USENIX Sec'05 - Ross - Stronger Password Authentication Using Browser Extensions

大概述： (S)由于送往的哈希方案不正确的实现，攻击者可以利用钓鱼攻击(结合哈希使用的特点)获取用户口令。(T)本文旨在提供一种浏览器扩展PwdHash，可以为不同的网站生成不同的口令，提供安全特性抵抗钓鱼攻击。(A)除了细致的网站代码设计，PwdHash采用PRF的方式以口令作为key，domain作为参数进行运算，采用特定的规则划定参与运算的domain，通过提供多种配置文件构建多种口令策略，使得生成的口令符合网站的口令策略。通过输入前缀或特定的按钮进入口令输入模式，减少很多潜在的攻击。(R)最后本文给出了抵抗离线字典攻击的方案，进行用户调研，并给出了本文的局限性

优点： ① 给出了一种可抵抗钓鱼攻击的口令哈希方案。
② 对潜在的攻击方式提供了一定的对抗方法

问题： ① 在针对domain的分析中应添加更常见的通过诱骗用户访问伪造URL的钓鱼攻击的分析
② 提供不同的配置文件后，若通过试错的方式匹配网站口令规则，则在登录时可能过于费时
③ 本文提供的防护离线字典攻击的方法不够好，如慢哈希同样大大降低了用户的运算速度，可通过分步执行缓解(如Password Multiplier).

---

1. 面对的问题

攻击者可以利用钓鱼攻击获取用户的口令，并可以利用口令重用的特点，通过攻击低安全性的应用获取访问高安全性应用的权限.

2. 设计的目标：为用户提供较好的安全实践，不需要服务器端的改变，且不需要(需要很少)用户行为的改变。可以抵抗钓鱼攻击

3. 设计挑战：

1> 如何避免钓鱼页面上的JS代码造成的攻击？

2> 应如何构造salt值，应使用URL的哪所有的来构造salt？

3> 如何构造encoding方法使得hash后的口令符合网站的口令策略.

4> 需与浏览器的auto complete等特征兼容.

5> 在安装PwdHash后应为用户提供更新口令的功能

6> 允许用户在无法安装PwdHash扩展时完成登录.

7> 当钓鱼攻击者获取了口令的哈希值，如何抵抗离线字典攻击？

②③① → 口令哈希函数决定. ①④ → 浏览器环境的问题 ⑤⑥ → 用户体验

⚠不考虑对用户机器的攻击的影响

4. 抵抗浏览器环境的问题

1> $hash(pwd, dom) = PRF_{pwd}(dom)$ → dom作为salt值.
可在浏览器扩展的配置文件下的控制下满足网站的口令策略.

2> 可能产生的JS攻击
① 获得登录记录. ② action中的domain重写 ③ 假的password字段(如text)
④ 在线的假的口令域(将用户输入的值发送给phisher) ⑤ 口令可能被输入到非敏感的字段，甚至被显示

3> 对抗的措施(使用浏览器扩展作为用户与应用之间的实体)
所有的输入可以经过浏览器扩展的保护.

① Password Prefix：在口令前添加任何知的字符，例如@@.
▲当输入前缀后，PwdHash会将随后输入的字符记录，使用固定的字符串替换后输入到浏览器(即无法检测到口令的字符内容，但可以获知口令的长度)→在扩展中存在一个转换表
若检测到前缀不在口令字段，会及时提醒用户.

▲客观的提供的好处
(i) 不会对无前缀的输入进行哈希.
(ii) 可以用于在修改口令时选择新旧口令.
(iii) 可以允许用户自行选择是否采用这种方法为口令提供安全防护.

② Password key：上述prefix的另一种方案，即通过特定的按钮进入

▲提供Password traffic light，当处于输入口令模式时，显示绿色；处于正常模式时显示红色

▲对于提供了auto-complete的浏览器，可保证存储的是哈希后的版本而不是明文

5、salting and encoding.(确定的salt值，不同网站不同且可以防止用户被欺骗)

1> domain name的选取(考虑的domain url仍然是正确的URL)

① current domain：除非攻击者可以在页面嵌入新的表单.
② target domain：尽管可以保证传输的是特定url对应的口令，但扩展需拦截表单数据，可能存在password reflection attack.
③ certificate hash：同一组织下的多个域的证书可能相同但：对证书的验证可能导致钓鱼攻击，难以手动计算(易出错)，无法兼容非SSL网站

2> 如何保证一个域下的口令生成值相同(即不能完全根据url)

↓

3> Encoding (如何满足不同网站的约束).
为不同的口令策略的网站创建special case.(使用configuration file)
格式为：<reg-exp, salt-rule, encode-algorithm> 例→ <x.com, use-top-2, encode-alg>
匹配的规则是首个匹配的rule(默认有5个)
其中包括encode-alg-0代表不进行hash.
使用two top-level domain作为salt.
▲配置文件定期更新：网站的新策略或更改后的策略，且该配置文件应该由可信实体签名防止篡改(例入修规则)(改为匹配下所有domain但不hash，则所有口令均为明文)

4> Dictionary attacks
防护方法：①slow hash function. → 增加运算消耗的资源
②short secret salt → 扩展需要多次计算(且未知哪次正确，可能被锁定).
③为扩展声明一个global password
④使用PAK协议防止离线字典攻击(但需浏览器和服务器的更改).

6. User interface and usability issues

1) 在扩展安装后更新所有网站口令(认为用户可能不需要更新所有网站口令)

2) Roaming：如何在没有安装扩展时计算口令的哈希值

▲ 提供一个页面用于生成口令哈希，

▲ 提供 JS 的 bookmark 或 bookmarklet(某些浏览器可能有空间的限制)

3) 口令更新.

可以设置不同的安全级别对应的口令，更改一个口令不会引起所有网站口令的修改.

▲ 口令字段在重新访问时会自动清空.

7. limitation.

1) Other applications：无法完全适用于所有的应用(如 Outlook 中的 HTML).

2) Spyware：无法抵抗在本地安装的恶意软件以及通过修改 HOST 文件完成攻击的钓鱼攻击者.

3) DNS Attacks：无法抵抗篡改 DNS 或 HTTP 响应的被动攻击

4) Flash：对于 Firefox 来说，当 embedded Macro-media Flash object 选中，Firefox 可能会允许 Flash 处理 keystroke 而 extension 无法处理，此时在 Flash 中的口令域将允许被动直接获取明文口令.

5) Focus Stealing：强制在用户输入时修改 focus，则用户可能会在非口令字段位置输入部分明文字段。可通过强制禁用 JS 或应用后缀来做.