COSE'19 - Ntantogian - Evaluation of password hashing schemes in open source web platforms

不足/缺点：(S) 口令破解敌手的计算能力日益增强，为对抗此类攻击，研究人员提出了例如 PBKDF2, BCRPT 及 SCRYPT 的哈希函数。然而，基于 Content Management System (CMS) 和 web application frameworks 的网站的默认哈希方案存在问题。(T) 本文旨在用量化的方式进一步评估不同网站的默认哈希方案。(A) 首先分析了流行 CMS 和应用框架的默认哈希方案，提出了可以量化口令破解时间的框架，并对流行 CMS 和应用框架的默认哈希方案进行了分析和比较。(R) 发现了神多实践存在的安全问题，并给出了最佳实践和改进建议。

优点：①本文分析了 CMS 和 web application framework 应用的哈希方案面对口令破解攻击和 DoS 攻击的抵抗能力，并给出了详细建议

②对观察的现象给出了较为详细和合理的解释

③在建议中，对 Distributed hashing 和 Sever relief (特别是 DoS 攻击) 的介绍较好.

问题：①在构建公式时，α代表 attack success factor，但并未分析 α=1 代表实际中的什么情况

②表6中，BCRYPT 和 SCRYPT，Hashrate 增加时却伴随 Runtime 也增加，应解释原因

③CMS 并非只有 open-source，研究范围与题目不相符

④数字存在逗号，连续几个数字出现时影响可读性。另外还存在其它几处笔误.

▲本文主要考虑离线猜测攻击，分为3类　　本文考虑√的情景

1) brute force：给定长度和字符集范围时尝试每一种可能性

2) dictionary：从 List 中取口令，按照使用概率大小

①pure：使用预定义的词作为候选口令　②PCFG-based：构建包含给定概率的口令的字典

③Markov model based：基于字符串的概率分布创建候选口令

④mangling rules：通过不同类型的修改规则创建口令变体

3) rainbow table：敌手预计算出候选口令的哈希值.

▲本文考虑的硬件平台 CPU，考虑 CMS 和 Github 中高 star 的 web application frameworks.

▲Password hashing schemes

1. 考虑3个参数

1) hash function 核心部件，如 MD5

2) iterations 可选，连续执行口令哈希函数的次数，用于减慢口令猜测攻击

3) salt 可选，与口令一起输入的随机串，彩虹表攻击失效.

2. 考虑不同的哈希函数

1) MD5, SHA1, SHA256, SHA512，默认不使用 salt 无 iteration

2) PBKDF2, BCRYPT, SCRYPT 和 Argon 在内部选择随机 salt，并默认应用 iteration

①PBKDF2 直接迭代 SHA256 等哈希函数，另外3种在算法的特定部分使用 iteration

②PBKDF2 与 BCRYPT 在使用时给定固定的 CPU 时间

③SCRYPT 和 Argon2 为 memory hard function (MHF)，可使用任意大且可调整的内存，硬件开销更大

3. 本文重点考虑不同哈希算法的 iterations.

▲口令猜测攻击的开销分析（提出一个框架）→成功破解1个口令哈希需要的平均哈希数目→对转换为时间

1) 参数

①Iterations (I)：计算哈希的次数，只与 MD5 等相关，默认作为内部参数，I=1

②Database passwords (D)：数据库中口令哈希的数目

③Salt (S)：数据库中的 salt 值（认为1个口令对应唯一1个 salt）；若数据库不使用 hash，S=1

④Hashrate (Hr)：每秒运算哈希值的数目.

⑤Password length (pwd-length)：敌手希望暴力破解的口令长度（包括最短和最长的长度）

⑥Charset (C)：只与暴力破解攻击有关，描述字符的个数，如数字包括 10 个

⑦Attempts in a dictionary attack (attempts)：字典攻击中敌手尝试的口令的数目.

⑧Effectiveness (E_{BF} or E_{DC})：完成攻击后被破解口令的比例.

2) Brute force attack

①$hashes_{BF}$：在此计算中需运算哈希的数目，通过字符集和 pwd-length 计算，另外还需猜测 salt 值，另外还包括 iterations 的值. $hashes_{BF} = \alpha \cdot I \cdot S \cdot \sum_{i=pwd\_length_{min}}^{pwd\_length_{max}} C^i$，α代表 attack success factor，与需要尝试所有口令的概率相关.

②$cracked\text{-}pass_{BF}$：被破解的口令哈希值的数目，$cracked\text{-}pass_{BF} = D \cdot E_{BF}$

③$cost_{BF}$：需要破解1个口令需要的运算哈希的平均数目 $cost_{BF} = \frac{hashes_{BF}}{cracked\text{-}pass_{BF}}$

④$cost\text{-}time_{BF}$：破解1个口令哈希需要的时间 $cost\text{-}time_{BF} = \frac{cost_{BF}}{Hr.}$

3) Dictionary attack.　与2)中的分析类似.

①$hashes_{DC} = \alpha \cdot I \cdot S \cdot attempts$，attempts 代表尝试次数.

②$cracked\text{-}pass_{DC} = D \cdot E_{DC}$（$E_{DC}$ 与采用的方法有关，pure, PCFG based 等）.

③$cost_{DC} = \frac{hashes_{DC}}{cracked\text{-}pass_{DC}}$　④$cost\text{-}time_{DC} = \frac{cost_{DC}}{Hr.}$

▲CMS、web application framework 的开销分析和比较

1) 分析考虑 hash function、iterations、usage of salt 和 minimum acceptable pwd-length.

包含 49 个 CMS 和 47 个 web application framework.

①59.18% 的 CMS 使用可利用石更件做并行运算的哈希方案，另外 40.82% CMS 默认使用 BCRYPT

②没有 CMS 使用 SCRYPT 或 Argon2 这样的 MHF（许多网站为了兼容性而没有更改算法）.

③14.29% 的 CMS 不使用 salt，易遭受彩虹表攻击，36.73% 的 CMS 不使用 iterations，使用的 iteration 值低

④38.78% 的 CMS 没有最小口令长度策略，可能导致弱口令如单个字符的口令.

⑤尽管部分 CMS 允许更改哈希方案，但管理员通常无足够安全意识，倾向于使用默认设置

⑥23.4% 的 web application frameworks 使用弱的哈希方案，12.77% 不使用 iterations，27.66% 使用 BCRYPT，但没有应用的默认设置为 SCRYPT 和 Argon2.　Naiakshina'17

⑦48.94% 的 web application framework 不提供默认口令哈希方案，开发者也无足够的安全意识

2. 开销比较：根据上述提出的框架做评估和数值比较，首先要计算值 测试可用性 → 高延迟 → 迟缓

i) 不同哈希函数和 iteration 的 hashrate（在CPU中测试而得，另外在web服务器上测试）

ii) 字典攻击的 effectiveness.

从相关工作中得到 pure，Markov model 和 PCFG 的参数，其中 Markov 比 PCFG 更高效

iii) 暴力攻击的 effectiveness

定义 $P_{pwd\text{-}length}$ 为特定长度口令的比例，$P_{c,pwd\text{-}length}$ 为具有特定charset和长度的口令的比例

$E_{BF} = \sum_{pwd\text{-}length_{min}}^{pwd\text{-}length_{max}} P_{pwd\text{-}length} \cdot P_{c,pwd\text{-}length}$ 作者通过真实的口令集计算得到

▲进行5组比较分析.（考虑 α=1）

i) 实现了口令策略和未实现口令策略的 CMS 的暴力攻击.

⑧ 口令策略可以提高抵抗暴力攻击的能力，但多数CMS未使用口令策略，甚至对长度无要求

ii) 比较使用 iterations 为256的 BCRYPT 的 CMS 和 iterations 为50000的 PBKDF2 的 web application framework 暴力破解

⑨ iterations 为256的 BCRYPT 比 iterations 为50000的 PBKDF2 提供的安全性更高，且表明NIST要求 PBKDF2 的 iterations 为10000太小了，且 PBKDF2 自身不够强大.

iii) 比较 BCRYPT 的 iterations 对字典攻击时间的影响

⑩ 多数CMS的 BCRYPT 使用1024 iterations（PHP默认），但比较结果表明增大 iterations 不会造成较大的登录延迟.

iv) 比较 MHF 和 BCRYPT 对字典攻击消耗时间的影响

⑪ 建议CMS可以将默认的哈希函数替换为MHF，不会增大登录延迟，且NIST也可以适时将PBKDF2更换为MHF

v) 比较最流行的3个CMS.

3. 口令哈希方案是否会因为占据CPU导致DoS攻击

使用 WordPress，用户可以配置参数（如哈希函数和iteration），服务器端使用htop监视

i) 参数包括：哈希函数，iterations，口令长度和登录频率 rate.

哈希函数包括 MD5，SHA512，BCRYPT 和 SCRYPT.

ii) 测试CPU利用率与 log in rate 的关系，4个哈希函数的 iterations 为应用默认或建议值

图像为线性.

⑫ 除了MD5之外，其余通过 iteration 降低速率的方案可能导致DoS攻击. BCRYPT-1024，20次/s.

根据前人研究，DDoS攻击可以每秒执行 100~1000 次请求，且DDoS不易被察觉

⑬ 在并行会话中使用 rate-limit 来对抗DoS攻击是重要的，且 rate-limit 在CMS和web application framework容易部署，NIST也建议使用 rate-limit 且推荐1个帐号至多100次失败.

iii) 口令长度和 iterations 能否在较低的 rate 下造成DoS攻击.

在ls尝试1个口令的情况下，哈希函数与ii)相同，观察口令长度增加是否会造成DoS攻击

⑭ 使用SHA1，SHA256，SHA512或PBKDF2（带有高iterations）的网站应限制最大的口令长度来防止DoS攻击，另外 BCRYPT 和 SCRYPT 不会因为大的口令遭受DoS攻击.

iv) CPU利用率与 iterations 的关系：

使用小的口令长度和低的 rate，可发现CPU利用率随着 iterations 的增加而增加

iteration 成为安全性与性能的权衡指标.

⑮ 与BCRYPT相比，SCRYPT在 iterations 增长时更具有扩展性，可提升安全性而不引入DoS攻击，且在足够的物理内存的情况下不会引入较大的登录延迟.

▲讨论和建议：增强抵抗猜测攻击的口令的鲁棒性.

1> Update NIST recommendations.

使用MHF替换PBKDF2，严格审查其安全性，并提供官方的实现库加速应用并促进软件测试

2> Use of secure default setting:

应设置更严格安全的默认设置，避免假设开发人员可以选择安全恰当的哈希方案.

3> Upgrade legacy hash functions.

由于替换掉现有过时哈希函数会导致用户重新注册，提供了两种方法无需重新注册即更新

① 使用两个口令哈希表，用户登录时会将新哈希值填充到新表中，直到完全替换完毕，删除旧表，但耗时太长

② layered hashing scheme，例 Facebook，即使用多层的hash，可以随时更新.

4> Distributed hashing

将运算哈希的任务委派给外部服务（Crypto as a service），利用外部服务的 key 执行HMAC运算，此时攻击者必须访问外部服务才可以猜测口令.（Pythia等）

5> Federation and FIDO

使用诸如OIDC的方案，网站不需要维护口令数据库.

另外，用户应仍使用长口令，并运用2FA，Password Manager和 FIDO 来抵抗口令猜测攻击

6> Server 'Relief（Argon2采用了此技术，避免DoS攻击）.

让客户端执行运算量大的运算得到中间值返回给服务器，不仅可以防止DoS攻击还可以阻止在线暴力攻击，达到了服务器负载、攻击消耗、安全性和可用性的权衡.

客户端执行的运算可能会造成信息泄漏，且可能会影响用户体验（客户端没有那么强的算力）

此方向仍需大量研究