CS'14-Zhaorui-Toward a Secure and Usable Cloud-based password manager for web browsers

概述：(S)当前最流行的5个浏览器内置的口令管理器没有对存储的口令做足够的保护，例如未使用mpw，采用的加密密钥可能被攻击获取等。(T)本文旨在揭示当前浏览器内置的口令管理器存在的安全问题，特别是攻击可以通过drive-by download等方式在用户计算机中偷偷安装恶意软件来获取口令，并且设计了一个新的不需要在本地存储的Cloud-based Storage-Free BPM(CSF-BPM)。(A)本文首先通过源代码分析、二进制文件分析和实验确定5个BPM的加密方式，并通过设计攻击获取key进而解密得到用户存储在本地的口令，设计了CSF-BPM，将用户数据存储在云端而非本地，用户需记忆SSMP并在SRS服务中设置一个帐户，生成key对存储的数据加密，并在后续登录时解密且实现autofill。(R)最后分析了方案的正确性、性能和可用性(用户调研)，相比于传统的BPM，用户更愿意使用CSF-BPM，最终给出了本文方案的局限性。

优点：①指出了5个流行浏览器BPM存在的问题。②提出了CSF-BPM，给出了详细的设计实现和评估

问题：①用户在无法访问云端服务器时将无法获取到口令。

②SSMP以及SRS帐户只提供1次，如何保证用户安全锁定或登出应付说。

③相比于BPM，本文方案需引入可信实体SRS会增加开销

---

1. 研究Browser-based Password Manager (BPM)，包括IE, Firefox, Chrome, Safari & Opera.

1> BPM的安全性分析(Threat Model and Assumptions)

- Threat Model：攻击者可以使用常见的攻击手段如"drive-by downloads"在用户的机器上偷偷安装恶意软件。→ 本文关注于BPM的漏洞可用于解密BPM存储的口令。

- Assumptions：

① 攻击者不能使用恶意软件进一步通过攻击OS等方式获取口令。

② 假设DNS是安全可靠的

③ 不考虑XSS和CSRF攻击(有各自的威胁模型和假设)。

2. 漏洞及安全问题

1> 在很多情况下，BPM的存储的加密口令不够安全

① 若无mpw保护，攻击者可以通过登录用户在网站的帐户解密口令(用于加密或解密的密钥可轻易被攻击获取或生成

② 若有mpw保护，攻击者可通过暴力攻击或钓鱼攻击获取mpw。

→ 用户应选择强的mpw，且BPM应提供足够的保护。

→ 这些漏洞可以被攻击者通过drive-by downloads攻击利用。

2> Chrome、IE、Safari、Opera未提供mpw，Firefox提供mpw，但非强制

3> 通过源代码文件分析，二进制文件分析和实验确定加密方式。

① 无mpw：Firefox使用3DES加密，

其它使用Windows自带的CryptProtectData和CryptUnprotectData执行加密和解密

除Firefox的另外4个浏览器可以设置dwFlags参数，将key与Windows用户绑定，则攻击者无法在另一台机器或另一个帐户解密(但通过恶意软件可以直接在用户的机器上尝试解密并发送回攻

② 有mpw：Firefox中master password和16bit的salt使用SHA-1生成master key，用于加密3DES的key，存储在key3.db文件中

→ 建议mpw输入框可以抵抗常见的攻击类攻击(例如钓鱼攻击)

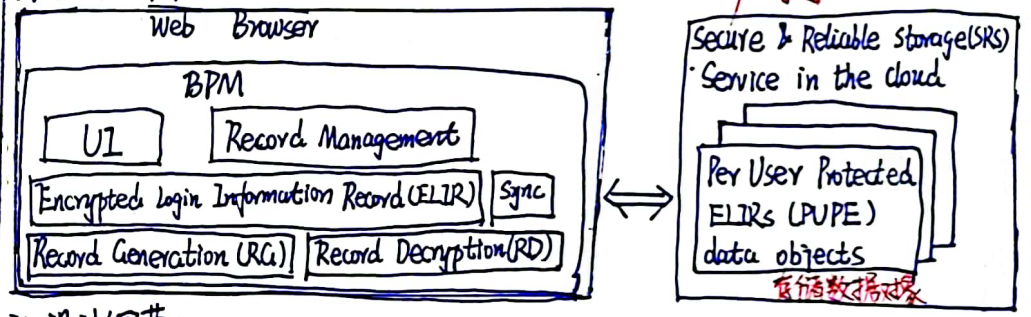
3. CSF-BPM设计

1> 受保护的数据存储在云端而非用户本地，两点原因：

① 相比于用户本地，云端存储可以更安全而且可以通过secret sharing的方式存储

② 用户可以在不同的位置用不同机器方便地登录。

2> 设计架构



3> 设计细节：

① Basic Usage

用户需记忆1个Single Strong Master Password (SSMP)。

- 用户需要在SRS服务中设置1个帐户(srsUsername, srsPassword)

- 每次会话需要用户使用帐户和口令向SRS服务认证，并把SSMP给BPM(只需1次)。

② ELIR Record

siteURL	siteUsername	encryptedSitePassword
recordSalt	...	

- recordSalt由BPM生成大且随机的值用于计算加密明文口令的key，通常由PBKDF1完成

③ key Derivation and Password Encryption. (5步)

mainkey = PBKDF2(SSMP, mainSalt, C1, dklen1) (输入后删除)

aekey = PBKDF2(mainkey, aeSalt, C2, dklen2)

recordkey = PBKDF2(mainkey, recordSalt, C3, dklen3)

encryptedSitePassword = E(recordkey, sitePassword) AES生成加密的口令

protectedELIRs = AE(aekey, concatenatedELIRs) Authenticated Encryption block cipher mode保护ELIR的机密性和可认证性

(i) mainSalt和aeSalt在用户首次认证至SRS时生成；

(ii) C1, C2, C3代表迭代次数，dklen1/dklen2/dklen3代表生成key的长度。

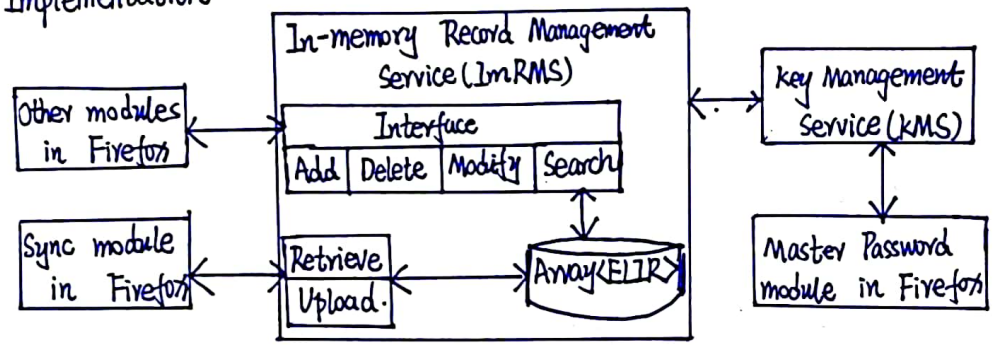
④ PUPE Data Object

可以以binary或encoded string object形式存储

mainSalt	aeSalt	PBKDF-id	PBKDF-params
E-id	E-params	AE-id	AE-params
protectedELIRs			...

⑤ Password Decryption (解密以提供autofill)
(i) 检索为SRS用户存储的PUPE data object
(ii) 生成mainkey和aekey
(iii) 使用CCM Decryption-Verification process 解密并验证 protected ELLRs → 获取所有ELLR的siteURL和siteUsername → 依次验证正确的SSMP完成
(iv) 获取ELLR的recordSalt，并生成recordkey
(v) 解密 encrypted Site Password.

4. Implementation.



5. Evaluation

1) 正确性
步骤：打开Firefox输入SRS帐户及SSMP；登录网站并确认网站记住口令；登出网站并通过autofill口令功能登录网站；关闭firefox，重新打开，输入SRS帐户及口令，输入SSMP并使用autofill功能登录网站.
▲即使在网站中有多个帐户也可以使用，且不会在Firefox本地存储额外的信息
▲另外两个观察：
① 多个网站可能共享一个登录的URL
② 一些网站设置了 autocomplete=off，故口令不会被存储到BPM中

2) 可用性，对比CSF-BPM与Firefox自身的BPM.
① 30位参与者，15男15女（5人使用过BPM（不一定为Firefox））
② 通过Likert Scale和t-test表明：
(i) 用户认为在原始Firefox BPM中使用Recovery key很麻烦
(ii) 大多数用户没有感觉到两种BPM的明显变化
(iii) 多数用户认为CSF-BPM更容易使用
(iv) 多数用户更愿意在未来使用CSF-BPM.

3) Limitation.
① 用户必须记住SSMP才可以解密网站口令；若SSMP丢失，则口令无法完成解密
② 在会话开始时，用户必须等待los (mainkey derivation)
③ 实现通过JS，可能不如用C++等开发的工具好
④ 用户必须注意保护SSMP以防泄漏
⑤ 用户需要防止malware的攻击，及时使用杀毒软件清理潜在危险.