【软件安全】 实验 7

密码技术应用实验*

指导教师：刁文瑞
山东大学 网络空间安全学院
Email: diaowenrui@sdu.edu.cn

December 21, 2022

1 实验概览

本实验的学习目标是让学生熟悉密钥加密与单向散列函数相关的概念。完成实验后，学生应该能够获得有关加密算法、加密模式、单向散列函数的第一手经验。此外，学生将能够使用工具和编写程序来加密/解密消息，为给定的消息生成单向散列值。

开发人员在使用加密算法和模式时会犯许多常见的错误。这些错误会削弱加密的强度，并最终导致出现漏洞。本实验向学生展示一些错误，并要求学生发起攻击以利用这些漏洞。

1. 密钥加密。
2. 加密模式、初始向量（IV）和填充（Padding）。
3. 使用加密算法的常见错误。
4. 使用密码库进行编程。

本次实验由助教李文智 负责答疑与实验报告批改。

2 实验报告

- 本次课程作业提交截止日期为 **2022 年 12 月 29 日 11:59PM**。
- 实验报告应附有**关键步骤解释与截图**，**回答思考题**。
- 实验报告应使用 **PDF 格式**，文件命名为（文件名无空格）：**【软件安全】实验报告 7−学号−姓名.pdf**。
- 实验报告模板已经提供下载。
- 使用山大云盘（https://icloud.qd.sdu.edu.cn:7777）提交作业，路径：群组文件-> 软件安全 2022-> 实验报告 7。截止日期后，文件上传权限将自动关闭。

*本次实验基于 SEED Labs 的 Crypto Lab − Secret-Key Encryption 部分：https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Encryption/与 MD5 Collision Attack Lab 部分：https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_MD5_Collision/。本中文翻译版得到山东大学教育教学改革研究项目（No. 2022Y285）资助。

3 实验任务

完成 Task 1（10 分）、Task 2（20 分）、Task 3（20 分）、Task 4（30 分）、Task 5（10 分）、Task 6（10 分）、Task 7（30 分），合计 130 分。

3.1 环境设置

在本实验中，我们将使用 OpenSSL 命令和库。我们已在 VM 中安装了 OpenSSL。

在本实验中，我们需要能够查看和修改二进制格式的文件，你可以使用任意一种十六进制编辑器。我们在 VM 中安装了一个名为 `Bless` 的十六进制编辑器。它允许用户从任何文件加载数据，以十六进制或 ASCII 格式查看和编辑它。

实验任务中所需文件已经提供，包括：**`plain.txt`**、**`pic_original.bmp`**、**`words.txt`**、**`original.txt`**。

3.2 Task 1：使用不同的密码算法和加密模式加密

在此任务中，我们将使用各种加密算法和模式。你可以使用以下 `openssl enc` 命令来加密/解密文件。要查看手册，你可以输入 `man openssl` 和 `man enc`。

我们提供了一个（明文）文本文件用作测试：**`plain.txt`**

```
$ openssl enc ciphertype -e  -in plain.txt -out cipher.bin \
-K  00112233445566778889aabbccddeeff \
-iv 0102030405060708
```

请将 `ciphertype` 替换为特定的加密算法，例如`-aes-128-cbc`，`-aes-128-cfb`，`-bf-cbc` 等。这个任务，你应该尝试至少 3 种不同的加密算法和模式组合。你可以通过键入“`man enc`”找到命令行选项和所有支持的密码类型的含义。我们在下面列出了 `openssl enc` 命令中包含的一些常用选项：

```
-in <file>     input file
-out <file>    output file
-e             encrypt
-d             decrypt
-K/-iv         key/iv in hex is the next argument
-[pP]          print the iv/key (then exit if -P)
```

3.3 Task 2：加密模式：ECB vs. CBC

文件 **`pic_original.bmp`** 包含一个简单的图片。我们想加密这张图片，所以没有加密密钥的人无法知道图片中的内容。请分别使用 ECB（电子密码本）和 CBC（密码块链接）模式加密文件。比如，你可以使用

- CBC 模式：`-aes-128-cbc`
- ECB 模式：`-aes-128-ecb`

然后执行以下操作：

(1) 让我们将加密的图片视为图片，并使用图片查看软件来显示它。但是，对于`.bmp` 文件，前 54 个字节包含有关图片的头部信息，我们必须正确设置它，因此加密文件可以被视为合法的`.bmp` 文件。我们将加密图片的头部替换为原始图片的头部。你可以使用十六进制编辑器工具（例如 `Bless`）直接修改二

进制文件。我们还可以使用以下命令从 `p1.bmp` 中获取 header，从 `p2.bmp` 中获取数据（从偏移量 55 到文件末尾），然后将 header 和数据组合到一个新文件中。

```
$ head -c 54 p1.bmp > header
$ tail -c +55 p2.bmp > body
$ cat header body > new.bmp
```

（2）使用任意图片查看软件显示加密图片。在 ECB 模式和 CBC 模式下，你分别能得到关于加密图片的原始图片的任何有用的信息吗？请解释你的观察。

请你自己选择一张图片，重复上面的实验并报告你观察到的现象。

3.4 Task 3：错误传播 – 被破坏的密文

为了理解各种工作模式的在错误传播上的性质，请做以下练习：

1. 创建一个至少 1000 字节长的文本文件。
2. 使用 AES-128 算法加密文件。
3. 不幸的是，加密文件中第 55 个字节的某一个 bit 已损坏。你可以使用 `bless` 十六进制编辑器来破坏该文件。
4. 使用正确的密钥和 IV 解密损坏的密文文件。

请回答以下问题：如果工作模式是 ECB 或 CBC ，你分别能从解密后的损坏文件中恢复出多少信息？请在做这个任务之前回答这个问题，然后在完成此任务之后看看你的答案是否正确。给出你的理由。

3.5 Task 4：寻找密钥

你得到一个明文和一个密文，你知道 `aes-128-cbc` 用于从明文生成密文。你得到的另一条线索是，加密此明文使用的密钥是一个少于 16 个字符的英语单词。这个单词可以从英语字典中找到。由于这个单词少于 16 个字符（即 128 bits ）[1]，在单词的结尾附加了一些井号（`#`：十六进制值是 `0x23`）构成一个 128 bits 的密钥。

你的目标是编写一个程序来找出这个密钥，你可以使用任意的编程语言（如 Java、C、Shell）来实现你的分析。你也可以在程序中直接调用 OpenSSL 命令来使用加密算法。

你可以从 Internet 下载英文单词列表，我们也提供了一个：**`words.txt`**。

明文、密文和 IV 如下：

```
Plaintext (total 21 characters): This is a top secret.
Ciphertext (in hex format):  7827baf49c2464156528d99a0fcda1c7
                             14d7cb591fbed2364c41a9fc03596cc4
IV (in hex format):          0102030405060708090a0b0c0d0e0f
```

Note 1： 如果你选择将明文消息存储在文件中，并将文件提供给你的程序，则需要检查文件长度是否为 21。某些编辑器可能会在文件的末尾添加一个特殊字符。如果发生这种情况，你可以使用十六进制编辑器工具删除特殊字符。

存储消息最简单的方法是使用下面的命令（`-n` 标志使 `echo` 不在末尾添加换行符）：

[1]在计算机的存储单元中，一个 ASCII 字符占一个字节（8 个二进制位），其最高位（b7）用作奇偶校验位。

```
$ echo -n "This is a top secret." > file
```

Note 2： OpenSSL 命令需要 16 进制的密钥，故单词需要进行 ASCII to Hex 转换后使用。

Note 3： 如上所述，除了使用 OpenSSL 命令，你也可以使用其他任意的编程语言和密码库来实现，但可能需要自行考虑填充问题与密钥格式问题。在下面的链接中可以找到调用 openssl 密码库的示例代码: https://www.openssl.org/docs/man1.0.2/crypto/EVP_EncryptInit.html。

Note 4： 请在实验报告中提供程序完整源代码（文本格式，非截图）。

3.6 Task 5：生成消息摘要

在此任务中，我们将使用各种哈希算法。你可以使用以下 `openssl dgst` 命令生成文件的哈希值。要查看手册，你可以输入 `man openssl` 和 `man dgst`。

```
$ openssl dgst dgsttype filename
```

请使用特定的哈希算法替换 `dgsttype`，例如`-md5`，`-sha1`，`-sha256` 等。在此任务中，你应该尝试至少 3 种不同的算法，并描述你的观察。你可以通过键入“`man dgst`”找到支持的哈希算法。

我们提供了一个（明文）文本文件用作测试：**`plain.txt`**

3.7 Task 6：哈希函数的输出特性

要了解哈希函数的输出特性，我们希望对 SHA1 执行以下练习：

1. 使用 SHA1 算法为 **`plain.txt`** 生成哈希值 H_1。
2. 修改输入文件的一位。你可以使用 `Bless` 来完成此修改。
3. 为修改后的文件生成哈希值 H_2。

请观察 H_1 和 H_2 是否相似。请在实验报告中描述你的观察结果。

3.8 Task 7：单向性与抗碰撞性

哈希函数的具有单向性与抗碰撞性。在这个实验里我们将使用暴力穷举方法来测试抗碰撞性，即找到具有相同哈希值的两个文件。**你的目标是，在给定一个文件，编写一个程序来找出另一个具有相同哈希值的文件**，你可以使用任意的编程语言（如 Java、C、Shell）来实现你的分析。你可以在程序中直接调用 OpenSSL 命令来使用哈希算法。

由于大多数哈希函数的抗碰撞性的抵御暴力攻击的能力非常强大，因此使用暴力穷举方法攻破它们需要数年时间。为了使任务可行，我们将哈希值的长度减少到 24 位。我们在此任务中**只使用 SHA1 哈希值的前 24 位**。即，可以理解成我们使用修改过的单向散列函数。请设计一个程序，找出以下内容：

1. 计算 **`original.txt`** 的 SHA1 哈希值，并记录前 24 位。
2. 使用暴力穷举方法（生成随机字符串），找到另外一个文件与上述文件具有相同的哈希值（前 24 位）。

在这个任务中，使用暴力穷举方法打破抗碰撞性需要多少次尝试？你应该进行多次实验，以获取平均值。

Note 1: 实验报告中需提供程序完整源代码（文本格式，非截图）。

Note 2: 实验报告中应展示你所找到的文件的文本内容。