Chapter 3

Interception of encrypted data, attacks on SSL sessions

Today, nearly everyone has had contact with a computer or the internet. It is now commonplace to order something off the internet or to take care of an important matter online. Because of this, data security is paramount. Every online retailer and online bank uses encryption; so, for that matter, do instant messaging programs. Our data are private, or so we are led to believe. In fact, as we shall soon see, encryption gives us a false sense of security.

At present the most popular encryption protocols are SSL (Secure Socket Layer) and SSH (Secure SHell). Both of these protocols use what is known as hybrid encryption.

A hybrid cryptographic system possesses advantages of both symmetrical and asymmetrical encryption. An asymmetrical key is used for exchanging a randomly generated key, which is used in turn to encode the remainder of the communication by means of symmetrical encryption. In this way the user is able to profit from the speed and performance of symmetrical encryption, while avoiding the problems related to the unsecured exchange of keys.

Because most applications using encryption are resistant to cryptanalysis, cracking this encryption by itself has little chance of success. However, if a hacker manages to intercept data sent between two people and impersonates one of them, it will be possible to attack the key exchange algorithm.

Cracking such a key within a short time without influencing the flow of communication is, in practice, impossible. That is exactly why the hacker does not attack the encryption mechanism but instead exploits the naivety of the user. This is called a “man in the middle attack.” The hacker intercepts one of the computers on the user’s communication node with the target server (usually a gateway). If, however, the hacker is located in the same network as the user, he can pretend to be the gateway, obtaining the same result. Attacks of this type were described in the previous chapter.

The man in the middle type of attack is not ideal and can be successful only and exclusively with the unknowing consent of the victim. Normally, communication using the SSH or SSL protocols starts by checking the “fingerprint” of the server (SSH) or a security certificate (SSL). If they are valid, the encrypted data exchange begins.

In the case of SSH, the fingerprints used during each connection are generated during installation and booting the server. If the user attempts to connect to an SSH session for the first time, he is informed that the server is unknown. In the next phase he supplies his fingerprint and is asked if he wishes to accept the sent keys.

Here is an example of this situation:

```
adam@devils:~$ ssh user@server
The authenticity of host 'server (66.66.66.66)' cannot be established.
RSA key fingerprint is 22:6f:32:7c:ef:78:6a:8c:f6:e2:b9:c1:9e:5f:66:f7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'server.com,66.66.66.66' (RSA) to the list of known hosts.
user@server.com's password:
```

The server is added to the list of known and authenticated addresses. Next time the question does not appear. If the server fingerprint changes in the future, we will receive a message notifying us about this.

```
adam@devils:~$ ssh user@server
#########################################################
WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!
#########################################################
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now
(man-in-the-middle attack)!
It is also possible that the RSA host hey has just been changed.
Please contact your system administrator.
```

The software, as we have seen, will inform us in the event we may have fallen victim to a possible man in the middle attack. The fingerprint could also have changed during system reinstallation or a change in the SSH version on the server. To check that the changed fingerprint is still valid, we should contact the administrator and ask to be given the correct key.

To carry out an attack of this kind we will use the sshmitm program from the dsniff-2.3 packet. To simplify matters, we will assume that we have successfully accessed the network gateway. We install the dsniff-2.3 packet on the computer used as a gateway and carry out the attack.

In the console window we enter the command:

```
root@gateway:~# sshmitm -I server
sshmitm: relaying to server.pl
```

The -I switch starts the program in interactive mode, meaning that it writes the intercepted data to the monitor in real time. Here, “server” is the address of the server whose session we wish to intercept. If the user has previously connected to the system, we can only hope that he will accept the connection despite the warning.

Not having access to the gateway, we can carry out the attack by changing the packet traffic between the gateway and the computer of the “victim” as described in the previous chapter (script arp.pl).

The SSL connection is created on the basis of a normal connection using the TCP protocol. The sent data are encrypted with the key negotiated at the beginning of the session. Next the server sends its certificate, and the client checks it and accepts or reports the problem if the certificate was not signed, is not valid, or the name on the certificate does not correspond to the domain name of the system.

As in the case of the wrong fingerprint in SSH we can continue or abort the connection. The socat program is ideal for carrying out the attack; it can be downloaded from:

```
http://www.dest-unreach.org/socat
```

The application provides the ability to transfer data in two directions between any two channels such as files or sockets (IPv4 and IPv6).

The program installation can proceed in the following way:

```
./configure
make
su
make install
```

If we want to eavesdrop on or to intercept the session between users and the remote server using the SSL encryption protocol, we need the certificate, access to the gateway, and the socat program described earlier. In the event we do not have access to the network gateway, it will be necessary to direct the packet traffic through our own computer in exactly the same way as described earlier in the case of SSH (script arp.pl).

The first step is to create a falsified certificate. It is necessary to install the OpenSSL packet, which is available from all the usual download sources, including as source code. We create the certificate as follows:

```
root@gateway:# /etc/ssl/misc/CA.sh -newca
```

We provide username and password protecting the certificate. These entries have to provide an appropriate match for the imitated server:

```
root@gateway:# /etc/ssl/misc/CA.sh -newreq
```

We give the information corresponding to the previous entries and sign the certificate with the command:

```
root@gateway:# /etc/ssl/misc/CA.sh -sign
```

The next task is to redirect the connections to the socat program. For these purposes, we use, for instance, iptables:

```
root@gateway:# iptables -A PREROUTING -t nat -i eth0 -p tcp -d
193.41.230.81 --dport 443 -j REDIRECT --to 443
```

All outgoing packets having the remote address corresponding to the falsified server (in this case 193.41.230.81) and the port 443 (standard SSL protocol) will be redirected back to the local port 443, where we start the socat program.

```
root@gateway:~# socat -v OPENSSL-
LISTEN:443,cert=newcert.pem,key=newreq.pem,verify=0,fork
OPENSSL:193.41.230.81:443,verify=0
```

The verbose (-v) option will cause the data to be displayed onscreen in real time. It is also possible to save the data in the file adding "2>output_file" to the end of the command, meaning the record of standard output will be sent to the file with the name "output_file."

The option:

```
OPENSSL-LISTEN:443,cert=newcert.pem,key=newreq.pem,verify=0
```

will start the SSL server eavesdropping on the port 443 with the newly generated key and certificate. The server that has been started will not verify the certificate of the client and will not end the operation after closing the connection. The data from the port end up in the computer at 193.41.230.81 at port 443 without checking the server certificate. This is defined in the option OPENSSL:193.41.230.81:443,verify=0. The person connecting to the server computer that we are impersonating will be informed about the possibility the session is being eavesdropped upon (invalid certificate, as was the case with SSH). However, most users usually do not read this when it appears in the browser, as they usually skip the warning and go to the sites that interest them.

Some programs using SSL (for example, instant messaging software), often have default settings to accept all certificates, or contain errors that mean they completely omit checking server certificates. Luckily for the user, there exist no means to force acceptance of invalid certificates without informing the user about it.

If we lack access to the network gateway, the packet traffic should be redirected using the description provided in the previous chapter. An alternative way to perform this task is presented below. It uses the arpspoof program. It is part of the dsniff-2.3 packet.

In the event the reader wishes to redirect traffic from the whole network to his own computer:

```
root@home:~# arpspoof -i eth0 <gateway_ip> &
```

If we wish to redirect the traffic from only a specific computer:

```
root@home:~# arpspoof -i eth0 -t <gateway_ip> <victim_ip> &
```

Before performing the above commands it is necessary to ensure the forwarding of the packets with the following command:

```
root@home:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```