USENIX'20 - Sean Oesch - That Was Then, This Is Now: A Security Evaluation of Password Generation, Storage, and Autofill in Browser-Based Password Managers

先前的研究揭露了口令管理器(特别是browser-based)的漏洞,本文对13个流行的口令管理器进行了生命周期三个阶段的新的评估,首次分析了口令管理器的password generation,并且对于password storage和autofill,重复了先前的实验,又观察到先前的问题有的已经解决,但仍存在一些安全问题,提出了一系列建议及未来工作方向

优点:① 首次对口令管理器的生命周期的全部三个阶段进行了安全性评估,研究较全面
② 文章结构很完整,研究动机、方法及具体的实验、结果分析等
③ 对未来的研究工作方向做了介绍。

问题:① 选用的研究对象本身具有一定的特征(包括未实现的安全特性),但没有评判安全性的标准
② 文章对于攻击场景及敌手模型刻画不够细致和精确
③ 图表自明性不够好,甚至原文中也无对表格中使用的符号的解释.

Password Manager:存储用户凭证(如用户名和口令)的工具,可减轻用户cognitive burden

Password vault:通常指口令的存储,理想情况下由用户选择的口令(master password)加密后存储;通常可以在线存储,还可以支持多个设备间的同步.

正确使用口令管理器可提供的好处: ①每网站使用不同的口令,可解决口令重用的问题 ②减轻用户记忆用户名和口令的负担. ③便于生成抵抗在线和离线猜测攻击的口令.

本文分析13个不同的口令管理器,根据与浏览器的集成程度分为3类:

①app-based:未与浏览器集成到一起的桌面客户端,包括KeePassX, KeePassXc

②extension-based:作为浏览器扩展部署,不依赖桌面应用程序,包括1Password X, Bitwarden, DashLane, LastPass, RoboForm

③ browser-based:相比于另外两种缺少许多安全特性,包括Chrome, Edge, Firefox, IE, Opera, Safari

▲关注的口令管理器的特性(feature)

utility and usability: ①对口令 generation和 autofill的支持 ②对extension setting及便携式的password vaults的同步的支持 ③允许从命令行使用口令管理器的能力

security: ①是否支持MFA ②Password vault是否可以lock. ③master password是否必须输入到当前tab和application中 ④是否提供评估帐户和口令安全性的工具. ⑤在拷贝口令后是否会清除clipboard的值 ⑥是否开源

Password Generation(使用Python脚本/dev/random和online Secure Password Generator(SPG)实现)

▲主要测试的特性: ①支持的长度,黑名单的长度及构成(letter, digit, symbol) ②是否去除用户难以记忆或识别的字符 ③字符集的构成▲ ④对generation setting的保存(最新or最旧)

▲password generation的参数(不同的参数对口令的随机性是否有影响)
- character classes: l, ld, ls, sd, all
- password length: 8, 12, 20 characters long.

⇒ 对于每个口令管理器,字符类及口令长度生成了100万口令.

采用以下方法来识别不同口令管理器生成的口令是否随机

① Shannon entropy:用于检查每个生成器生成字符的频率是否异常.

② χ^2 test for randomness: a simple statistical test for determining whether the difference between two distributions can be explained by random chance. 用于评估n个集,并使用Bonferonni correction来修正p-values.

③ zxcbvn password analysis tool:检测生成口令中是否包括字典词汇和简单模式,用于评估口令是否可以抵抗在线和离线猜测.(LISA'14,抵抗在线:10^6次猜测,离线:10^{14}次猜测)

④ recurrent neural net-based password guesser.:用于检测zxcvbn未检测到的pattern

结果:① Password length: ≥12足够抵抗在线和离线攻击.(弱口令除了长度,也与字符集相关)

② Randomness:多数存在不随机的情况(字符集存在频率不同等情况),部分可以根据自身的策略解释,但也有无法解释的

③ Random but Weak Passwords:为truly random generator应有的行为,但这些口令给带来安全影响基于数据分析,使用长度为10的口令抵抗在线攻击,长度为18的口令抵抗离线攻击.

Password Storage:检查本地存储口令的数据库,确定是否加密以及是如何加密的(master password是如何构造的)

▲Password Vault Encryption:①app/extension-based使用AES-256加密数据库,管理器使用不同的KDF加密master password(AES-KDF, PBKDF2, Argon2D(memory-hard KDF)).不同管理器对于master password的构造要求不同

②brower-based,多数依赖于OS加密password vault,也因此不具备master password,所以只能通过锁定账户来锁定password vault.

▲Metadata Privacy: ① app-based加密了所有的metadata ② extension-based仍存在一些问题,如未加密或未全部加密,泄露URL等. ③ 依赖OS的browser-based的metadata仅做加密,其它的管理器则有许多未加密的metadata.

Password Autofill: ① User Interaction Requirements:在autofill发生前应与用户进行交互,而不同管理器的策略不同,例如允许静默或无需交互,这与网站是否应用HTTPS相关. ② Autofill for iframes,无论是否有用户交互,在iframes中autofill口令都是危险的 ③ Fill Form Differing from Saved Form:口令管理器应检查填充时与存储时的方式是否相同,如协议 ④ Non-Standard Login Fields:调查口令管理器是否会填充type="text"的字段以及具有用户名和口令两个字段但不是登录框的场景. ⑤ Potential Mitigation:研究人员提出在填充时使用随机值替换口令,在传输时替换为口令,可防止XSS攻击,而实践起来发现extension-based的口令管理器无法修改request body,故无法实现. ⑥ Web Vault Security & Bookmarklets:由于在extension-based中,在线password vault可能放大autofill的问题,观察是否使用CSRF token阻止CSRF攻击,以及CSP是否有效防止了XSS攻击.

建议：①去除较弱的口令(password generation)

②推荐口令管理器使用更严格的条件筛选 master password，防止口令管理器成为 single point of failure

③推荐在 autofill 时强制要求用户交互，且应禁用在 iframe 中的 autofill.

未来工作：① 浏览器应使口令管理器具备以下特征：1)允许填充时用随机数代替口令，在传输时再替换成口令. 2)允许访问 system keyring. 3)在 autofill 前做用户交互且确保 clickjack resilient 4)为网站口令策略增加 HTML 属性.

②面对不同情景的口令字符集的选择的研究

③增加 HTML 属性来帮助口令管理器识别、生成口令的策略

④对移动端口令管理器的分析.