2000년 한국정보처리학회 추계 학술논문발표집 제7권 제2호

자바 기반의 일회성 변형 암호 기법을 이용한 사용자 인증 구현

이승헌*, 이병욱*
*경원대학교 전자계산학과
e-mail:volcanom@web.kyungwon.ac.kr

Implementation of User Authentication using Java-based One-Time Transformation Password Mechanism

Seung-Heon Lee*, Byung-Wook Lee*
*Dept. of Computer Science, Kyungwon University

요 약

사용자 인증과정에서 나오는 패킷 정보 유출은 보안상 큰 문제가 되고있다. 따라서, 사용자 인증과정에 대한 보안 강화 대책이 요구된다. 본 논문에서는 변형 암호 인증 방식을 응용하여 사용자 인증 방식을 강화한다. sniffer와 같은 패킷 단위의 공격이 있을 경우 암호키를 모르면 해독이 불가능하다. 로그인을 위한 페이지에서 파악한 IP와 입력받을 때 파악한 IP가 같아야만 인증을 허용함으로써 로그인 화면이 아닌 다른 경로에 의한 접근을 원천적으로 막는다. 암호화에 사용되는 키를 항상 새로 생성함으로써 키 유출에 관한 문제를 해결한다. 마지막으로 안전성이 해쉬 알고리즘의 의존도를 낮추어 안전성을 강화한다.

1. 서론

현재 인터넷을 통한 정보 유출은 많은 문제를 갖는다. 특히 개인 정보에 대한 유출은 개인의 부주의에 의해 발생하지만 사용자의 아이디와 암호를 가로채는 악성 프로그램에 의해 유출되기도 한다. 이런 프로그램들은 대부분 패킷 단위로 정보를 알아낸다. 인증을 받기 위해 보내는 패킷 정보는 암호화되지 않기 때문에 누구든지 쉽게 받아 볼 수 있는 문제점을 가지고 있다.

패킷을 이용해 정보를 빼내는 프로그램으로 많이 알려진 스니퍼(sniffer)는 네트워크의 한 호스트에서 실행되어 호스트 주위를 지나다니는 패킷들을 획득하는 프로그램이다. 이더넷 LAN 환경에서 네트워크 트래픽을 분석하여 사용자 이름, 데이터의 내용을 수집하고 그 데이터의 내용을 다시 조회할 수 있도록 하는 프로그램이다. 즉 네트워크에서 TCP/IP를 통과하는 패킷 중 찾고자 하는 키워드를 가진 패킷을 추출하여 관리자용 클라이언트 PC에 저장함으로써 유입하고 유출되는 패킷 트래픽을 모니터링 하는 것이다. 이런 문제는 해커가 어디에서 스니퍼를 설치하여 패킷을 훔쳐보고 있는지 알지 못하고 설사 같은 지역 네트워크 망에 스니퍼를 설치하여도 그 사실을 보안 관리자가 알기 힘들다.

본 논문에서 IP와 키를 이용하여 변형 암호 인증 방식을 응용한 사용자 인증을 구현한다. 패킷에 의한 유출을 막기 위해 아이디와 암호를 클라이언트 내에서 암호화하여 전송함으로써 최초의 아이디와 암호 유출을 방지한다. 암호키의 유출에 관한 문제를 해결하기 위하여 접근할 때마다 새로운 키를 생성하여 사용한다. 또한 암호화한 내용이라도 같은 내용이 계속 전송되면 결국 암호화된 내용은 해독이 가능하기 때문에 인증 과정에서 암호화를 몇 번 할 것인지 결정할 난수를 서버에서 생성하고 생성된 난수를 이용하여 클라이언트에서 그 수만큼 암호화를 함으로써 항상 다른 값을 전송한다.

마지막으로 변형 암호 방식이 해쉬함수에 의존하는 것을 극복하기 위한 방법으로 IP를 이용한다. IP는 로그인 화면 요청과 로그인 시 탐지되는 IP 그리고, 암호화 작업에 필요한 킷값을 요청할 때 이를 비교하여 같은 IP로 접근시 계속적인 작업을 허용한다.

2. 관련연구

2장에서는 암호화에 관련한 사용자 인증의 기존 방식과 암호화 알고리즘에 관해 살펴본다.

2.1 암호 인증 방식

현재 대부분의 사이트들이 이 방식을 이용하고 있지만 악의적인 공격에 매우 취약한 보안 문제를 안고 있다. 가장 많이 나타나는 문제점으로는 암호 전송 과정에서 스니퍼와 같은 프로그램의 공격에 정보 유출이 생긴다.

스니퍼는 호스트에서 작동하며 호스트로 들어오는 통신 패킷을 이용하여 아이디와 암호를 가로챈다. 암호는 기본적인 암호화가 이루어진다 해도 항상 고정된 결과를 보내기 때문에 해석이 가능하다.

이런 문제점을 해결하기 위해 해쉬 함수를 이용한 변형 암호 인증 방식이 도입되었다. 변형 암호 방식은 해쉬함수로 암호를 암호화하여 서버로 보내고 서버에서 복호화작업을 한 후 저장된 값과 비교한 후 같을 경우 접근을 허용한다. 그러나 아이디와 난수의 값이 노출되면 암호가 유출될 가능성이 크다.

2.2 일회용 암호 인증 방식

이 방식은 인증과정에서 항상 새로운 암호를 사용하여 접근하는 방식으로 불법적으로 암호를 알아내도 사용이 불가능하다. 암호를 생성할 때 seed 값을 생성하고 seed 값에 따라 난수 n을 결정한 후 클라이언트에서는 난수의 값으로 해쉬하는 횟수를 결정한다. 클라이언트는 암호를 난수 값만큼 암호화를 한 후 서버로 아이디와 암호화된 암호를 보낸다. 서버는 받은 아이디와 암호를 미리 난수만큼 해쉬 함수를 이용하여 생성한 임시 암호와 비교하여 같은 경우 인증을 해준다.

이 방식은 사용하기는 쉽지만 seed 값에 의해 난수가 제한적이고 사용자에 의해 수행된 단 방향 함수의 수가 하나씩 줄어들기 때문에, 어느 시점에 다다르면 사용자는 시스템을 재 초기화해야만 한다. 또한 해쉬 알고리즘에 너무 의존하는 문제점을 가지고 있다.

2.3 DES

DES는 대칭형 블록 암호 시스템의 대표적인 방식으로 평문 64bit를 암호문 64bit 블럭으로 바꾸는 암호화를 한다. 이때 사용되는 키도 64bit인데 여기서 8bit는 패리티 비트이므로 실제로 키의 길이는 56bit이다.

DES의 암호화는 다음 두 가지 기본변환을 통해 이루어진다. 첫 번째는 대치(substitution) 작업을 하고 두 번째는 상위비트와 하위비트를 치환(permutation)하는 작업을 한다. 64bit의 평문을 64bit의 키를 이용하여 64bit의 암호문으로 만들어낸다.

먼저 평문을 초기치환한후 기본변환을 거쳐 암호화를 수행한다. 마지막에는 처음에 했던 초기치환의 역치환을 해서 원래대로 되돌린다. 이런 방식으로 DES의 암호화가 이루어진다. 복호화는 역초기치환을 제일 먼저 한 다음에 위의 과정을 역으로 수행하면 복호화가 이루어진다.

3. 자바를 이용한 암호화된 사용자 인증

본 논문에서는 Java를 이용하여 아이디와 암호를 암호화하고 복호화된 데이터베이스의 결과 값을 비교하여 사용자를 확인하는 절차를 갖는다. 자바 암호화 구조(JCA)는 쉽게 메시지 축약을 사용할 수 있게 해준다. 사용자 인증은 두 가지 요소에 의해 보안을 유지한다. 첫 번째는 암호화와 복호화에 필요한 키를 이용한 방법이고 두 번째는 IP를 이용한 방법이다.

3.1 사용자 등록

사용자 등록 과정은 클라이언트의 입력작업과 서버에서의 암호화 과정 및 데이터베이스 입력으로 이루어진다. 클라이언트에서 사용자 정보를 입력하면 서버에서 키 파일인 SecretKey 파일을 다운받고 사용자 정보를 1회 암호화한다. 암호화된 내용은 서버에 전송하고 전송된 값은 복호화 과정을 거쳐 데이터베이스에 입력한다.

3.2 인증 과정

① 클라이언트가 서버에게 인증 페이지를 요청한다.
② 서버에서 인증에 필요한 키와 난수를 발생하고 IP를 체크한 후 키, 난수, IP를 임시 저장한다.
③ 클라이언트에 암호화된 난수를 보낸다.
④ ID와 암호를 입력하고 전송을 요청한다.(로그인)
⑤ 클라이언트에서 서버로 난수를 복호화하고 ID와 암호를 암호화하기 위한 키를 요청한다.
⑥ 서버에서 요청 클라이언트의 IP를 확인한 후 키를 보낸다.
⑦ 클라이언트는 난수를 복호화하고 난수만큼 암호화한다.

⑧ 클라이언트에 저장된 키를 제거하고 서버로 ID와 암호를 전송한다.
⑨ 서버에서는 IP를 확인하고 키를 제공하고 복호화 한다.
⑩ 복호화가 끝나면 해당 IP의 임시 저장 정보를 제거한다.
⑪ 복호화된 결과와 서버에 저장된 결과를 비교하여 인증한다.

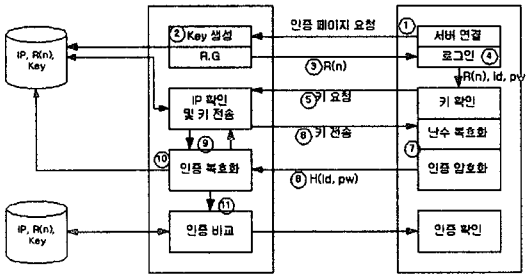

그림1 인증 절차

3.3. 알고리즘 구현

암호 시스템에 사용되는 알고리즘은 크게 3부분으로 나눌 수 있다. 암호화와 복호화 그리고, 이 둘을 진행하기 위해 필요한 Key 생성 부분이다.

3.3.1 Key 생성 구현

암호는 키를 사용하기 때문에 키가 다른 경우 같은 내용이더라도 서로 다른 결과를 가져온다. 여기서는 SecretKey라는 파일에 키를 저장하여 사용함으로써 클라이언트와 서버는 같은 키를 사용하여 암호화, 복호화 작업을 한다. 대칭 암호화는 키 쌍이 아닌 하나의 키를 사용한다. 키는 DES 알고리즘에 의해 생성된다.

클라이언트로 키 전송 방법은 IP를 확인하고 임시 저장소의 IP를 비교하여 같은 경우 전송한다. SecretKey는 인증 암호화가 끝나면 자동 삭제된다. 서버에서도 복호화 과정에서 사용된 후 파일을 삭제한다. 이 키 파일은 일회용으로써 키가 항상 다르기 때문에 파일이 유출되지 않고는 암호화된 아이디와 패스워드의 해독은 불가능하다. 이 작업시 클라이언트 IP 주소를 함께 얻어낸다.

```
Key key;
// DES 암호 알고리즘을 위한 키를 생성
KeyGenerator gen = KeyGenerator.getInstance("DES");
gen.init(new SecureRandom()); //새 키 생성 난수 초기화
key = gen.generateKey();
ObjectOutputStream out = new ObjectOutput_Stream(new
FileOutputStream("Key.ser"));
out.writeObject(key);
out.close();
//클라이언트 IP 획득
String remoteAddress = request.getRemoteAddr()
```

그림 2 키 생성 및 IP 얻기 루틴

3.3.2 암호화 구현

암호화는 서버의 난수 암호화와 클라이언트의 패스워드 암호화에 사용된다. 키를 먼저 생성한 후 이 키를 이용하여 암호화한다. 암호문의 시스템 호환성을 유지하기 위하여 UTF8 표준 인코딩을 사용한다.

```
//암호화에 사용될 키 획득
try {
 ObjectInputStream in = new ObjectInputStream(new
FileInputStream("Key.ser"));
 key = (Key) in.readObject();
 in.close();
}
//암호화
Cipher cipher = Cipher.getInstance("DES", "SunJCE");
cipher.init(Cipher.ENCRYPT_MODE, key);
String amal1 = args[0];
for ( int i = 1 ; i < args.length ; i++ ) amal1 += " " + args[i];
//바이트 배열로부터 암호화
byte[] id = amal1.getBytes("UTF8");
byte[] id = cipher.doFinal(id);
String amal2 = args[1];
for ( int i = 2 ; i < args.length ; i++ ) amal2 += " " + args[i];
byte[] password = amal2.getBytes("UTF8");
byte[] password = cipher.doFinal(password);
BASE64Encoder encoder = new BASE64Encoder();
String base64 = encoder.encode(password);
```

그림 3 암호화 루틴

3.3.3 복호화 구현

복호화는 클라이언트의 난수 복호화와 서버의 패스워드 복호화에 사용된다. 이 역시 미리 생성된 키를 가지고 복호화한다.

```
//복호화에 사용될 키 획득
try {
 ObjectInputStream in = new ObjectInputStream(new
FileInputStream("Key.ser"));
 key = (Key) in.readObject();
 in.close();
}
```

```
//복호화
Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5 Padding");
cipher.init(Cipher.DECRYPT_MODE, key);
BASE64Decoder dec = new BASE64Decoder();
byte[] id = dec.decodeBuffer(args[0]);
byte[] id = cipher.doFinal(id);
String id = new String(id, "UTF8");
byte[] password = dec.decodeBuffer(args[1]);
byte[] password = cipher.doFinal(password);
String password = new String(password, "UTF8");
```

그림 4 복호화 루틴

3.4 불법 접근 방지를 위한 3가지 대책

사용자 인증과정은 크게 3가지 형태로 보안을 유지한다.

첫째, 사용자의 IP를 추적하여 인증절차에 사용한다. 사용자의 IP에 따라 암호키를 제공하고 복호화에 필요한 난수를 제공한다. 불법 접근일 경우 임시 저장소에 해당 IP의 정보가 없기 때문에 난수와 암호키는 제공되지 않는다.

둘째, 키를 이용하는 것이다. 키는 암호화 절차에 필요한 것으로 DES를 알고리즘으로 사용한다. 같은 문장을 암호화해도 키가 다르면 서로 다른 값을 갖기 때문에 본 문에서는 일회성 키를 사용하여 전송시 항상 다른 값을 가지도록 한다.

셋째, 인증절차에 시간을 제한하는 방식을 적용한다. 인증절차에 시간제한을 두어 시간을 넘길 경우 임시 저장소에 저장되어있는 IP, 난수, 키가 제거되어 인증절차를 따르지 않는 다른 접근이 있을 경우 인증에 필요한 내용이 없기 때문에 인증을 허가하지 않는다.

본 논문에서는 이들 세 가지 방식을 적용하여 기존의 변형 암호 방식의 보안을 강화했다.

기존 암호 방식과 제안 방식을 비교 분석하면 변형 암호 방식은 위장 공격에 대해 방어가 어렵다. 제안 방식은 IP와 키를 이용하여 위장 공격에 대한 판단이 용이하다. 변형 암호 방식과 일회용 암호 방식은 안전성에 대해 해쉬 함수에 대한 의존도가 크지만 제안 방식은 IP, 키, 해쉬 함수를 함께 사용하여 의존도를 낮추었다. 또한 기존 방식들이 기밀성 유지가 힘든 반면 제안 방식은 암호 알고리즘을 이용함으로써 문제를 해결했다.

4. 결론 및 향후 연구 과제

대부분의 시스템에서 적용하는 사용자 인증과정은 정보유출 가능성이 크다. 본 논문에서는 자바 기반의 변형 암호 인증 방식를 응용하여 사용자 인증과정의 안전성을 강화하였다. 본 연구의 사용자 인증과정은 항상 다른 키를 사용함으로써 패킷정보가 유출되더라도 실제 사용자 아이디와 암호를 모르면 접근이 불가능하다. 또한 IP와 키를 사용하여 불법 사용자의 불법 경로를 통한 접근을 막는다. 기존 일회용 암호는 해쉬 함수에 대한 의존도가 크지만 제안 기법은 해쉬 함수와 함께 IP와 키를 사용함으로써 해쉬 함수에 대한 의존도를 낮추었다.

향후 연구 과제로는 사용자가 아이디와 암호를 입력하는 과정에서 일어날 수 있는 정보 유출에 대한 해결이 필요하고 인증정보가 사용자 인증에 제한되지 않고 전체 시스템에 적용할 수 있는 방법이 필요하다. 또한, 키를 파일로 저장하지 않고 사용할 수 있는 기술과 키 생성과정에서 과부하를 줄이는 기술 개발이 필요하다.

참고문헌

[1] 이강수, 최성자 "Java환경의 보안문제에 관한 고찰" 한국통신정보보호학회지 9 v.6, n.3, 1996

[2] 조인준, 정회경, 김동규 "인터넷 보안 메커니즘에 관한 연구" 한국통신정보보호학회지 6 v.8, n.2, 1998

[3] 박희운, 이임영 "기밀성을 제공하는 상호 인증 일회용 패스워드 메커니즘 설계" 한국정보처리학회 13회 춘계학술대회논문집, 2000

[4] "일회용 패스워드 기술," 한국정보보호센터, www.kisa.or.kr

[5] Jonathan Knudsen "자바 보안과 암호학" O'Reilly

[6] 위장현, 임인채, 김한경 "원타임 패스워드시스템을 적용한 지불시스템 모델" 한국정보처리학회 13회 춘계학술대회논문집, 2000