COL-759
Assignment3
Report

2019MCS2554

August 2020

PROTOCOL: SSLV3

VULNERABILITY: POODLE

About SSL

SSL stands for **Secure Sockets Layer** and is a transport layer protocol that is supposed to ensure secure transmission of information between the client and the server. SSL works by implementing a handshake between the the client and the server, which involves verification of the server's identity via a digital certificate, and ends with generation of the symmetric key to be used for further encryption and decryption of the data, after the handshake has been completed.

In summary, SSL takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, and transmits the result. Received data is decrypted, verified, decompressed, and reassembled, then delivered to higher level clients.

Below is the high level overview of the steps that take place in a SSL handshake:

- The client connects to an SSL enabled server and requests a secure connection, and also presents a list of supported cipher suites.
- The server picks a suitable cipher and hash function from this list and notifies the client.
- The server then provides identification in the form of digital certificate, which contains the server name, the trusted certificate authority and the server's public encryption key.

- The client confirms the validity of the certificate before proceeding.
- To generate the session(symmetric) keys used for secure connection, the client either encrypts a random number with the server's public key or uses the Diffie-Hellman exchange.
- This concludes the handshake and begins the secured connection, which is encrypted and decrypted with the session key until the connection closes.

If any of the above steps fails, then the handshake fails as well and the connection is not created.

SSLV3 and what makes it vulnerable

SSLV3 is the third iteration of SSL and was released in 1996 in hopes of overcoming the numerous security flaws in SSLV2. SSL went through a complete redesign for SSLV3.
The features of SSLV3 are listed below:

- SSLV3 achieves encryption via **cipher block chaining**, which involves segmenting the message into manageable blocks, and the encryption of one block depends directly on the encryption of all the previous blocks.
- SSLV3 uses **MAC i.e. Message Authentication Code** which is a set of algorithms that allows the receiver to check the integrity of the message and the identity of the sender. MAC involves the generation of a key, selected randomly from the key space, encrypting the ciphertext with this key and appending this new message to the ciphertext payload. This new key is made available to the receiver so that the receiver can encrypt the ciphertext it receives and compare it with the MAC encryption appended to the ciphertext. If both match, then the data has not been tampered with and the receiver proceeds to process the data further, otherwise, the data has been tampered with and the receiver drops the connection.
- SSLV3 uses the concept of **padding** to make the size of the payload a multiple of a certain number of bytes(usually, 8 or 16), as is standard for many protocols. To achieve this, random numbered bytes are added in the form of padding, and at the end, a number is added which represents the size of the padding.
- A typical SSLV3 payload is sequenced as follows:- Ciphertext->MAC->Padding.

It was found that the padding scheme used in SSLV3, while simple and easy to deal with, contained a subtle flaw that can be exploited by a man in the middle

in various harmful ways.
If the attacker/man in the middle can get his hands on the message from the sender to the receiver, he can easily tamper with the message in such a way that replaces the last character of the message containing the padding size of the message with a character from the ciphertext of the message. If the replaced character matches the size of the padding, then the attacker has managed to figure out one character from the encrypted message, and this process can be repeated multiple times. This the POODLE attack and has been explained in detail below.

POODLE

POODLE stands for **Padding Oracle on Downgraded Legacy Systems**. It is a man in the middle attack that takes advantage of the flawed padding scheme of SSLV3 to illegally glean private information about the client, which can later be used to access the client's accounts or act on the client's behalf.

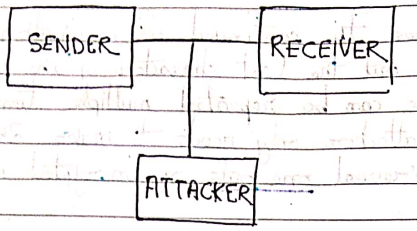
How SSLV3's padding scheme enables a POODLE attack

- Suppose there's a man in the middle eavesdropping on the conversation between the sender and the receiver and manages to obtain a message sent from the sender.
- Since SSLV3 follows cipher block chaining for encryption, the attacker can pick one a block at random from the message and replace the last block of the message with the picked block.
- We know that the last block of the message contains the padding and the last character is the size of the padding, which effectively gets replaced by the last character of the replacing block picked by the attacker.
- The attacker now reroutes the message back to the receiver. The receiver, upon receiving the message, will check the validity of the message using MAC. Since the client knows that the MAC of the message lies just before the padding block, he will read the last character of the block to obtain the size of the padding so that the padding block can be skipped and the MAC can be obtained.
- However, since the attacker replaced the last block of the message, it is likely that the client will read an incorrect size for the padding, and read the wrong bytes for the MAC, which will lead to the MAC not matching with the receiver's own computed MAC, which will lead to the receiver terminating the connection, as it senses tampering of the message.
- However, there's a small but significant chance of the **last character in the last block of the tampered message actually matching the**

size of the padding. In that case, the receiver will obtain the correct MAC, and will sense no tampering, thus continuing the communication with the sender.

- The attacker will sense the continued communication, and will thus **learn the last character of the block it replaced the padding block** of the original message with. This process can be continued. By repeated shifting of characters, the attacker can learn the contents of the entire block, and over a course of many iterations, learn the contents of multiple blocks.
- In this way, the attacker can get hands on the personal information of the sender embedded in the message.

Below is an on-paper implementation of the steps the attacker will take, with a few pictorial representations as well (For simplicity, it uses a "sender-receiver" nomenclature, but it can easily be extended to a "client-server" system):



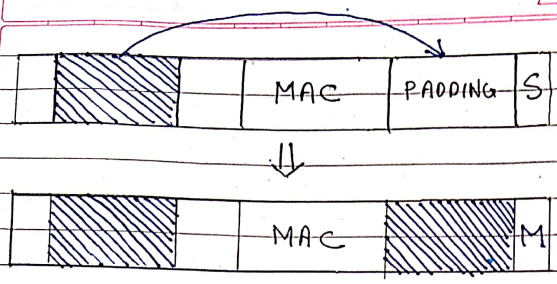
-) Suppose an attacker is eavesdropping on the conversation between a sender and a receiver
-) SSLV3 is deprecated and not used anymore, however many browsers still support it, to ensure backwards compatibility. So the first step the attacker needs to take is to trick the sender and the receiver during the handshake process to use SSLV3 for communication.

-) Once the connection has been established, the attacker can intercept the messages the sender sends to the receiver. Here's what an intercepted payload looks like.

CIPHERTEXT	MAC	PADDING	S

Here, 'S' represents the last character of the padding which denotes the size of the padding in bytes.

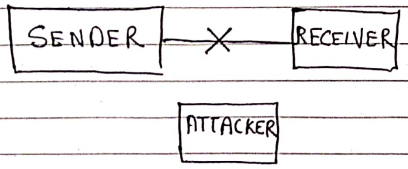
-) SSLV3 uses cipher-block chaining to encrypt the messages, and it involves dividing the payload into manageable blocks. Using this knowledge, the attacker can take a block from the ciphertext and replace the padding block with this block.



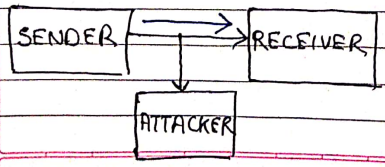
'M' denotes the last character in the replaced block.

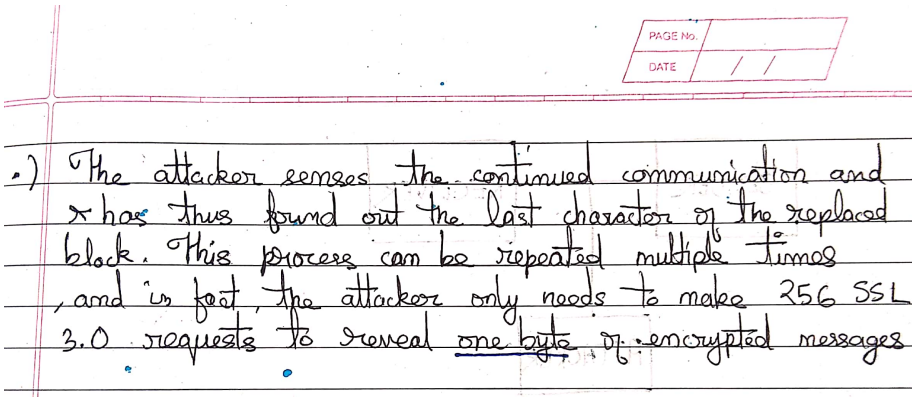
- The attacker now re-routes this tampered message back to the receiver.

- There's a 255 in 256 chance that $M \neq S$, which means the client will read the wrong bytes for the MAC, which will lead to a MAC mismatch and the client / receiver terminates the connection.



- However, there's a 1 in 256 chance that $M = S$, which means that the receiver will read the correct bytes for the MAC. In this case, the receiver will be satisfied and the communication will continue.





Impact of POODLE and measures to safeguard against it:

- If a POODLE attack is successful, then the attacker can gain access to sensitive information about the sender. For example, a lot of websites support **cookies**, which stores session information for the user and allow the user to access their accounts on websites without having to log in repeatedly. If the man-in-the-middle/attacker intercepts the transmission of a message cookie from the client to the server, then he'll gain access to the account info of the client.
- This allows the attacker to act on behalf of the client, which can cause a lot of trouble for the client.
- To prevent a POODLE attack, the most straightforward measure is to completely disable SSLV3 on both the client and server sides, to prevent them from getting tricked into using SSLV3 by the attacker. However, some old clients and servers do not support TLS 1.0 and above. It is encouraged in that case to use TLS_FALLBACK_SCSV, which will make downgrade attacks impossible.
- Another way to mitigate POODLE attacks is to implement "anti-POODLE record splitting". It splits the records into several parts and ensures none of them can be attacked. However the problem of the splitting is that, though valid according to the specification, it may also cause compatibility issues due to problems in server-side implementations.