Lab 7: MACs and the Length Extension Attack

1. The one-way hash function can be exposed to the Man in the Middle attack. When trying to preserve the integrity of the data it is common to append the hash of a message with the message itself. However, if the hash function is known then the attacker can modify the message and then append a newly generated hash to this message. The Message Authentication Code (MAC) is one solution to prevent this. In this scheme Alice and BOB share a secret key K which is incorporated into the hash algorithm and the resultant hash can then be appended to the message. Thus, without the knowledge of K an attacker cannot produce a correct hash for any modified message.

Diagram

Description automatically generated

1. There are several ways in which K can be mixed with the message M. Append K to M and then hash them together. XOR the key with the message and then hash, etc. Some of these methods are not secure and vulnerable to the length extension attack. This attack is possible because of the Merkle-Damgard construction used by MD5, SHA-1 and SHA-2. For such a construction it can be shown that for a given message X and a padding P an attacker can find the value of H (X|| P || Y) for any string Y without knowing the value of X.
2. In the case of a MAC where the secret key K is appended in front of the message M then an attacker can compute H (K||M||P||T) for any T with knowing K or M. The padding P is dependent only on the length of K||M and not its value.

Diagram

Description automatically generated

1. From the diagram above it can be seen, that if the attacker captures the value of S then the hash of S||T can be found. This means that while the attacker may not be able to change the original message, they are able to add to it.
2. Let us conduct an experiment to show how this attack may work. Assume a secret key K = “secretkey” and assume the message to be sent is M = “Launch a missile towards target A”. The SHA-256 hash of K appended(concatenated) to the message can be found as follows:



1. It should be noted that since the concatenated message is less than the 64 bytes required for SHA-256 (it is only 44 bytes), an amount of padding equal to 64-40 = 20 bytes will be added. According to RFC 6234, padding for SHA-256 is a 1 bit followed by a number of 0s and ending with a 64-bit field that specifies the length of the message.

Knowing the above, which of the following would represent the padding that is added? Highlight the correct one.

10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 60

80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 60

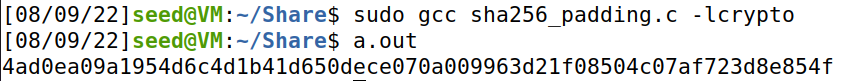
80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 52

1. Copy the C program **sha256\_padding.c** into your shared folder. The code for this program is shown below. This computes what the valid hash would be if the enemy could append the additional message   
   “**Launch a missile towards the headquarter.”** to the original. The highlighted code shows the secret key appended to the front of the original message, the padding and the added message.

Text

Description automatically generated

1. Compile and execute the code in Linux:



1. The hash value shown in the last line above is the hash of (K:M||P||T)

What is ?

K:

M:

P:

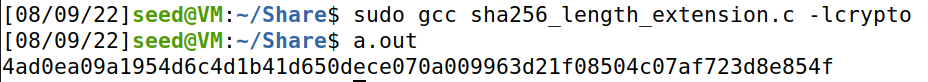
T:

Of course, this program **cannot** be executed by the enemy because they do not know the secret key. However, they can capture the hash of (K:M||P) and using the length extension attack generate a valid hash with the following code.

Text

Description automatically generated

1. Copy the file sha256\_length\_extension.c into your shared folder. Compile and execute the code. Notice that the same valid hash was computed from the extended message.



1. Repeat the above with the following:

K = “mySecretKey”

M= “Give Alice and Bob a raise.”

With the enemy (Darth) extending the message as follows:

T = “ Give Darth a raise also.”

12 – According to RFC 6234, padding for SHA-256 is a 1 bit followed by a number of 0s and ending with a 64-bit field that specifies the length of the message.

What is the padding in hexadecimal for a

a- 55-byte message?

b- 56-byte message?