[ESORICS 12-Custi] - On the Security of Password Manager Database Formats

相关述：(S) PM存储着吃以及其他有价值且敏感的数据，但很少有研究探索这方面的安全性 (T) 本文研究了3个主流PM的存储格式 (A) 定义了两个实际的安全模型，用于标识实际攻击者的能力。从理论角度进行分析并给出了漏洞对应的实际攻击场景。(R) 结果表明大多数PM的存储格式均是易受攻击的 (即使攻击者能力较弱)，建议选择PM时仔细检查DB格式。

优点：①首次从存储格式的角度探索了流行PM的安全性
②给出了两种敌手能力和安全模型的形式化定义。
③通过Game的方式定义了安全性

问题：①本文虽给出了理论对应的实践场景，但没有给出对应的攻击实践或案例，若可提供则文章更有说服力
②对于发现的问题，应通报厂商并收集反馈。
③应汇总本文出现问题的改善措施

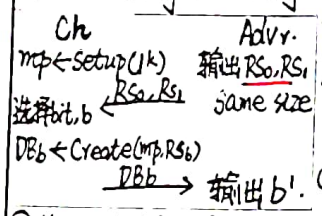
▲用户解决帐户数增多，吃又注记的两种常用方法：① reuse ② 使用PM.
→其中，对于PM的存储，通常依赖于加密进行数据保护，密钥key通常由用户输入的master password生成，进而可以防止非授权的访问 (甚至在未受保护的环境中，例如敌手获取了存储的DB)
→如果存储不安全，则可能存在：隐私信息泄漏 (例如浏览习惯，或者cookies，用户选择删除)
理想情况下，PM应只存储吃 (不需要删除)，而若以明文存储url等则会暴露隐私。

▲PM在许多方面不同：① DB格式 ② 功能 ③ 是否提供源码 ④ 支持的平台 ⑤ 是否可访问云存储
本文为PM使用自己私有的存储格式。
→本文关注于数据库格式和提供的安全性 (不考虑侧信道攻击和针对实现的攻击)
另外R关注了提供本地存储的PM.
- 3个浏览器PM的DB格式：Chrome, Firefox和IE
9个独立安装PM的DB格式：1Password, KDB, KDBX4, PasswordSafe, PIN, RoboForm.

▲Adversary Model.
1>Adv_r：拥有对吃数据库的读权限，目标为获取尽可能多的信息
2>Adv_{rw}：拥有对吃数据库的读-写权限，除了获取信息，还可以产生不是用户创建的但打开后不会弹出警告或错误信息。
→两者均可获得PM在不同时间段的快照。
→假设master password足够强，底层的密码原语不被破坏，且认为PM使用的是安全设定
→本文不考虑重放攻击，PM可通过维护local state或在可信硬件中存储来防护，但通过存储格式无法抵抗。

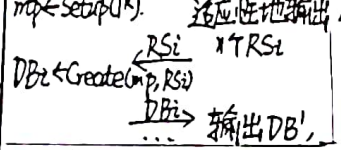
▲安全定义
1>PM包含以下4个算法：
①$Setup(\cdot)$：概率算法，给定安全参数1^k，输出master password mp.
②$Create(\cdot)$：……，输入mp和记录三元组的集合$RS=\{(r_1,n_1,v_1),\dots,(r_l,n_l,v_l)\}$.
③$Open(\cdot)$：确定性算法，输入mp和数据库DB，输出记录集合RS或$\perp$.
④$Valid(\cdot)$：确定性算法，给定mp和DB，若$Open(mp,DB)\neq\perp$则返回1，否则返回0.
→$Valid(\cdot)$应在$Open(\cdot)$内部执行，若验证失败，$Open(\cdot)$应返回error.

2>四个Game.
① Indistinguishability of database game $IND\text{-}CDBA_{Adv_r,PM}(k)$



若b=b'，输出1，Adv_r获胜。
▲IND-CDBA security:
$Pr[IND\text{-}CDBA_{Adv_r,PM}(k)=1]\leq\frac{1}{2}+negl(k)$
(不考虑攻击者根据不同size的RS进行区分的情况)

② Malleability of chosen database game. $MAL\text{-}CDBA_{Adv_{rw},PM}(k)$



当且仅当$Valid(DB')=1$且$DB'\neq DB_i$，这n时，输出1，Adv_{rw}获胜
▲MAL-CDBA security
$Pr[MAL\text{-}CDBA_{Adv_{rw},PM}(k)=1]\leq negl(k)$
⇕
existential unforgeability of ciphertexts

当不满足时，存在如下的攻击：
①Adv_{rw}将Alice的DB替换为包括Adv_{rw}创建的amazon.com的凭证的DB'
②Adv_{rw}诱导Alice通过PM自动登录到amazon.com ③Alice进行购物等活动
④Adv_{rw}使用DB替换回DB'，此时Adv_{rw}可使用Alice的帐户购物啦

▲DB格式漏洞
1>Google Chrome：在用户配置文件夹的SQLite数据库文件中存储用户名和吃，允许存储其它信息，且提供同步功能。但未提供安全性和完整性保护，可访问文件的用户可以看到其中的内容并修改
2>Mozilla Firefox：存储在SQLite DB中，用户可使用master password加密，但url未被加密
Adv_r可通过区分url来胜利。在实践中，攻击者可获取用户拥有帐户的url，有目标其它攻击
由于Firefox不提供完整性保护，Adv_{rw}可以更换其中的url，进而使用户向错误domain提供凭证
3>Microsoft IE：在注册表中存储用户名和吃，每条记录为单独的表项，且通过系统登录凭证加密
存储键值对$(h:c)$，其中$h=SHA\text{-}1(url)$，$C=E_k(metadata\|username\|0x00\|password\|0x00)$
metadata包括额外信息如加密元素的大小。
→加密通过系统调用，使用3-DES (CBC)及hash-based MAC，k由salt (也存储在密文中)，url和Windows登录凭证生成
- 由于url不加密，故Adv_r可以赢得IND-CDBA，实践中可以获知用户是否访问了特定域名
虽然不能修改表项，但可以删除对应的表项，故Adv_{rw}可以赢得MAL-CDBA.

4>1Password：在多个文件中存储，每个文件包含JSON格式的数据库项，所有的项存储在一个索引文件中。
最高安全级别为：使用master password生成的key加密，但无完整性保护 (可检测JSON解析失败)
Adv_r可通过条目别title, location等信息赢得IND-CDBA，(敏感数据)，攻击者可获得敏感数据
Adv_{rw}可以更换相应文件中的url，只要不破坏JSON格式，就可以做钓鱼攻击

5>KDB (KeePass 1.x)：KDB包含一个独立的文件，分成两个文件，an unencrypted header (hdr) 和 an encrypted body (bdy)，bdy中存储数据项的加密值。hdr存储group和entry的数目，以及bdy在加密时的哈希值

其中，哈希值用于保证完整性，解密后，PM可验证旧哈希是否与hdr存储值相等
若不相等则DB被破坏或master password错误。
Adv_r 可通过对比RS的哈希 $h_2 = H(RS_2)$，进而确定属于哪个DB，在实践中，这允许攻击者在密文不相等的情况下区分两个DB是否相等，
另外，hdr未被认证，攻击者可改其中的值如entries的数目，不影响哈希值

6> KDBX4 (aka keePass 2.X)：由1个口令保护的文件组成；hdr和bdy
hdr 包含 mseed, tseed (用于计算加解密key)，IV，pskey，ssbytes (用于secrecy和integrity保护)
bdy包含XML串，通过AES-256加密，后32字节包含XML串的哈希值。另外XML串中的口令均经过与伪随机串的XOR运算得到 $S_2 = pwd_i \oplus Salsa_{20}(k, IV)$
→ 修补了KDB的问题，但攻击者仍可以在MAL-CDBA中胜利
由于完整性检查在XOR之前，故pskey中的值被随机替换也无法检测！

7> PINs 存储在1个文件中，使用AES加密，其中每条记录分别进行加密
第2行存储的加密值用于验证用户提供的master password是否正确
→ IND-CDBA 安全，攻击者除了记录数和估计的长度，其它均无法得出
不提供数据完整性保护，Adv_{rw} 可获得MAL-CDBA胜利

8> Password Safe V3：由单个的文件构成，hdr和bdy。
hdr包含IV，和对256 bit的keys：K和L，分别用于加密bdy和用在HMAC中。
K和L由口令生成的密钥加密。
→ IND-CDBA-secure 且 MAL-CDBA-secure

9> Roboform：在多个文件中存储，每个文件包含header (编码2个url：goto和match)
另外包含 a short header 和 encrypted payload。
未加密
由于url未加密且未受完整性保护，故Adv可在两个game中获胜

△讨论
1> 3类存储格式
① 可用于不安全的存储介质中：Password Safe v3
② 可用于底层存储机制提供了完整性和数据认证：KDBX4 和 PINs，不恢复其中的信息
③ 只能用于底层存储机制提供了integrity，authenticity和secrecy。

2> For this reason, users should carefully consider whether a particular database format is acceptable for storing data in the cloud, on a USB drive or on a machine shared with other users. ☆