the fifth message flow of the protocol. Surprisingly, both FDR and AVISPA said that there is no attack. However, Protocol VIII is vulnerable to a special kind of off-line attacks (so-called *e*-residue attacks) [17].

4 Discussions

We summarized the results from Section 3 in Table1. As Table1 indicates, FDR was able to detect in many cases, especially those not using the Diffie-Hellman protocol, offline attacks. The reason is that Casper has an imbedded model for guessing off-line attacks so that one can activate this model by declaring that the password is "Guessable"; this command can be included in the protocol description.

Table1. Summary of the results

Tools / Protocols	FDR	AVISPA	Theoretical Result
Protocol I	Off-line	Secure	Off-line
Protocol II	No check	Secure	Off-line
Protocol III	No check	Secure	Off-line
Protocol IV	No check	Secure	Secure
Protocol V	No check	Secure	Off-line
Protocol VI	Off-line	Secure	Off-line
Protocol VII	Off-line	Secure	Off-line
Protocol VIII	Secure	Secure	*e*-residue attack

However, AVISPA was not able to detect any of the off-line attacks known to exist in the protocols. In order to verify whether AVISPA really has ability to detect off-line attacks, we change some password-based protocols listed in the library of protocols verified by the AVISPA project team [13] in such a way to be vulnerable to off-line attack and checked with the AVISPA tool. Nevertheless, AVISPA showed that there is no attack in the modified protocol meaning that the results we got in Section 3 are reliable. Here is the original SPEKE protocol listed in [13].

1. A → B: *exp(kab, Na)*
2. B → A: *exp(kab, Nb)*
3. A → B: *{Ca}_K*
4. B → A: *{Cb,Ca}_K*
5. A → B: *{Cb}_K*

In SPEKE, *K=exp(kab, Na*Nb)* is the Diffie-Hellman key, *kab* is the password shared between Alice and Bob, *Na* and *Nb* are nonces, and *Ca* and *Cb* are challenges, generated respectively by Alice and Bob. We replaced the Diffie-Hellman key *K* in the third and fourth message flows with the password *kab*, resulting in insecurity again off-line attacks. However, AVISPA was not able to detect such an attack.

5 Conclusion

In this paper, we verified typical password–based security protocols via well-known formal verification methods in order to investigate their real abilities, far from what is written in papers and manuals. We think this work can serve for people, who designs security protocols and need to verify their proposed protocol

by formal verification tools, in the sense that it shows the weaknesses of two well-known and strong verification tools. We also verified other authentication protocol where algebraic attacks like *e*-residue attack exist. And we are trying to propose a model to detect such an attack.

References:

[1] J.C. Mitchell, M. Mitchell, and U. Stern, Automated analysis of cryptographic protocols using Murphi, IEEE Symposium on Security and Privacy, 1997

[2] W. Marrero, E. Clarke, and S. Jha. A model checker for authentication protocols, Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997.

[3] C.A.R. Hoare, Communication sequential processes, 1985.

[4] G. Lowe, Breaking and fixing the Needham-Schroeder public-key protocol using FDR, Proceedings of Tools and Algorithms for the Construction and Analysis of Systems, volume 1055 of LNCS, pages 147-166. Springer-Verlag, 1996.

[5] Formal Systems (Europe) Ltd, FDR2 user manual, Aug. 1999.

[6] G.Lowe, Casper: A compiler for the analysis of security protocols, 10th IEEE Computer Security Foundation Workshop, 1997

[7] ISO (1989), ESTELLE: A formal description technique based on an extended state transition model, International Standard ISO 9074.

[8] C. Meadows, The NRL protocol analyzer: an overview, Journal of Logic Programming, vol. 26, no. 2, pp. 113-131, 1996.

[9] C. Meadows, Applying formal methods to the analysis of a key management protocol, Journal of Computer Security, vol. 1, no. 1, 1992.

[10]S. Stubblebine and C. Meadows, Formal characterization and automated analysis of known- pair and chosen-text attacks, IEEE Journal on Selected Areas in Communications, vol. 18, no. 4, pp. 571-581, 2000.

[11]G. Lowe and A.W Roscoe, Using CSP to detect errors in the TMN protocol, IEEE transactions on Software Engineering, 1997.

[12] A. Armando et al., The AVISPA tool for the automated validation of Internet security protocols and applications, 2005.

[13] AVISPA-project team, Deliverable D6.2: specification of the problems in the high-level specification language, 2005.

[14] P. Ryan and S. Schneider, Modeling and analysis of security protocols, 2001.

[15] Y. Chevalier, et al., High level protocol specification language for industrial security-sensitive protocols, Proc. SAPS04, Austrian Computer Society, 2004.

[16] IETF (Internet Engineering Task Force), Challenge handshake authentication protocol, IETF RFC 1994.

[17] S.M. Bellovin and M. Merritt, Encrypted key exchange: password-based protocols secure against dictionary attacks, Proc. of IEEE Symposium on Security and Privacy, pp. 72-84, 1992.