Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.5, Oct. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.5.1047

패스워드 매니저의 패스워드 저장소 보안 취약점 분석

정 혜 라,† 소 재 우‡
서강대학교

Security of Password Vaults of Password Managers

Hyera Jeong,† Jaewoo So‡
Sogang University

요 약

웹사이트 이용이 증가하면서, 패스워드들을 암호화하여 데이터베이스에 저장 관리해주는 패스워드 매니저의 사용이 증가하고 있다. 브라우저 기반 패스워드 매니저와 로컬 기반 패스워드 매니저는 암호화된 데이터베이스를 로컬 컴퓨터에 저장한다. 웹 기반 패스워드 매니저는 암호화된 데이터베이스를 클라우드 서버에 저장하지만 사용자는 클라우드 서버에 접속하는데 사용하는 마스터 패스워드를 로컬 컴퓨터에 저장할 수 있다. 공격자가 사용자의 컴퓨터에서 패스워드 데이터베이스를 훔쳐 복호화에 성공한다면, 사용자의 모든 패스워드들이 노출되는 심각한 문제가 있다. 본 논문에서는 로컬 컴퓨터에 저장된 패스워드 저장소의 보안 취약점을 분석하는 절차를 제시하고, 패스워드 저장소를 공격하는 시나리오를 제시하며, 공격 프로그램을 개발하여 패스워드 저장소를 공격함으로써 패스워드 저장소의 보안 취약점을 확인한다.

ABSTRACT

As the number of services offered on the Internet exponentially increases, password managers are increasing popular applications that store several passwords in an encrypted database (or password vault). Browser-integrated password managers or locally-installed password managers store the password vault on the user's device. Although a web-based password manager stores the password vault on the cloud server, a user can store the master password used to sign in the cloud server on her device. An attacker that steals a user's encrypted vault stored in the victim's device can make an offline attack and, if successful, all the passwords in the vault will be exposed to the attacker. This paper investigates the vulnerability of the password vault stored in the device and develops attack programs to verify the vulnerability of the password vault.

Keywords: Password Manager, Password Vault, Vulnerability Analysis, Password Decryption, Password Attacks

I. 서 론

대부분의 웹사이트들은 로그인 과정을 통해 사용자의 아이디와 패스워드를 확인하여 웹 서비스를 제공한다. 따라서 사용자는 이용하는 웹사이트들이 많아질수록 기억해야 하는 패스워드들도 증가한다. 또한 웹사이트의 패스워드 해킹 사례가 증가하면서 패스워드 설정이 더욱 복잡해지고 있다. 이에 웹사이트별로 아이디와 패스워드들을 저장 관리해주는 패스워드 매니저 프로그램 사용이 크게 증가하고 있다[1].

패스워드 매니저는 사용자의 웹사이트별 아이디와 패스워드들을 암호화된 데이터베이스에 저장하고, 사용자는 하나의 마스터 패스워드만을 사용하여 해당 암호화된 데이터베이스에 접근한다. 따라서 사용자는

Received(06. 01. 2018), Modified(09. 19. 2018), Accepted(09. 20. 2018)
† 주저자, hrjeong@sogang.ac.kr
‡ 교신저자, jwso@sogang.ac.kr(Corresponding author)

웹사이트별 아이디와 패스워드들을 기억할 필요 없이 하나의 마스터 패스워드만을 기억하면 된다.

패스워드 매니저는 브라우저 기반 패스워드 매니저, 웹 기반 패스워드 매니저, 그리고 로컬 기반 패스워드 매니저로 구분할 수 있다. 브라우저 기반 패스워드 매니저는 크롬, 인터넷 익스플로러 등이 있으며, 브라우저가 패스워드들을 직접 저장 관리한다. 웹 기반 패스워드 매니저는 LastPass, Dashlane, 1Password, Roboform 등이 있으며, 사용자는 마스터 패스워드 하나만을 사용하여 클라우드 서버에 저장된 암호화된 데이터베이스에 접근한다. 그러나 암호화된 데이터베이스는 클라우드 서버에 저장되지만, 사용자는 클라우드 서버에 접속하는데 사용하는 마스터 패스워드를 로컬 컴퓨터에 저장할 수 있다. 로컬 기반 패스워드 매니저는 KeePass Password Safe가 있으며 데이터베이스를 로컬 컴퓨터에 저장한다.

패스워드 매니저가 안전하게 관리하여야 할 로컬 컴퓨터에 저장된 패스워드 저장소의 위치, 파일 포맷, 암호화 방식 등이 알려지면서 패스워드 저장소의 보안 취약점이 심각히 드러나고 있다[2-4]. 그러나 패스워드 저장소의 보안 취약점이 심각하지만, 체계적인 분석 절차가 미흡하고, 구체적인 공격 시나리오에 기반한 보안 취약점 연구가 부족하다.

본 논문에서는 패스워드 매니저가 로컬 컴퓨터에 저장한 패스워드 저장소의 보안 취약점을 체계적으로 분석하고, 패스워드 저장소를 공격하는 시나리오를 제시하며, 실제 공격 프로그램을 개발하여 패스워드 저장소를 공격하는 사례를 제시함으로써 패스워드 저장소의 보안 취약점에 대한 개선의 시급함을 알리고자 한다. 본 논문의 구성은 다음과 같다. 2장에서는 패스워드 매니저의 보안 취약점에 대한 기존 연구를 소개하고, 3장에서는 패스워드 저장소의 보안 취약점을 분석한다. 4장에서는 패스워드 저장소 공격 시나리오를 제시하고, 실제 공격 프로그램을 개발하여 패스워드 저장소를 공격하는 사례를 제시한다. 그리고 5장에서 결론을 맺는다.

II. 관련 연구

패스워드 저장소에 대한 보안 취약점이 연구되었다[2-4]. 논문 [2]는 11종의 패스워드 매니저에 대해 패스워드 저장소의 위치 및 포맷을 분석하여 패스워드 저장소의 포맷 취약점을 분석하였다. 논문 [3]과 [4]는 패스워드 저장소의 안정성을 위한 설계 지침을 제시하였다. IT 보안 그룹인 TeamSIK는 안드로이드 스마트폰에서 공격자가 샌드박스를 가로질러 로컬 앱 폴더에 접근하여 패스워드 저장소를 열람할 수 있는 패스워드 저장소의 취약점을 보고하였다[5, 6].

웹 기반의 패스워드 매니저는 사용자가 웹사이트를 방문하였을 때, 로그인 아이디와 패스워드를 자동으로 완성하거나 자동 로그인하는 기능을 제공한다. 이러한 자동 완성 또는 자동 로그인 기능은 웹 브라우저의 확장프로그램 방식으로 지원되거나 또는 북마크릿 기반으로 지원된다. 그러나 자바스크립트를 실행하는 북마크릿은 공격에 취약할 수밖에 없다. 논문[7]은 LastPass에서 북마크릿을 이용한 자동 로그인 과정을 분석하고 공격 기법을 제시하였다. 또한 자동 로그인은 iFrame 공격 또는 Sweep 공격을 통해 패스워드를 가로챌 수 있다는 취약점이 지적되었다[8, 9]. 패스워드 매니저는 웹에서 동작하므로 웹기반 호스트 공격을 통해 패스워드를 가로챌 수 있는 문제점이 지적되었고[10], CSRF(Cross-Site Request Forgery)와 XSS(Cross-Site Scripting) 취약점이 지적되었다[7, 11].

공격자가 사용자의 컴퓨터에서 패스워드 저장 파일을 탈취하여 암호화된 데이터베이스를 열람할 수 있다면, 사용자의 모든 웹사이트별 아이디와 패스워드가 공격자에게 노출될 것이다. 그러므로 본 논문에서는 패스워드 저장소의 보안 취약점을 연구하였다. 기존의 패스워드 저장소에 대한 보안 취약점 연구는 개별적인 분석으로 체계적인 분석 절차가 부족하였고, 공격 시나리오가 제시되어 있지 않으며, 공격 프로그램 개발을 통한 검증이 부족하였다. 본 논문에서는 패스워드 저장소의 분석 절차, 공격 시나리오, 공격 프로그램 개발을 통해 패스워드 저장소의 보안 취약점을 구체적으로 제시한다.

III. 패스워드 저장소 보안 취약점 분석

3.1 패스워드 저장소 보안 취약점 분석 절차

브라우저 기반 패스워드 매니저와 로컬 기반 패스워드 매니저는 사용자의 웹사이트별 접속 아이디와 패스워드 정보가 담긴 데이터베이스를 로컬 컴퓨터에 저장한다. 그리고 웹 기반 패스워드 매니저는 데이터베이스를 클라우드 서버에 저장하지만, 사용자는 클

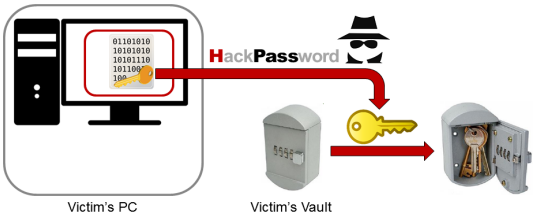

Fig. 1. Security of Password Vaults

라우드 서버에 접속하는데 사용하는 마스터 패스워드를 로컬 컴퓨터에 저장할 수 있다. Fig. 1과 같이 공격자가 사용자 컴퓨터에서 패스워드 데이터베이스 파일을 탈취하여 암호화된 데이터베이스를 열람할 수 있다면, 사용자의 모든 웹사이트별 아이디와 패스워드가 공격자에게 노출될 것이다.

패스워드 저장소 보안 취약점을 분석하기 위한 절차로 Table 1과 같이 4단계 분석 절차를 제안한다. 제안하는 분석 절차는 각 단계의 첫 자를 사용하여 LoFE(LOcation, Format, Encryption) 분석(Analysis)이라고 지칭한다. 1단계는 패스워드 저장소가 저장된 파일 경로를 파악하고, 2단계는 패스워드 저장소의 파일 포맷을 분석한다. 3단계는 패스워드 저장소의 암호화 방식과 복호화 방법을 분석한다. 4단계는 공격 시나리오에 따른 보안 취약성을 확인한다.

Table 1. Vunerability analysis of password vault

Step 1	**Lo**cation of a password vault
Step 2	**F**ormat of a password vault
Step 3	**E**ncryption and decryption algorithm
Step 4	Vulnerability Analysis

3.2 브라우저 기반 패스워드 매니저의 보안 취약점 분석

3.2.1 크롬 브라우저 패스워드 매니저의 동작

크롬, 인터넷 익스플로러, 파이어폭스, 사파리 등의 브라우저는 자체적으로 사용자의 웹사이트별 아이디와 패스워드 정보를 저장 관리한다. 본 논문에서는 가장 널리 사용하고 있는 크롬 브라우저의 패스워드 저장소 보안 취약점을 분석한다.

3.2.2 크롬 브라우저의 패스워드 저장소 위치 및 포맷

크롬 브라우저는 사용자의 웹사이트별 아이디와 패스워드 정보를 Table 2에 보이는 파일 경로에 "Login Data" 라는 파일 이름으로 저장한다. 파일 포맷은 SQLite 데이터베이스 형식이다.

크롬의 패스워드 데이터베이스 파일을 "DB Browser for SQLite" 프로그램[12]을 사용하여 열면 Fig. 2와 같이 웹사이트별 아이디와 패스워드 관련 정보를 확인할 수 있다. 여기서 웹사이트 주소 및 아이디는 암호화되어 있지 않아 정보가 모두 노출됨을 확인할 수 있다. 패스워드는 암호화되어 있어 BLOB(Binary Large OBject)로 표시된다.

Table 2. Location and format of Chrome password vault

Location	⟨user profile dir⟩\AppData\Local\Google\Chrome\User Data\Default\
Format	SQLite format 3

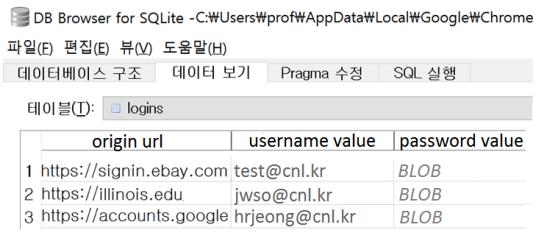

Fig. 2. Format of Chrome password vault

3.2.3 크롬 브라우저의 패스워드 암호화 및 복호화

크롬 브라우저는 웹사이트 로그인에 사용하는 패스워드를 윈도우즈에서 제공하는 DPAPI(Data Protection Application Programming Interface) 함수를 사용하여 암호화하여 저장한다. 따라서 Fig. 2의 패스워드 필드에 "BLOB"로 표시된다[13]. 윈도우즈에서 제공하는 DPAPI 암호화 함수는 `CryptProtectData()` 이고, 복호화 함수는 `CryptUnprotectData()` 이다. DPAPI는 데이터를 암호화하기 위해 세 가지 키를 순차적으로 생성한다. 먼저, 윈도우즈 계정 패스워드를 사용하여 "Pre Key"를 만들고, "Pre Key"를 사용하여 "Master Key"를 생성하며, "Master Key"를 사용하여 BLOB를 암호화하기 위해 사용할 "Blob Key"를 생

성한다[14]. 키 생성은 PBKDF2 알고리듬을 사용하고, 대칭키 암호화는 윈도우 7부터 AES-256 방식을 사용한다. 해쉬 함수는 SHA512 방식을 사용한다. BLOB를 복호화할 때는 윈도우즈 계정 패스워드를 사용하여 "Master Key"가 있는 금고를 해제하여 BLOB의 Master GUID 정보에 해당하는 "Master Key"를 추출하고, 해당 "Master Key"를 사용하여 Blob Key를 생성하여 BLOB를 복호화한다. Fig. 3은 BLOB 데이터를 보여주고 있으며, BLOB 데이터는 Provider GUID 정보와 Master GUID 정보를 포함하고 있다. GUID 정보의 위치를 Table 3에 표기하였다.

윈도우즈 DPAPI 함수를 사용하여 데이터를 암호화한 경우, 암호화가 수행된 동일한 컴퓨터에서 복호화를 수행하는 경우라면 `CryptUnprotectData()` 함수를 사용하여 쉽게 복호화할 수 있지만, 다른 컴퓨터에서 복호화를 수행하려면 윈도우즈 계정 패스워드 및 GUID 정보를 알아야 한다.

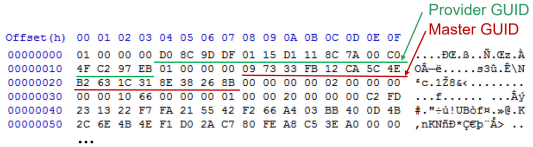

Fig. 3. BLOB HEX data

Table 3. Location of provider and master GUIDs

Provider GUID	HKEY_LOCAL_MACHINE\Software\Microsoft\Cryptography\Protect\Providers
Master GUID	⟨user profile dir⟩\AppData\Roaming\Microsoft\Protect

3.3 웹 기반 패스워드 매니저의 보안 취약점 분석

3.3.1 LastPass 패스워드 매니저의 동작

웹 기반의 패스워드 매니저는 LastPass, Dashlane, 1Password, Roboform 등과 같은 상용 프로그램이 있다. 본 논문에서는 널리 사용하고 있는 LastPass 프로그램의 패스워드 저장소 취약점을 분석한다.

웹 기반의 패스워드 매니저는 사용자의 웹사이트별 아이디와 패스워드를 클라우드 서버에 저장하지만, 사용자는 클라우드 서버에 접속하는데 사용하는 마스터 패스워드를 로컬 컴퓨터에 저장할 수 있다. 크롬 확장 프로그램으로 LastPass 프로그램을 설치하고, Fig. 4의 LastPass 로그인 화면에서 "암호 기억" 옵션을 선택하면 로컬 컴퓨터에 사용자의 마스터 패스워드가 암호화되어 저장된다. 이후 사용자는 웹 브라우저를 활성화할 때마다 클라우드 서버 접속을 위해 마스터 패스워드를 매번 입력해야 하는 번거로움을 피할 수 있다.

LastPass 프로그램의 동작은 Fig. 5와 같다. 사용자는 클라우드 접속을 위한 전자메일과 마스터 패스워드를 입력하고 클라우드 서버에 있는 데이터베이스에

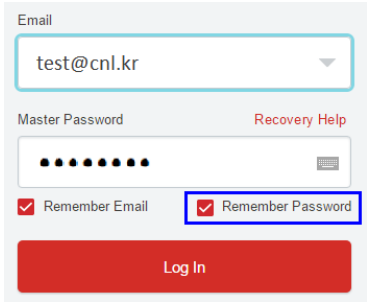

Fig. 4. Login option of LastPass

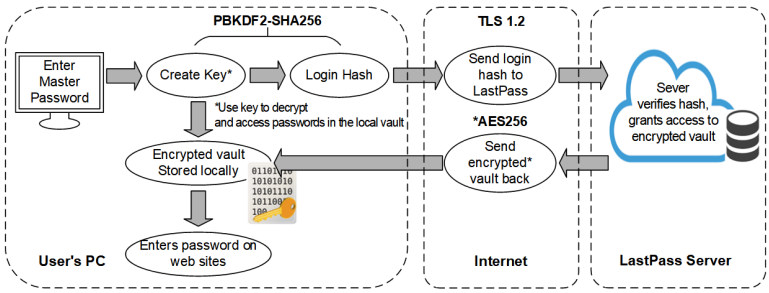

Fig. 5. Overall operation of LastPass

접근하여 웹사이트별 아이디와 패스워드를 가져온다.

3.3.2 LastPass의 패스워드 저장소 위치 및 포맷

LastPass는 클라우드 서버 로그인 정보를 Table 4와 같이 파일 경로에 파일 이름을 숫자로하여 저장한다. 파일 포맷은 SQLite 데이터베이스 형식이다.

LastPass의 마스터 패스워드 저장 파일을 "DB Broswer for SQLite" 프로그램을 사용하여 열어보면 Fig. 6과 같이 클라우드 서버 로그인 계정 정보를 확인할 수 있다.

Table 4. Location and format of LastPass password vault

Location	• Chrome Browser ⟨user profile dir⟩\AppData\Local\Google\Chrome\User Data\Default\databases\chrome-extension_hdokiejnpimakedhajhdlcegeplioagd_0\ • IE Browser ⟨user profile dir⟩\AppData\LocalLow\LastPass • FireFox Browser ⟨user profile dir⟩\AppData\Local Settings\Applicaion Data\LastPass
Format	SQLite format 3

Fig. 6. Format of LastPass password vault

3.3.3 LastPass의 패스워드 암호화 및 복호화

LastPass는 마스터 패스워드를 AES-256 CBC 모드 또는 ECB 모드로 저장한다[15]. 따라서 마스터 패스워드 복호화는 Fig. 7과 같다. 여기서 AES에서 사용하는 Key는 Fig. 6의 사용자 이름 필드에 있는 것을 SHA-256 해쉬한 것이다.

Fig. 6의 AES-CBC로 암호화된 마스터 패스워드를 복호화하는 방법을 Fig. 8에 도시하였다. 암호화된 패스워드는 길이가 50자이며, '|'를 구분자로 앞에서 '!'를 제외한 24자는 IV(Intialization Vector)로 사용되고, 뒤의 24자는 마스터 패스워드의 암호문이다.

```
AES-CBC(encryptedPassword, Key, IV)
or AES-ECB(encryptedPassword, Key)
```

Fig. 7. Decryption of LastPass master password

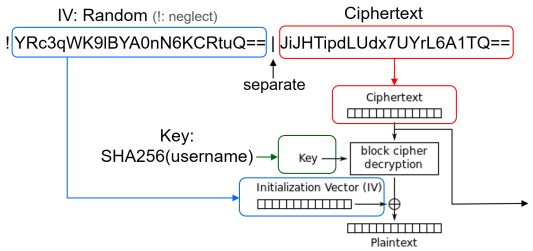

Fig. 8. Decryption of AES-CBC encrypted password

3.4 로컬 기반 패스워드 매니저의 보안 취약점 분석

3.4.1 KeePass 패스워드 매니저의 동작

로컬 기반 패스워드 매니저는 KeePass Password Safe(이하 KeePass)가 있으며, 사용자의 웹사이트별 아이디와 패스워드들을 암호화하여 로컬 컴퓨터에 저장한다[16]. 본 논문에서는 프리웨어로 널리 사용하고 있는 KeePass 프로그램의 패스워드 저장소 취약점을 분석한다.

3.4.2 KeePass 패스워드 저장소 위치 및 포맷

KeePass는 프로그램 설치시 사용자가 직접 패스워드 저장소의 파일 경로 및 파일 이름을 설정할 수 있다. 그리고 해당 데이터베이스 파일 포맷은 KeePass 자체 규격으로 1.x 버전은 ".kdb" 확장자를 가지며, 2.x 버전 이상은 "*.kdbx" 확장자를 가진다. 본 논문에서는 KeePass 2.x를 기준으로 패스워드 저장소 파일을 분석한다.

KeePass 2.x 데이터베이스 파일 포맷은 KeePass 버전을 알려주는 8 바이트 "File Signature", 데이터베이스 암호화 방식 및 정보를 담고 있는 암호화되지 않은 "Header", 그리고 사용

자의 웹사이트별 계정 정보를 담고 있는 암호화된 XML 형식의 "Body"로 구성된다. Header는 Type(1 바이트)-Length(2 바이트)-Value 형식으로 Table 5와 같은 항목들을 저장한다.

Table 5. KeePass database fields

Header	• End of Header, Comment, Cipher ID, Compression Flags • Master Seed, Transform Seed, Tramsform Rounds for Key Generation • Encryption IV, Protected Stream Key, SartBytes, Inner Random Stream ID for Encryption
Body	• StoredStartBytes • Encrypted ContentBlock

3.4.3 KeePass의 패스워드 암호화 및 복호화

KeePass는 AES-256 또는 ChaCha20 알고리듬을 사용하여 데이터베이스를 전체적으로 암호화하여 저장한다. 암호화 알고리듬에 사용하는 Key 생성 알고리듬은 AES-KDF와 SHA-256을 사용한다. 자세한 Key 생성 과정을 Fig. 9에 도시하였다.

KeePass는 사용자가 데이터베이스에 접근할 때, 마스터 패스워드 검증 절차를 가진다. KeePass는 보안을 위해 데이터베이스에 마스터 패스워드 암호문을 저장하지 않고, Fig. 10에서와 같이 Header에 있는 StartBytes 값을 암호화하여 Body의 StoredStartBytes 필드에 저장한다. 여기서, StartBytes와 Encryption IV는 Header에 저장된 값을 사용한다. 그리고 사용자가 데이터베이스를 열려고 할 때, 사용자가 입력한 마스터 패스워드를 사용하여 Key를 생성하고, Body의 StoredStartBytes 값을 복호화하여 Header의 StartBytes와 비교하여 사용자가 입력한 마스터 패스워드를 검증한다.

KeePass의 암호화된 데이터베이스를 복호화하기 위해서는 마스터 패스워드가 필요하다. 그러나 데이터베이스는 마스터 패스워드를 추출할 수 있는 직접적인 정보를 담고 있지 않으므로, KeePass의 데이터베이스를 복호화할 수 있는 방법은 무차별 대입 공격(Brute force attack) 이외에는 찾기 어렵다.

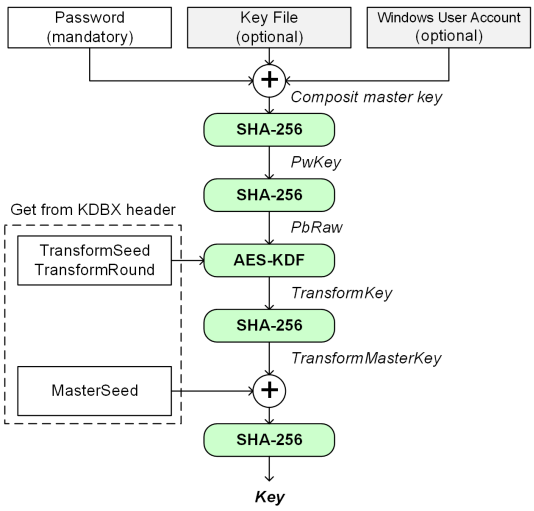

Fig. 9. KeePass Key generation

StoredStartBytes
= AES-CBC(StartBytes, Key, Encryption IV)

Fig. 10. Encryption for the verification of user's master password

IV. 패스워드 저장소 공격 시나리오 및 프로그램 개발

4.1 브라우저 기반 패스워드 매니저 공격

4.1.1 크롬 브라우저 패스워드 저장소 공격 시나리오

Fig. 11과 같이 공격자(Attacker)는 피공격자(Victim)의 컴퓨터에 공격 프로그램을 설치한다. 공격 프로그램은 피공격자의 컴퓨터에서 Table 2에 기술한 크롬 브라우저의 패스워드 저장소 파일을 복호화하여 사용자의 웹사이트별 아이디와 패스워드 정보를 공격자의 컴퓨터로 전송한다.

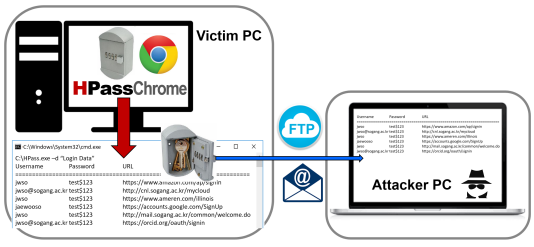

Fig. 11. Attack scenario on a Chrome password vault

4.1.2 크롬 브라우저 패스워드 공격 프로그램 개발

본 논문은 크롬 브라우저의 패스워드 저장소 보안 취약점을 확인하기 위해 패스워드 저장소를 공격하는 프로그램을 개발하였으며, 프로그램을 "HPass Chrome"이라고 지칭한다. 개발한 프로그램의 동작 순서를 Fig. 12에 도시하였다.

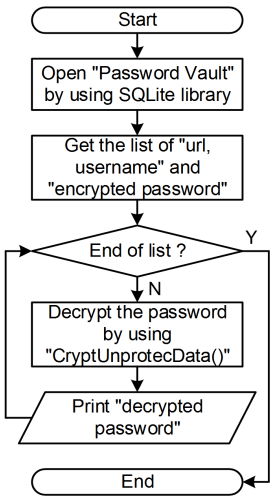

Fig. 12. Flow chart of the HPassChrome

```
void main(int argc, char *argv[])
{
   char *query = "SELECT origin_url, username_value,
                          password_value FROM logins";
   …
   sqlite3_open(databasePath, &db);
   sqlite3_prepare_v2(db, query, -1, &stmt, 0)
   …
   while (sqlite3_step(stmt) == SQLITE_ROW) {
      url = (char *)sqlite3_column_text(stmt, 0);
      username = (char *)sqlite3_column_text(stmt, 1);
      password = (BYTE *)sqlite3_column_text(stmt, 2);
      …
      DATA_BLOB cipher, plain;
      cipher.pbData = password;
      …
      CryptUnprotectData(&cipher, NULL, NULL, NULL, NULL, 0, &plain);
      for (i=0; i<plain.cbData; i++)
         decryptedPassword[i] = plain.pbData[i];
      …
      printf("%s\t%s\t%s\n", username, decryptedPassword, url);
   }
   …
   sqlite3_close(db);
}
```

Fig. 13. C++ pseudo code of HPassChrome

Fig. 13은 개발한 HPassChrome 프로그램의 C++ 코드 일부를 보여주고 있다. HPassChrome 프로그램은 SQLite 라이브러리를 사용하여 Table 2에 기술한 크롬 브라우저의 패스워드 저장소 파일을 열어, "logins" 테이블로부터 "웹사이트, 아이디, 패스워드" 데이터를 순차적으로 추출하고, `CryptUnprotectData()` 함수를 사용하여 암호화된 패스워드를 복호화한다.

개발한 HPassChrome 프로그램을 사용하여 크롬 브라우저의 패스워드 저장소를 복호화한 사례를 Fig. 14에 도시하였다. HPassChrome은 크롬 브라우저의 패스워드 저장소를 자동으로 찾아 사용자의 "아이디, 패스워드, 웹사이트" 데이터를 순차적으로 보여준다.

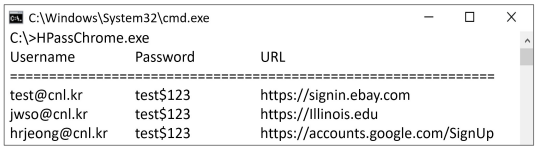

Fig. 14. Execution result screen of the HPassChrome

4.2 웹 기반 패스워드 매니저 공격

4.2.1 LastPass 패스워드 저장소 공격 시나리오

Fig. 15와 같이 공격자는 피공격자의 컴퓨터에서 Table 5에 기술한 LastPass 패스워드 저장소 파일을 훔친다. 공격자는 LastPass 패스워드 저장소 파일을 복호화하여 클라우드 서버 로그인 계정 정보를 얻고, 클라우드 서버에 접속하여 피공격자의 웹사이트별 아이디와 패스워드 정보를 열람한다.

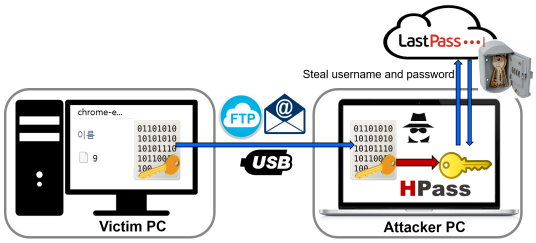

Fig. 15. Attack scenario on a LastPass password vault

4.2.2 LastPass 패스워드 저장소 공격 프로그램 개발

본 논문은 LastPass의 패스워드 저장소 보안 취약점을 확인하기 위해 패스워드 저장소를 공격하는 프로그램을 개발하였으며, 프로그램을 "HPass"라고 지칭한다. 개발한 프로그램의 동작 순서를 Fig. 16에 도시하였고, 개발한 C# 코드 일부를 Fig. 17에 도시하였다. HPass 프로그램은 SQLite 라이브러리를 사용하여 Table 5에 기술한 LastPass 패스워드 저장소 파일을 열어, "LastPassSavedLogins2" 테이블로부터 "username, password" 데이터를 추출하고, C# AES 라이브러리를 사용하여 패스워드를 복호화한다.

개발한 HPass 프로그램을 사용하여 LastPass 패스워드 저장소 파일을 복호화한 사례를 Fig. 18에 도시하였다.

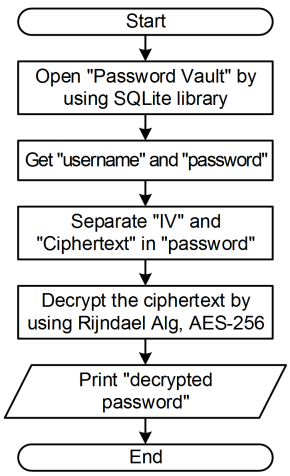

Fig. 16. Flow chart of the HPass

```
public void DecryptPassword(string filePath)
{
    SQLiteConnection sqlite = new SQLiteConnection(
                        "Data Source=" + filePath + ";Version=3;");
    sqlite.Open();
    string sql = "SELECT username, password
                  FROM LastPassSavedLogins2";
    SQLiteCommand comm = new SQLiteCommand(sql, sqlite);
    SQLiteDataReader reader = comm.ExecuteReader();
    if (reader.Read()) {
        username = (string)reader["username"];
        password = (string)reader["password"];
    }
    sqlite.Close();

    string IV = password.Substring(1, 24);
    string CText = password.Substring(26, 24);
    SHA256 shaM = new SHA256Managed();
    byte[] bytes = Encoding.ASCII.GetBytes(username);
    byte[] Key = shaM.ComputeHash(bytes);
    this.decryptedPassword = AES_CBC(CText, Key, IV);
}
```

Fig. 17. C# pseudo code of HPass

Fig. 18. Execution result screen of the HPass

4.3 로컬 기반 패스워드 매니저 공격

4.3.1 KeePass 패스워드 저장소 공격 시나리오

공격자는 피공격자의 컴퓨터에서 KeePass 데이터베이스 파일을 훔친다. 공격자는 기존에 알려진 공격 프로그램을 사용하여 KeePass 데이터베이스 복호화를 시도한다. 그러나 KeePass 데이터베이스는 마스터 패스워드에 대한 직접적인 정보를 저장하지 않으므로 마스터 패스워드를 직접적으로 찾을 수 없다. 따라서 현재까지는 KeePass 데이터베이스를 복호화하기 위해서는 무차별 대입 공격(Brute force attack)만이 가능하다.

4.3.2 KeePass 패스워드 저장소 공격

공격자는 KeePass 데이터베이스로부터 해쉬 값을 추출한다. 이때 해쉬 값을 추출하기 위해 "keepass2john.py" 프로그램[18]을 이용할 수 있다. Fig. 19는 KeePass 데이터베이스로부터 해쉬 값을 추출하는 화면이다.

Fig. 19. Extraction of a crackable hash from a KeePass database

공격자는 추출한 해쉬 값을 사용하여 무차별 대입 공격을 통해 데이터베이스 마스터 패스워드를 알아낸다. 그러나 256 비트 범위의 키를 무차별 대입 공격을 통해 해독하기 위해서는 2^{256} 가지의 범위를 조사하여야 하므로 실제 환경에서는 거의 불가능하다[17]. 그러나 사전에서 사용하는 단어를 조합하여 짧은 길이의 마스터 패스워드를 만든 경우 "hashcat" 프로그램[19]에서 사전 공격(Dictionary attack)을 옵션을 사용함으로써 해독이 가능하다. Fig. 20은 hashcat 프로그램을 사용하여 무차별 대입 공격을 실행한 화면이다. 그리고 Fig. 21은 "libkeepass" 프로그램[20]을 사용하여 KeePass 데이터베이스를 해독하여 사용자의 웹사이트별 아이디와 패스워드가 포함된 XML 데이터를 추출하는 화면이다.

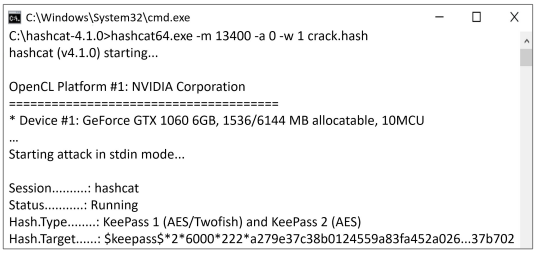

Fig. 20. Brute force attack by using the hashcat

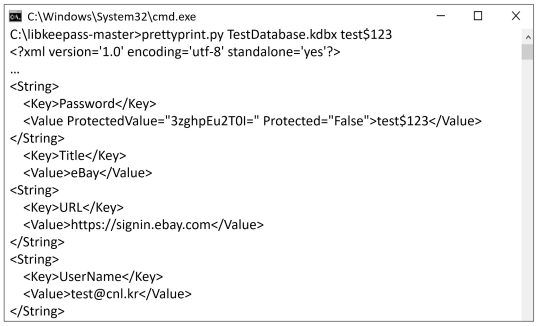

Fig. 21. Decryption of a KeePass database

V. 결 론

본 논문에서는 패스워드 매니저 프로그램을 브라우저 기반 패스워드 매니저, 웹 기반 패스워드 매니저, 로컬 기반 패스워드 매니저로 분류하였고, 각 분류별 패스워드 저장소의 보안 취약점을 분석하고, 실제 공격 시나리오를 제시하였으며, 이를 확인하기 위해 패스워드 저장소를 공격하는 HPass 프로그램을 개발하였다.

먼저, 패스워드 저장소의 보안 취약점을 분석하기 위한 4단계 절차를 제안하였고, 이를 LoFE Analysis(Location, Format, Encryption and decryption, Vulnerability Analysis)라고 지칭하였다. 그리고 분류별 대표적인 패스워드 매니저 프

Table 6. Vulnerability Comparison of Password Managers

	Chrome	LastPass	KeePass
Strength	User's passwords are encrypted with the Windows per-user DPAPI and therefore the protected passwords are hard to be decrypted by attackers or on other machines.	User's vault is protected using 256-bit AES encryption based on the user-created master password.	KeePass encrypts the whole database, i.e., not only passwords, but also user names, URLs, etc., where the key for the encryption is generated using SHA-256 from the user-crated master password.
Weakness	User's encrypted passwords can be easily decrypted on the machine the user logged in.	User-created master password can be easily found if the user enables the policy "Remember Password".	If the user forgets the master password, he will lost all the data.
Assumption or limitation of attacks	An attacker needs to access the victim's computer.	An attacker needs to steal the DB file located at the 〈user profile dir〉 of the victim's computer.	Brute-force attacks is required.

로그램을 선정하여 패스워드 저장소, 데이터베이스 포맷, 암호화 및 복호화 방식을 분석하였으며, 패스워드를 알아내는 공격 시나리오를 제시하였다. Table 6은 패스워드 취약점 관점에서 패스워드 매니저들의 장단점을 분석하고, 패스워드 공격 시나리오의 가정 및 한계를 정리하였다. 대표적인 브라우저 기반 패스워드 매니저인 크롬 브라우저는 윈도우즈 DPAPI 함수를 사용하여 패스워드를 암호화하여 저장하므로, 다른 컴퓨터에서는 복호화가 어려운 장점이 있다. 그러나 암호화가 수행된 동일 컴퓨터에서는 공격자가 쉽게 패스워드를 복호화할 수 있고, 아이디 및 웹사이트 주소 등은 암호화되지 않은 상태로 저장되므로 보안이 취약한 단점이 있다. 따라서 크롬 브라우저에서 패스워드 공격이 가능하기 위해서는 동일 컴퓨터에 접근할 수 있거나 윈도우즈 계정 패스워드가 필요한 한계점이 있다. 웹 기반 패스워드 매니저인 LastPass는 사용자가 만든 마스터 패스워드를 사용하여 AES-256 암호화를 수행한다는 장점이 있다. 그러나 사용자가 LastPass 로그인 화면에서 "암호 기억" 옵션을 선택하면, 공격자가 쉽게 사용자의 마스터 패스워드를 알아낼 수 있다는 단점이 있다. 이때 공격자는 사용자의 컴퓨터에서 마스터 패스워드 저장 파일을 탈취하기만 하면 된다. 로컬 기반 패스워드 매니저인 KeePass는 패스워드 저장소에 마스터 패스워드의 해쉬 값만을 저장하므로 보안이 뛰어나다. 그러나 사용자가 마스터 패스워드를 잊어버린 경우 저장된 모든 데이터들을 잃어버리는 단점이 있다. 본 논문에서 제안한 패스워드 저장소 분석 절차 및 공격 시나리오는 패스워드 저장소의 보안 중요성을 일깨우고 패스워드 매니저를 개선하는데 기초 연구 자료로 활용할 수 있을 것이다.

References

[1] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in Proc. IEEE Symposium on Security and Privacy, pp. 553-567, May 2012.

[2] P. Gasti and K. B. Rasmussen, "On the security of password manager database formats," in Proc. European Symposium on Research in Computer Security, pp. 770-787, Sep. 2012.

[3] M. Golla, B. Beuscher, and M. Dürmuth, "On the security of cracking-resistant password vaults," in Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 1230-1241, Oct. 2016.

[4] R. Chatterjee, J. Bonneau, A. Juels, and T. Ristenpart, "Cracking-resistant password vaults using natural language encoders," in Proc. IEEE Symposium on Security and Privacy, pp. 481-498, May 2015.

[5] S. Huber, S. Arzt, and S. Rasthofer, "Extracting all your secrets: Vulnerabilities in android password managers," Hack In The Box Security Conference, pp. 1-50, Apr. 2017.

[6] S. Huber, S. Rasthofer, and S. Arzt, "Bypassing android password manager apps without root," DEF CON, pp. 1-58, Jul. 2017.

[7] Z. Li, W. He, D. Akhawe, and D. Song, "The emperor's new password manager: Security analysis of web-based password managers," in Proc. USENIX Security Symposium, pp. 465-479, Aug. 2014.

[8] D. Silver, S. Jana, D. Boneh, E. Chen, and C. Jackson, "Password managers: Attacks and defenses," in Proc. USENIX Security Symposium, pp. 449-464, Aug. 2014.

[9] M. Blanchou and P. Youn, "Password managers: Exposing passwords everywhere," Whitepaper, iSEC partners, pp. 1-6, Nov. 2013.

[10] K. Bhargavan and A. Delignat-Lavaud, "Web-based attacks on host-proof encrypted storage," in Proc. USENIX Workshop on Offensive Technologies, pp. 1-8, Aug. 2012.

[11] X. Li and Y. Xue, "A survey on server-side approaches to securing web applications," ACM Computing Surveys, vol. 46, no. 4, pp. 1-29, Apr. 2014.
[12] SQLite, "DB Browser for SQLite," http://sqlitebrowser.org, Mar. 2018.
[13] NAI Labs, "Windows Data Protection", https://msdn.microsoft.com/en-us/library/ms995355.aspx, Oct. 2001.
[14] E. Burzstein and J. M Picod, "Recovering windows secrets and EFS certificates offline," in Proc. USENIX Workshop on Offensive Technologies, pp. 1-9, Aug. 2010.
[15] LastPass, "Technical Whitepaper," http://enterprise.lastpass.com, pp. 1-20, Mar. 2018.
[16] KeePass Password Safe, "KeePass Password Safe," https://keepass.info, Apr. 2018.
[17] H. Zhang, J. Hong, and J. Hu, "Analysis of encryption mechanism in KeePass Password Safe 2.30," in Proc. IEEE International Conference on ASID, pp. 1-4, Sep. 2016.
[18] John the Ripper suite, "keepass2john.py - Python module to extract a hash from KeePass databases," https://gist.github.com, Apr. 2018.
[19] Hashcat, "hashcat - advanced password recovery," http://hashcat.net/hashcat, Apr. 2018.
[20] Python module to read KeePass, "libkeepass - Python module to read KeePass files," available from https://gist.github.com, Apr. 2018.

〈저자소개〉

정 혜 라 (Hyera Jeong) 학생회원
2017년 2월: 연세대학교 미래교육원 소프트웨어개발학과 졸업
2017년 9월~현재: 서강대학교 정보통신대학원 석사과정
〈관심분야〉 암호 프로토콜, 블록체인 보안 취약점 분석, 인증 및 키 교환

소 재 우 (Jaewoo So) 정회원
1997년 2월: 연세대학교 전자공학과 졸업
1999년 2월: 한국과학기술원 전기 및 전자공학과 석사
2002년 8월: 한국과학기술원 전기 및 전자공학과 박사
2001년~2005년: ㈜아이피원 수석연구원
2005년~2007년: ㈜삼성전자 책임연구원
2007년~2008년: Stanford University 전기공학과 박사후연구원
2008년 9월~현재: 서강대학교 전자공학과 교수
2014년~2015년: UIUC ECE 방문교수
2014년~2017년: 한국연구재단 ICT·융합연구단 전문위원
〈관심분야〉 무선 보안, 블록체인 보안, 머신러닝, 5G 커넥티비티