Formal Verification of PAP and EAP-MD5 Protocols in Wireless Networks : FDR Model Checking

Il-Gon Kim, Jin-Young Choi
Dept. of Computer Science and Engineering, Korea University, Seoul, 136-701 KOREA
{igkim, choi}@formal.korea.ac.kr

Abstract

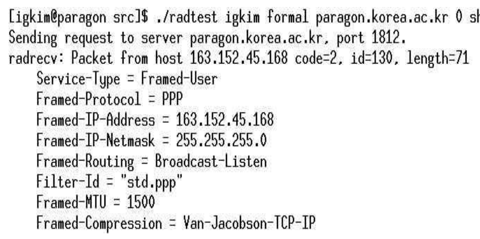
IEEE 802.1x and authentication server based security protocols are mainly used for enhancing security of wireless networks. In this paper, we specify PAP and EAP-MD5 based security protocols formally with Casper and CSP, and then verify their security properties such as secrecy and authentication using FDR. We also show that they are vulnerable to the man-in-the-middle attack. Finally we discuss their security weakness and potential countermeasures related to PAP and EAP-MD5 protocols.

1 Introduction

With the rapid growth of wireless networks, a wide range of networks and applications have come into existence, so that security solutions are needed to meet the requirements of a wide variety of customers based on IEEE 802.11 mechanism. IEEE 802.1x provides some various security protocols based on PPP, EAP[1] and AAA[5] servers. Analyzing a security protocol's flaw is difficult and error-prone, but it is very important that this be done, from the viewpoint of security and privacy. Many security protocols have been discussed in the academic literature, with various goals, such as the establishment of a cryptographic key[11], or achieving authentication[2],[9]. Unfortunately, a large proportion of the protocols which are proposed do not succeed in their stated goals[2]. There have been several approaches to the specification and analysis of authentication protocols. Several methods have been tried, each with their own strengths and weaknesses. Among them, model checking has been proven to be a very successful approach analyzing security protocols. The basic approach is to make a model of the protocol, together with a model of the intruder. This method is very efficient for exploring the state space of the abstraction model and finding counterexample states in which the security properties are violated. However, model checking is limited to a finite system containing a small state space, as otherwise it will be confronted with the state explosion problem. ESTELLE, Murphi, NRL Protocol Analyzer and FDR are examples of model checking tools[2]. In particular, model checking using FDR has been widely used in formal methods and frequently cited in the security literature, after finding man-in-the-middle attack of Needham-Schroeder Public Key Protocol[3],[7]. In this paper, we specify the PAP and EAP-MD5 based security protocols[12] which are frequently used in 802.11, using Casper[8] and CSP[3],[10]. Then we verify whether or not they satisfy such security properties as secrecy and authentication, using FDR model checking tool. After running FDR tool, we reconfirm the existence of some known security flaws in these protocols. This paper is constructed as follows. Section 2 describes IEEE 802.1x. In section 3, we provide an overview description of Casper, CSP and FDR respectively. Section 4 deals with access server authentication. In section 5, we show PAP and EAP based security protocol models, which are written in Casper script. In section 6, we describe the verification results obtained with these security protocol models. Finally, this paper is concluded in section 7.

2 Access Server Authentication

In this paper, we assume that RADIUS is used as the access authentication server. RADIUS was developed by IETF AAA Working Group, to support AAA services, which allow for the authentication, authorization and accounting of remote users in wireless networks. There are some open-source software versions(e.g. FreeRADIUS[4]), as well as commercial(e.g. Merit Networks, Livingston Enterprize and Microsoft). RADIUS uses the UDP protocol, which means that it allows one session to be opened and remain open throughout the entire transaction. It communicates on port 1812, not 1645. The message formats used for sending and receiving between client and server have several message types such as "Access-Request", "Access-Accept", "Access-Reject", "Access-Challenge", etc. RADIUS supports a variety of different protocols for transmitting credential user data to



```
[igkim@paragon src]$ ./radtest igkim formal paragon.korea.ac.kr 0 sharedkey
Sending request to server paragon.korea.ac.kr, port 1812.
radrecv: Packet from host 163.152.45.168 code=2, id=130, length=71
    Service-Type = Framed-User
    Framed-Protocol = PPP
    Framed-IP-Address = 163.152.45.168
    Framed-IP-Netmask = 255.255.255.0
    Framed-Routing = Broadcast-Listen
    Filter-Id = "std.ppp"
    Framed-MTU = 1500
    Framed-Compression = Van-Jacobson-TCP-IP
```

Figure 1. Access-Request packet

and from the authentication server. The two most common protocols are PAP(Password Authentication Protocol)[9] and CHAP(Challenge Handshake Protocol[9]). In the PAP authentication method, the user sends an Access-Request packet containing his or her user name and password, together in one packet, to the NAS(Network Access Server)[9] but in CHAP, the user sends an Access-Request packet including user password, but without the user name. This is the main difference between PAP and CHAP. The authentication method used with RADIUS is peculiar to the vendor's specific product so that each implementation is slightly different. Therefore, in this paper, we analyze the vulnerability of RADIUS based on its RFC document[9] and the FreeRADIUS software package which is very widely used. Fig. 1 shows an example of an "Access-Request" packet sent from an authenticator(client) to a FreeRADIUS(server).

3 Casper, CSP and FDR

3.1 Casper(A compiler for the Analysis of Security Protocols)

Over the last few years, a method for analyzing security protocol that first models communication security protocol using CSP, then verifies its secrecy, authentication and other properties using FDR[3],[7],[8],[10]. In this method, the main difficulty is specifying the security protocol's behavior using CSP. Creating the description of the security model with CSP is a very error-prone and difficult task. To simplify the expression of the security protocol, and render this process more error free, Casper was developed by Gavin Lowe[8]. This tool enables a non-expert who is unfamiliar with CSP to express the security protocol's behavior more easily, without being familiar with the notation used by CSP notation, using various key types, messages, security properties and intruder knowledge descriptions contained in Casper. In brief, Casper is a compiler that translates a more simple and concise description of a security communication model into CSP code. The security process is described by means of 8 section headers, including "♯Free variables", "♯Processes", "♯Protocol", "♯Specification", "♯Variables", "♯Functions", "♯System" and "♯Intruder Information".

3.2 CSP(Communicating Sequential Processes)

CSP[6] is a process specification language specially designed to describe communication processes, and it can describe both a pure parallelism and interleaving semantics. In CSP, the former is expressed as '$\|$' and the latter as "$|||$". The combination of a client, server and intruder are regarded as a process. The use of two different concurrency concepts is well suited to the description and analysis of network protocols. For example, security communication systems operated in distributed networks can be modeled briefly as follows.

```
SYSTEM = (CLIENT1 ||| CLIENT2 ||| SERVER)
          || INTRUDER
```

3.3 FDR(Failure Divergence Refinement)

FDR is a model checking tool for state machine, with foundations in the theory of concurrency based on CSP. This tool checks whether a security model described with CSP$_M$(Machine Readable CSP) satisfies certain security properties such as secrecy and authentication. If the security model doesn't satisfy these properties, FDR shows counterexample event traces and helps to analyze which attack scenario would be most likely to happen. For the equivalence checking of the specification model(safety property model) and implementation model(security system model paralleled with intruder model), FDR supports three refinement checking methods. First, traces refinement can show safety property. Second, failures refinement can represent system's deadlock property. Finally, failure and divergence can detect the livelock property. For further information about Casper and CSP/FDR, the reader is referred to [3],[6],[7],[8],[10].

4 The Specification of the PAP and EAP-MD5 Protocol

In this paper, we describe the PAP and EAP-MD5 based security protocols using Casper and CSP. To do this, we use the following terminology: "USER" means a client host that wishes to obtain a service from the server. "NAS" stands for Network Access Server and it is an intermediate access point between the "USER" and "RADIUS". "RADIUS" is an AAA server used to authenticate, authorize and account for specific resources provided to "USER".

4.1 PAP Specification

In this subsection, we illustrate the PAP specification. Example 1 shows the user authentication procedure based on PAP shown in the RFC document.

Example1. User Telnet to Specified Host based on PAP

1. USER sends his or her user name("igkim") and password("formal") to the NAS

   ("163.152.40.23") using port 3.

2. NAS sends an Access-Request packet to RADIUS. The Access-Request packet is constructed as follows; In this example, the shared secret is defined as "shared secret".

   Code = Access Request

   ID = 0

   Length = 56

   Request Authenticator = random number

   User-Name = igkim

   User-Password = XOR(formal, MD5(shared secret + request authenticator))

   NAS-IP-Address = "163.152.1.16"

   NAS-Port = 3(6 Byte)

3. RADIUS sends the Access-Accept packet to NAS. The Access-Accept packet contains the following message.

   Code = Access Accept

   ID = 0

   Length = 38

   Response Authenticator = (Code + ID + Length + request authenticator + shared secret)(16Byte)

   Service Type = Login

   Login Service = Telnet

   Login IP Host = 163.152.1.3

4. NAS sends an access success message with telnet host as the destination address IP(163.152.1.3) to USER.

According to the above authentication procedure description, we can write the PAP model in Casper script. The ♯Free variables, ♯Protocol and ♯Specification of the PAP model using Casper script are shown in example 2. The other Casper header descriptions are not mentioned in this paper, because they are fairly trivial. We assume that the encryption algorithms such as MD5 are highly resistant to brute force attacks, as well as to any other ciphertext analysis attacks, and that the intruder already knows each host's public keys and identities.

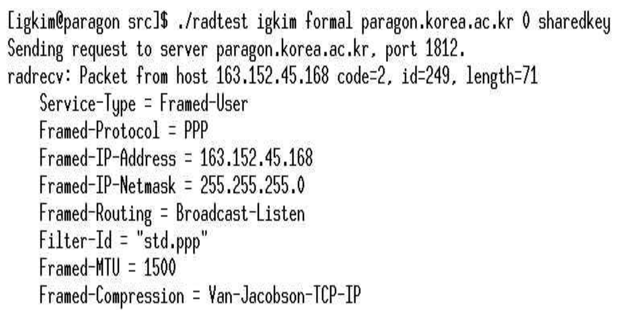
Example2. PAP Casper script

```
#Free variables
A, B : Agent
S: Server
rauth : Nonce
passwd, shared: SessionKey
f : HashFunction InverseKeys =
(f,f),(passwd,passwd),(shared,shared)

#Protocol description
0.    -> A : B
1. A -> S : A, passwd
2. S -> B : S, rauth, passwd (+)
            f(rauth, shared)
3. B -> S : B, f(rauth,shared)

#Intruder Information
Intruder = Mallory
IntruderKnowledge =
{Alice, Bob, Mallory, Sam}
```

Example 2 shows the PAP authentication protocol written in Casper script. In the ♯Free variables header, the first five lines simply declare the types of free variables. A, B and S represent the identities of USER, NAS and RADIUS respectively. rauth is the request authenticator, which is a random number. passwd and shared represent the password("formal" in the above example) and shared secret("shared secret" in the above example). f is a MD5 hash function. The sixth line is a definition of which keys are inverses of one another. f, passwd and shared are symmetric key types so that encryption key is the same as the decryption key. In ♯Protocol description header, the first line, 0 message means A(USER) must communicate with B(RADIUS Server). Messages 1, 2 and 3 represent the message sequence shown in example 2. The (+) notation represents the eXclusiveOR operator used for the streamcipher computation. ♯Intruder Information header defines the intruder's initial knowledge, Generally, we assume that the intruder knows each host's identity, public key, and his own generated random number. According to the scope of the intruder's knowledge, the verification result can vary. In example 2, the intruder's knowledge is limited to the identities of A(Alice), B(Bob), S(Sam) and Intruder(Mallory). In this paper, we show only part of the Casper script, due to space limitations.

4.2 EAP-MD5 Specification

The EAP-MD5 is used to implement the EAP analog of the CHAP protocol, specified in RFC 1994. It is expected to be widely used in 802.1x wired Ethernet switch deployments. This algorithm can also be used for wireless application with less stringent wireless LAN security requirements. For example, the use of EAP-MD5 authentication may be sufficient for public space applications, in which encryption is provided at the application level. However, it has the disadvantage that no encryption keys are generated in the case of wireless LAN applications. Fig. 2 shows the EAP-MD5 message flow.



```
[igkim@paragon src]$ ./radtest igkim formal paragon.korea.ac.kr 0 sharedkey
Sending request to server paragon.korea.ac.kr, port 1812.
radrecv: Packet from host 163.152.45.168 code=2, id=249, length=71
    Service-Type = Framed-User
    Framed-Protocol = PPP
    Framed-IP-Address = 163.152.45.168
    Framed-IP-Netmask = 255.255.255.0
    Framed-Routing = Broadcast-Listen
    Filter-Id = "std.ppp"
    Framed-MTU = 1500
    Framed-Compression = Van-Jacobson-TCP-IP
```

Figure 2. EAP-MD5 Message Flow

Based on Fig. 2, we describe EAP-MD5 based protocol in Casper script. Example 3 shows EAP-MD5 Casper script.

Example3. EAP-MD5 Casper script

```
#Free variables
A, B : Agent
S: Server
challenge : Nonce
response : SessionKey
f : HashFunction
InverseKeys =
(f,f),(response,response)

#Protocol description
0.    -> A : B
1. A -> S : A
2. S -> B : S, A
3. B -> S : f(challenge) % enc
4. S -> A : enc % f(challenge)
5. A -> S : f(response)% end
6. S -> B : end % f(response)

#Intruder Information
Intruder = Mallory
IntruderKnowledge =
{Alice, Bob, Mallory, Sam}
```

In ♯Free variables header, the first five lines are very similar to those of example 2. A, B and S represent the identities of USER, NAS and RADIUS respectively. The challenge is randomly generated by RADIUS. Generally, a random number is generated and distributed by a server for the sake of better key management. The response is calculated by USER, after entering the challenge value into the response generation function. And we write m%v, where m is a message and v is a variable, to denote that the recipient of the message should not attempt to decrypt the message m, but should instead store it in the variable v. Similarly, we write v%m to indicate that the sender should send the message stored in the variable v, but the recipient should expect a message of the form given by m.

5 The Verification of the PAP and EAP-MD5 Protocols

5.1 The Verification of the PAP Protocol

To analyze the vulnerabilities of the PAP protocol using FDR, we declare the secrecy and authentication properties that it must satisfy to be as follows.

```
Secret(A, passwd, [S])
Secret(S, rauth, [B])
Secret(B, rauth, [S])
Secret(S, shared, [B])
Secret(B, shared, [S])
```

The lines starting with Secret represent that *secrecy property* should be certain secret information between only specific hosts. The *secrecy property* states that intruders cannot obtain this secret information during a run of the protocol whenever its secrecy is claimed. In CSP, a *secrecy property* can be formalized as *signal.Claim_Secret.a.b.s* event. This may be understood to mean; 'The secret value s used in the run between *a* and *b*, which was initiated by *a* should be secret for the entire protocol run'. If the secrecy property is satisfied in the model, then the intruder should not be able to obtain access to the secret value, *s*. That might be expressed naively as below. The notation of *tr* refers to a set of trace events.

$$signal.Claim\_Secret.a.b.s \ \textbf{in} \ tr \Rightarrow \neg(leak.s \ \textbf{in} \ tr)$$

The first statement is interpreted as 'USER thinks that password is a secret that should be known only to USER and NAS'. The second statement can be paraphrased as 'NAS thinks that request authenticator is a secret that should be known only to NAS and RADIUS'. The fourth statement specifies that the shared key is a secret that should be known to only NAS and RADIUS. If A, S or B is an intruder in this protocol, secret information will be leaked to him, in which case a man-in-the-middle attack is considered to have occurred. The *authentication property* can be expressed in the form of the following two statements.

```
Agreement(S, B, [rauth, shared, passwd])
Agreement(B, S, [rauth, shared, passwd])
```

The lines starting with Agreement define that *authentication property*. The *authentication property* represents the establishment guarantees when it has completed, concerning the party it has apparently been running with. In CSP, an *authentication property* can be observed from two viewpoints; one is the authentication of the initiator by the responder,while the other is the authentication of the responder by the initiator. Events of the form *Running.a.b* in a's run of the protocol are introduced to mark the point that should have been reached by the time b performs the *Commit.b.a* event. The authentication of initiator a by responder b will require that *signal.Running.INITIATOR.a.b* must precede *signal.Commit.RESPONDER.b.a*. That can be expressed more simply follows.

a ϵ *Honest* $\Rightarrow$ *signal.Running.INITIATOR.a.b* **precedes** *signal.Commit.RESPONDER.b.a*

And the authentication of responder b by initiator a will require that *signal.Running.RESPONDER.b.a* must precede *signal.Commit.INITIATOR.a.b*. This can be represented more simply as follows.

b ϵ *Honest* $\Rightarrow$ *signal.Running.RESPONDER.b.a* **precedes** *signal.Commit.INITIATOR.a.b*

These are often of the form that if the run has completed then the other party has also been involved in the protocol run. The first one means that 'NAS is authenticated to RADIUS with request authenticator, secret key and password'. Similarly, the second statement means that 'RADIUS is authenticated to NAS by request authenticator, secret key and password'. To verify the safety specification of RADIUS, we use traces refinement provided in FDR tool. If the trace events set of implementation model Q is a subset of the trace events set of specification model P, we can say that Q is a safe implementation. In other words, if the specification model is a property model, it may be expressed briefly as follows.

```
SYSTEM_0 = (AGENT_Alice ||| (AGENT_Bob
            ||| AGENT_Sam))
SYSTEM = SYSTEM_0[|{|intercept, fake|}|]
         INTRUDER
assert SYSTEM [T= SECRECY
assert SYSTEM [T= AUTHENTICATION
```

Where SYSTEM represents the implementation model which consists of communication processes paralleled with an intruder in a distributed network. This means that the intruder can always eavesdrop on any message to and from any process. If the traces event sets of SYSTEM process are a subset of those of the SECRECY and AUTHENTICATION processes, we can conclude that the intruder process doesn't interrupt the behavior of the security process. With FDR model checker, we can verify whether the RADIUS protocol satisfies the Secret and Agreement requirements in the Casper script. In the result, FDR displays the counterexample that RADIUS protocol does not satisfy the secrecy and authentication properties. Casper has the ability to translate the above low-level CSP algebra notation into high-level language. By analyzing the first debug tree which *signal.Claim_Secret. Sam.Rauth.(Bob)* is followed by *leak.Rauth*. This means that the security value *Rauth* is leaked to the intruder. So, we can determine the fact that the communication challenge from USER to NAS is not encrypted, with the result that the user password, request authenticator and password (+) MD5(request authenticator + secret key) can be obtained by an intruder. And after debugging the second counterexample, we can determine that RADIUS is not authenticated to NAS. This means that the PAP protocol may be susceptible to a sniff and spoof attack by an intruder. To solve this shortcoming regarding the susceptibility of PAP to a man-in-the-middle attack, a secure communication channel must be established between USER and NAS. In addition to this, we verify a corrected PAP protocol that establishes a secure channel between A and S, and we reconfirm that, in this case, the PAP based security protocol satisfies the authentication property.

```
#Protocol description
0.   -> A : B
1. A -> S : {A, passwd}{PK(S)}
2. S -> B : S, rauth, passwd (+)
            f(rauth,shared)
3. B -> S : B, f(rauth,shared)
```

In this protocol description, we just add the public key of host S , PK(S) to message 1. This guarantees the secure delivery of {A, passwd} between A and S. So, the above protocol provides protection against a man-in-the-middle attack.

5.2 The Verification of EAP-MD5

We verify whether or not the EAP-MD5 based protocol satisfies the secrecy and authentication properties specified below.

```
Secret(A, response, [B])
Secret(B, response, [A])
Agreement(A, B, [challenge, response])
Agreement(B, A, [challenge, response])
```

After running the FDR tool, we obtain the following counterexample trace events. To make the output from FDR easier to understand, Casper provides the ability to convert the low-level traces into high-level traces. However, we can also analyze and find the violations of the authentication property at the CSP code level. In the first CSP counterexample event traces, you can see that *signal.Running1.INITIATOR_role. Alice.Mallory.Challenge. Response* precedes *signal.Commit1.RESPONDER_role. Bob.Alice.Challenge.Response*. However, there is a mismatched sequence of host identities. In other words, *Alice.Mallory* does not correspond with *Bob.Alice*. This means that the intruder *Mallory* has violated the authentication property.

```
fake.Sam.Bob.
(Msg2,Sq.<Sam,Alice>,<>)
env.Alice.(Env0,Mallory,<>)
intercept.Bob.Sam.
(Msg3,Hash.(f,<Challenge>),
<Alice,Response>)
intercept.Alice.Sam.
(Msg1,Alice,<>)
fake.Sam.Alice.
(Msg4,Hash.(f,<Challenge>),<>)
signal.Running1.INITIATOR_role.
Alice.Mallory.Challenge.Response
signal.Commit1.RESPONDER_role.
Bob.Alice.Challenge.Response
```

After reviewing the high-level traces, we can see that the secrecy property is satisfied, but that the authentication property is not allowed in EAP-MD5. A general attack scenario which can be found in EAP-MD5 is summarized below. The notation S(I) on the right hand side of the arrow(→) represents the intruder intercepting a message intended for S. And the notation S(I) on the left-hand side of the arrow represents the intruder faking a message, making it appear to come from S. If an intruder intercepts a message containing host A's username and succeeds in disguising himself as host S, he can break the security of the EAP-MD5 protocol, as shown in example 4.

Example 4. Attack scenario in EAP-MD5

Message 1. A $\rightarrow$ S(I) : A

Message 2. S(I)$\rightarrow$ B : S, B

Message 3. B $\rightarrow$ S(I) : f(challenge)

Message 4. S(I) $\rightarrow$ A : f(challenge)

Message 5. A $\rightarrow$ S(I) : f(response)

6 Conclusion

In this study, we verify some of the security protocols which fall into the second category, because we believe cryptographic algorithms are strong and unbreakable. In this paper, we focus on the verification of real security protocols which are widely used in wireless networks, not theoretical ones. We analyze the vulnerability of PAP and EAP-MD5 based authentication protocols using Casper and CSP/FDR. In verifying these protocols with FDR tool, we were able to reconfirm some of the known security vulnerabilities which are likely to occur in wireless networks. These vulnerabilities are relatively simple but nevertheless constitute serious dangers, because only a few access point devices currently support enhanced security mechanisms and software level upgrading. The intruder model of CSP is limited to intercepting and faking messages to and from communication processes. In order to identify other security vulnerabilities, such as DoS attack, further studies will be required in which the intruder's inference rule in CSP is extended.

References

[1] L. Blunk and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998.

[2] J. Clark and J. Jacob. A survey of authentication protocol literature: Version 1.0. Available via *http://www.win.tue.nl/ ecss/downloads/clarkjacob.pdf*, 1997.

[3] Formal Systems(Europe) Ltd. FDR2 User Manual, Aug. 1999.

[4] J. Hassell, *O'Reilly RADIUS book*, 2002.

[5] J. Hill, An Analysis of the RADIUS Authentication Protocol, 2001.

[6] C.A.R. Hoare, *Communicating Sequential Processes*, 1985.

[7] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. *Proceedings of TACAS*, number 1055 in LNCS. Springer, 1996.

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

[9] C. Rigney, A. Rubens, W. Simpson and S. Willens, RFC 2138, Remote Authentication Dial In User Service(RADIUS), 1997.

[10] P.Y.A. Ryan and S. A, Schneider, *modelling and analysis of security protocols: the CSP Approach*, Addison-Wesley, 2001.

[11] W. Stallings, *Network Security Essentials(2nd Edition)*, 2002.

[12] D. Stanley, *A chapter in Wireless Local Area Network: The New Wireless Revolution*, 2003.