4. Hash - Mã hóa chứng thực thông điệp

4.1 TỔNG QUAN

4.1.1 Kiến thức nền tảng

Trong bài thực hành này, sinh viên sẽ tìm hiểu về mã chứng thực thông điệp và các loại hàm băm (hash), chủ yếu là MD5 và SHA kết hợp với các ứng dụng cụ thể. Để thực hành tốt, sinh viên cần tìm hiểu và nắm được các kiến thức về:

- Xác định giá trị checksum CRC
- Mã chứng thực thông điệp (MAC) & Hàm băm (Hash - MD5 và SHA)
- Các bài toán và ứng dụng liên quan: Bài toán ngày sinh nhật (

4.1.2 Môi trường - Công cụ

1. Ngôn ngữ lập trình tự chọn (C#, Java, Web, Python,...)
2. Công cụ CrypTool 2.1, pwdump7, gói OpenSSL,... *(mở rộng)*
3. Máy ảo SEED Ubuntu (Pre-built Virtual Machine Images).
   `http://www.cis.syr.edu/~wedu/seed/lab_env.html`
   Đây là máy ảo Ubuntu 12.04 đã được cài đặt sẵn các công cụ cần thiết phục vụ cho các bài thực hành trong bộ SEED Labs. Sử dụng VMWare hoặc VirtualBox để chạy máy ảo trên.

4.2 NỘI DUNG THỰC HÀNH

Lưu ý: Bài thực hành gồm 2 phần gồm phần 4.2.1 và phần 4.2.2 là nội dung từ SEEDs Lab: One-Way Hash Function and MAC. Sinh viên lưu ý hoàn thành đầy đủ các yêu cầu của Lab.

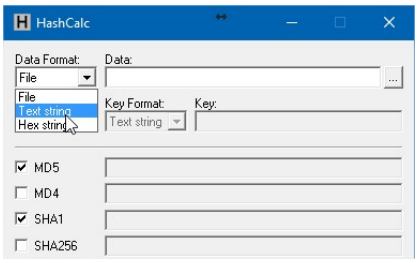
4.2.1 Hash - Hàm băm (MD5 - SHA)

Task 4.1 Tìm hiểu về phương pháp sử dụng hash MD5 - SHA-1 trong thư viện hỗ trợ của các ngôn ngữ lập trình, viết chương trình tính giá trị hash MD5 và SHA-1 cho dữ liệu dạng:

- Text
- Hex
- Tập tin trên máy

Với trường hợp tập tin trên máy, kiểm tra thêm như sau:
- Chọn 1 file bất kỳ, tính hash và gửi file đó qua email cho bạn khác. Bạn nhận được file tính giá trị hash file và so sánh với hash file ban đầu. Nếu 2 giá trị hash khác nhau thì có thể kết luận gì? Khi nào có thể xảy ra trường hợp khác nhau?
- Chọn 1 file .txt để tính giá trị Hash trước và sao khi thêm 1 ký tự khoảng trắng vào cuối file và so sánh. ■

Gợi ý 4.1. *Có thể tham khảo hoạt động của phần mềm HashCalc*



Hình 4.1: Phần mềm HashCalc

Task 4.2 Từ năm 2005, người ta đã bắt đầu phát hiện về trường hợp 2 nội dung khác nhau nhưng lại có cùng 1 hash khi sử dụng MD5, trường hợp này còn gọi là *MD5 Collision*.

1. Xét 2 thông điệp sau, sinh viên hãy tìm các bit khác nhau của 2 thông điệp (dạng hex) này. Sau đó, tính giá trị hash MD5 tương ứng của từng thông điệp và so sánh.
   - *Thông điệp 1*
     d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
     55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5b
     d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
     e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
   - *Thông điệp 2*
     d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
     55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
     d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
     e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70
2. Sử dụng CrypTool 2 với template tại *Template > MD5 Collision Finder* để thực hiện mô phỏng tạo 2 nội dung data khác nhau nhưng cùng có chung 1 hash MD5. Mô tả nguyên tắc hoạt động và tạo ra 5 mẫu data khác nhau nhưng có cùng MD5 bằng Cryptool 2.
3. Xét 2 ứng dụng **hello.exe** và **earse.exe** được cung cấp tại `https://goo.gl/fFN704`. Kiểm tra hoạt động và so sánh MD5 hash của 2 ứng dụng đó.
4. Hiện tại (tháng 5/2018), hàm băm SHA-1 có gặp phải tình trạng đụng độ (collision) như MD5 như trên hay không? Nếu có, hãy đưa ra minh chứng, hậu quả của đụng độ và đề xuất giải pháp thay thế, đảm bảo an toàn hơn khi dùng hash.

■

Mở rộng 4.1 Tìm hiểu và xác định nguyên nhân có 2 nội dung khác nhau nhưng cùng giá trị MD5 hash *(MD5 Collision)*. Đưa ra các ví dụ khác để tạo ra MD5 Collision *(nếu có)*.

Mở rộng 4.2 Tìm hiểu giải thuật hash/mã hóa để lưu trữ mật khẩu của Windows. Khi có được giá trị hash đó, có cách nào để tìm ra mật khẩu ban đầu hay không?.

Giả sử trích xuất được 3 giá trị hash mật khẩu của 3 user **u1, u2, u3** như sau, thử tìm mật khẩu tương ứng của các user đó:
u1:1003:NO PASSWORD**********:8846F7EAEE8FB117AD06BDD830B7586C:::
u2:1004:NO PASSWORD**********:C705696627D5DE4C57E4E78E01A14EAA:::
u3:1005:NO PASSWORD**********:03008F60BA0146D0B9E7B21221A722C3:::
(kết quả trích xuất từ pwdump7)

4.2.2 Khảo sát tính chất One-Way và Collision-Free của MD5

Sinh viên xem nội dung *SEED Lab: One-Way Hash Function and MAC* kèm theo

4.3 YÊU CẦU - ĐÁNH GIÁ

4.3.1 Yêu cầu

Sinh viên thực hiện đầy đủ các nội dung trong phần thực hành theo nhóm (tối đa **2** sinh viên/nhóm) hoặc cá nhân *(đăng ký từ buổi 1)* và báo cáo kết quả thực hiện như sau:

- File báo cáo .pdf kết quả thực hiện các nội dung thực hành (theo mẫu).
- Đối với các ứng dụng, đặt vào thư mục có tên tương ứng với Task *(source code và file thực thi)*

Báo cáo trình bày trực quan việc thực hành có kèm hình ảnh và chú thích rõ ràng theo cách tiếp cận của bản thân, không sao chép lẫn nhau. Nếu có tham khảo cần ghi nguồn.

Thời gian hoàn thành Lab 4:
Trong tối đa **12** ngày từ buổi thực hành tại lớp.

Lưu ý 4.3.1 Đặt tất cả báo cáo và thư mục liên quan vào 1 tập tin nén (.zip,.rar) có tên:

[MSSV1]-[Tên SV1]_[MSSV2]-[Tên SV2]_MMH_LabX

Ví dụ: 16520901-Nhut_16521516-Lam_MMH_Lab4. Kích thước tối đa: 10MB.
Nộp báo cáo theo thời gian quy định tại website môn học.

4.3.2 Đánh giá

- Sinh viên hiểu và hoàn thành tốt nội dung thực hành, đúng hạn: 80%
- Sinh viên trình bày cụ thể, rõ ràng kết quả thực hiện: 5-10%
- Khuyến khích sinh viên có tìm hiểu, thực hiện các nội dung mở rộng: 15-20% hoặc cộng trực tiếp vào điểm tổng kết thực hành nếu hoàn thành xuất sắc. *Nội dung mở rộng khuyến nghị thực hiện với sinh viên lớp ANTN.*
- Sao chép bài của sinh viên khác, tham khảo không ghi nguồn: -50% đến -100%.
- Nộp bài trễ: -20% mỗi ngày nộp trễ.

Lưu ý 4.3.2 Báo cáo nộp trễ, sao chép sẽ được xử lý tùy theo mức độ. Các nội dung báo cáo có thể được vấn đáp để đánh giá kết quả tại buổi thực hành tiếp theo, sinh viên vắng thực hành không có lý do thì nhóm sinh viên tương ứng sẽ được trừ tối thiểu 30% điểm bài thực hành đó.

4.4 TÀI LIỆU THAM KHẢO

[1] W. Stallings, *Cryptography and network security: Principles and practice, 6th ed*. Boston, MA, United States: Prentice Hall, 2013. *Chapter 11 -13*
[2] MD5 Collision, Available: `http://www.mscs.dal.ca/~selinger/md5collision/`.

Lab 4.2.2 – One-Way Hash Function and MAC

(Nội dung này được trích từ SEED Lab, dùng cho sinh viên UIT)

1 Overview

The learning objective of this lab is for students to get familiar with one-way hash functions and Message Authentication Code (MAC). After finishing the lab, in addition to gaining a deeper undertanding of the concepts, students should be able to use tools and write programs to generate one-way hash value and MAC for a given message.

2 Lab Environment

Installing OpenSSL. In this lab, we will use `openssl` commands and libraries. We have already installed `openssl` binaries in our VM. It should be noted that if you want to use `openssl` libraries in your programs, you need to install several other things for the programming environment, including the header files, libraries, manuals, etc. We have already downloaded the necessary files under the directory `/home/seed/openssl-1.0.1`. To configure and install `openssl` libraries, go to the `openssl-1.0.1` folder and run the following commands.

```
You should read the INSTALL file first:

  % sudo ./config
  % sudo make
  % sudo make test
  % sudo make install
```

Installing a hex editor. In this lab, we need to be able to view and modify files of binary format. We have installed in our VM a hex editor called `GHex`. It allows the user to load data from any file, view and edit it in either hex or ascii. Note: many people told us that another hex editor, called `Bless`, is better; this tool may not be installed in the VM version that you are using, but you can install it yourself using the following command:

```
% sudo apt-get install bless
```

Note: You can use SEED pre-built Ubuntu virtual machine image (with OpenSSL and GHex are already installed) at `http://www.cis.syr.edu/~wedu/seed/lab_env.html`

3 Lab Tasks

3.1 Task 1: Generating Message Digest and MAC

In this task, we will play with various one-way hash algorithms. You can use the following `openssl dgst` command to generate the hash value for a file. To see the manuals, you can type `man openssl` and `man dgst`.

```
% openssl dgst dgsttype filename
```

Please replace the `dgsttype` with a specific one-way hash algorithm, such as `-md5`, `-sha1`, `-sha256`, etc. In this task, you should try at least 3 different algorithms, and describe your observations. You can find the supported one-way hash algorithms by typing "`man openssl`".

3.2 Task 2: Keyed Hash and HMAC

In this task, we would like to generate a keyed hash (i.e. MAC) for a file. We can use the `-hmac` option (this option is currently undocumented, but it is supported by `openssl`). The following example generates a keyed hash for a file using the HMAC-MD5 algorithm. The string following the `-hmac` option is the key.

```
% openssl dgst -md5 -hmac "abcdefg" filename
```

Please generate a keyed hash using HMAC-MD5, HMAC-SHA256, and HMAC-SHA1 for any file that you choose. Please try several keys with different length. Do we have to use a key with a fixed size in HMAC? If so, what is the key size? If not, why?

3.3 Task 3: The Randomness of One-way Hash

To understand the properties of one-way hash functions, we would like to do the following exercise for MD5 and SHA256:

1. Create a text file of any length.
2. Generate the hash value H_1 for this file using a specific hash algorithm.
3. Flip one bit of the input file. You can achieve this modification using `ghex` or `Bless`.
4. Generate the hash value H_2 for the modified file.
5. Please observe whether H_1 and H_2 are similar or not. Please describe your observations in the lab report. You can write a short program to count how many bits are the same between H_1 and H_2.

3.4 Task 4: One-Way Property versus Collision-Free Property

In this task, we will investigate the difference between hash function's two perperties: one-way property versus collision-free property. We will use the brute-force method to see how long it takes to break each of these properties. Instead of using `openssl`'s command-line tools, you are required to write our own C programs to invoke the message digest functions in `openssl`'s crypto library. A sample code can be found from `https://www.openssl.org/docs/man1.0.2/crypto/EVP_DigestInit.html`. Please get familiar with this sample code.

Since most of the hash functions are quite strong against the brute-force attack on those two properties, it will take us years to break them using the brute-force method. To make the task feasible, we reduce the length of the hash value to 24 bits. We can use any one-way hash function, but we only use the first 24 bits of the hash value and ignore the rest in this task. Namely, we are using a modified one-way hash function. Please design an experiment to find out the following:

1. How many trials it will take you to break the one-way property using the brute-force method? You should repeat your experiment for multiple times, and report your average number of trials.
2. How many trials it will take you to break the collision-free property using the brute-force method? Similarly, you should report the average.
3. Based on your observation, which property is easier to break using the brute-force method?
4. (10 Bonus Points) Can you explain the difference in your observation mathematically?

4 Submission

You need to submit a detailed lab report to describe what you have done and what you have observed; you also need to provide explanation to the observations that are interesting or surprising. In your report, you need to answer all the questions listed in this lab.
Remember to use report template and follows the same structure as previous reports.