소유 매체 정보를 이용한 패스워드 강화방법

이창희*, 이희조
*고려대학교 컴퓨터정보통신대학원
e-mail : guldook@korea.ac.kr

Password stretching method based on what you have

Chang-Hee Lee*, Heejo Lee
*Graduate School of Computer and Information Technology
Korea University

요 약

사용자들은 기억력의 한계로 인하여 동일한 패스워드를 많은 사이트에 사용하고 있다. 이로 인해 한 사이트에서 획득한 패스워드를 사용하여 다른 사이트를 공격하는 것이 용이하다. 패스워드 강화 방법은 이러한 공격을 막기 위해 엔트로피가 낮은 패스워드로부터 엔트로피가 높은 패스워드를 만드는 방법이다. 본 논문에서는 신용카드, 신분증과 같이 사용자가 가지고 다니는 소유 인증 매체의 정보를 이용하여 사용자의 패스워드를 강화하는 방법을 제안한다. 실험결과 제안하는 방법은 기존 연구와 같은 보안성을 유지하면서도 강화된 패스워드 생성시간을 1/370 배로 단축하는 효과가 있다.

1. 서론

패스워드 시스템은 단순정보 제공 서비스에서 전자상거래 서비스까지 가장 널리 사용되고 있는 사용자 인증 방법이다. 하지만, 사람들은 기억력의 한계로 인하여 다수의 사이트에 동일한 패스워드를 사용하고 있다. 이러한 점을 악용하여, 허위사이트를 만들어서 패스워드 및 사용자의 중요 정보를 획득한 후, 해당 정보를 다른 사이트를 공격하는 데 사용하는 피싱공격이 큰 문제가 되고 있다. Anti-Phishing Working Group(APWG)의 2006 년 6 월 보고서에 따르면, 2006 년 6 월에만 28,571 번의 피싱공격과 9,255 개의 피싱 사이트가 있었으며, 130 개의 브랜드가 피싱공격의 피해를 입었다 [1].

패스워드 강화 방법은 엔트로피가 낮은 패스워드로부터 엔트로피가 높은 패스워드를 생성하는 방법이다. 이를 응용하여 하나의 약한 패스워드로부터 각 사이트마다 서로 다른 강한 패스워드를 생성하는 것이 가능하며, 이를 통해 피싱공격을 막을 수 있다.

기존의 패스워드 강화 방법은 아이디, 사이트 도메인 명 등 상수 값을 사용하여 패스워드를 강화하는 방법과 랜덤값을 추가하여 패스워드를 강화한 후 랜덤값은 삭제하는 방법 등이 제시되었다 [3, 4, 5, 6]. 하지만, 상수 값은 공격자도 알 수 있고 따라서 무작위 공격이 가능하다. 랜덤값을 사용하는 경우 공격자의 무작위 공격을 어렵게 하지만 사용자 또한 삭제된 랜덤값을 찾는 과정을 수행해야 하므로 패스워드 강화 빈도가 높은 환경에서는 사용하기 어렵다.

본 논문에서는 신용카드, 신분증과 같이 사용자가 가지고 다니는 인증 매체 정보를 추가하여 패스워드를 강화함으로써, 기존 연구와 비교하여 공격자가 패스워드를 찾기 위해 수행해야 하는 무작위 공격 횟수는 더 크게 하고, 사용자가 입력한 패스워드로부터 강화된 패스워드를 생성하는 데 필요한 시간은 더 짧게 만드는 패스워드 강화 방법을 제안한다.

본 논문의 2 장에서는 패스워드 강화 방법과 관련된 기존 연구들을 분석하여 그 문제점을 파악하고, 3 장에서는 사용자의 인증 매체 정보를 이용한 새로운 패스워드 강화방법을 제안하며, 4 장에서는 기존 연구들과 본 논문의 제안을 비교 분석하고, 5 장에서는 분석 결과를 바탕으로 결론을 도출한다.

2. 관련 연구

패스워드 강화 방법은 엔트로피가 낮은 키, K_{short} 로부터 무작위(brute force) 공격이 어려운 키 K_{long} 을 유

도하는 방법이며, 다음과 같이 정의할 수 있다 [2].

$K_{long} = F(K_{short}, Salt)$

위 식에서 *Salt* 는 키를 강화하기 위한 변수 값이며, 시스템마다 다른 값, 시간마다 다른 값, 메시지에 기반한 값 등이 될 수 있다. 위 식의 함수 F()는 무작위 공격에 소요되는 시간을 늘리기 위한 일방향 함수로써 해쉬함수 기반한 것과 블록 암호화에 기반한 것이 있다.

일방향 함수, F()는 출력으로부터 입력을 찾는 것이 가능하지 않으므로, 공격자는 강화된 패스워드, K_{long} 을 획득한 후, K_{short} 를 찾기 위해 무작위(brute force) 공격을 시도해야 한다. 따라서 무작위(brute force) 공격에 소요되는 시간이 클수록 패스워드 강화 방법은 안전하다고 할 수 있다.

Ross 의 연구에서는 Salt 값으로써 도메인 명을 사용하고 F()로는 HMAC 을 사용했다 [3]. 또한, 패스워드 강화 기법을 브라우저 확장 모듈로써 구현함으로써 HTML 폼을 이용한 로그인 시스템에 적용할 수 있도록 했다. Ross 의 패스워드 강화 알고리즘은 다음과 같다.

$K_{long} = HMAC(K_{short}, dom)$

Ross *et al.*의 알고리즘에서 salt 로 사용한 dom(도메인 명)은 공격자 또한 알 수 있는 값이다. 따라서 공격자가 K_{short} 를 찾기 위해 수행해야 하는 총 공격 시도 횟수는 $2^{length\ of\ Kshort}$ 가 된다. 즉, 강화된 패스워드를 획득한 후 오프라인 상에서 $2^{length\ of\ Kshort}$ 번의 공격시도를 하면, 사용자의 패스워드를 찾을 수 있다.

Halderman *et al.*의 연구에서는 Ross *et al.*의 연구가 HMAC 에 기반함으로써 무작위 공격에 소요되는 시간이 짧다는 단점을 보완하여 다음과 같은 알고리즘을 제안하고 있다 [4].

$V = f^{k1}(salt1, K_{short})$
$K_{long} = f^{k2}(K_{short}, salt2, V)$

여기서, $f^k()$는 f 를 k 번 반복해서 적용한 것을 의미하며, *salt1* 은 사용자 아이디, *salt2* 는 도메인 명이다. Halderman *et al.*의 연구는 패스워드 강화 알고리즘을 한 번 수행하는 데 드는 시간을 늘림으로써, 전체 무작위 공격에 소요되는 시간을 늘린 것이다. 또한, k1 변수 값을 크게 설정하고, k2 값을 작게 설정한 후, 생성시간이 오래 걸리는 V 값을 캐쉬함으로써, 공격에 필요한 시간은 커지고, 강화된 패스워드 생성시간은 짧아지도록 했다. 하지만, 여전히 사용자 아이디와 도메인은 공격자가 알고 있는 값이며, 따라서 필요한 공격시도 횟수는 늘어나지 않는다. 그렇다고 일회 공격에 소요되는 시간을 더 늘리기 위해 K1 과 K2 값을 계속 증가시키는 경우 사용자가 강화된 패스워드를 생성하는 데 걸리는 시간도 증가하게 됨으로써, 사용자의 편의성을 떨어트린다.

Abadi *et al.*는 패스워드 강화 방법을 파일 암호화에 적용했다. 패스워드와 랜덤한 salt 값으로부터 강화된 패스워드, K_{long} 를 생성하여 저장하고, 패스워드와 salt 값을 연접한 값으로부터 암호화키를 유도하여 파일 암호화를 한 후, salt 값을 삭제한다 [5]. 이렇게 함으로써, 공격자가 파일을 복호화하기 위해 수행해야 하는 공격 시도 횟수는 $2^{length\ of\ (패스워드\ +\ salt)}$ 가 된다. 하지만, 파일을 복호화하기 위해서는 원래의 salt 값을 찾아야 하기 때문에 사용자 편의성에 문제가 있다.

Manber 의 연구에서는 기존 UNIX 패스워드 시스템을 강화하는 방법을 제안했는데, 공개 salt 와 비밀 salt 를 사용한다 [6]. 12 비트 길이의 공개 salt 는 파일에 저장해두고, 100 개의 값으로부터 선택된 비밀 salt 는 비밀번호를 생성한 후 파기한다. 이렇게 함으로써 공격자가 암호화된 패스워드 파일을 얻은 후에 오프라인에서 패스워드를 얻기 위한 공격 시도 횟수는 $2^{length\ of\ 패스워드} * 100$ 으로 증가하게 된다.

Abadi *et al.*와 Manber 의 알고리즘을 사용하면, 공격 시도 횟수는 증가하지만, 정당한 사용자의 패스워드 생성 시간도 증가하기 때문에 많은 서비스에 패스워드를 사용해야 하는 경우에는 적합하지 않다.

3. 제안하는 패스워드 강화방법

본 논문에서는 패스워드를 공격하는 데 필요한 시간은 굉장히 크고, 패스워드를 강화시키는 데 소요되는 시간은 짧은 알고리즘을 제안한다. 제안하는 방법은 크게 두 가지 절차로 수행된다. 그림 1 에서 보는 것처럼 단계 1 은 사용자가 소유하고 있는 인증 매체의 정보로부터 salt 값을 생성하는 과정이고, 단계 2 는 생성된 salt 를 사용하여 사이트별 강화된 패스워드를 생성하는 과정이다.

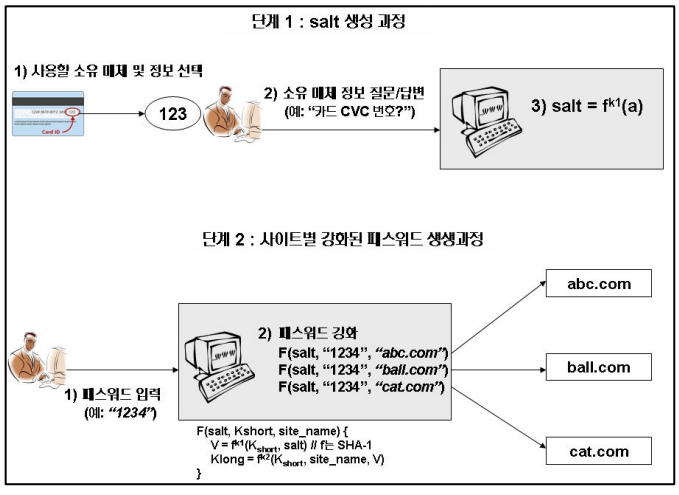

그림 1 제안하는 패스워드 강화 방법

3.1 패스워드 강화 알고리즘

첫 번째 단계에서는 salt 생성을 한다. 사용자는 먼저 자신이 보유하고 있는 인증 매체(예: 신용카드, 신분증, 은행 보안카드 등)에 표기된 특정 정보를 요구

하는 질문을 설정한다(예: “XX 카드의 CVC 값을 입력하세요”, “주민등록증 발급월일을 입력하세요” 등). 설정한 질문을 q 라고 하고, q 에 대한 사용자 자신의 답을 a 라고 했을 때, salt 를 다음 식에 따라 생성한다.

$salt = f^{k1}(a)$, (a: 특정 매체 정보 요청에 대한 답변)

패스워드 강화 시마다 사용자에게 인증 매체 정보를 요청하여 salt 값을 생성하는 경우 사용자가 굉장히 불편하다. 따라서 salt 값은 생성 후 캐쉬한다. 그리고 나서, 두 번째 단계에서는 생성된 salt 값을 가지고 Halderman *et al.*의 알고리즘을 적용한다.

$V = f^{k2}(salt, K_{short})$
$K_{long} = f^{k3}(K_{short}, dom, V)$

Halderman *et al.*의 알고리즘에서는 V 값을 캐쉬하는데, 제안하는 방법에서는 V 값을 캐쉬하지 않는다. 만약, 사용자가 캐쉬 파일이 없는 다른 단말기를 사용하는 경우에는 다시 한번 사용자에게 비밀정보를 요청한 후 salt 를 생성하여 캐쉬함으로써, 제안하는 알고리즘을 적용할 수 있다.

제안하는 방법을 OpenSSL 을 사용하여 커맨드 라인에서 사용할 수 있도록 구현했으며, 그림 2 은 실행화면을 캡쳐한 것이다.

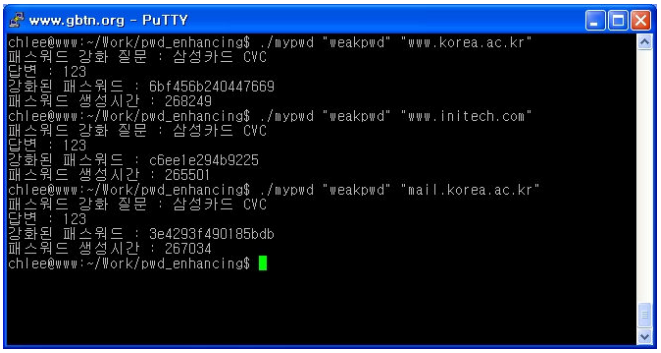

그림 2. 패스워드 강화 프로그램 실행화면

3.3 보안성 분석

제안하는 패스워드 강화 방법은 사용자 단말기는 안전하다는 가정을 하고 있다. 따라서 공격자는 특정 사이트로부터 사용자의 강화된 패스워드, K_{long} 을 획득한 후 패스워드 K_{short} 를 찾기 위해 무작위 공격을 해야 한다. Halderman *et al.*의 연구에서는 공격자가 아무런 정보도 없이 하는 공격, 강화된 패스워드를 획득한 후 공격, 캐쉬된 파일을 획득한 후 공격, 강화된 패스워드와 캐쉬 파일을 모두 획득한 후 공격 등 4 가지 공격 유형을 언급하고 있는 데, 제안 알고리즘을 이 네 가지 경우에 대해서 분석했으며, 표 1 은 분석결과를 정리한 것이다.

공격유형 1) 정보없음 : 공격 대상자에 대한 아무런 정보도 없이 공격 대상 시스템의 특정 사용자의 패스워드를 찾기 위해 무작위 시도를 하는 공격. 하지만, 대부분의 시스템은 패스워드 오류 횟수를 제한하고 있기 때문에, 패스워드를 찾기 위한 시도를 계속할 수 없다. 따라서 이 경우의 공격은 성립하지 않는다.

공격유형 2) 강화된 패스워드 : 공격자가 보안이 약한 사이트 또는 피싱사이트에서 강화된 패스워드를 획득한 후 오프라인 공격을 할 수 있다. 이 경우 사용자 매체 정보와 패스워드를 모두 추측해서 무작위 공격을 해야 한다. 따라서 공격에 필요한 무작위 공격 횟수는 $n(a)*2^{length\ of\ Kshort}$ 가 된다(여기서 n(a)는 사용자 매체 정보가 될 수 있는 값들의 가지 수이다).

공격유형 3) 캐쉬 획득 : 공격자가 캐쉬 파일을 획득하여 salt 값을 알고 강화된 패스워드는 모르는 경우이다. 제안하는 방법은 K_{short} 로부터 유도된 어떤 값도 캐쉬하지 않는다. 따라서 캐쉬파일을 획득한다고 해서 오프라인 공격을 할 수는 없다.

공격유형 4) 캐쉬 및 패스워드 획득 : 강화된 패스워드와 캐쉬 파일을 모두 획득한 경우, 즉 공격자가 강화된 패스워드와 salt 값을 모두 아는 경우이다. 이 경우 가능한 모든 패스워드를 추측하여 패스워드 강화 알고리즘에 입력한 후 그 결과를 K_{long} 값과 비교함으로써 오프라인 공격이 가능하다. 하지만, 제안하는 패스워드 강화 방법의 두 번째 단계에 소요되는 시간을 늘리거나, 소유 매체 정보를 일정 시간마다 변경함으로써 오프라인 공격을 피할 수 있다.

<표 1> 제안방식의 보안성 분석 결과

공격유형	필요한 공격 시도 횟수	공격당 해쉬 수
정보없음	공격불가	공격불가
강화된 패스워드	$n(a)*2^{length\ of\ Kshort}$	k1+k2+k3
캐쉬 획득	공격불가	공격불가
캐쉬 및 패스워드 획득	$2^{length\ of\ Kshort}$	k2+k3

4. 비교 및 분석

사용자 단말기의 안전성이 높아질수록, 캐쉬파일을 획득하는 것은 어려운 일이 될 것이다. 따라서 점차로 강화된 패스워드를 획득한 후의 오프라인 공격이 늘어날 것이다. 즉, 공격 유형 2 가 가장 빈번한 공격이 될 것이므로, 공격 유형 2 에 대해서 Ross *et al.*, Halderman *et al.*의 알고리즘과 비교한다. 제안 알고리즘에는 a, k1, k2, k3 등의 변수 값이 있으며, 비교를 위해 a 는 “주민등록증 발급월일”에 대한 답변으로 고정했다. 즉, n(a)는 365 가 된다. 그리고, k1, k2, k3 는 두 가지 경우로 나누어서 설정했는데, 설정 1 은 패스워드 생성 시간은 Halderman *et al.*의 알고리즘과 유사하고, 공격에 필요한 횟수는 커지도록 k1 을 크게 한 것이며, 설정 2 는 패스워드 생성 시간은 짧고, 공격에 필요한 시간은 Halderman *et al.*의 알고리즘과 유사하도록 k1 값을 작게 설정한 경우이다. Halderman *et al.* 알고리즘의 k1, k2 변수 값은 각각 $9*10^7$, $9*10^4$ 이고, 설정 1 의 k1, k2, k3 변수 값은 각각 $9.5*10^7$, $5*10^4$, $5*10^4$ 이며, 설정 2 의 k1, k2, k3 변수 값은 각각

$1.5*10^5$, $5*10^4$, $5*10^4$ 이다.

1) 필요 공격 시도 횟수 : brute-force 공격을 시도하는 경우 필요한 공격 시도 횟수

2) 단위공격 해쉬연산 수 : brute-force 공격 1 회에 필요한 해쉬 연산 수

3) 단위공격 시간 : brute-force 공격 1 회를 수행하는 데 소요되는 시간으로써, 단위공격 해쉬연산 수와 해쉬 1 회에 소요되는 시간의 곱이다.

4) 오프라인 공격 소요 시간 : 필요 공격 시도 횟수 * 단위공격 시간

5) 캐쉬한 경우 강화된 패스워드 생성시간 : 본 알고리즘에서 salt 값, Halderman *et al.*의 알고리즘에서 V 값이 캐쉬된 상황에서 정당한 사용자가 강화된 패스워드를 생성하는 데 소요되는 시간

설정 1 과 Halderman *et al.* 알고리즘의 비교결과를 보면 제안하는 방법의 무작위 공격에 필요한 공격 횟수는 $n(a)*2^{\text{length of Kshort}}$ 이므로, Halderman *et al.*의 알고리즘에 비해 n(a)배 만큼 크다

새로운 사이트의 강화된 패스워드를 생성하는 데 소요되는 시간은 단위공격 시간과 같은 데, 설정 2 의 결과를 살펴보면 오프라인 공격에 소요되는 시간은 유사하지만, 새로운 사이트의 강화된 패스워드를 생성하는 데 소요되는 시간은 0.27 초로 Halderman *et al.*의 연구 결과인 100 초에 비해 1/370 이 된다.

<표 2> 공격유형 2 번에 대한 비교결과

단위 : 초

기준	Ross	Halderman	설정 1	설정 2
1)	2^{56}	2^{56}	$365*2^{56}$	$365*2^{56}$
2)	1	k1+k2	k1+k2+k3	k1+k2+k3
3)	10^{-6}	100	100	0.27
4)	$6.4*10^{10}$	$6.4*10^{18}$	$365*6.4*10^{18}$	$6.4*10^{18}$
5)	10^{-6}	0.1	0.1	0.1

그림 3 은 패스워드 생성시간 대비 공격소요시간을 그래프로 나타낸 것이다. 그래프에서 알 수 있는 것처럼 같은 패스워드 생성 시간에 대해, 제안하는 알고리즘이 기존 알고리즘에 비해 공격소요시간이 더 크다는 것을 알 수 있으며, n(a)값이 클수록 공격소요시간은 더 커진다.

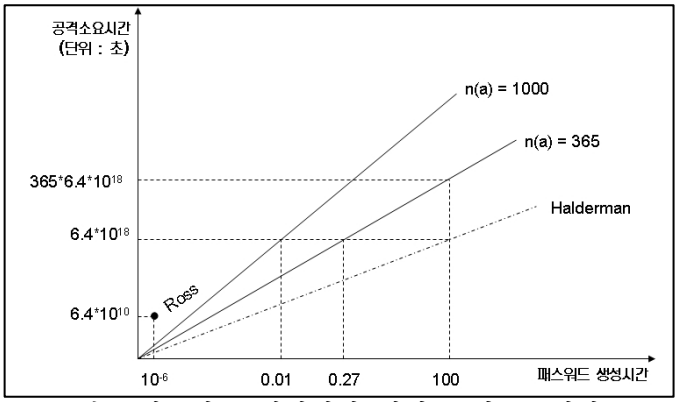

그림 3 패스워드 생성시간 대비 공격소요시간

Halderman *et al.* 알고리즘에서 id 와 dom 은 상수 값이므로, 가능한 K_{short} 값에 대해서 사전에 강화된 패스워드를 연산해두는 것이 가능하다. 따라서 실제로 공격에 소요되는 시간은 훨씬 더 짧을 수 있다. 이에 비해 제안 알고리즘을 사용하는 경우 salt 값을 획득한 경우에도, 해당 사용자의 소유 매체 정보를 모르는 경우 사전 연산을 하는 것이 불가능하다. 따라서 Ross *et al.*, Halderman *et al.*의 연구에 비해서 사전 연산 공격에 대해서도 높은 보안성을 가진다.

5. 결론 및 향후과제

본 논문에서는 보안이 취약한 인터넷 사이트로부터 획득한 사용자의 패스워드를 다른 사이트를 공격하는 데 사용하는 것을 막기 위해서, 사용자가 입력한 패스워드로부터, 사이트마다 서로 다른 강화된 패스워드를 생성하는 방법을 제안하고, 기존의 연구와 보안성을 비교, 분석했다. 제안하는 알고리즘은 강화된 패스워드를 획득한 후 오프라인 공격을 하는 경우와 캐쉬파일을 획득한 후 오프라인 공격을 하는 경우에 대해서 기존 연구에 비해 월등한 보안성을 나타내었다. 또한, 알고리즘의 변수 값을 적절히 설정함으로써 공격소요시간을 유지하면서도 패스워드 생성시간을 줄일 수 있다. 따라서 제안하는 알고리즘은 기존 연구에 비해 공격소요시간을 늘리면서도 패스워드 생성시간은 줄일 수 있는 패스워드 강화 알고리즘이다.

사용자가 캐쉬파일이 없는 새로운 단말기로 이동하는 경우 salt 값을 생성하기 위해서 다시 사용자에게 인증 매체 정보에 관한 질문을 해야 한다. 이 때 사용자에게 같은 질문을 할 수 있어야 한다. 이를 위해서 질문을 서버에 보관하는 절차, 미리 질문의 유형들을 코드화하는 방법 등은 추가로 연구가 되어야 한다.

참고문헌

[1] Anti-Phishing Working Group. Phishing Activity Trends Report, June 2006. http://antiphishing.org/reports/apwg_report_june_2006.pdf

[2] J. Kelsey, B. Schneier, C. Hall, and D. Wagner. Secure applications of low-entropy keys. Lecture Notes in Computer Science, 1396:121–134, 1998.

[3] Blake Ross, Collin Jackson, Nicholas Miyake, Dan Boneh, and John C. Mitchell. Stronger Password Authentication Using Browser Extensions, Proeedings of the 14th Usenix Security Symposium, 2005

[4] J.A.Halderman, B.Waters, and E.Felten. A Convenient method for securely managing passwords. Proceedings of the 14th International World Wide Web Conference (WWW 2005), 2005

[5] Mart´ın Abadi, T. Mark A. Lomas, and Roger Needham. Strengthening passwords. Technical Report 1997 - 033, 1997.

[6] U. Manber. A simple scheme to make passwords based on one-way functions much harder to crack, 1996.