효율적인 토큰 업데이트를 이용한 인증 기술에 관한 연구

이정호° 장인주 유형선
인하대학교 컴퓨터공학부
ljh7915@nate.com ° jangij@inhaian.net hsyoo@inha.ac.kr

Efficient Token-Update scheme for Site Authentication

Joung Ho Lee° InJoo Jang, Hyeong Seon Yoo
School of Computer Science and Engineering, Inha University

요 약

본 논문에서는 사이트 인증을 위해 효율적으로 토큰을 업데이트하여 등록된 사용자만을 대상으로 인증하는 것을 목적으로 하고 있다. 일반적인 OTP(one-time password)와 스마트카드를 이용한 인증은 다양한 공격에서 서버와 클라이언트를 보호하는데 사용이 되고 있으며, KIC(key information center)를 통해 서버와 클라이언트를 상호 인증을 한다. 제안하고 있는 기법에서 KIC를 거치지 않고 대체 사용이 가능한 컴퓨터 안에 있는 랜 카드의 고유한 값인 맥(media access control) 주소와 난수를 사용하여 토큰을 업데이트하는 방법은 사이트 인증에 매우 효율적인 알고리즘이다.

1. 서 론

현재 대중적으로 사용하고 있는 다중 사용자 시스템에서는 시스템에 접근하기 위해서 사용자에게 유일한 아이디와 패스워드를 요구하며, 평문이나 암호문의 형태로 전송이 이루어진다. 사용자가 서버에 인증을 받기 위해서 데이터를 전송하는 과정에서 평문으로 전송될 경우, 공격자에게 노출 될 가능성이 있기 때문에 완벽한 인증이 불가능하다. OTP는 로그인을 하기 위해서 예측 불가능한 난수를 발생시켜 패스워드를 만들어 제공하기 때문에, 매번 로그인이 이루어질 때마다 인증 값이 달라지므로, 스니핑의 공격을 효과적으로 막을 수 있다 [1,2,3].

최근에는 스마트 카드를 활용한 인증 기법들이 많이 제안되어 지고 있다. 스마트 카드의 특성은 저장과 연산이 가능하며, 사용자가 선택한 아이디는 KIC에서 별도의 인증을 통해 발급받은 CID(card identifier)를 스마트 카드에 저장을 하고 정해진 연산을 통해 서버에서 인증이 이루어 진다.

효율적으로 패스워드 인증키를 받기 위해서는 다음과 같은 4가지가 요구된다. 첫째, 인증 테이블이 없어야 하며, 둘째, 패스워드를 자유롭게 선택해야 하고, 셋째, 적은양의 통신과 연산을 수행해야 하며, 마지막으로 상호인증이 이루어 져야 한다 [4]. 일반적으로 서버에서는 등록된 사용자의 아이디와 패스워드를 평문 혹은 암호문의 형태로 저장을 하고 있지만, 변경을 하지 않고 오랜 기간 사용하게 되면, 암호문으로 이루어져 있어도 공격자에게 아이디와 패스워드의 공격을 받을 수 있다.

이 논문에서 제안하는 방법은 하드웨어와 사용자를 하나의 쌍으로 생각을 하여, 지정된 사용자가 아니고서는 로그인과 인증을 받을 수 없으며, 아무런 제약 없이 자유롭게 선택한 아이디와 패스워드를 사용할 수 있다. 그리고 전송되는 데이터는 공개키 방식과 해쉬 함수를 사용하기 때문에 공격자에게 쉽게 노출되지 않으며, 또한 서버에 저장되는 인증 값은 사용자가 로그인 할 때마다 새롭게 업데이트가 된다. 그래서 최근 사용 빈도가 높아지고 있는 스마트 카드, 노트북, 개인용 휴대 단말기 등과 같이 개인을 대상으로 사용하는 휴대용 기기에 적용 할 경우 안전성을 크게 높여 줄 것으로 기대된다.

2. 패스워드 인증 기법

OTP의 일반적인 기법은 시간 혹은 클라이언트와 서버에서 한번만 선택하여 사용하는 임시의 수(nonce)와 난수를 사용한다. 서버에서는 N을 선택하고 N-1의 값을 클라이언트에 보내주면, 클라이언트는 난수를 N-1번만큼 해쉬 연산을 한다. 클라이언트에서 연산된 해쉬 값을 서버에 보내면, 서버에서는 클라이언트로부터 받은 해쉬 값을 1회 더 연산하여 처음의 난수를 N번 해쉬한 값과 비교 후 인증을 결정한다. 만약 서버에서 N의 값이 규칙적인 형식에 의해서 선택이 된다면, 공격자는 해킹을 통해 서버에 접근해서 N을 획득한 후 클라이언트를 속일 수 있다 [2,3].

알고리즘1. OTP 인증 기법

$$OTP\ Authentication:$$

$$S\ selects\ N, computes\ h(R)^{(N)};$$

$$1.\ S \rightarrow C:\ \ N-1, R;$$

$$2.\ C \rightarrow S:\ \ h(R)^{(N-1)}$$

$$3.\ S\ calculates\ h(h(R)^{(N-1)}),$$

$$accepts\ \ if\ \ h(R)^{(N)} = h(h(R)^{(N-1)})$$

$$rejects\ \ \ otherwise$$

Peyravian과 Zunic는 원격 사용자를 위한 패스워드 전송 기법을 제안하였다 [6]. 사용자는 클라이언트에 아이디와 패스워드를 전송을 한다. 클라이언트는 서버에 아이디와 함께 난수(rc)를 조합하여 전송을 한다.

서버는 다른 난수(rs)를 클라이언트에 반환을 한다. 클라이언트는 아이디와 패스워드를 해쉬로 조합을 하고 서버에 한번 사용할 수 있는 토큰과 함께 아이디를 전송한다. 서버는 아이디를 비교하고 새로운 토큰을 생성하여 클라이언트로부터 받은 토큰과 비교 후 인증을 결정한다. 이 기법에서 서버는 아이디와 패스워드를 조합한 해쉬를 저장하고 있으며, 패스워드의 형태로 특별히 저장을 하지 않는 장점을 가지고 있다.

알고리즘2. Peyravian과 Zunic의 패스워드 전송 기법

$$
\begin{aligned}
&1.\ U \rightarrow C: \ \{id, pw\}\\
&2.\ C \rightarrow S: \ \{id, rc\}\\
&3.\ S \rightarrow C: \ \{rs\}\\
&4.\ C \quad calculates \quad token = h(h(id, pw), rc, rs)\\
&5.\ C \rightarrow S: \ \{id, token\}\\
&6.\ S \quad calculates \quad token*\\
&\qquad accepts \quad if \quad token* = token\\
&\qquad rejects \quad otherwise
\end{aligned}
$$

그리고 서버에 접근을 하기 위해서 전송되는 토큰에 서버와 클라이언트에서 서로 다른 난수를 가지고 변경을 하게 된다. 그러나 만약 공격자가 암호화 되지 않은 id, pw, rc, rs를 얻을 수 있는 가능성이 있기 때문에, 암호화 혹은 해쉬를 사용할 필요가 있다.

3. 개선된 토큰 업데이트 기법

만약 개인 컴퓨터의 고유한 숫자를 사용한다면 KIC와 같은 인증기관이 불필요하다. 본 논문에서는, KIC를 대체하기 위해서 랜 카드의 맥 주소를 사용한다. 맥 주소는 랜카드를 생산할 때부터 부여되는 고유의 하드웨어 주소를 말한다. 등록된 사용자에게만 인증을 하기 위해서 맥 주소를 토큰 업데이트에 사용한다. 그리고 평문 형태로 보내어지는 난수와 아이디, 패스워드의 약점을 보완하기 위해서 암호화 알고리즘 및 해쉬 함수를 다음과 같이 적용한다.

- 등록

사용자(User)는 아이디(id)와 맥 주소(ma), 패스워드(pw)를 포함하여, 아래와 같이 token1을 구성한다.

$$U \ construct \quad token1 = (h(id \,\|\, ma) \,\|\, pw) \tag{1}$$

사용자는 token1을 가지고 h(token1)으로 구성한다. 그리고 서버의 공개키를 사용하여 암호화한 token1을 암호화 한 후 h(token1), id와 함께 서버에 보낸다.

$$U \rightarrow S: \{id, h(token1), \varepsilon_{ku}(token1)\} \tag{2}$$

서버는 암호화된 token1을 복호하여 h(token1)을 구성한다. 만약 두개의 해쉬가 동일하면, 서버는 등록을 완료하고, 서버는 난수(rs)를 생성하여 h(rs)를 구성한 후 사용자의 공개키로 암호문의 형태로 사용자에게 보낸다.

$$
\begin{aligned}
&S \ decrypts \ and \ reconstructs \quad h(token1)*\\
&\qquad accepts \quad if \ h(token1) = h(token1)*,\\
&\qquad returns \ \varepsilon_{ku}(h(rs))\\
&\qquad rejects \quad otherwise.
\end{aligned} \tag{3}
$$

서버는 token1과 h(rs)와 함께 token2를 구성하고, 업데이트 된 token2의 해쉬 값을 유지한다.

$$
\begin{aligned}
&S \ constructs \quad token2 = (h(id \,\|\, ma) \oplus h(rs)) \,\|\, pw,\\
&\qquad\qquad h(token2)
\end{aligned} \tag{4}
$$

- 로그인과 인증

로그인과 인증은 서버로부터 받은 h(rs)와 함께 token2를 구성한다. 사용자는 전송받은 암호문을 복호화하고, token1과 h(rs)와 함께 token2를 구성한다. 그리고 서버에 id와 h(token2)를 보낸다.

$$
\begin{aligned}
&U \ decrypt \quad \varepsilon_{ku}(h(rs))\\
&\qquad construct \quad token2 = (h(id \,\|\, ma) \oplus h(rs)) \,\|\, pw\\
&\rightarrow S: \ \{id, h(token2)\}
\end{aligned} \tag{5}
$$

서버는 h(token2)의 해쉬를 저장되어 있는 h(token2)와 비교를 한다. 만약 두개의 해쉬가 동일하면, 서버는 로그인과 인증을 결정한다. 그리고 서버는 새로운 난수(rs*)를 생성하고, h(rs*)로 구성 후 암호화하여 사용자에게 보낸다. 서버에는 오래된 token2은 아래와 같이 token2*로 업데이트 하여 해쉬 값을 유지한다.

$$
\begin{aligned}
&S \ accepts \ if \ h(token2) = h(token2)*\\
&\qquad updates \quad token2* = (h(id \,\|\, ma) \oplus h(rs*)) \,\|\, pw\\
&\qquad returns \quad \varepsilon_{ku}(h(rs*))\\
&\quad rejects \quad otherwise.
\end{aligned} \tag{6}
$$

- 패스워드 복구

사용자는 id, ma 그리고 마지막으로 받은 h(rs)를 조합하여 token3를 구성한다. 사용자는 서버에 아이디와

h(token3), 그리고 암호화된 token3를 보낸다.

$$U \quad constructs \quad token3 = \{h(id \| ma) \| h(rs)\} \rightarrow S: \quad \{h(token3).\varepsilon_{k_s}(token3)\} \tag{7}$$

서버는 token3를 재구성하고 두개의 해쉬가 동일하면 사용자에게 id, pw, h(rs*)의 암호문으로 보낸다.

$$S \quad reconstructs \quad h(token3)^* \quad accepts \quad if \; h(token3) = h(token3)^*, \quad returns \quad \varepsilon_{k_u}(id, pw, h(rs)) \quad rejects \quad otherwise. \tag{8}$$

4. 보안 분석

• 패스워드 추측 공격

사용자가 기억하기 쉬운 패스워드를 선택하면, 공격자에게 패스워드 추측만으로 재등록과 로그인, 인증을 가능하게 할 수 있다. 하지만 제안된 기법에서는 평문의 형태로 아이디와 패스워드 그리고 맥 주소를 전송하지 않고, 해쉬 값과 암호문으로 전송이 된다. 만약 공격자가 아이디와 패스워드를 알고 있어도, 서버에 등록 및 로그인 하기 위해서는 맥 주소와 암호화된 h(rs) 복호하기 위한 복호키도 필요하게 된다. 그리고 맥 주소는 리눅스 혹은 유닉스 시스템에서는 원격으로 얻기는 매우 힘들다.

• 재전송 공격

공격자는 아이디와 이전에 사용한 h(token2)를 가지고 로그인을 시도를 해도, 업데이트 된 token2이 아니기 때문에 로그인이 불가능하다. token2는 매번 로그인을 할 때 업데이트 된다. 그리고 token2의 구성은 등록된 token1과 한번만 사용이 가능한 난수를 해쉬한 값으로 구성이 되어 있다.

• 패킷 스니핑

공격자가 사용자의 아이디와 h(token1), 암호화된 token1의 패킷을 획득했다라고 가정을 하면, 공격자는 서버의 복호키를 얻기 힘들기 때문에 token1을 획득하는 것은 불가능하다. 그리고 획득한 h(token2)는 매번 업데이트가 되므로 사용이 불가능 하다.

• 위조 공격

공격자가 등록된 사용자의 아이디와 패스워드 그리고 주소를 위조하였다. 그러나 이미 등록된 사용자가 존재하므로 공격자의 재등록은 불가능하며, token1을 완벽히 위조해도 사용자의 복호키 없이 token2를 구성하는 것은 불가능하다.

5. 결론

본 논문에서, 원격 사용자가 획득하기 어려운 맥 주소와 난수를 이용한 토큰 업데이트의 기법을 제안 하였다. 인증에 사용하는 데이터들은 공개키 방식과 해쉬 함수의 사용으로 안전성을 높였으며, 사용자는 서버로부터 전송 받은 h(rs)를 이용하여 인증 데이터를 구성하기 때문에, 매번 다른 인증 데이터를 생성하여 사용할 수 있다. 이는 패스워드 추측공격, 재전송 공격, 패킷 스니핑, 위조 공격 등으로부터 안전하다. 그리고 패스워드 복구에서는 서버로부터 마지막으로 받은 h(rs*)가 필요하기 때문에, 등록된 사용자가 아니면 복구가 불가능하다. KIC를 대신할 수 있는 맥 주소와 토큰을 업데이트 가능하게 하는 h(rs)로 구성된 인증 데이터는 사이트 인증에 효과적이다.

6. 참고문헌

[1] M. Bishop, "Password Management," Proceedings of COMPCON, 167-169, 1991

[2] B. Soh and A. Joy, "A Novel Web Security Evaluation Model for a One-Time-Password System," Proceedings of the IEEE/WIC, WI('03), IEEE Computer Society, 2003

[3] J. Archer Harris, "OPA: A One-time Password System," ICPPW'02, IEEE Computer Society, 2002

[4] W.S. Juang, "Efficient Password Authenticated Key Agreement Using Smart Cards," Computers & Security, 23, 167-173, 2004

[5] W. Yang and S. Shieh, "Password Authentication Schemes with Smart Cards,"Computers & Security, 18, 727-733, 1999

[6] M. Peyravian and N. Zunic, "Methods for Protecting Password Transmission,"Computers & Security, 19, 466-469, 2000

[7] T.H. Chen, W.B. Lee and G. Horng, "Secure SAS-like password authentication schemes,"Computer Standards & Interfaces, 27, 25-31, 2004

Acknowledgements.

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성 지원사업의 연구결과로 수행되었음.