COMP2700 实验室 3

识别和认证

在本实验室中，我们将研究基于密码和生物特征的身份验证的理论方面（第一部分）和密码破解的实践方面（第二部分）。完成本实验后，学生应该能够

- 评估密码方案对抗暴力攻击的安全性、
- 区分生物识别技术在身份验证中的不同用途、
- 使用最先进的密码字破解工具，进行简单实用的密码破解练习。

本实验要求您完成练习 1 至练习 8。

实验结束后将进行实验测验（实验 3 测验）。实验测验将测试你的理论知识和实践知识。实验室 3 测验将包括密码破解挑战，因此请务必做下面的练习以熟悉 hashcat。实验室的 Wattle 页面将提供更多详细信息。

第一部分：基于密码的身份验证、密码计数、熵和生物识别技术

练习 **1**

请看下图所示的申请人与验证人之间的另一种密码验证方案，其中 h 代表单向散列函数。

请注意，在验证过程中，申请人必须输入明文口令，口令的散列由申请人而不是验证人完成。这种验证方案会有什么潜在的安全问题？

在本练习中，可以假设申请人和验证人之间的通信渠道是安全的，因此攻击者无法在传输过程中截获密码哈希值。但可以假设攻击者可以通过某种方式从验证者那里获得包含哈希值的表。

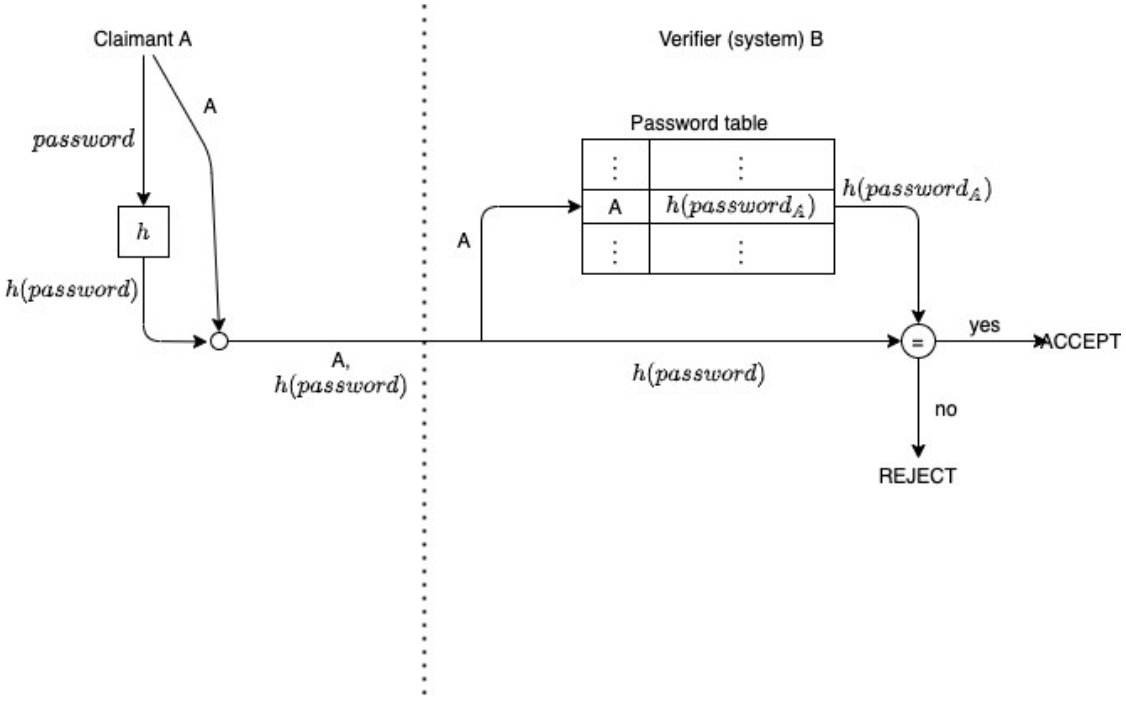

图 1：密码验证协议示意图。

练习 **2**

针对以下每种密码策略，计算有多少可能的密码符合给定的策略。

1. 有效密码长度为 5 个字符，每个字符都是小写字母（a-z）。
2. 有效的密码最长为 5 个字符，每个字符都是小写字母 (a-z) 或大写字母 (A-Z)。
3. 有效的密码长度为 10 个字符，每个字符可以是小写字母 (a-z)、大写字母 (A-Z) 或数字 (0-9)，且必须至少包含一个数字。

练习 **3**

考虑使用 FMR=0.5% 和 FNMR=0.8% 的指纹扫描仪。为简单起见，我们假设采集失败率（FTA）为 0%，因此设备每次扫描都能完美捕捉指纹特征。

假设该指纹扫描仪用于一个身份验证服务器。为了防止一个人创建多个账户，认证服务器要求每个新用户注册时通过指纹扫描仪提供指纹样本，然后与当前注册用户的指纹进行核对；如果发现匹配，则拒绝注册。

要登录服务器，用户需要提供用户名并扫描手指。服务器将查询用户存储的指纹，并与扫描的指纹进行比对--只有当两者匹配时，身份验证才会成功。

假设我们目前有 200 个注册的不同个人，每个人的指纹都存储在注册系统中。

1. 注册用户无法验证服务器身份的概率是多少?
2. 假设新用户 Alice 想要创建一个账户。服务器拒绝 Alice 注册的概率是多少?
3. 假设鲍勃是注册用户。鲍勃试图创建一个新的用户账户。鲍勃成功的概率是多少?

练习 **4**

在某些 Windows 操作系统中，为了保持向后兼容性，计算机中本地用户的密码哈希值以两种不同的形式存储：LM 哈希值和 NT 哈希值。NT 哈希值是直接根据用户密码计算出来的，而 LM 哈希值则是根据不区分大小写的版本计算出来的，也就是说，在哈希值计算之前，用户密码会被转换成大写版本。我们假设密码中只使用英文字母。

1. 假设攻击者成功破解了密码的 LM 哈希值，从而获得了用户密码的非大小写敏感版本。攻击者猜测实际（大小写敏感）密码所需的尝试次数上限是多少？假设密码长度为 n。

2. 一般来说，对于长度为 n 的密码，哪种暴力破解密码的策略更好：先暴力破解 LM 哈希值，然后猜测大小写敏感性，还是直接暴力破解 NT 哈希值？通过比较两种策略所需的尝试次数上限来证明你的答案。

第二部分：使用哈希猫破解密码

在本部分中，我们将使用著名的工具 Hashcat 进行密码破解的实际操作。Hashcat 拥有非常丰富的功能，本实验不可能涵盖所有功能，但希望它能为你提供足够的基础知识，以便你自己尝试更高级的功能。

我们将使用您在实验室 1 中设置的实验室虚拟机。在下面的练习中，您需要以用户 admin2700 的身份登录。

Hashcat 可以使用 GPU 和 CPU 来破解密码。默认情况下，它会尝试使用 GPU，因此如果你在实验室虚拟机中使用默认设置运行它，可能会遇到 "找不到/没有设备 "的错误提示。要消除这个错误并强制 hashcat 使用 CPU 来破解密码，你需要在每个 hashcat 命令中添加选项"-force"。

在本练习中，如果你有自己的 Linux 安装，就不需要使用实验室虚拟机。Hashcat 可以在任何 Ubuntu 安装中运行，因此并不局限于实验室虚拟机。

Hashcat 是一款高度可定制的密码破解工具。我们在此仅介绍其一小部分功能：字典攻击、暴力攻击（通过模式）和基于规则的攻击。

下载实验室文件

本实验为您提供了一些需要破解的密码哈希值。使用以下命令下载文件 `md5_hashes.txt`，并将其保存到 admin2700 的主目录下。

```
$ cd ~/
$ wget http://users.cecs.anu.edu.au/~tiu/comp2700/md5_hashes.txt
```

文件 `md5_hashes.txt` 包含使用 MD5 哈希函数生成的密码哈希值列表，MD5 是早期用于密码存储的哈希函数之一，现在已被认为不安全。在接下来的几个练习中，我们将使用这个简单的密码哈希函数，以便利用实验室虚拟机中有限的计算资源尝试不同的密码破解策略。

我们接下来将看到的攻击方法适用于更复杂的哈希函数，如 linux crypt，但性能会（大大）降低（我们将在下面的练习 8 中再次讨论）。

在虚拟机中运行 **hashcat** 的重要注意事项

Hashcat 的设计旨在利用 GPU 的优势，由于本实验室使用的虚拟机不支持 GPU 虚拟化，因此可能存在一些兼容性问题。

如果在运行 hashcat 时遇到 "非法指令 "等错误，可以尝试使用传统版本，该版本也安装在实验室虚拟机中（Azure Labs 版本和 VirtualBox 版本）。只需将命令 "hashcat "替换为 "hashcat- legacy "即可。

就破解密码而言，它们应该可以互换，但状态屏幕上显示的信息略有不同。

词典攻击

顾名思义，这种攻击方法使用的是 "字典"（通过各种渠道（如密码泄露）获取的常用密码列表）。在本实验室中，我们将使用 cracklib 中的字典，该字典已安装在实验室虚拟机中，位于

```
/usr/share/dict/cracklib-small
```

这种攻击方法的语法如下：

```
hashcat -a 0 -m 0 target_hashes dictionary_file -o output_file --force
```

我们将逐一讨论这些选项：

- 选项 `-a 0` 用于指定攻击模式，这是一种 "直接 "攻击模式，只需使用字典文件即可破解密码。可以使用

  敲击
  ```
  hashcat --help
  ```

  查看所有可用选项。例如，帮助页面显示了以下攻击模式选项。

  ```
  - [ 攻击模式 ] - # |
    模式
  ===+======
    0 | 直线
    1 | 组合
  ```

3 | 蛮力
6 | 混合词表 + 掩码
7 | 混合掩码 + 词表

- 选项 `-m 0` 指定了要破解的哈希值类型。数字 0 表示 MD5 哈希值。MD5 是一种哈希函数。Hashcat 支持多种不同类型的哈希函数，详情请查看帮助页面。
- `target_hashes` 文件包含我们要破解的密码哈希值。需要将其替换为目标文件。
- 文件 `dictionary_file` 包含用于破解密码哈希值的单词列表。请将其替换为实际使用的字典文件。
- 选项 `-o output_file` 指定了包含成功破解的密码的文件名。
- 如前所述，选项 `--force` 用于强制 hashcat 使用 CPU 进行搜索。只有在本实验室中才需要这个选项，因为我们是在虚拟机中运行 hashcat。如果在带 GPU 的实际硬件上运行，可能就不需要这个选项了。

  注意：如果使用 `hashcat-legacy`，请不要使用 `--force` 选项。

练习 **5**

使用 `/usr/share/dict/cracklib-small` 中的 cracklib 字典，使用字典攻击恢复 `md5_hashes.txt` 中提供的哈希值中的密码。您成功破解了哪些密码？

Hashcat 会输出许多统计数据，其中一个有趣的数字是 "速度"，它给出了哈希率（每秒生成的哈希数）。您在尝试中观察到的哈希率是多少？

注意：Hashcat 会将目前找到的散列和密码组合缓存在文件 `/.hashcat/hashcat.potfile`，因此如果重复运行相同的查询，第二次和随后的运行可能会很快终止，并且不会生成输出文件（因为它假定哈希值已经被破解）。如果想将哈希猫运行恢复到初始状态，只需删除 `/.hashcat/hashcat.potfile` 文件即可。

暴力攻击

这种攻击模式只需尝试所有字符组合来破解哈希值。这只适用于使用字典不容易破解的短密码。

这种攻击模式的语法如下（假设我们仍在破解 MD5 哈希值）：

敲击
```
hashcat -a 3 -m 0 target_hashes pattern -o output_file --force
```

请注意，选项 `-a` 现在使用了 `3`（而不是 `0`），表示暴力模式。另一个新参数是模式（`pattern`），用于指定暴力模式。

暴力模式指的是哈希猫在暴力攻击中使用的字符集。Hashcat 已定义了以下字符集（摘自 hashcat 帮助页面）：

```
- [ 内置字符集 ] -

  ?| 字符集
 ===+=========
  L | ABCDEFGHIJKLMNOPQRSTUVWXYZ
  D | 0123456789
  H                  |
  0123456789abcdef   H
  | 0123456789ABCDEF
  s | !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
  a | | | | | | | | | | | | | l?
  b | 0x00 - 0xff
```

每个字符集都以"? "作为前缀。例如，要指所有小写字母的字符集，我们使用 ?l 。有些字符集，如"?a"，是由其他字符集组合而成的。例如，"?a "由"?l（小写字母）"、"?u（大写字母）"、"?d（数字）"和"?s（符号）"组合而成。

我们现在可以指定模式，例如

- `l?l?l?l`：指长度为 4 的密码，每个字符均为小写。
- `?u?l?l?l`：指长度为 4 的密码，以一个大写字母开头，后面跟三个小写字母。

例如，以下命令搜索长度为 4 个字符的密码，每个字符都是大写字母。

敲击
```
hashcat -a 3 -m 0 target_hashes ?u?u?u?u?u -o output_file --force
```

也可以通过组合内置字符集来定义自定义字符集。例如，下面的命令使用选项 -1 ?l?d定义了自定义字符集 ?1，它是小写字母和数字的组合。

敲击
```
hashcat -a 3 -m 0 target_hashes -1 ?l?d ?u?1?1?1 -o output_file --force
```

该命令强制输入四个字符的密码，其中第一个字符为大写字母，其余三个字符均为小写字母或数字。

练习 **6**

使用暴力破解模式破解 md5_hashes.txt 中的更多密码哈希值。你恢复了哪些新密码？请注意，暴力破解的复杂程度与模式的长度成指数关系，因此您可能需要限制暴力破解模式的长度，并限制用于较长模式的字符集的大小。

基于规则的攻击

这也许是最先进的方法。其基本思想是根据某些启发式方法，使用现有的字典单词生成新的候选密码。启发法可以是一些众所周知的密码变换，例如，在单词末尾或开头添加数字，用数字代替某些字母等。

这种攻击模式的语法与字典攻击相同，但增加了一个选项 -r：

敲击
```
hashcat -a 0 -m 0 target_hashes dictionary_file \
-r rule_file -o output_file --force
```

选项 -r rule_file 表示应根据 rule_file 文件中的规则来转换 dictionary_file 中的单词。

注意：bash 命令末尾的反斜杠字符 \（后跟换行符）会告诉 bash 该命令继续到下一行。如果你想把（很长的）命令分成两行或更多行以增加可读性，这个命令就很有用。如果你愿意，上面的命令也可以输入为一行（不带换行符）。

下面是一些简单规则的示例。更多详情，请参阅 hashcat 维基页面 (https://hashcat.net/wiki/doku.php?id=rule_based_attack)。

名称	功能	说明	示例规则	输入字	输出字
无		什么也不做（直通）	:	p@ssW0rd	p@ssW0rd
小写	l	所有字母小写	l	p@ssW0rd	大写字母 u
	全部大写字母		u	p@ssW0rd	P@SSW0RD
大写	c	第一个字母大写，其余字母小写	c	p@ssW0rd	P@ssw0rd
向左旋转	{	将单词向左旋转。	{	p@ssW0rd	@ssW0rdp
向右旋转	}	将字向右旋转	}	p@ssW0rd	dp@ssW0r
附加字符	$X	在末尾添加字符 X	$1	p@ssW0rd	p@ssW0rd1
替换	sXY	替换所有实例的 X 与 Y	ss$	p@ssW0rd	p@$$W0rd
预置字符	^X	将字符 X 添加到前面	^1	p@ssW0rd	1^ p@ssW0rd

这些函数可以组合成更复杂的变换。例如

```
c ^2^1
```

会将 "password "转换为 "12Password"，而

```
sa4 $1$2
```

会将 "password "转换为 "p4ssword12"。

规则文件只是一个规则列表。规则文件中的每条规则都适用于字典中的每个单词；因此，如果有 N 条规则，那么 hashcat 将生成字典中单词数量的 N 倍。

在 /usr/share/hashcat/rules/ 目录中可以找到一些规则文件示例。但请尽量通过编写自己的规则来解决下一个练习，只有在遇到困难时才参考这些示例规则。

练习 **7**

使用基于规则的攻击从 `md5_hashes.txt` 中恢复剩余的密码。您使用了哪些规则?

练习 **8**

为了了解良好的密码散列算法（如 linux crypt）对密码攻击的影响，我们现在将尝试破解使用基于 SHA512 的 linux crypt 算法生成的密码散列。为此，我们首先为一个非常简单的密码（"hello"）生成密码哈希值，并将其保存到名为 `crypt_hash.txt` 的文件中。

敲击

```
$ openssl passwd -6 "hello" > crypt_hash.txt
```

这将生成一个加盐密码哈希值。盐值是随机生成的，因此重复执行该命令可能会得到不同的哈希值。现在使用 hashcat 尝试破解刚刚生成的 `crypt_hash.txt` 中的哈希值：

敲击

```
$ hashcat -a 0 -m 1800 crypt_hash.txt /usr/share/dict/cracklib-small \
-o crypt_out.txt -O --force
```

这里我们使用 `-m 1800` 选项，告诉 hashcat 哈希类型是 SHA512 加密。我们使用 `-O` 来告诉 hashcat 使用优化选项来加快哈希计算速度（但代价是它能处理的盐的长度会减少）。

比较这次尝试中的哈希率和练习 5 中的哈希率。SHA512 加密的哈希率慢多少？

请注意，您可能会看到 hashcat 显示这一行：

敲击
`[状态 [P]ause [B]ypass [C]heckpoint [Q]uit =>`

这意味着 hashcat 认为密码破解需要一段时间，所以会给你一个选项，例如检查当前破解状态 (s) 或退出 (q)。在这种情况下，你只需等待即可，因为破解时间不会超过 3-4 分钟。

更多阅读

下列教程

- https://laconicwolf.com/2018/09/29/hashcat-tutorial-the-basics-of-cracking-passwords- with-hashcat/
- https://laconicwolf.com/2019/03/29/hashcat-tutorial-rule-writing/

中包含了一些更多的技巧和窍门，尤其是混合攻击，你可能会发现这些技巧和窍门很有用。hashcat wiki (https://hashcat.net/wiki/) 包含更高级攻击的全面资源列表。