实验 - 散列操作

目标

第 1 部分：使用 OpenSSL 创建散列值

第 2 部分：验证散列值

1. 背景/场景

散列函数是一种数学算法，旨在将数据用作输入并生成固定大小的唯一字符串（也称为散列）。散列函数运算速度快且很难逆转；根据单独的散列值，很难恢复用于创建任何给定散列值的数据。散列函数的另一个重要特性是，即使对输入数据进行最小程度的更改，也会生成完全不同的散列值。

虽然 OpenSSL 可用于生成和比较散列值，但也可以使用其他工具。本实验中也囊括其中的一些工具。

1. 所需资源

* CyberOps Workstation VM
* 互联网接入

1. 使用 OpenSSL 创建散列值

OpenSSL 可以用作创建散列值的独立工具。要创建文本文件的散列值，请按照下列步骤操作：

* 1. 创建文本文件的散列值
     1. 在 CyberOps Workstation 虚拟机中，打开一个终端窗口。
     2. 由于要创建散列值的文本文件位于 /home/analyst/lab.support.files/ 目录中，因此请更改为该目录：

[analyst@secOps ~]$ **cd /home/analyst/lab.support.files/**

* + 1. 键入下面的命令，以在屏幕上列出 letter\_to\_grandma.txt 文本文件的内容：

[analyst@secOps lab.support.files]$ **cat letter\_to\_grandma.txt**

嗨，奶奶：

我写这封信是为了感谢同学们送给我的巧克力曲奇饼。我今天早上收到的，已经吃了一半了！它们实在是太好吃了！

祝同学们一切都好。爱同学们，

吃饼干的孙子。

* + 1. 在同一个终端窗口中，发出下面的命令来创建文本文件的散列值。该命令将使用 MD5 作为散列算法，以生成该文本文件的散列值。OpenSSL 计算出散列值后，散列值将显示在屏幕上。

[analyst@secOps lab.support.files]$ **openssl md5 letter\_to\_grandma.txt**

MD5(letter\_to\_grandma.txt)= 8a82289f681041f5e44fa8fbeeb3afb6

注意输出的格式。OpenSSL 显示使用的散列算法 MD5，后跟用作输入数据的文件的名称。MD5 散列值本身显示在等号（“=”）之后。

* + 1. 散列函数对于验证数据的完整性非常有用，无论是图像、歌曲还是简单的文本文件。即便是最小程度的更改也会生成完全不同的散列值。可以在传输前后计算散列值，然后进行比较。如果散列值不匹配，则表明数据在传输过程中被修改。

让我们修改 letter\_to\_grandma.txt 文本文件并重新计算 MD5 散列值。发出下面的命令以打开命令行文本编辑器 **nano**。

[analyst@secOps lab.support.files]$ **nano letter\_to\_grandma.txt**

使用 **nano**，将第一句**“Hi Grandma”**更改为**“Hi Grandpa”**。注意，我们只更改了一个字符，即将**“m”**更改为**“p”**。进行更改后，按 **<CONTROL+X>** 键保存修改后的文件。按**“Y”**以确认名称并保存文件。

* + 1. 现在已修改并保存了该文件，再次运行同一命令以生成文件的 MD5 散列值。

[analyst@secOps lab.support.files]$ **openssl md5 letter\_to\_grandma.txt**

MD5(letter\_to\_grandma.txt)= dca1cf6470f0363afb7a65a4148fb442

新的散列值与在项 (d) 中计算的散列值是否不同？有什么不同？

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + 1. MD5 散列值被认为较弱，容易受到攻击。更强大的散列算法包括 SHA-1 和 SHA-2。要生成 letter\_to\_grandma .txt 文件的 SHA-1 散列值，请使用下面的命令：

[analyst@secOps lab.support.files]$ **openssl sha1 letter\_to\_grandma.txt**

SHA1(letter\_to\_grandma.txt)= 08a835c7bcd21ff57d1236726510c79a0867e861

[analyst@secOps lab.support.files]$

**注：**还有其他工具可生成散列值。即，md5sum、sha1sum 和 sha256sum 可用于分别生成 MD5、SHA-1 和 SHA-2-256 散列值。

* + 1. 使用 **md5sum** 和 **sha1sum** 生成 letter\_to\_grandma.txt 文件的 MD5 和 SHA-1 散列值：

[analyst@secOps lab.support.files]$ **md5sum letter\_to\_grandma.txt**

dca1cf6470f0363afb7a65a4148fb442 letter\_to\_grandma.txt

[analyst@secOps lab.support.files]$ **sha1sum letter\_to\_grandma.txt**

08a835c7bcd21ff57d1236726510c79a0867e861 letter\_to\_grandma.txt

[analyst@secOps lab.support.files]$

使用 md5sum和sha1sum 生成的散列值是否分别与项 (g) 和 (h) 中生成的图像相匹配？请说明原因。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**注**：虽然SHA-1 尚未受到有效的影响，但计算机正变得越来越强大。预计这种自然演进将很快使攻击者能够破解 SHA-1。在积极的举措中，SHA-2 现已成为散列的推荐标准。同样值得注意的是，SHA-2 实际上是一个散列算法的大家族。SHA-2 家族由六个散列函数组成，即 SHA-224、SHA-256、SHA-384、SHA-512、SHA-512/224、SHA-512/256。这些函数分别生成长度为 224、256、384 或 512 位的散列值。

**注**：CyberOPS 虚拟机只提供对 SHA-2-224 和 SHA-2-256（分别为 **sha224sum** 和 **sha256sum**）的支持。

1. 验证散列值

如前所述，散列值的一个常见用途是验证文件的完整性。按照以下步骤，使用 SHA-2-256 散列值来验证 sample.img（从互联网下载的文件）的完整性。

* + 1. 下载 sample.img 的同时也下载了 sample.img\_SHA256.sig。sample.img\_SHA256.sig 是一个包含网站计算的 SHA-2-256 的文件。首先，使用 cat 命令显示 sample.img\_SHA256.sig 文件的内容。

[analyst@secOps lab.support.files]$ **cat sample.img\_SHA256.sig**

c56c4724c26eb0157963c0d62b76422116be31804a39c82fd44ddf0ca5013e6a

* + 1. 使用 SHA256sum 计算 sample.img 文件的 SHA-2-256 散列值:

[analyst@secOps lab.support.files]$ **sha256sum sample.img**

c56c4724c26eb0157963c0d62b76422116be31804a39c82fd44ddf0ca5013e6a sample.img

是否正确下载了 sample.img？请说明原因。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**注**：虽然比较散列值在检测传输错误方面是相对有效的方法，但还有更好的方法来确保文件没有被篡改。诸如 **gpg** 之类的工具提供了一种更好的方法来确保下载文件未被第三方修改，就是发布者意图发布的文件。