사용자 정보를 이용한 인증 절차 자동화

황우섭[*1)], 박지수[**], 손진곤[*2)]
[*]한국방송통신대학교 대학원 정보과학과
[**]동국대학교 융합교육원
E-Mail: wooseob@knou.ac.kr, zetsaver@gmail.com

Automate authentication processes with user information

Woo Seob Hwang[*], JiSu Park[**], Jin Gon Shon[*]
[*]Dept. of Computer Science, Graduate School, Korea National Open University
[**]Convergence Institute, Dongguk University

요 약

사용자가 인터넷을 사용할 때 화면에 표시되는 텍스트나 그래픽 등을 웹 문서라고 하며 HTML5는 웹 문서를 제작하는 표준 언어의 일종이다. HTML5 중에서 web storage는 사용자가 인터넷을 통한 서비스를 받을 때 데이터를 저장하기 위한 기능으로 키와 값의 형태로 저장한다. web storage는 서버 측에서 사용되는 session storage와 클라이언트에서 사용되는 local storage가 있다. local storage 사용 시 데이터를 클라이언트에 평문 형태로 저장하며 만료 기간 없이 영구적인 특징을 갖고 있다. 이러한 특징은 공격자로부터 XSS 등의 공격에서 저장된 데이터의 접근 및 수정 그리고 탈취할 수 있어 공격자의 의도에 따라 데이터 가공 및 재사용이 가능하다는 문제가 있다. 보안 취약점 문제를 해결하기 위한 최근 연구들은 local storage에 저장된 데이터들을 암호화하여 기밀성을 높였다. 그러나 데이터 암호화를 사용하려면 잦은 암호 입력이나 온라인에서만 사용할 수 있다는 또 다른 문제점을 가지고 있다. 기존 보안 취약점 문제와 기존 연구의 문제점을 동시에 해결하기 위해 운영체제 사용자 정보와 기기의 정보를 활용하여 암호화에 필요한 사용자 인증을 자동화하였으며 검증을 위해 코드를 구현하고 테스트하였다.

Keyword : 사용자, 정보, 인증, 자동화, 암호화, HTML5, Local Storage

1. 서론

HTML은 Hyper Text Mark-up Language의 약자로서 사용자가 인터넷을 사용할 때 볼 수 있는 웹 문서의 모습을 기술하기 위한 표준 규약이다. HTML은 텍스트와 명령어 집합인 태그로 구성되며 HTML5는 HTML 표준의 5번째 버전으로 사용자가 인터넷 브라우저를 통하여 그래픽과 음악·동영상 등을 편리하게 접할 수 있는 표준 마크업 언어이다[1].

HTML5에서 web storage는 웹 문서의 데이터를 저장하기 위한 기능으로 키와 값의 형태를 가진다. web storage에 데이터가 저장되는 장소는 session storage와 local storage가 있으며, 데이터는 평문 형태로 저장된다. session storage에 저장된 데이터는 PC의 운영체제나 웹 브라우저가 종료할 때 자동으로 제거된다. 그러나 Local storage에 저장된 데이터는 반영구적으로 브라우저를 종료해도 데이터가 유지된다[2].

웹 브라우저의 local storage에 반영구적으로 저장되는 데이터는 평문 형태이다. 만약, 개인 정보가 웹 브라우저의 local storage에 평문 형태로 저장되어 있다면, 공격자가 XSS 등의 공격으로 저장된 데이터에 쉽게 접근하여 수정 및 탈취하여 재사용이 가능하다[3].

이러한 보안 취약점 문제를 해결하기 위해서 최근 연구들은 local storage에 저장된 데이터들을 암호화하여 정보의 기밀성을 높였으나 이러한 연구들이 또 다른 문제점을 가지고 있었다. 온라인에서만 암호화 이용할 수 있고 인증 서버를 구성해야 하는 문제점과 사용자 인증을 위한 잦은 암호 입력 요구로 인해 편리성을 갖추지 못하고 key-logger 등의 다른 공격에 취약할 수 있다는 문제점이 있다[4-9].

보안 취약점 문제와 기존 연구의 문제점을 해결하기 위해 운영체제 사용자 정보와 기기 정보를 암호화에 활용하여 사용자와 기기 인증을 자동화하였다.

2. 관련 연구

2.1 HTML5 로컬 스토리지의 문제점

HTML5에서 web storage의 기본적인 정의는 클라이언트에서 영구 저장 목적으로 키와 값의 데이터를 이용하며 API 형태로 기능을 제공하고 있다. web storage는 서버에서 사용되는 session storage와 클라이언트에서 사용되는 local storage로 나눌 수 있다[2].

특히 local storage는 사용자가 웹에서 사용했던 데이터

1) 한국방송통신대학교 대학원 재학생
2) 교신저자

들을 저장하는데 평문 형태로 저장하며 만료 기한 없이 영구 저장함으로서 보안 취약점 문제가 발생한다. 보안 취약점 문제는 저장된 데이터를 악의적인 의도가 있는 공격에 노출되면 변조 및 탈취를 할 수 있다는 문제가 있다[3].

우선 local storage의 연구 중 응용하려는 다양한 연구가 있었는데 local storage의 활용하기 위한 연구가 있었다[10, 11]. 또한, Local storage를 이용한 웹 사이트 인증 연구도 찾아볼 수 있었다[12].

local storage의 보안 취약점 문제를 연구한 기존 논문에서는 문제점을 해결하였으나 온라인 상태에서만 암호화를 사용할 수 있고 별도의 인증 서버가 필요하다는 문제점이 있다[5, 6]. 또 다른 연구에서는 local storage에 접근 시 사용자 인증을 위한 암호 입력 요구가 잦아 XSS 공격이 아닌 key-logger 등의 다른 공격에 노출될 수 있으며 편리성이 떨어지는 문제점이 있다[4, 7-9].

2.2 로컬 스토리지 보호 및 인증

이 논문에서는 local storage의 보안 취약점을 해결하면서 기밀성과 편리성을 높이기 위해 사용자가 이용하는 운영체제의 현재 사용자 정보와 기기 정보를 이용한다. 특히 기기 정보를 얻어오는 메인보드(MB)는 기판으로 구성된 컴퓨터의 핵심 부품 중 하나이다. 메인보드에는 BIOS라는 칩이 있는데 입출력을 담당하는 시스템이며 Basic Input Output System의 약자이다. 그리고 UUID(Universally Unique IDentifier)는 고유 식별자, SN(Serial Number)은 일련번호라는 뜻을 가지고 있다. 또한 많은 사용자가 이용하는 윈도즈 운영체제에서 registry의 현재 사용자 정보를 사용하여 암호화하므로 기기와 사용자를 인증할 수 있다[13]. 이로써 보안 취약점 문제를 해결하고 기밀성과 편리성을 높일 수 있도록 메인보드의 SN, BIOS SN, UUID 값과 운영체제의 현재 사용자 정보를 이용하여 암호화할 때 사용자 인증을 자동화할 수 있는 기법을 제안한다.

3. 사용자 인증 및 암호화 적용 설계

이 논문에서 제안하는 기법은 local storage에 저장될 평문 데이터를 암호화할 때 사용자로부터 암호 입력 없이 사용자 정보를 이용하여 자동화하는 방식이다. 사용자의 운영체제에서 로그인한 현재 사용자 값과 하드웨어의 고윳값을 얻어 암호화를 진행한다. 앞서 암호화된 사용자 정보와 평문 데이터를 한 번 더 다른 방식으로 암호화하여 인증 절차를 자동화하였다. 사용자 정보와 기기 정보를 같이 인증할 수 있어 데이터가 다른 기기로 이동되어도 안전성을 보장할 수 있다. 제안하는 기법은 (그림 1)과 같은 흐름으로 처리된다.

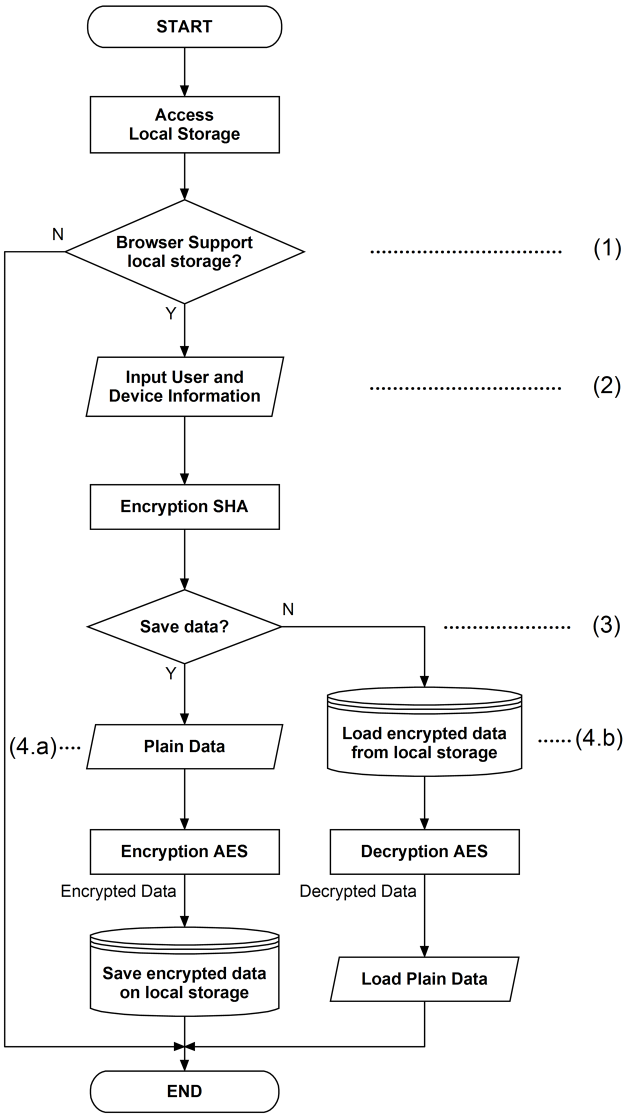
(1) 사용자의 인터넷 브라우저가 local storage를 지원하는지 확인한다.

(2) 저장과 불러오기를 할 때 사용자 정보를 가져와 암호화하는 과정은 같다. 사용자 정보들은 윈도즈 기준으로 운영체제에서 현재 로그인된 사용자 값을 Registry의 HKey_CURRENT_USER에 속해 있는 Identities\User ID에서 얻어오고 기기 값은 메인보드 내에서 BIOS SN, MB SN, UUID 값을 얻어온다. 가져온 값들은 SHA 256의 파라미터를 이용하여 암호화한다. 데이터를 저장하거나 불러올 때 인증할 수 있는 값으로 사용된다.

(3) 데이터를 저장하거나 혹은 불러오는지 확인한다. 이벤트 리스너의 상태에 따라 저장하거나 불러오기를 감지한다.

(4.a) 데이터를 저장할 경우 SHA로 암호화된 사용자 값과 평문 데이터 값을 불러와서 AES 암호화를 통해 파라미터로 사용하여 암호화한다. 두 번째 암호화가 모두 완료되면 local storage에 암호화된 데이터를 저장한다.

(4.b) 데이터를 불러올 경우 local storage에서 암호화된 데이터 값을 얻어오고 SHA로 암호화된 사용자 값으로 AES 복호화 한다.



(그림 1) 제안 기법 흐름도

기기에서 고윳값을 불러올 때 각각 기기마다 BIOS 및 MB의 SN 값이 있다. 그러나 제작사에 따라 제로 값이 있거나 비어 있는 기기가 존재하므로 UUID 값도 활용하여야 기기를 인증할 수 있다. 기기 정보를 얻어올 때 LAN과 같이 장비를 추가하거나 변경 가능한 기기의 정보 값을 가져오면 안 된다. 추가하거나 변경할 수 있는 장비를 사용하면 장비 이동에 따른 데이터 유실이나 유출 가능성이 있기 때문이다.

4 성능 평가

4.1 테스트 환경

본 논문에서 제안하고 있는 기법을 검증하기 위하여 구현과 성능 테스트를 하였다. 여기에서 사용된 컴퓨터 시스템은 <표 1>과 같으며 사용자가 접근하기 쉬운 환경을 고려하여 윈도즈 운영체제와 함께 포함된 IE(Internet Explorer)를 기준으로 구현을 진행하였다. 여기에서 사용된 암호화는 SJCL(Stanford Javascript Crypto Library)를 사용하여 암호화 및 복호화를 진행하였다[7, 8, 14].

<표 1> 시스템 사양

	System 1	System 2	System 3
CPU	Intel Pentium G630	Intel Core i5-4670	Intel Core i3-4160
RAM	8 GB	16 GB	4 GB
OS	Win7 64Bit		
Browser	IE11	IE 9	IE11

local Storage는 javascript를 이용하여 데이터에 접근하고 제어하므로 같은 스크립트를 이용해 구현하였다. 여기에서는 검증을 위한 구현이므로 실제 사용 가능한 형태의 배포 목적 구현에서는 코드를 최적화하여 개발해야 하며 javascript라는 점을 고려하여 보안상 전역 변수 사용하지 않는 등의 secure 코딩이 필요하다.

성능 평가를 진행하기 위해 제안하고 있는 방법과 기존 연구에서 패스워드 기반 키 유도 함수(PBKDF-2)로 구현된 내용을 토대로 비교하여 평가를 진행하였다. 추가로 PBKDF-2(SHA)와 AES을 사용하여 구현한 연구와 비교하였다. 사용자에게 별도의 암호 입력을 받아 정상 사용자를 판별하고 데이터 암·복호화를 수행하는 기존 연구 내용을 최대한 구현하였으며 SHA-1에 대한 collision이 발견되어 SHA-2 이상 사용할 것을 권고하고 있기 때문에 성능 평가를 시행한 모든 소스의 SHA는 SHA-2에 해당하는 SHA 256으로 변경하였다. 성능 평가를 진행하기 위해 <표 1>과 같은 기기를 사용하였다. 기존 연구에서의 성능 평가는 암호화되지 않은 상태와 비교 평가하였으므로 사용자에게 요구하는 암호 입력에 대한 시간을 제외하여 평가하였지만 본 논문에서는 정상 사용자와 정상 기기를 판별하므로 전체 수행 시간을 비교하여 평가해야 한다.

4.2 성능 결과

제안하는 기법을 시행한 결과 다양한 컴퓨터 시스템에서 구현된 소스 코드를 실행하였고 모두 정상 작동하였으며 기존 연구들과 비교하여 성능 측정을 진행하였다.

기존 연구에서 PBKDF-2를 사용하여 사용자에게 암호 입력을 받았을 때 암호의 길이에 따라 입력 시간이 증가하였으며 입력 시간이 ms를 넘으므로 <표 2, 3>에서 t라고 하였다. 반면 본 논문에서 제시하는 방법은 별도의 사용자 입력을 요구하지 않으므로 <표 2, 3>과같이 사용자가 암호 입력하는 시간(t) 만큼의 수행 시간 차이를 보였다.

<표 2> 단일 암호화와 자동화 비교 (단위:ms)

저장 시간 비교			
	사용자 인증	암호화	총 시간
자동화 방식	65.17	6.52	71.69
암호 입력 방식	사용자 암호 입력시간(t)	5.39	t + 5.39

불러오는 시간 비교			
	사용자 인증	암호화	총 시간
자동화 방식	64.35	5.97	70.30
암호 입력 방식	사용자 암호 입력시간(t)	4.82	t + 4.82

기존 연구들과 같이 암호화하여 저장하거나 저장 데이터를 불러올 때 암·복호화를 수행 시간만 측정하면 약 1ms의 시간 차이가 발생하였지만 큰 차이가 없었다. 오히려 본 연구에서의 암호화 방식이 SHA와 AES 이 두 가지 알고리즘을 사용하여 모듈 로드 시간이 추가된 것이다. 추가로 비교한 대상은 PBKDF-2(SHA)와 AES 두 가지 암호화 모듈이 사용되었고 성능 평가는 <표 3>과같이 큰 차이가 없었다. 앞서 비교한 것과 같이 PBKDF-2를 사용하여 사용자에게 요구한 암호 입력 시간(t) 만큼의 빠른 수행 시간을 나타낸다.

<표 3> 이중 암호화와 자동화 비교 (단위:ms)

저장 시간 비교			
	사용자 인증	암호화	총 시간
자동화 방식	66.51	6.59	73.10
암호 입력 방식	사용자 암호 입력시간(t)	6.42	t + 6.42

불러오는 시간 비교			
	사용자 인증	암호화	총 시간
자동화 방식	65.71	5.85	71.56
암호 입력 방식	사용자 암호 입력시간(t)	5.77	t + 5.77

전체적으로 암·복호화에 걸리는 수행 시간이 큰 차이를 보이지 않고 사용자에게 암호 입력을 요구하여 수행되는 시간과 운영체제의 사용자 정보와 기기 정보를 자동화하

여 수행되는 시간에서 큰 차이를 보였으며, 수행 시간의 차이만큼 더 빠르다. 성능 평가는 IE 환경에서 구현하여 검증한 것이므로 서로 다른 브라우저에서 구현할 때 코드가 달라질 수 있는데 Chrome과 같은 브라우저에서는 Native Message(메시지교환)나 NaCl(Native Client)의 PPAPI(Pepper API)를 이용하여 구현할 수도 있다[15-17]. 추후 WebAssembly를 이용하게 된다면 코드 변화 없이 처리 속도 향상과 함께 좋은 결과를 기대할 수 있을 것이다[18].

5. 결론

local storage에서 다루는 데이터들은 평문 형태로 만료기간 없이 저장되어 XSS 등의 공격으로부터 보안 취약점 문제가 있었다. 기존 연구에서 보안 취약점 문제를 해결하였으나 온라인에서만 암호화 사용이 가능하며 인증 서버의 설치가 필요한 문제점이 있었다. 다른 연구에서는 local storage 접근 시에 잦은 사용자 암호 요구에 불편함과 key-logger 등의 다른 방식의 공격에 취약점을 가지고 있다는 문제점이 있다.

이 논문에서는 local storage의 기능을 사용하면서 추가 절차 없이 사용자가 자연스럽게 사용하지만, 정보를 암호화하고 사용자와 기기 정보를 이용하여 인증 절차를 자동화하였다. 불필요한 인증 서버 없이 오프라인에서도 암호화가 가능하면서 잦은 사용자 암호 요구의 문제점까지 해결하여 편리성을 갖추었다. 성능 평가에서도 기존 연구들은 사용자에게 암호를 요구하여 입력될 때까지의 대기시간이 있고 그 시간 차이만큼 논문에서 제안하는 방법이 더 빠르다. 데이터를 암·복호화하여 저장과 불러오는 시간만 측정한다면 큰 차이 없이 비슷한 성능을 보였다.

암호화는 서버 측에서 많이 이루어지지만 많은 기업에서 정보의 중요성을 인지하고 각 클라이언트를 암호화하여 외부 유출을 막으려고 조처를 하고 있으며 개인 클라이언트에서도 사용자 정보는 중요하다.

추후 연구로서는 운영체제의 사용자 정보뿐만 아니라 브라우저의 고윳값이 있으면 브라우저 인증까지 활용한 암호화에 관한 연구가 필요하다.

참고문헌

[1] W3C Recommandation, "HTML 5.2", Available : https://www.w3.org/TR/html5/, Access : May 2019.

[2] W3C Recommandation, "Web Storage (Second Edition)", Available : https://www.w3.org/TR/webstorage/, Access : May 2019.

[3] OWASP(The Open Web Security Project), "Cross-site Scripting (XSS)", Available : http://www.owasp.org/index.php/Cross-site_Scripting_(XSS), Access : Jun 2019.

[4] 곽재권, “HTML5 기반 로컬스토리지의 보안 취약점 분석 및 암호화 알고리즘의 웹 브라우저별 성능평가에 관한 연구”, 석사학위논문, 고려대학교, 2017.

[5] 김지수, “A Study on Data Security of Web Local Storage”, 석사학위논문, 고려대학교, 2016.

[6] 김지수, 문종섭, “웹 로컬 스토리지 데이터 보안을 위한 연구”, 인터넷정보학회 논문지, 17권 3호, 55p ~ 66p, 2016.

[7] 명희원, “Study on implementation of secure HTML5 local storage”, 석사학위논문, 고려대학교, 2013.

[8] 명희원, 백정하, 이동훈, “안전한 HTML5 로컬스토리지 구현에 대한 연구”, 인터넷정보학회 논문지, 13권 4호, 83 ~ 93p, 2012.

[9] Meltem Sonmez Turan, Elaine Barker, William Burr, Lily Chen, "Recommendation for Password-Based Key Derivation Part 1 : Storage Applications”, NIST Special Publication 800-132, 2010.

[10] 이강근, “HTML5의 Web Storage 활용에 대한 연구 = A study on the use of the web storage HTML5”, 석사학위논문, 고려대학교, 2012.

[11] 이강근, 최진영, “HTML5의 Web Storage 활용에 대한 연구”, 한국컴퓨터종합학술대회 논문집, Vol 38, 430p ~ 433p, 2011.

[12] 이규석, “다중 디바이스에서 HTML5의 keygen과 Web Storage 기반의 2-factor 인증연구”, 석사학위논문, 고려대학교, 2013.

[13] Micosoft, "Microsoft Developer Network”, Available : https://msdn.microsoft.com/, Search : Registry API, Access : May 2019.

[14] Stanford University Cryptography Group, "Stanford Javascript Crypto Library”, Available : https://crypto.stanford.edu/sjcl/, Access : June 2019.

[15] Google Chrome, "Native Messaging”, Available : https://developer.chrome.com/extensions/nativeMessaging/, Access : June 2019.

[16] Google Chrome, "Welcome to Native Client”, Available : https://developer.chrome.com/native-client/, Access : Jun 2019.

[17] Google Chrome, "Pepper API Reference (Stable)”, Available : https://developer.chrome.com/native-client/pepper_stable/, Access : Jun 2019.

[18] WebAssembly, "WebAssembly”, Available : https://webassembly.org/, Access : Jun 2019.