CS 577 Cybersecurity Lab

Lab 1 – due 9/17/14 6:00pm

Subject: Man-in-the-Middle Attack Proxy

A client **C** is communicating with a server **S** through a proxy **P** to circumvent network filtering in **C**'s network. **C** does not trust **P**, so it uses SSL encryption to prevent eavesdropping. The problem is that even though **C** already has the certificate of the **S**, it is not correctly verifying it.

Deliverables

1. Your goal is to create an evil proxy **P** that eavesdrops traffic. For simplicity, the proxy should relay all TCP connections using the HTTP CONNECT method. All exchanged data should be eavesdropped and logged. For example, for each connection the data sent from client to server and vice-versa, should be logged into files for later examination. [80%]
2. Include a report.txt file with a paragraph explaining your choices when building the proxy and including instructions for building and running your software. Also describe, what would an attacker require to launch a MiTM attack, if the client did correctly verify the certificate presented by the server and certificate authorities are used for the client to verify server certificates. [20%]

Submission information

The code you submit for grading must build and run on the linux-lab, even if you use a different machine/environment for developing. Note that linux-lab.cs.stevens.edu is just an alias that connects you to one out of many hosts. Always, make sure you are not messing with anyone else's assignment.

The assignment must be done in C, C++, or Java. All the code must belong to you with the exception of certain built-in libraries (e.g., Java packages part of the JDK) and standard libraries (e.g., for C++). You can use crypto libraries like OpenSSL or equivalent for Java. The elegance of the solution will also be taken into account when grading. Permission by the instructor is required for deviating from these rules.

Submit all your files as a tar.gz archive through Canvas.

Helpful information

Generating an RSA public/private-key pair

```
Command: openssl genrsa  -out private.pem 2048
```

Generating a self-signed certificate

```
Command: openssl req -new -x509 -key private.pem -out cacert.pem -days
1095
```

Emulating a server with openssl

```
Command: openssl s_server -cert cacert.pem -key private.pem -accept
8900
```

Emulating a client with openssl and using a proxy

```
Command: openssl s_client –proxy 127.0.0.1:8000 -connect 127.0.0.1:8900
-state -verify 1 -CAfile cacert.pem -verify_return_error
```

Note that if you wish to use the –proxy command line parameter with the s_client tool of openssl, you will need to download and build the most recent version of openssl from their git repositoty. E.g., you can get it by running:

```
git clone git://git.openssl.org/openssl.git
```

Build and install it on your home directory to use the tool or go ahead and build your own client **C**.

Emulating a client that does not verify the server's certificate correctly

```
Command: openssl.exe s_client –proxy 127.0.0.1:8000 -connect
127.0.0.1:8900 -state -verify 1 -CAfile cacert.pem
```

HTTP CONNECT method

A proxy supporting this method essentially supports HTTP. You only need to support a tiny part of the protocol, essentially consisting of the CONNECT command.

E.g.,

CONNECT *target_IP*:*target_port* [http version]\n

[http headers terminated using \n]

\n

Entries in [] are optional in the protocol, so the client may supply them but you can ignore them, but should still be able to parse a request that supplies them. An example value for http version is "HTTP/1.0" and for an http header "Accept-Encoding: gzip,deflate,bzip2,lzma,lzma2". An HTTP request ends when an empty line is read.

Simple proxy request:

```
CONNECT 122.44.55.66:8888


--ends here
```

More elaborate request:

```
CONNECT 122.44.55.66:8888 HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0

Proxy-Connection: keep-alive

Connection: keep-alive

Host: 122.44.55.66:8888


--ends here
```

Assignment too easy?

Try implementing the following attacks against SSL:

The POODLE Attack

The FREAK Attack

The CRIME Attack