Hash Length Extension Attack

1st Junfei Wang
UMass Amherst

ECE 644 Final project
Instructor: Arman Pouraghily

junfeiwang@umass.edu

***Abstract*—In this project, I will simply discuss the Hash Length Extension Attack and use it to attack the MD5 system. I will show how to achieve the Hash Length Extension Attacks through the MD5 system, try to produce a demo to simulate the attack step by step, and record what I learn in the final report. Screenshots and detailed descriptions should be included.**

***Keywords*—hash, security, attack, MD5**

I. Introduction

Hash is to transform an input of any length (also called pre-image) into a fixed-length output through a hashing algorithm, and the output is the hash value. This conversion is a compression mapping, that is, the hash value space is usually much smaller than the input space, and different inputs may be hashed into the same output, so it is impossible to uniquely determine the input value from the hash value. Simply put, it is a function that compresses messages of any length to a fixed-length message digest.

Hash and Encrypt are completely different concepts, and the correct distinction between the two is the basis for the correct choice and use of hash and encryption. The biggest difference between hashing and encryption is:

Hash converts the target into an irreversible hash string of the same length, while encryption converts the target into a reversible ciphertext of different lengths. The length generally increases as the plaintext grows. If the protected data is only used for comparison verification and does not need to be restored to plaintext in the future, the hash is used; if the protected data needs to be restored to plain text in the future, encryption is required.

But Nowadays, Most network connection protocols need to maintain the confidentiality of messages and protect their integrity. For example, a file download interface uses the SSL record protocol to connect to the target server and uses the MD5 algorithm to authenticate the validity of its download. The SSL(security socket layer) connection provides two services of confidentiality and message integrity. The asymmetric encryption algorithm RSA is used to provide a key exchange, the symmetric encryption algorithm DES is used to encrypt messages, and the hash algorithm MD5 is used to provide message authentication codes. If we use this interface to download other files, the download may fail because of the mismatched message authentication code. But attacker uses a hash length expansion attack to counterfeit exactly the correct MAC(MD5 value) when using different length inputs, it may make this attack successful.

In a real attack environment, the attacker cannot know the length of the message and needs to guess its length. Continuing the previous example, suppose that this vulnerable interface will return an error message (HTTP response code) when MAC authentication fails. An error message will also be returned when the verification is successful, but the file does not exist. If the two error messages are not the same, the attacker can calculate different extension values, each corresponding to a different length, and then send them to the server separately. When the server returns an error message indicating that the file does not exist, it means that there is a hash length extension attack. The attacker can calculate a new extension value at will to download unlicensed sensitive files on the server.

II. MD5 Algorithm

The algorithm takes as input a message with 512-bit blocks and produces as output a 128-bit message digest. The processing consists of the following steps.

Step1: Append padding bits

The message is padded so that its length is congruent to 448 modulo 512 [length $\equiv$ 448 (mod 512)]. Even if the message is already the desired length, it is always necessary to add padding. Therefore, the range of padding bits is 1~512. The padding part is composed of a single bit 1 followed by the required number of bits 0.

Step2: Append length

Append a 64-bit data block to the message. The data block is regarded as a 128-bit unsigned integer (high byte first), and it also contains the length of the original message (before padded). The first two steps generate a message whose length is an integer multiple of 512 bits.

Step3: Initialize hash buffer

A 128-bit buffer is used to store the intermediate and final results of the hash function. The buffer can be four 32-bit registers (a, b, c, d). These registers are used to initialize 32-bit integers.

Step4: Process message in 512-bit (64-word) blocks

The core of the algorithm is a module composed of 16 rounds. In each round, the 128-bit buffer value a,b,c,d is used as input, and the buffer type content is updated.

Step5: Output

When all N 512-bit data blocks are processed, the output from the Nth stage is a 128-bit message digest.

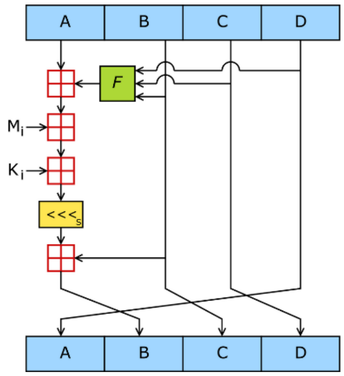


Fig. 1. MD5 algorithm

III. Hash Length Extension Attack

Hash length extension attack means when only know the length of the ciphertext and the hash of the ciphertext, the MD5 algorithm can be used to predict the hash after the ciphertext is concatenated with another message.

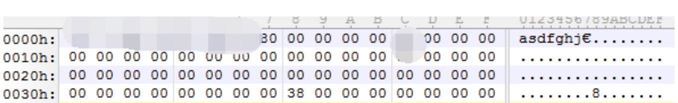
```python
# coding:utf-8
import hashlib
import sys
from urllib.parse import unquote


def login(password, hash_val):
    m = hashlib.md5()
    secret_key = "message"
    m.update((secret_key + password).encode('utf-8'))
    if (m.hexdigest() == hash_val):
        print("Login Successful!")
    else:
        print("Login Failed")


if __name__ == "__main__":
    password = unquote(sys.argv[1])
    hash_val = unquote(sys.argv[2])
    login(password, hash_val)
```

Fig. 2. Authentication program

If the length of a message is greater than 512 bits, the message will be divided into 512-bit blocks, and the last message block will be appended with pad bits and length. Suppose we know that the MD5 value of a 7-byte or 56-bit message is 78e731027d8fd50ed642340b7c9a63b3. When the MD5 algorithm operates on it, it will append pad bit and length first. Since we don't know the content of the message, the result of the bit appended is as shown in the figure below:



Then appended result will perform complex function operations with the initial hash buffer, because the MD5 value is 78e731027d8fd50ed642340b7c9a63b3, so the result is:
a = 0x0231e778
b = 0x0ed58f7d
c = 0x0b3442d6
d = 0xb3639a7c

If a message string "admin" is added to the appended message, the message will append the pad bit and length, and then the value calculated by the previous operation will be used as the initial hash buffer to perform the function operation. The hash value obtained is e53a681a30ff99e3f6522270ca7db244.

This completes the calculation of the hash value of the message + padding and length + appended message without knowing the message.

We can verify it, assuming that the program for verifying user login is like Fig.2.

Now we only know a set of user names and hash values: "root" and f3c36e01c874865bc081e4ae7af037ea

```
~/md5hashlengthextensionattack » python example.py root f3c36e01c874865bc081e4ae7af037ea
Login Successful!
```

From the analysis, we know that when we know the length of the message, we can forge the padding and length, and then we can calculate the MD5 value of message+padding and length+additional string by appending the string. Assuming that the string we append is "admin", the hash expansion attack is used to calculate md5(message+padding and length+additional string) = e53a681a30ff99e3f6522270ca7db 244. Then we test it and show that the login is successful:

```
~/md5hashlengthextensionattack » python example.py %80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%38%00%00%00%00%00%00%00admin e53a681a30ff99e3f6522270ca7db244
Login Successful!
```

IV. Summary

The principle of this attack method is that if someone knows the hash value of ciphertext and the length of the ciphertext, he can calculate the hash value after the ciphertext and the extended message are concatenated together. The prevention method is also very simple:

1. Using HMAC algorithm

Hash functions such as MD5 are not specifically designed for MAC, because the hash functions do not rely on keys. There are already many schemes to incorporate the key into the existing hash algorithm. The most widely accepted solution is HMAC. The way to prevent hash length expansion attacks is to use the HMAC algorithm. The algorithm is roughly like this: MAC = hash(key + hash(key + message)), instead of simply hashing the value after the key is connected to the message. It uses a standard algorithm to mix the key into the process of calculating the hash value. Since this algorithm performs double digests, the key is no longer affected by the hash length expansion attack.

REFERENCES

[1] https://www.liaoxuefeng.com/wiki/1016959663602400/1017686752491744

[2] http://www.jinglingshu.org/?p=7122

[3] http://blog.nsfocus.net/hash-length-extension-attack/

[4] https://xz.aliyun.com/t/2563

[5] William Stallings, “Network Security Essentials: Applications and Standards, 6th Edition”