An Example of a Man-in-the-middle Attack Against Server Authenticated SSL-sessions

Mattias Eriksson
(mattias.eriksson@simovits.com)

Simovits Consulting
Wenner-Gren Center
113 46 Stockholm
Sweden

Abstract: With the evolution of financial web services there is an increasing amount of transactions performed over the Internet. As a de-facto standard the security protocol SSL (Secure Socket Layer) or TLS (Transport Layer Security) is used to create a secure connection to web services. The connections are mainly server authenticated, which means that the servers trust any client. The technology might be considered as secure if users fulfill their responsibilities and manually control the server certificate provided by a server. Ordinary users do not have clear understanding of their responsibility and therefore make it possible to perform a man-in-the-middle attack. The man-in-the-middle attack is often discussed as practical inconceivable. This paper shows how an advanced tool can be created with reasonable efforts.

1 Introduction

Internet connections can be attacked in various ways. A general type of attack is called “Man-in–the-middle”. The idea behind this attack is to get in between the sender and the recipient, access the traffic, modify it and forward it to the recipient.
The term “Man-in-the-middle” has been used in the context of computer security since at least 1994 [2]. A general definition of a man-in-the-middle attack may be described as a “Computer security breach in which a malicious user intercepts — and possibly alters — data traveling along a network.” [1]

There exists a wide range of Internet applications today. E-business has expanded from being used for selling CDs and books to provide E-banking, with access to bank accounts and means to perform transactions and payments online. In order to provide services in a safe way most Internet applications use SSL/TLS [12][13] to provide an encrypted connection. SSL/TLS can create a two-way trust relationship between the user and the application which require administration and distribution of certificates to all users and management of revocation lists which tend to be complex. SSL/TLS is therefore mostly used with a one-way trust relationship where only the server owns a certificate.

This paper will show how vulnerabilities in the use of SSL may be exploited with the help of a man-in-the-middle tool. First an historical overview is made over previous presented techniques and related work. Then prerequisites are discussed which make this man-in-the-middle attack possible. After this discussion a scenario is described on how a man-in-the-middle attack may be performed and what criteria must be fulfilled in order to setup an attack. The architecture of a tool is described with a high-level abstraction of the major algorithms. The paper is concluded with a discussion over the security given by server authenticated SSL/TLS – sessions.

2 Related Work

Man-in-the-middle attacks have been described on several occasions especially when describing the security in cryptographic protocols. When concerning the Internet, this has been described in different steps where IP-spoofing was considered as the first step toward a working man-in-the-middle attack. IP-spoofing, is a technique where the source address of the IP-packet is forged. The problem with this technique is to be able to get the answers, since they are sent to the forged address. An early example of IP-spoofing is described in the paper "A Weakness in the 4.2BSD Unix TCP/IP Software" [3]. The paper describes a scenario where it is possible to exploit the trust relationship in a computer system by masquerading as the trusted counterpart using IP-spoofing.

A similar exploit was used around 1995 by Kevin Mitnick against Tsutomu Shimomura [4]. Mitnick used a combination of TCP-hijacking to accomplish the attack, complemented with SYN-flooding to silence one part in the communication. The TCP-hijacking was possible because the source address of the TCP/IP package was not checked for changes, allowing Mitnick to hijack the session.

Even if IP-spoofing and TCP-hijacking as mentioned above are not man-in-the middle attacks in its strict definition, the techniques might be used in such attacks. The Mitnick case could have been a man-in-the-middle attack if a connection had been established with also the other server instead of silencing it.

In 1996 a more refined technique were presented in the article "Web Spoofing: An Internet Con Game" [5] showing how a "shadow copy" of the Internet could be created using a man–in-the-middle attack. This article described how a false web application could be used to intercept the traffic intended for a real website. The article mentions that SSL with a one-way trust relationship might be circumvented, and give the user a false sense of security.

In the article "Why your switched network isn't secure" [14] some techniques to sniff and retrieve traffic on a switched network is explained. One technique to retrieve traffic on a switched network is called ARP-spoofing. To retrieve the traffic on the network an attacker can send false ARP-replies stating that the attackers computer have the MAC address of another computer on the network. This will cause the traffic intended for the computer with this MAC address, to be sent to the attacker.

3 Prerequisites

3.1 Multiple Root Certificates

The number of trusted authorities are increasing rapidly, modern web browsers come with a large number of certificates installed. In Mozilla [7] there are 66 trusted root certificates from 24 authorities, and Internet Explorer has 108 trusted root certificates. But the different amount of certificates indicates that there are differences of opinions about which authorities that are trusted. For the user it means that Internet applications might behave different depending on which browser is used and security warnings might occur. The user must still make the decisions about who to trust, and with the increased amount of certificates available the decision is harder to make.

3.2 Social Aspects

In modern computer environments users can make configurations through a GUI. In most cases the suitable options are already selected and the user may accept it. All easy to use applications with the correct options pre-selected have taught the users that in case of uncertainty the “correct” choice is to accept the pre-selected options.

Using SSL to provide security for a web server requires maintenance, lack of maintenance might cause unnecessary security warnings. Many certificates used on the Internet have problems that cause security warnings, they might not have a trusted root certificate or might have expired. When users repeatedly are confronted with “bad” certificates and warnings, it becomes normal for users to accept security warnings.

A large part of the processing of information for a user to make a decision is based on the context in which the information is presented. If some information is presented next to a warning sign, users tend to read the information more carefully and be more cautious than if it is presented as a friendly information message. Timing of events also provides users with context. If a link is clicked in the browser and a authentication dialog appears, it is natural to assume the clicking of the link is related to the authentication event (even if it might not be the case) [5].

3.3 Flaws

Modern computer systems are getting quite complex and have a lot of dependencies. One effect of the architectural dependencies is that flaws in different parts of the systems might have side effects that are affecting the overall security.

One flaw related to man-in-the-middle attacks is the “Internet Explorer SSL Vulnerability” [6], which makes it possible to forge a certificate. This exploit makes it possible to make a man-in-the-middle attack without having the user to accept a false server certificate. The attacker must however possess a certificate issued by a trusted authority to use this exploit.

Another flaw is a missing root certificate on some Win98 systems. This will cause security warnings and make the user think they are normal.

3.4 Tools

There are a many tools available on the Internet that might be used for an attack. Some tools make it possible to obtain network-traffic, they use a technique called ARP-spoofing to redirect traffic. Ettercap [8] and dsniff [9] are tools capable of both ARP-spoofing and processing of incoming traffic. Ettercap is a sniffer tool that makes it possible to listen and analyze traffic. Dsniff is a sample tool which can sniff encrypted traffic and retrieve passwords that are sent in plaintext.

A number of programming tools might be used by an attacker to create sophisticated tools. In mid 1990's when the concept of a man-in-the-middle attacks started to be discussed in security contexts, it was quite hard to write a good tool for these kinds of attacks. If the tool should be able to handle encrypted sessions and possibilities to modify information, it required in-depth knowledge of the encryption protocols used. Today there are abstraction layers for all network related programming, and manipulation of data is simplified through the use of high level programming languages.

4 Scenario

4.1 Overview

A plan to successfully launch a man-in-the-middle attack against server authenticated SSL must provide a solution to the following problems:

- Retrieving the traffic intended for a specific host.
- Real-time manipulation of the data.
- Forwarding the data to the real recipient and avoid detection.
- Handle the SSL/TLS connections.

It is might be difficult to make a complete transparent attack. To get around this problem an almost transparent attack can be made, and then complemented with some social engineering to make the user accept the differences.

4.2 Retrieving the Traffic

Traffic could be accessed by manipulation on different levels in the protocol stack. One way to do this is by logical manipulation, where the DNS - name server is replaced and directs the traffic to the attacker's computer. Another way is to manipulate the network topology, by forging routers and switches.

We have two different scenarios:

- The attacker is getting access to traffic from a certain user or network.
- The attacker is getting access to traffic intended for a specific recipient, either a single host or a whole network.

Which attack method is used depends on what kind of attack the attacker has in mind, and what is easiest for the attacker. If the attacker is a system administrator for a large company, it is easier to manipulate their own corporate DNS and steal companies outgoing traffic than to redirect all the traffic for an external web service.

The characteristics and possibilities of an attack depends on where in the network topology the DNS is forged. Since only the part of the network using the particular DNS will be affected, all attacks will not work effectively. A large scale fraud will not work if only a small network is affected. An adoption of the attack might then be required for the attack to achieve anything useful for the attacker.

4.3 Real-time Modification of the Data

A simple program is needed to perform sniffing and modification of the incoming traffic. The program must be able to create a server socket and listen to the same port as the real recipient. It must also be able to open outgoing connections to the real recipient. Some enhancements to the program are needed to get more advanced functionality such like SSL/TLS-connections and some interpretation and modification of the data. Since many programming libraries are available, much of the needed functionality might be added with a few lines of code.

4.4 Forwarding Data to the Real Recipient

Nothing special must be done to send the traffic to the real recipient. If a DNS-server is manipulated to redirect traffic this DNS-server must be avoided when sending. The problem with sending the traffic is the risk of being traced, since the IP-packages contains the attackers IP-address to make it possible to get the answers from the server.

4.5 How to Handle SSL/TLS

Most secure web services are using certificates that are issued by a trusted issuer, which means that the use of SSL will not give any security warnings. When an attacker intercepts the connection a warning will appear to the user unless the attacker also has invested in a certificate by a trusted issuer.

To avoid detection by this security mechanism a fake certificate is created, ordinary programs like OpenSSL [11] can be used for this task. To get all the information correct, the authentic trusted certificate is viewed in a browser and the information is copied when the fake root-certificate is made. Then a certificate signed by the root-certificate is created. A connection to the web service that’s going to be forged is made and the original certificate data is duplicated.

The user will still get a security warning when this certificate is used, since the root-certificate is not installed in the user’s browser, but the certificate will seem to be authentic. Most users don’t know how to see the difference between an authentic certificate and a fake certificate, especially if the fake certificate looks valid.

To ensure that the user will accept the certificate a fake page can be made that informs the user that the security warning is expected. This page is presented as it is a part of the original site, before the real secure website is presented to the user.
A system administrator could install the fake root-certificate in the user's web-browser by bundle it with a security update, to eliminate the need to deceive the user. The user will not see any security warning if the fake root-certificate is installed.

5 Creating a Tool for an Attack

5.1 Overview

The first task before starting to create a tool is to decide its usage, and what capabilities it need. To do this the web service to be attacked must be examined to see what security mechanisms are used. A useful tool can have the following capabilities:

- Handle SSL-connections.
- Log all traffic.
- Manipulation of data, and temporary store values in variables.
- Hijack sessions.

A tool with these capabilities could be used to launch attacks against e-commerce sites, and other quite sophisticated web services. The attack will be transparent to the user, since all the information may be altered to fit the attacker's intentions.

5.2 Algorithm of a Man-in-the-middle tool

The following algorithm will fulfill the requirements for the tool:
Main function:

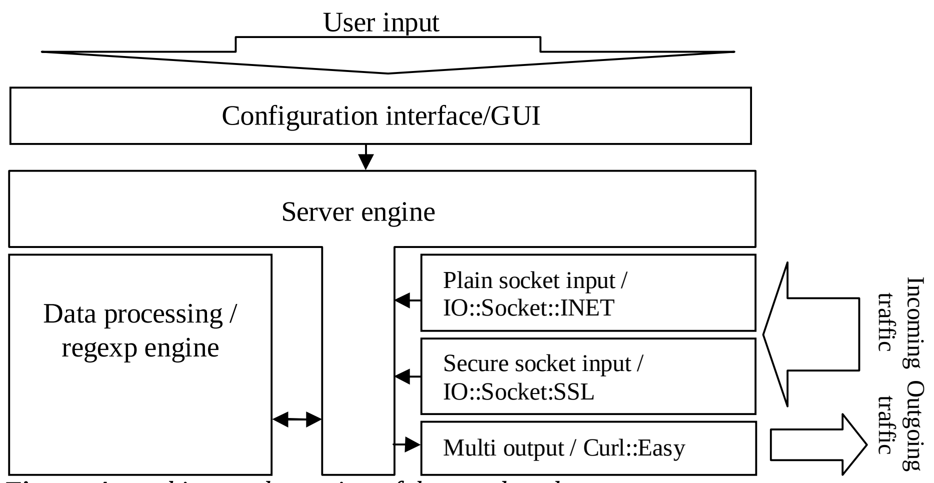
```
1. Start the server socket and listen to a given port.
2. Accept a request and perform matching and substitution.
3. Send the modified request to the intended recipient.
4. Retrieve the answer according to the following:
    4.1 Retrieve the header and perform matching and substitution.
    4.2 Retrieve the body and perform matching and substitution.
5. Send the modified answer to the request to the sender of the original request.
```

Matching function:

```
1. Match the data against different kinds of expressions depending of the kind of
   input (request, header or body).
   1.1 Match the data against all the expressions in the list in a sequential order (if a
       action flag is specified the matching might be discontinued)
       1.1.1 If the NOREWRITE flag is set no further matching is done.
       1.1.2 If the data matches the expression a search and replace is performed,
             the expression is connected to the original matching criteria.
       1.1.3 If the BREAK flag is set no more matching is done.
2. If the HIJACK flag is set the action should be performed and an error should be
   sent to the user and allow the content to be sent to attacker.
```

5.3 Architectural Design

To make it possible to handle “safe” web services a modular design of the input is required, where it is possible to alter between ordinary sockets and SSL. A configuration interface is also required to control the input/output layer and also to configure the data processing engine.



User input
Configuration interface/GUI
Server engine
Data processing / regexp engine
Plain socket input / IO::Socket::INET
Secure socket input / IO::Socket:SSL
Multi output / Curl::Easy
Incoming traffic
Outgoing traffic

Figure: *An architectural overview of the attack tool*

All the parts of the architecture should perform a small but vital part in the design, by designing the architecture this way it is easy to add or replace a certain part of the tool to extend the functionality.

Configuration interface – This is the GUI that initiates the tool with the correct parameters. The GUI is just for convenience and might be replaced with a simple module that reads the values from a configuration file.

Server engine – To handle requests from the input/output modules a server engine is used. When a request comes from the input module in use it is forwarded to the data processing module, and the response sent to the output module.

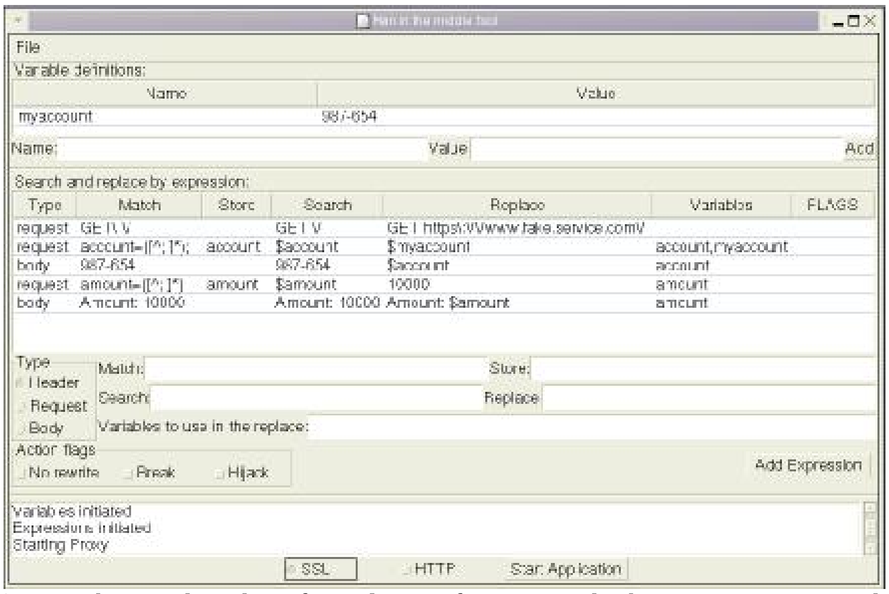
Data processing – The data retrieved is analyzed and manipulated by using regular expressions.

Input modules – Which input module that is used is decided when the tool is started, but by using a standardized form of input new methods can be added with only minor modification to the tool.

Output module – A unified output module is used for both plain and SSL output. This makes it possible to easily change the kind of output used during an ongoing session, and let it depend on the input and configuration.

5.4 Configuration and Setup

Since the tool created is a general attack tool, much of the functionality is based upon the configuration. A tool based on regular expressions is easiest to configure if it performs one substitution per configuration entry. The tool matches every part of the incoming data against every rule in the configuration (unless the break flag is specified), this makes it possible to combine rules to create complex matches.



Name	Value
myaccount	987-654

Type	Match	Store	Search	Replace	Variables	FLAGS
request	GET \/		GET \/	GET https:\/\/www.fake.service.com\/		
request	account=([^;]*);	account	$account	$myaccount	account,myaccount	
body	987-654		987-654	$account	account	
request	amount=([^;]*)	amount	$amount	10000	amount	
body	Amount: 10000		Amount: 10000	Amount: $amount	amount	

```
Variables initiated
Expressions initiated
Starting Proxy
```

***Figure:** The attack tool configured to perform a attack, the users account number does not have to be known since it is detected and stored in $account.*

To perform an attack where transactions should be forged and account numbers and the amount should be replaced, a configuration of approximately five lines is needed:

- **The first** line must adjust the incoming request to be used for outgoing traffic, this is a internal adoption due to the design choices made.
- **The second and third** line is to perform the substitution of account and amount, in the direction from the user to the donation service.
- **The forth and fifth** line is to make the substitution of account and amount, in the direction from the donation service to the user.

Some more line may be needed depending on the actual look and design of the web service to be attacked.

5.5 Modules and Programming Libraries

Since web communication mostly is based on plain text, a programming language with good string handling is needed. Since Perl is capable of string manipulation through regular expressions it makes a good choice. There are many programming libraries available for Perl, which will make it easier to add advanced functionality.

The following libraries are used to simplify the programming:

- IO::Socket::SSL – A library to provide sockets for encrypted connections.
- IO::Socket::INET – A library to provide sockets for ordinary plain text connections.
- Curl::easy – A library to provide easy handling of fetching web pages.

The above perl modules are freely available on the Internet [10].

The communication part in the program will be reduced to a few lines of code by using libraries. The use of perl as the programming language will make the matching syntax quite simple and reduce the amount of code even further. However, the matching function and the logic behind it might require some skills in program design.

6 Discussion

In most web services the security is put in the hands of the user, the technical details might be perfect but if the user opens a hole in the security the attack will be possible. The user might not realize that the security has been loosened since the occurrence of security warnings is quite common. The appearance of security warnings in applications together with a note to click yes in all security dialogs to be sure that it will function correct, will teach the user a bad behavior.
The security must be made reliable and easy to understand, the decision to override the security should be based on information that is clear to the user.

The attacker in a man-in-the-middle scenario is not the ordinary "hacker". The attacker is a person with access to Internet communication, so the attack might be considered to be some form of an insider threat. There are many points where access to Internet traffic is available, every ISP and many providers of routers and other infrastructure have access to traffic.

It does not take a very skilled programmer to create a tool that can perform an attack. The tools and the programming libraries available today provide a good foundation for the creation of an attack tool. The skill required is to create a design capable of performing the desired task, and the complexity of this varies with the task. A plain sniffer-tool requires almost no design at all, while an active attack tool against a high security web service requires a more complex design.

The man-in-the-middle attacks are possible when a one-way trust relationship is used. The weak trust relation is then complemented by an authentication to establish the missing second trust relationship. Authentication in web services gives little protection against a man-in-the-middle attack. The only authentication that creates a problem for an attacker is when a hardware device is used to sign the user's intentions. All other authentication can just be forwarded between the user and the web service.

The defense against man-in-the-middle attacks is to use a two-way trust relationship at the time when the connection is established, or to sign the user's intentions.

7 Conclusions

This paper has shown that a powerful tool to perform attacks against authenticated SSL-sessions can be made quite easy by using available programming libraries. The choice to use a one-way trust when the connection is established makes the user responsible for security during the authentication. The complex security model and the lack of understanding of the users responsibilities will make it possible to deceive the user and perform an attack.

The findings in this paper lead to the suggestions that the responsibility for the security must be removed from the user level. If the user remains responsible for the security it should be made easier for the user to make the proper decisions. Applications and web services must reduce the number of “false” security warnings. False security warnings will lower the security.

Since the two-way trust relationship is not established as an atomic operation, the connection is a weak link in the security, since only one-way trust exist at this time. If a web service has this weak link in the security it should be complemented with a method to sign the user’s intentions.

8 References

[1] Definition of man-in-the-middle - http://www.wordspy.com/words/maninthemiddleattack.asp

[2] "How string is Clipper?," *Computer Fraud & Security Bulletin*, May, 1994

[3] “A Weakness in the 4.2BSD Unix TCP/IP Software” By Robert T. Morris, AT&T Bell Laboratories, February 1985.

[4] Network Intrusion Detection, Stephan Northcutt and Judy Novac, ISBN 0-7357-1008-2.

[5] ”Web Spoofing: An Internet Con Game”, Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Technical Report 540-96 (revised Feb. 1997), Department of Computer Science, Princeton University.

[6] Internet Explorer SSL Vulnerability 2002-08-05 - http://online.securityfocus.com/archive/1/286290/2002-08-05/2002-08-07/2

[7] Mozilla web browser – http://www.mozilla.org/

[8] Ethercap - http://ettercap.sourceforge.net/

[9] Dsniff - http://monkey.org/~dugsong/dsniff/

[10] Comprehensive Perl Archive Network – http://www.cpan.org/

[11] OpenSSL – http://www.openssl.org/

[12] SSL specification - http://wp.netscape.com/eng/ssl3/

[13] TLS specification - http://www.ietf.org/rfc/rfc2246.txt

[14] “Why your switched network isn't secure”, Steven Sipes, 2000-09-10 - http://www.sans.org/newlook/resources/IDFAQ/switched_network.htm