Casper, FDR을 이용한 SSH 프로토콜의 취약성 분석 및 수정

김일곤*, 최진영

*고려대학교 컴퓨터학과

The Vulnerability Analysis and Modification of SSH Protocol using Casper, FDR

Il-Gon Kim*, Jin-Young Choi

*Dept. of Computer Science & Engineering, Korea University

요 약

보안 프로토콜을 이용하여 사용자 상호 간의 정보의 비밀성, 무결성, 부인방지를 보장하는 일은 반드시 필요한 과제가 되었다. 이에 따라 해당 서비스에 대해 적합한 사용자를 인증하고 인가하기 위해 다양한 보안 프로토콜(EKE, S/KEY, Kerberos등)이 사용되고 있으며, 이와 더불어 보안 프로토콜을 정형적으로 명세하고 검증하기 위한 다양한 방법들이 연구되고 있다. 본 논문에서는 Casper와 FDR을 이용하여 SSH 프로토콜의 취약성을 분석하고, 수정하여 SSH 프로토콜의 안전성을 향상시키고자 하였다.

I. 서론

클라이언트와 서비스 제공자 사이의의 비밀성, 무결성, 부인방지를 제공하기 위해 다양한 보안 프로토콜(EKE[1], S/KEY[2], Kerberos[3]등) 및 보안 장비(방화벽, 지문 인식기등)들이 개발되고 있으며, 보안 시장도 점차 커져나가고 있는 추세이다. 이런 보안 프로토콜들의 대부분은 처음에는 안전하다고 믿었지만 시간이 지남에 따라 취약점이 발견되고 있다. 이에 따라 보안 프로토콜의 안전성을 보장하는 문제도 큰 관심사항으로 떠오르고 있다. 하지만 보안 프로토콜의 문제점을 직관적으로 찾아내기란 쉽지 않다. 그래서 이미 외국에서는 수학적 논리 방법을 토대로 한 정형기법을 이용하여 설계단계에서부터 보안 프로토콜의 안전성을 분석하고자 하는 연구가 진행되어 오고 있다. 예를 들어, 처음에는 안전하다고 여겼던 Needham-Schroeder 인증 프로토콜[4]은 약 17년이 지나서야 자동화된 검증 방법을 사용하여 프로토콜의 취약점을 찾아 낼 수 있었다. 이에 따라 보안 프로토콜을 검증하기 위한 다양한 검증 방법과 도구들이 개발되고 있다. 그 중에서도 Casper와 FDR은 보안 프로토콜을 명세하고 검증하기 위한 전용 도구로서, 다양한 보안 프로토콜을 검증하여 그 효율성을 인정받고 있다. 본 논문에서는 Casper, FDR을 사용하여 SSH 프로토콜의 취약성을 분석하고 수정하여 SSH 프로토콜의 보안성을 향상시키고자 하였다. 본 논문의 구성은 다음과 같다. 본 논문의 1장에서는 SSH 프로토콜에 대해 간략히 설명하고, 2장에서 Casper와 FDR을 이용한 명세 및 검증 방법에 대해 살펴보고, 3장에서는 Casper와 FDR을 이용하여 SSH 프로토콜을 명세하고, 4장에서는 SSH 프로토콜의 취약성을 분석하고, 5장에서는 보다 안전한 통신을 위해 SSH 프로토콜을 수정하여 그 안전성을 검증하였으며, 마지막으로 6장에서는 결론 및 향후 연구방향을 제시하고자 한다.

II. 본론

1. SSH(Secure Shell)

SSH 보안 프로토콜[5]은 RSA[6] 암호 매커니즘을 사용하여 암호화 호스트 인증을 통해 클라이언트와 서버간에 안전한 통신 채널을 제공한다. SSH의 장점은 공개키 기반의 암호화 방식을 사용하여 안전하지 않은 통신 채널을 갖고 있는 사용자들간에 보다 안전한 암호화 통신을 해주기 때문에, 악의적인 공격자가 스니핑 도구[7]를 이용해 사용자의 아이디와 패스워드를 쉽게 가로채지 못하게 한다는 것이다. SSH 프로토콜의 호스트 인증은 크게 5종류로 나눌 수 있다. 첫번째 방식은 RSA-Host Based Authentication, 두번째 방식은 Trusted Host Authentication, 세 번째 방식은 Rhosts and RSA Authenitcaion, 네번째 방식은 RSA Authentication이며, 이 4가지 인증 방식은 패스워드 기반 인증 방식과 달리 클라이언트와 서버간의 공개키와 비밀키, 난수를 이용하여 상호 인증하는 비패스워드 인증 방식이고, 마지막으로 다섯번째 방식은 클라이언트로 서버간에 미리 설정해 놓은 공유키인 패스워드를 통해 인증하는 패스워드 기반 인증 방식을 사용하고 있다.

PK(c) : 클라이언트의 공개키
Ns : 서버에서 생성되는 임의 수
Hash : MD5 해쉬 함수

본 논문에서는 이 4가지 인증 방법 중에서 그림 1에 나타나 있는 RSA Authentication 방법을 선택하여, 그 취약성을 분석하고 수정하였다.

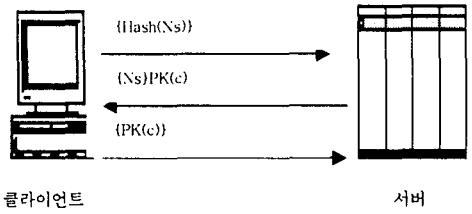

그림 1 RSA Authentication의 동작 절차

2. Casper와 FDR

Casper(A Compiler for the Analysis of Security Protocols)[8]는 CSP[9][10]로 프로토콜을 명세하는데 걸리는 시간을 단축시켜주고 보다 모델링을 쉽게 하도록 개발되어진 컴파일리이다. Casper에서 입력 파일은 프로토콜의 동작과 체킹해야할 시스템을 표현한다. Casper에서 위의 사항들을 8개의 세부항목으로 분류하고 있으며, 각 항목의 헤디 부분은 #로 시작한다.

FDR(Failure Divergence Refinements)[11] 도구는 CSP(Communication Sequential Process)를 입력 언어로 받아, 모델이 특수한 속성을 만족하는지 않는지를 체킹하는 모델 체킹 도구이다. 즉 보안 프로토콜 모델을 프로세스 알제브라 언어인 CSP로 명세한 후, 보안 프로토콜이 반드시 갖추어야 하는 요구사항인 비밀성, 무결성, 인증, 부인방지와 같은 보안 속성을 만족하는지 검사하는 도구라고 할 수 있다. FDR은 다음과 같은 세 가지 모델을 지원해 준다.

1. trace model
2. failure model
3. failures/divergence model

FDR은 Refinement checking 방법을 사용하여 보안 프로토콜을 검증한다. Refinement checking 방법이란 보안 프로토콜의 구현 모델이 명세 모델을 만족하는지를 검사하는 방법을 의미한다. 명세 모델이 S1이고, 구현 모델이 S2 라고 하면, S2⊆S1 인지 검사하게 된다. 즉, 명세 모델은 보안 프로토콜을 의미하며, 구현 모델은 공격자가 개입해서 동작하는 보안 프로토콜 모델)을 의미하게 된다. 그러므로 만일 두 모델이 같다면, 공격자는 보안 프로콜상에서 어떤 악의적인 행위도 취할 수 없다는 것이고, 따라서 보안 프로토콜은 안전하나는 의미가 된다.

3. SSH 프로토콜의 명세

본 논문에서는 Casper를 사용하여 RSA Authentication의 동작을 명세하였다. 아래 그림 2와 3은 각각 RSA Authentication의 Free variables와 Processes 부분을 보여주고 있다.

```
#Free variables
 a, b : Agent
 PK : Agent -> PublicKey
 SK : Agent -> SecretKey
 F: HashFunction
 nb: Nonce
 InverseKeys = (PK,SK),(F,F)
```

그림 2. RSA Authentication의 Free variables

```
#Protocol description
 0.    -> a : b
 1.  a -> b : PK(a)
 2.  b -> a : {nb}{PK(a)}
 3.  a -> b : F(nb)
```

그림 3. RSA Authentication의 Free variables

4. SSH 프로토콜의 취약성 분석

Casper의 *.spl 형식의 입력 파일을 컴파일하여 생성된 *.csp 형식의 코드를 FDR 모델체커에 입력한 후, 디버거를 통해 다음과 같은 행위를 찾아낼 수 있었다.

env.Alice(Env0,Mallory,<>)
intercept.Alice.Mallory(Msg1,PK_.Alice,<>)
fake.Alice.Bob(Msg1,PK_Alice,<>)
intercept.Bob.Alice(Msg2,Encrypt.(PK_.Alice,<Nb>),<>)
fake.Mallory.Alice(Msg2,Encrypt.(PK_Alice,<Nb>),<>)
signal.Running1.INITIATOR_role.Alice.Mallory.Nb
signal.Commit1.RESPONDER_role.Bob.Alice.Nb

위의 내용에 대한 공격 시나리오는 그림 4에 나타나 있다.

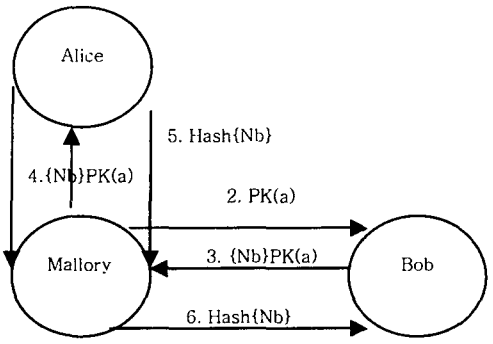

그림 4 RSA Authentication의 공격 시나리오

위의 분석 내용을 토대로 보면 Alice는 공격자 Mallory가 Bob인줄 알고 자신의 공개키 PK(a)를 보내게 되며 Mallory는 자신이 가로챈 공개키 PK(a)를 Alice로 가장하여 Bob에 전달하고 Bob은 자신이 생성한 임의 수 Nb를 PK(a)로 암호화하여 전달하게 되고 Mallory는 이 암호화된 메시지를 Alice에게 전달하게 되면 Alice는 인증을 위해 Bob의 임의 수를 해쉬화한 메시지 Hash{Nb}를 Bob인줄 알고 전달하지만 이 역시 Mallory가 가로채게 됨으로써 Mallory는 Bob에 인증을 받을 수 있게 된다. 이런 취약점은 Needham shroeder 프로토콜에서 발견된 man-in-the-middle-attack 방법[7]과 유사하다. 공격자의 스니핑 및 스푸핑 [7] 기술을 방지하기 위한 보안책이 요구된다.

5. 수정된 SSH 프로토콜의 안전성 검증

RSA Authentication에서 발생하는 보안 취약점은 기존에 Needham Shroeder에서 발생한 취약점과 매우 유사하다, 따라서 RSA Authentication의 취약성을 해결하기 위해서는 크게 2 가지 문제점을 개선해야한다.

1. 공격자가 PK(a)를 가로챔
2. 공격자가 적합한 클라이언트로 위장함

첫 번째 문제점을 해결하기 위해서 PK(a)를 서버의 PK(b)를 이용하여 암호화해서 전달하도록 한다. 클라이언트와 서버 초기 통신 설정을 위해 상호간의 공개키 PK(a)와 PK(b)를 네트워크상으로 전달하는 방법을 피한다. 피치 못할 경우에는 한번 더 암호화하여 전달한다.

두 번째 문제점을 해결하기 위해서는 패킷 헤더에 해당 호스트의 식별자를 추가하여, IP 스푸핑을 방지하도록 한다. 물론 IP 스푸핑을 방지하기 위해서는 DNS 보안이 요구된다.

수정된 RSA Authentication의 Processes 부분 그림 6에 나타나 있으며, 그림 7은 Intruder Information을 나타내고 있다.

```
#Protocol description
0.    -> a  : b
1.  a -> b : {na, PK(a)}{PK(b)}
2.  b -> a : {na, nb, b}{PK(a)}
3.  a -> b : {F(nb)}{PK(b)}
```

그림 5. RSA Authentication의 Free variables

```
Intruder = Mallory
IntruderKnowledge = {Alice, Bob,
Mallory, Nm, PK, SK(Mallory)
```

그림 6. RSA Authentication의 Free variables

수정된 RSA Authentication 방식이 만족해야 하는 보안 속성은 그림 7과 같이 작성하였다.

```
#Specification
Secret(a, na, [b])
Secret(b, nb, [a])
Agreement(a, b, [nb])
Agreement(b, a, [nb])
```

그림 7. RSA Authentication의 Free variables

Secret는 비밀성을 의미하고, Agreement는 인증을 의미한다. 즉, Secret(a, na, [b])는 A 호스트는 b 호스트만이 na를 알고 있다고 믿는 것을 뜻하고 Secret(b, nb, [a])는 b 호스트는 a 호스트만이 nb를 알고 있다고 믿는 것을 나타낸다. 그리고 Agreement(a, b, [nb[)는 a 호스트는 b호스트에 nb를 이용해서 인증 받는다는 것을 의미하게 하고Agreement(b, a, [nb])는 b호스트는 a호스트에 nb를 이용해 인증 받는 다는 것을 나타낸다. 수정된 RSA Authentication에 대한 CSP 코드를 FDR 검증 도구에 입력한 결과 그 안전성을 확인할 수 있었다. 그림 8은 FDR 검증 도구를 실행하였을 때의 화면을 보여주고 있다.

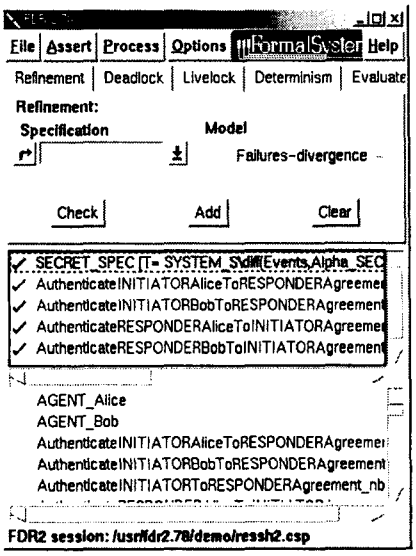

그림 8 FDR 도구를 이용한 검증 결과

Ⅲ결론

보안 프로토콜의 안전성을 검증하는 일은 그리 간단하지 않다. 보안 프로토콜의 특수성과 더불어 정형 기법의 수학적 복잡성 때문이기도 하다. 본 논문에서는 Casper와 FDR을 이용하여, 분산 시스템 환경에서 많이 사용하는 SSH 프로토콜의 취약성을 분석하였을 분만 아니라, 보다 안전한 새로운 수정 프로토콜을 제시하고 검증함으로써 SSH 프로토콜의 안전성을 향상시킬 수 있었다. 향후 연구 방향으로는 기존의 정형적 설계에는 공격자의 공격 유형이 스니핑과 스푸핑 두 가지로 한정되어 있기 때문에, 보다 다양한 취약점을 찾아내기 힘들다 따라서 다양한 공격 행위를 정형적으로 모델링하는 방법을 연구하고자 한다.

참고문헌

[1] Steven M.Bellovin & Michael Merritt Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks, AT&T Bell Laboratories, 1992

[2] N.Haller, A One-Time Password System, rfc1938,,1996

[3] William Stallings, NETWORK SECURITY ESSENTIALS, Application and Standards, 1999

[4] Gavin Lowe, Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR, 1996

[5] Scott Mann - Ellen L. Mitchell, Linux System Security. The Administrator's Guide to Open Source Security Tools, 2000

[6] William Stallings, Cryptography And Network Security, Principles and Practice, 1999

[7] Brian Hatch, James Lee, George Kurtz, HACKING LINUX EXPOSED, Network Security Secrets & Solutions, 2001

[8] Gavin Lowe, Casper User Manual and Tutorial, version 1.3, 1999

[9] C.A.R. Hoare, Communicating Sequential Processes, PrenticeHall, 1985

[10] Peter Ryan & Steve Schneider, modelling and analysis of security protocols], 2001

[11] Formal Systems(Europe) Ltd. Failure Divergence Refinement-FDR2 User Manual, Aug. 1999