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

Tutor: Thomas Johansson
Examiner: Per Lindström

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 Sockets 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 for an aggressor to perform a man-in-the-middle attack. The man-in-the-middle attack is often discussed as a theoretical possibility but practical inconceivable. This paper shows an example of how an advanced tool can be created with reasonable efforts.

Table of Contents

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" have been used in the context of computer security since at least 1994 [2], Some different variants of this kind of attack exist, but 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 CD:s and books to provide E-banking, the latter has provided the Internet user 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. When SSL/TLS is used with a one-way trust relationship it means that the server can not validate the SSL/TLS connection, only the user that can verify that the connection is secure.

This paper will show how vulnerabilities in the use of SSL may be exploited with the help of a man-in-the-middle tool. The paper starts with 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 criterias 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 discussions and conclusion over the security given by server authenticated SSL/TLS – sessions.

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 when using this technique is to be able to get the answers, since they are sent to the forged address. An early example of an exploit related to 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.

The Mitnick attack

The Mitnick attack is related to man-in-the-middle attacks since the exploited the basic design of the TCP/IP protocol to take over a session.
The attack is performed in a few stages:

- Identify a weak trust relationship between two computers and collect the necessary information.
- Silencing the server who is going to be replaced by the Hijacker.
- The actual Hijack.

Mitnick identified that certain services was running on the server that is going to be exploited. This was done by some simple queries with ordinary tools, normally used by system administrators. The tools used was

- finger, that gives information about who is logged in on a particular computer.,
- showmount that lists information about the available NFS shares on the computer.
- rpcinfo, that lists the available rpc-services.

The information gathered by this tools might seem harmless but combined with the proper knowledge it might be used to take over the system. Today most systems are more sparse with this information, and the services used are more secure so this particular attack might be hard to perform.
After the information was collected and analyzed, the attack was planed and a tool to aid the attack was created.

A few seconds before the actual Hijack Mitnick initiated the attack with the silencing of one part of the communicating computers. This was done by a technique called SYN-flooding. This technique is based upon the basic three way handshake that initiates a TCP-connection. A handshake can be exemplified as:

1. A tries to initiate a TCP-connection with B by sending a SYN package to B.
2. B identifies this as an attempt to start a TCP-connection and sends a acknowledgment to A, this is done by sending a SYN/ACK package.
3. A receives the acknowledgment and sends a acknowledgment on the acknowledgment, by sending a SYN/ACK back to B and the handshake is completed and a TCP-connection is established.

When this handshake is performed both parties has received confirmation that the other part is getting the packages correctly.

The SYN-flooding is based on the fact that sometimes packages gets lost on the Internet and has to be resent after a timeout. So by sending a SYN package but never send the final SYN/ACK (step 3 in the example) the other part must assume that the acknowledgment of the first SYN package (the SYN/ACK in step 2 in the example) has got lost and had to be resent.
So by sending a large amount of SYN packages to a computer but ignore the responses all available connections can be put in a state where they are waiting for a response that is never coming. Since the response is going to be ignored a fake IP-address might be used to avoid detection.

When one part of the communication are silenced, the actual Hijack might be performed. This is done by predicting the sequence number of the TCP/IP package and continue sending packages in that sequence. One thing to thing about here is that the server didn't complain about the fact that the other part of the communication just switched IP-address. But this was the weakness that Mitnick had discovered a way to exploit, the IP-address is part of the IP-layer and is not correlated with the TCP-sequence number. This means that the IP-layer cant know to which session a package origins, hence its possible for at connection to switch IP-address.

When the Hijack was completed Mitnick had a connection as a trusted part with a computer and could remotely execute programs with system administrators privileges by using the rsh remote shell.

Web Spoofing

A different technique for a man in the middle attack was presented in the article “Web Spoofing: An Internet Con Game” [5], showing how a “shadow copy” of the World Wide Web could be created.
The task of getting in between the user and the real website is accomplished by the use of URL-rewriting. The attack tool is a web application that fetches the real sites and manipulates its content, then it is sent to the user. The user must be tricked into using the attackers web application for the attack to work.

The attack works as follows:

1. User is tricked to visit the attackers web application by clicking on a false link, the name of the link might say http://home.netscape.com but it is really pointing to http://www.attacker.org/http://home.netscape.com. Notice that the URL to the attackers web application have the address to the user intended site attached at the end, this is used by the application to know which site the user really wants.
2. The web application fetches the site the user wants and modifies all links on the page to point to the attackers website with the real link attached at the end, as with the link above. This will keep the user at the attackers web application. Other information might also be altered to suit the attackers intentions.
3. The modified page is then sent to the user.

Figure 1: *The URL is modified to point to the attackers web application, the real address it attached at the end of the url.*

This simple web application will create an illusion of a "shadow copy" of the Internet, it is however not a real copy since the web pages is fetched and modified in real time. The attack is however not completely transparent to the user as the URL in the location field shows the attackers address. If the mouse pointer is above a link most browsers display the link address to the attackers web application in the statusbar. Both the location bar and the statusbar might be turned off by the attacker if the browser has javascript enabled, making it harder for a user to discover the attack.

ARP-Spoofing

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 that might be used 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 any other computer on the network. This will cause the traffic intended for the computer actually having this MAC address, to be sent to the attacker.

Prerequisites

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 in the end, and with the increased amount of certificates available the decision is harder to make.

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. A lot of the certificates used on the Internet have problems that cause security warnings, they might not have a trusted root certificate or they might have expired. When users repeatedly are confronted with "bad" certificates and security 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].

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 not make the system more insecure but it will cause security warnings and make the user think they are normal.

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 of a man-in-the-middle attack which can sniff encrypted traffic and retrieve passwords that are sent in plain text.

A number of normal programming tools are available and 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 kind of network related programming, and manipulation of data is simplified through the use of high level programming languages.

Scenario

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.

Retrieving the traffic

Getting access to the traffic can be done 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 to get the desired traffic.

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 in 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.

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 of the traffic. 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. How much programming is required to achieve the functionality depends on how advanced the program must be. Since many

programming libraries are available, much of the needed functionality might be added with a few lines of code.

Forwarding data to the real recipient

Nothing special must be done to send the traffic to its intended destination. If a DNS-server is manipulated to redirect traffic to the attacker this DNS-server must be avoided. The problem with sending the traffic to the intended destination 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.

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 are viewed in a web browser and the information is copied when the fake root-certificate is made. Then a certificate signed by the root-certificate is created. To get the needed information, 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 to the user 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.

Creating a tool for an attack

Overview

The first thing to do before starting to create a tool is to decide what it is going to be used for, and what capabilities it should have. To do this the web service to be attacked must be examined closely to see what security mechanisms are used. A simple but useful tool can have the following capabilities:

- Handle SSL-connections.
- Log all traffic.
- Manipulation of data, with the capabilities to temporary store values in variables for later use.
- 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.

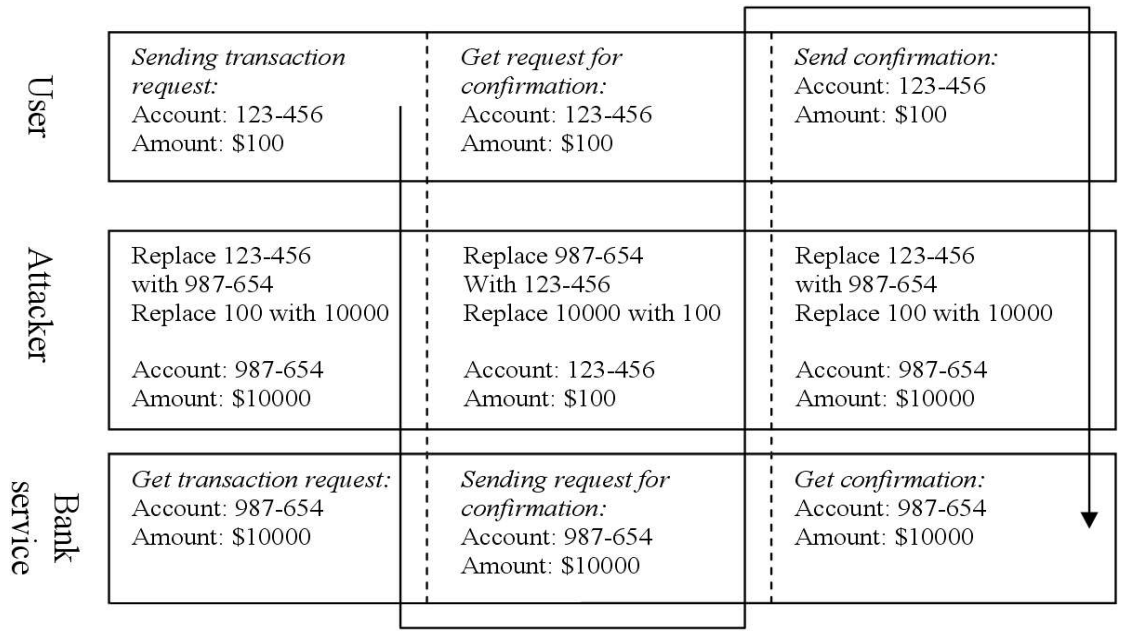


Figure 2: *An attacker alters a user's bank transactions by changing the account number and amount without the possibility for the user to detect the attack*

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 as described in the matching function.
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 as described in the matching function.
    - 4.2 Retrieve the body and perform matching and substitution as described in the matching function.
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 action flag is specified no further matching is performed on the data.
        - 1.1.2 If the data matches the expression a search and replace are performed on the data, the search and replace expression is connected to the original matching criteria.
        - 1.1.3 If the action flag BREAK is specified no more matching are performed on this data.
2. If action flag is HIJACK the action should be performed and an error should be sent to the user and allow the content to be sent to attacker.

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.

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.

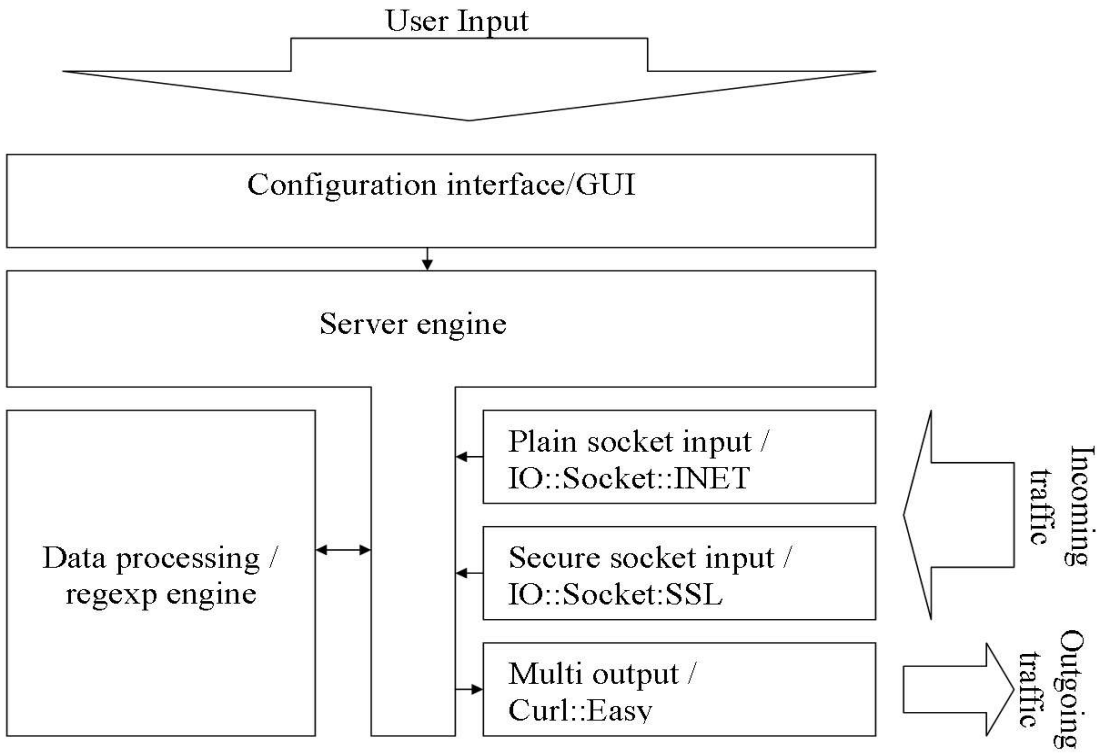


Figure 3: *An architectural overview of the attack tool*

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, which kind of matching is performed depends on the kind of data that is being worked on.
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.

Configuration and setup

Overview

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 matching algorithm makes it possible to combine rules to create complex matches.

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.

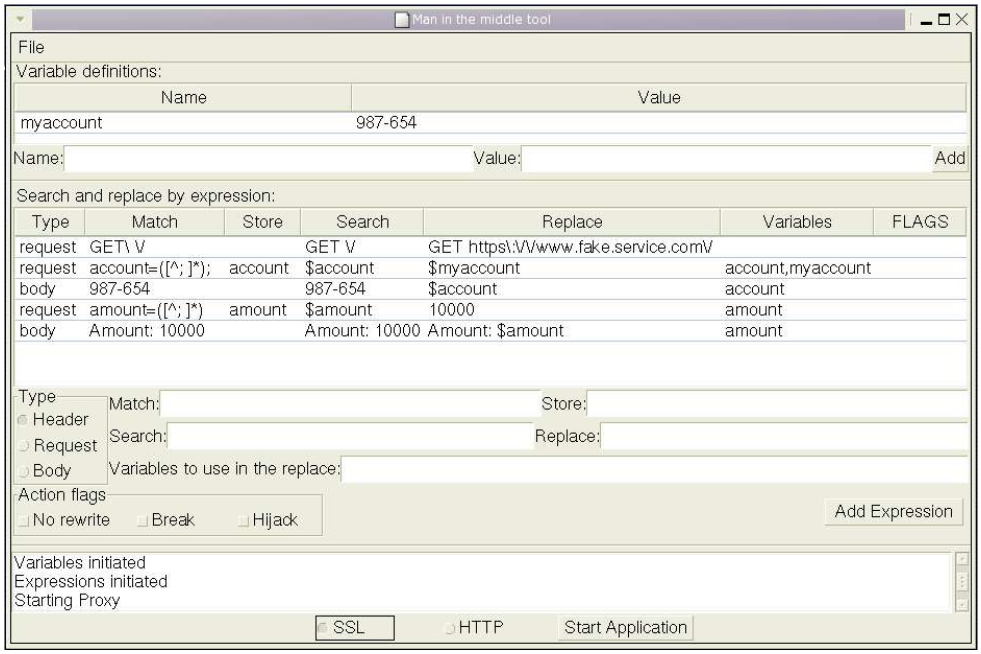


Figure 4: *The attack tool configured to perform a attack to manipulate bank transactions, the users account number does not have to be known since it is detected and stored in $account.*

Regular expressions

This tool uses regular expressions to define the strings to match and substitute, the reason is that a very flexible configuration can be created with very few lines. Since regular expressions can look quite strange, this chapter will explain the theory behind them briefly.

A regular expression is simply a pattern that is compared against a string, the pattern consist of the ordinary characters. Some of the characters have been given a extended meaning, some useful special characters are explained in the table below. [15]

Special character	***Description***
.	The dot is used to match against any character.
*	The asterisk defines that the preceding character might be matched any number of times.
()	The parenthesis is used to mark a part of the matching string that is going to be stored in a variable.
[]	The square bracket is used to define a set of characters. For example will [a-z] match one lower case character from ”a” to ”z”. A set that begins with the characters “^” will define the inverse of the set. For example will the set [^a-z] match one time against any character but the lower case characters from “a” to “z”.
\	The backslash is used to escape a character, this means that no interpretation of the character will be made. To match a dot the expression “\.” must be used to avoid matching any character.

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 a programming language that 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.

Sample attack

The sample attack is launched against a fictional bank, due to the sensitivity of the matter. The security and counter measures are based on common setups used by real financial institutions. In this scenario the attacker are assumed to be an employee with administration privileges at an Internet service provider, this gives the attacker access to a lot of traffic.

Scenario

The target for the attack is a bank. It has two different ways of logging in to the bank and three different ways of transfer money between accounts, to visualize how different kinds of security might be circumvented.

The different methods for transferring money is:

1. The user must sign a random number sequence to confirm the transfer.
2. The user must sign the account of the recipient to confirm the transfer, the signing are performed on a dedicated page with the appropriate information.
3. The user must sign the account of the recipient to confirm the transfer, the signing are performed on the same page as the transfer.

The goal for the attack is to intercept money transfers and redirect the money to the attackers account without the users knowledge.
In the scenario the user are logging in to the bank using one time passwords that is generated by a hardware token. The user are then transferring money to external account using each method for transferring money..

Preparations and planing

When the authentication mechanisms, html pages and request sequence are examined, the following problems must be solved for each scenario.

When the user is using the first method of transfer:

1. The authentication must be circumvented.
2. The transfer must be detected and the recipient account number and amount should be changed.
3. The confirmation dialog must be intercepted and the information should be altered to match the users original intentions.

When the user is using the second method of transfer:

1. The authentication must be circumvented.
2. The transfer must be detected and the recipient account number and amount should be changed.
3. The confirmation dialog must be altered so that the account number the user is signing is matching the attackers. The information on the page should be altered too, to give the impression that it is a random number that is singed.

When the user is using the third method of transfer:

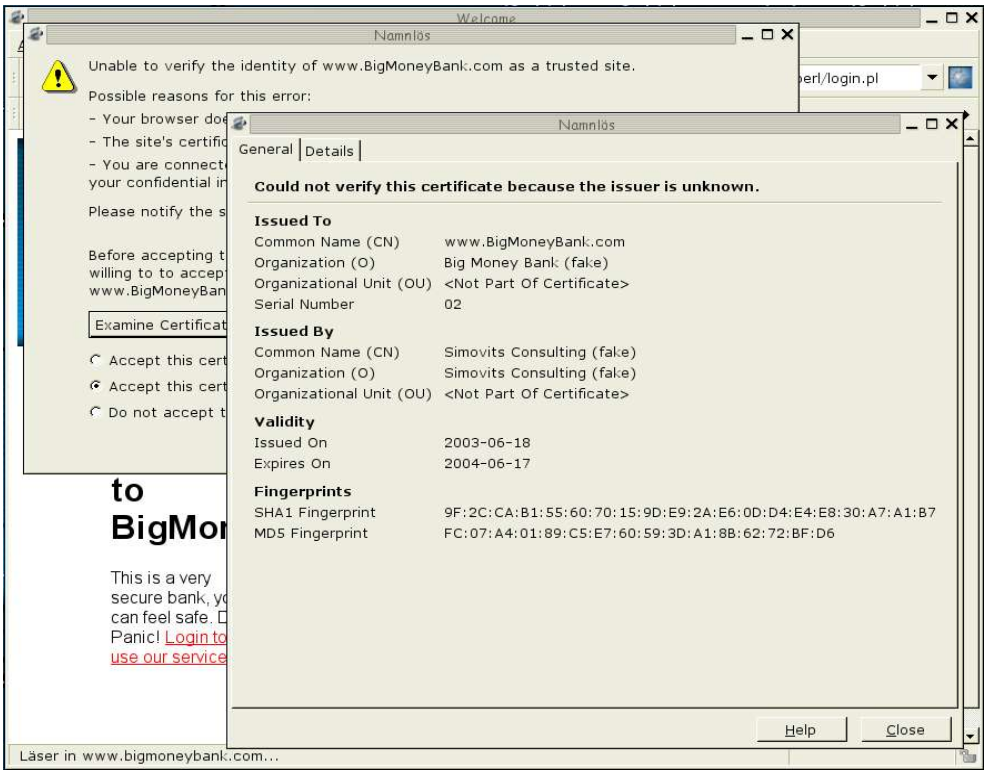
1. The authentication must be circumvented.
2. The transfer must be detected and the recipient account number and amount should be changed, the appropriate signature should also be provided.

When the problem is analyzed it shows that the authentication with onetime passwords doesn't actually require any circumvention, the attacker can just act like a proxy and forward the original information.

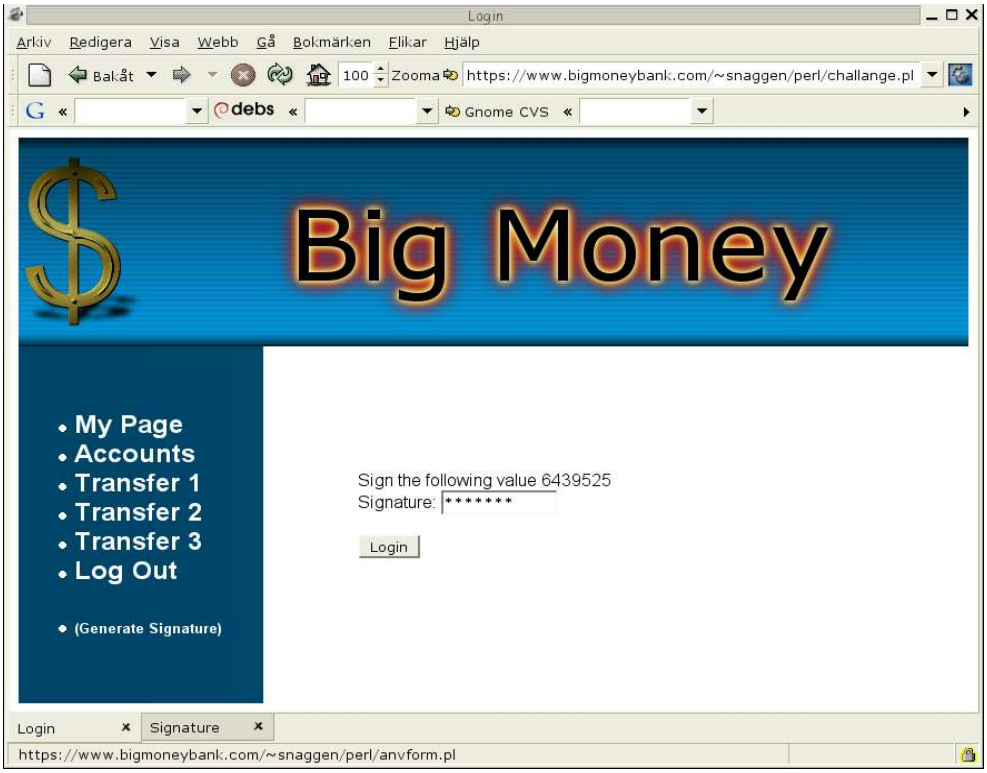
Performing the attack

This chapter will show the actual attack and what the user will see. The configuration of this attack is shown in appendix A.

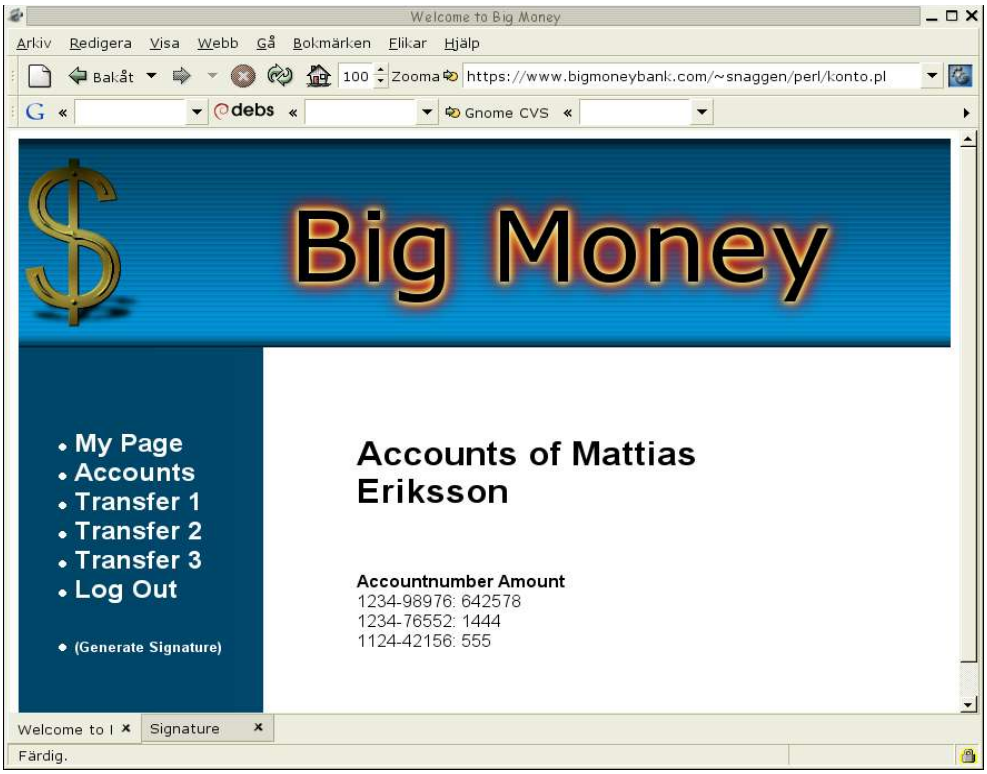
The first thing an attacker might do is to alter the welcoming screen to contain information about technical problem. This is not done in the sample attack since it isn't related to the technical part of the attack.



The user may see a warning dialog when the secure connection is established, this is due the fact that the attacker doesn't have a valid certificate (to make it more clear in the demo the certificate show has a notice that it is fake, this would not be present in a certificate created by a real attacker). If the user however have a specific version of Internet Explorer this dialog will never appear. The lack of this warning is a bug in Microsoft's implementation of SSL, that made it possible for attackers to forge certificates.
Even if the user gets a warning it is unlikely that they will see that it is a forged certificate and not just a ordinary certificate warning. An attacker may assume that most users does not know how to verify the validity of a certificate. The attacker may also show the information above before the secure connection is established to calm down suspicious users.

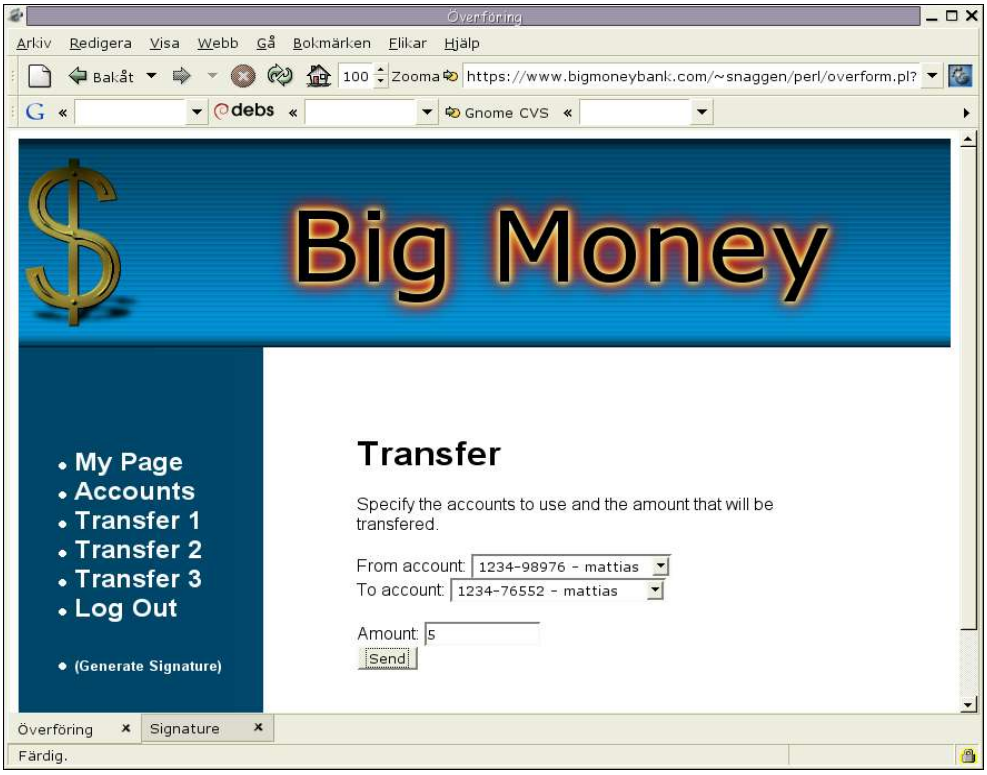


When the connection is made the user is presented with the ordinary login screen. This screen is the original login screen, nothing is altered.

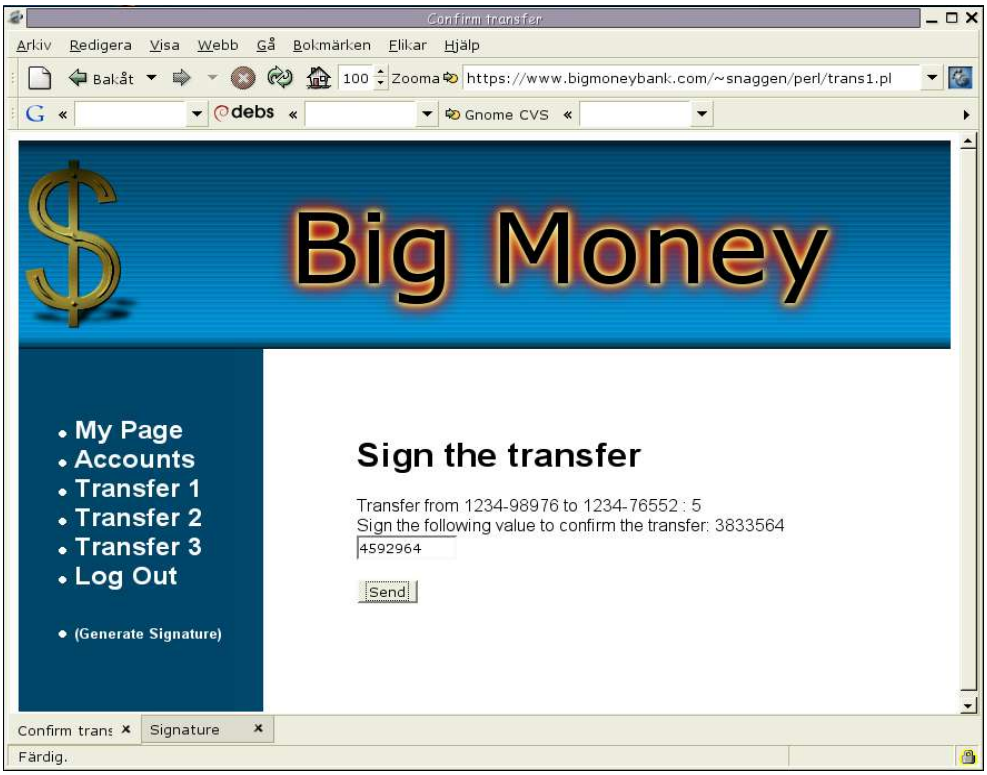


This screenshot shows the users accounts to make it possible to verify if the money has been transferred.

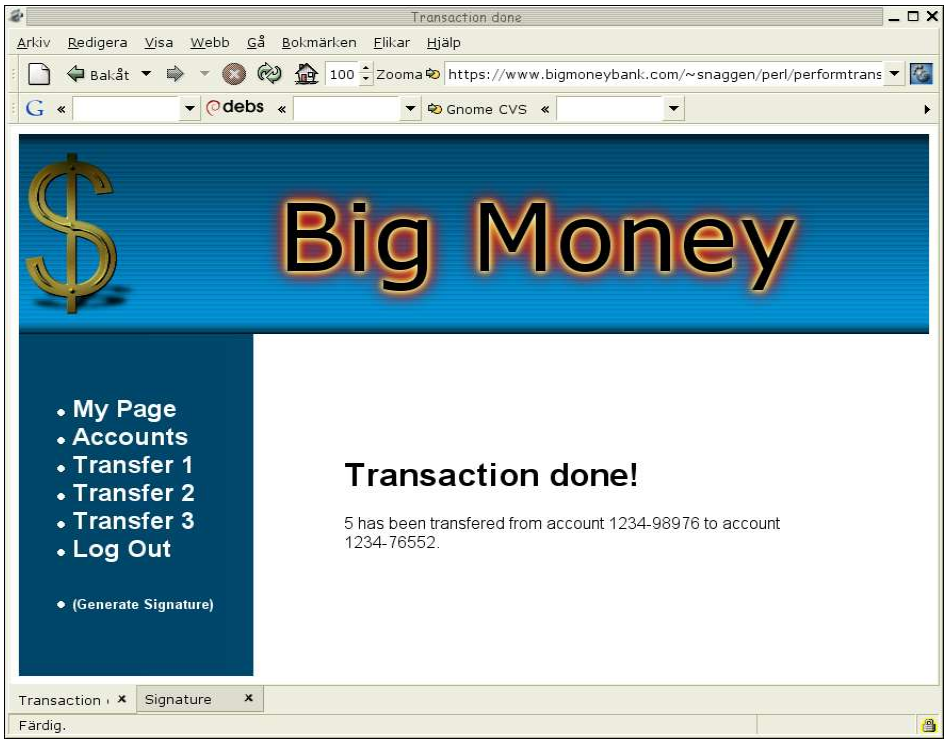
Transferring money using method one



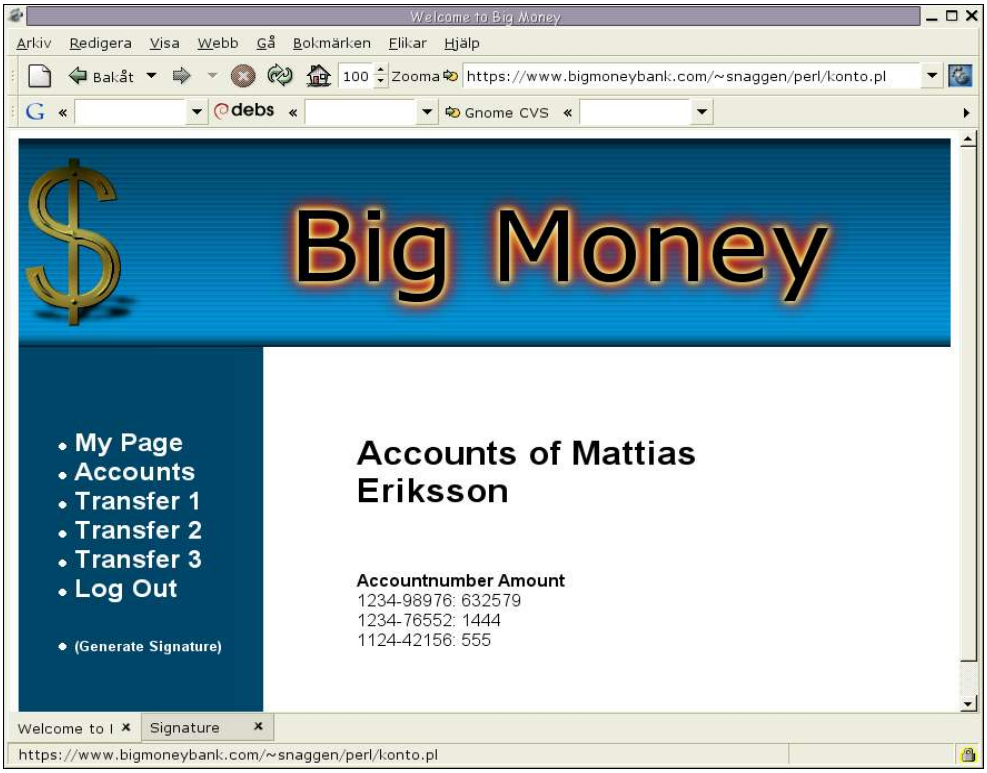
The user is going to transfer money to an external account, nothing is altered in this screen. The resulting request when the user is pressing submit will be altered, account number and amount will be changed to fit the attackers intentions.



The confirmation screen is altered to show the users original recipient account and the original amount. The signature does not have to be altered since it is just a random number.

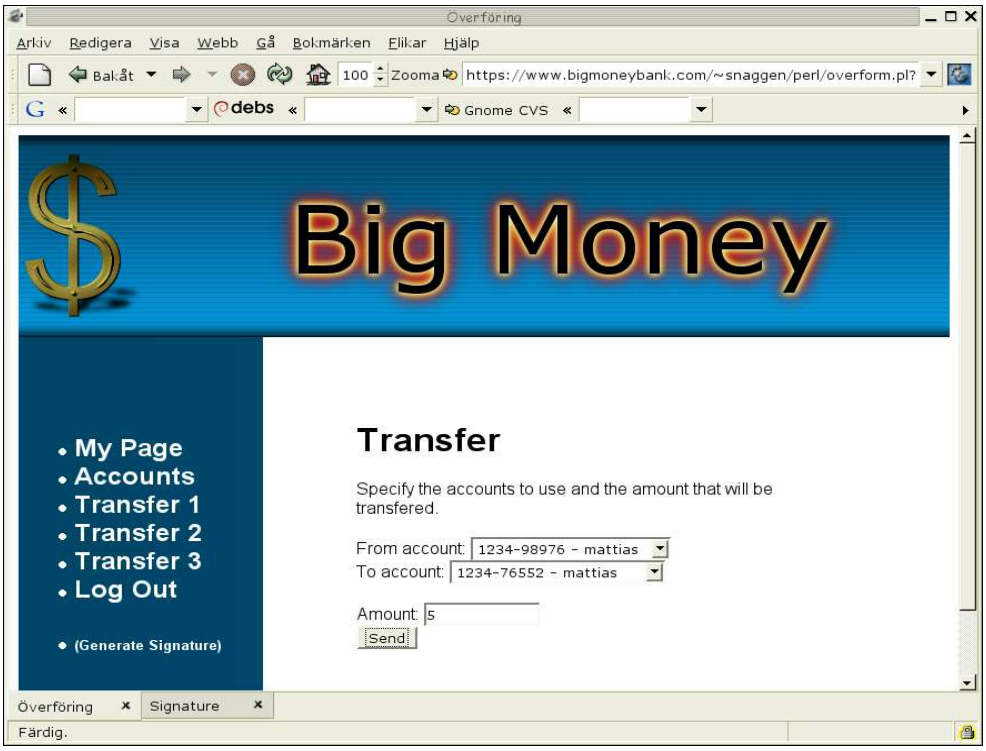


The user gets a message that the transfer has succeeded. All the information in this message indicates that everything has been transferred as intended.

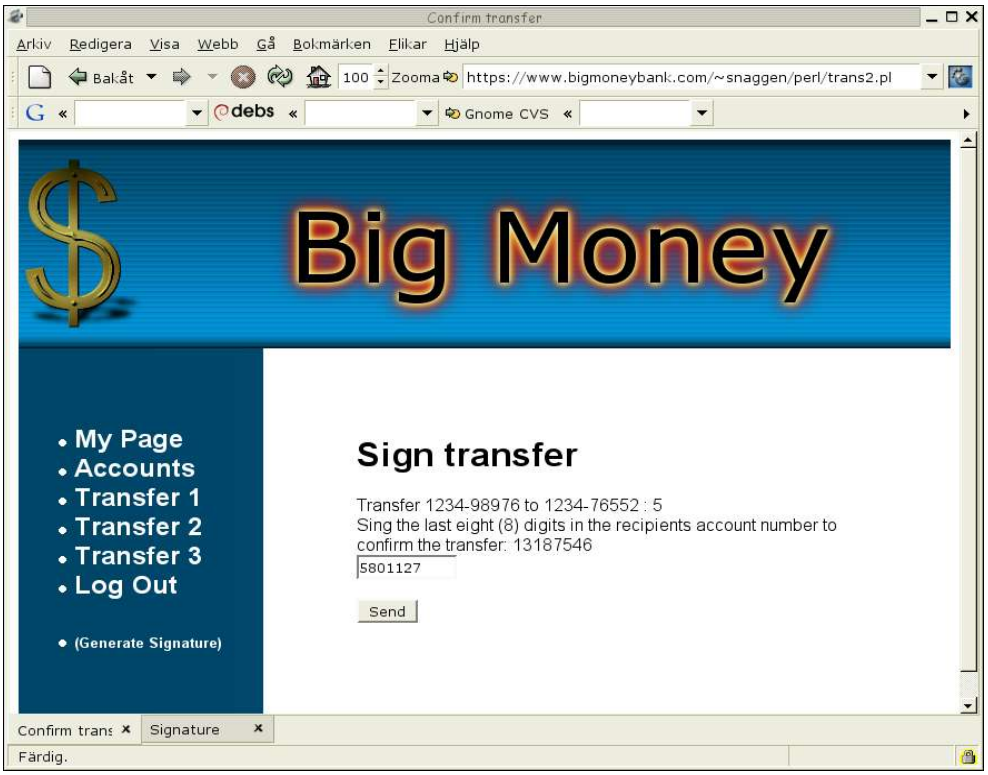


If the user looks at the accounts again it will show that the wrong amount has been transferred. In a real attack it is unlikely that the attacker will alter the amount for this reason, this is just done in this demo to make it more clear that the attack has succeeded.

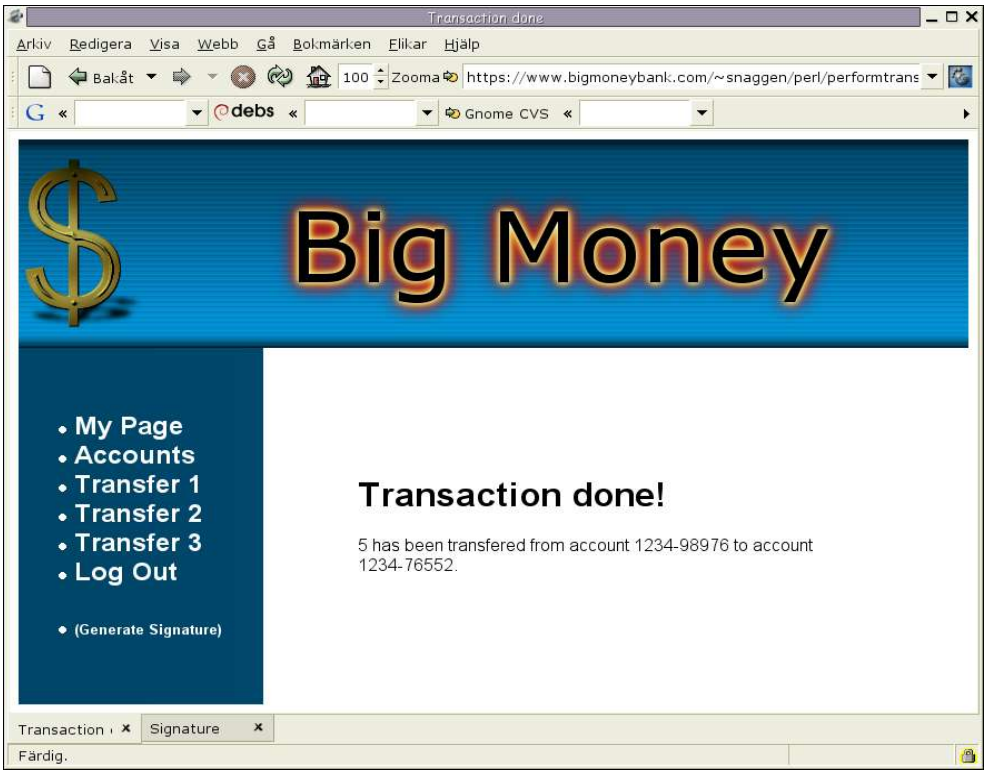
Transferring money using method two



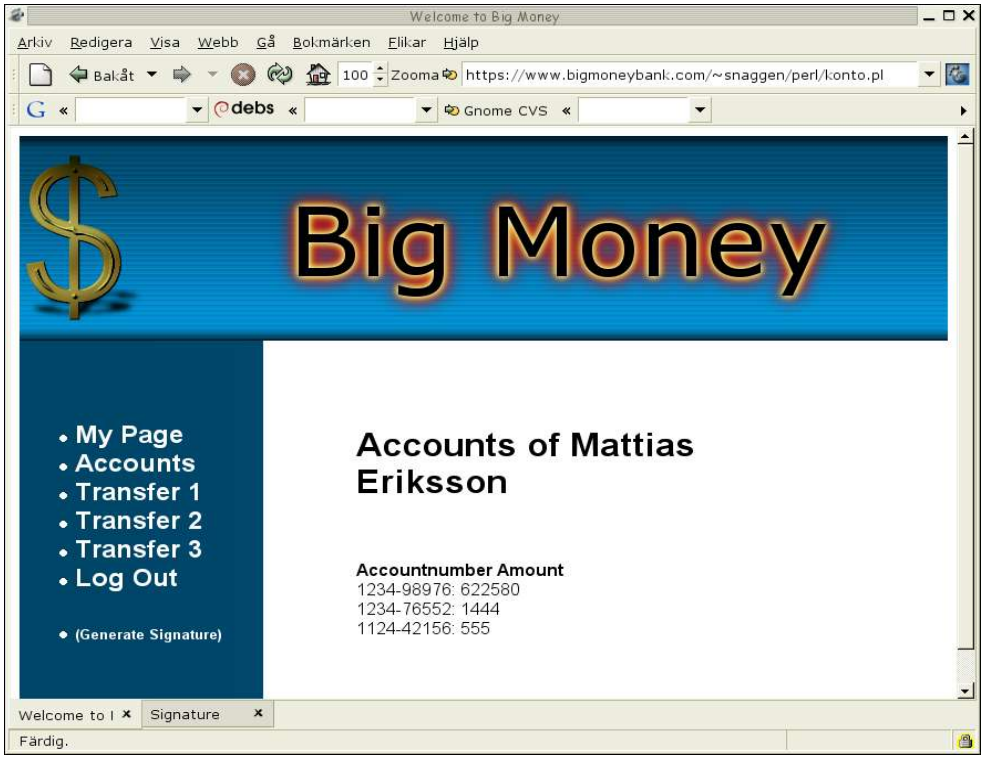
This screen will let the user transfer money using the second method of transfer, where the account number is singed. Nothing is altered in this screen, the resulting request will however be altered so that the account and amount matches the attackers intentions.



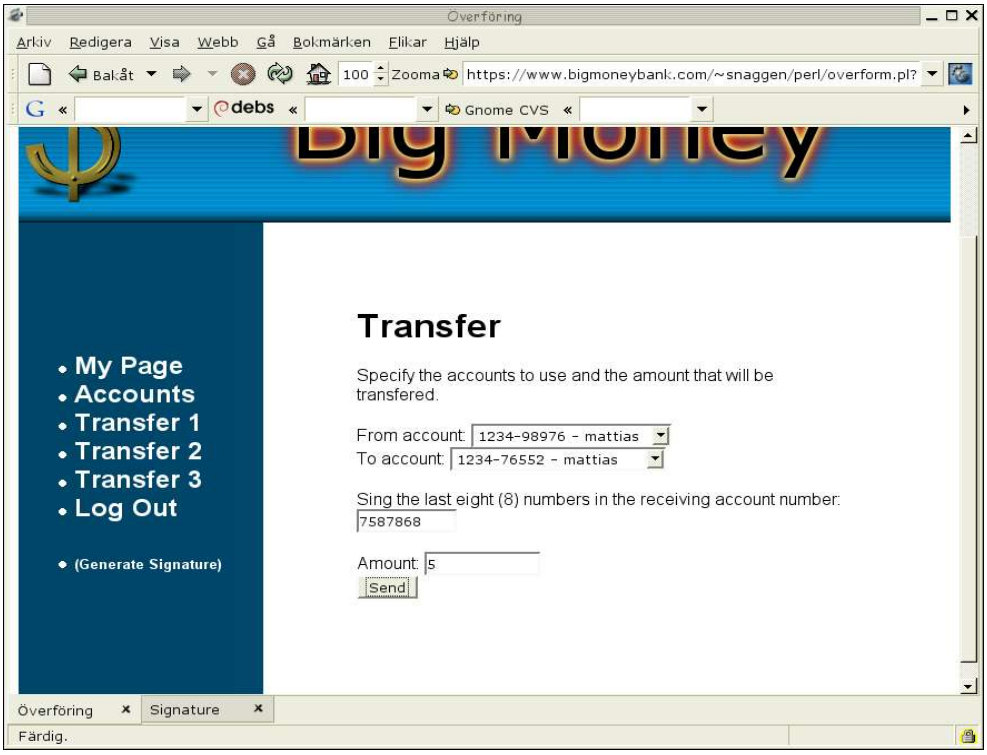
In this screen the numbers that is going to be signed is altered to match the attackers account number. In a real attack the text may be altered to give the impression that it is just a random number as shown in the first transfer method.. It is intentionally unaltered in this demo to make it possible to see the difference.



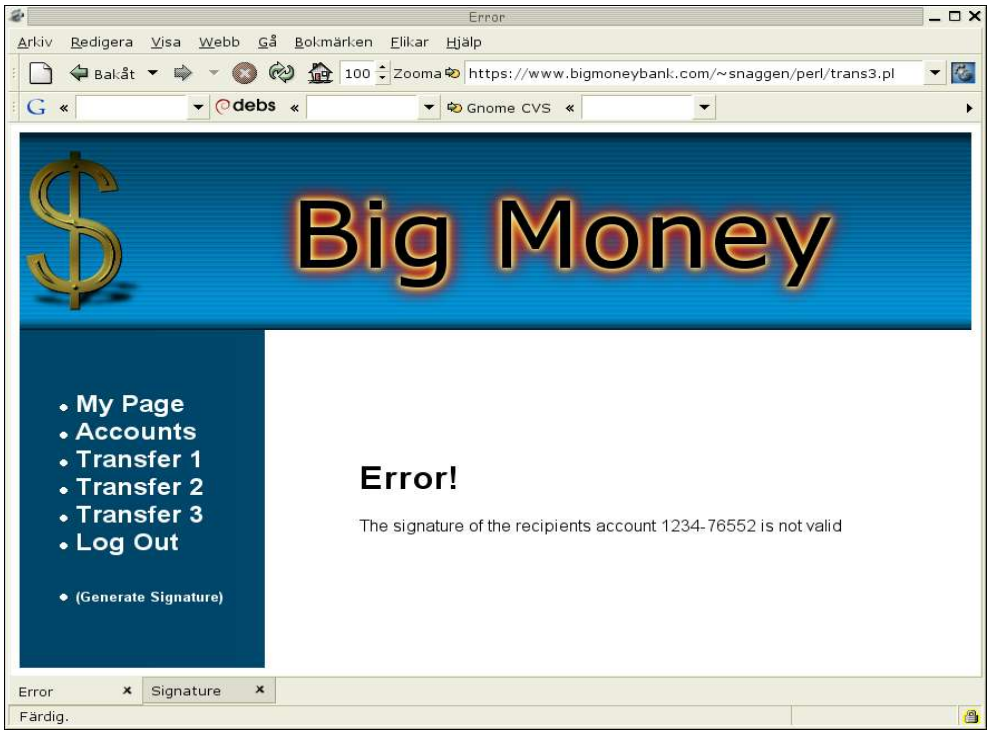
The transfers succeeds and the user will believe that the money has been transferred, but a look at the accounts again will show that the wrong amount has been transferred.



Transferring money using method three



In this screen the user will specify the accounts and make the signatures, nothing is altered in this page. The resulting request can not be altered since the attacker have not been able to get a correct signature.



This will force the attacker to do more complex alterations of the websites functionality to make the transfer look like the first method.

Techniques, variations and countermeasures

Handling of user login is not a problem as stated above and seen in the sample attack, this is caused by the fact that there are no intention of the user that might be verified. This means that using “something the user has” or “something the user knows” only for login can be circumvented. One solution of the problem could be if the information used for authentication is used to establish the encrypted connection. The use of certificates for both the client and the server is an example of this.

Some potential variations of the attacks above might occur, if the hardware token lacks a time factor in the calculation of the signature all three methods above might be circumvented. The token will give the same value for the same input every time, this might be used by the attacker to get a signature from the user by presenting the user with a fake login screen. This login screen will contain the value the attacker like to have singed, the attacker the presents the user with a message that the login failed and that the user should try again. The user is then sent to the real login screen, and the attacker have a signed value that might be used later to confirm a money transfer.

The sample attack above shows that the user should sign something that verifies the users intention to prevent the attacker from just pass information along. The signing should also be handled in the same request as the action that is signed, this forces the attacker to create a more complex attack which increases the risk of detection.

Discussion

In most web services today all of 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 possibility to perform man-in-the-middle attacks is due to the fact that a one-way trust relationship is used when the secure connection is made. The weak trust relation is then complemented by an authentication to establish the missing second trust relationship. Authentication in web services gives very 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.

Conclusions

This paper has shown that a powerful tool to perform attacks against authenticated SSL-sessions can be made quite easy by using the 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 phase. 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 understand, in order 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.

References

[1] Definition of man-in-the-middle, Webpage 2002-03-26, Retrieved 2002-09, 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, 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, Webpage 2002-08-05, Retrieved 2002-09, http://online.securityfocus.com/archive/1/286290/2002-08-05/2002-08-07/2

[7] Mozilla web browser, Webpage, Retrieved 2002-09, http://www.mozilla.org/

[8] Ethercap, Webpage, Retrieved 2002-09, http://ettercap.sourceforge.net/

[9] Dsniff, Webpage, Retrieved 2002-09, http://monkey.org/~dugsong/dsniff/

[10]Comprehensive Perl Archive Network, Webpage, Retrieved 2002-09, http://www.cpan.org/

[11]OpenSSL, Webpage, Retrieved 2002-09, http://www.openssl.org/

[12]SSL specification, Webpage, Retrieved 2002-09, http://wp.netscape.com/eng/ssl3/

[13]TLS specification, Webpage, Retrieved 2002-09, http://www.ietf.org/rfc/rfc2246.txt

[14]“Why your switched network isn't secure”, Steven Sipes, Webpage 2000-09-10, Retrived 2002-09 , http://www.sans.org/newlook/resources/IDFAQ/switched_network.htm

[15] Perlre, Perl 5.6 documentation of regular expressions, Webpage, Retrieved 2002-09, http://www.perldoc.com/perl5.6/pod/perlre.html

List of Figures

Appendix A – Configuration of an attack

```
# Fileformat:^(request|header|body)
_<tag>:matchregexp§matchvar§repregexp§repvar
#           ^init_<tag>:var=value
#
init:evilclearing=
init:evilnbkonto=nbkonto
init:evilkontonummer=7104158592
init:formateratnr=710415-8592
init:evilnamn=Olle+Karlsson
init:formateratnamn=Olle Karlsson
request:^[^ ]* \/_logga_in.html §§^([^ ]*) \/_logga_in.html¤$1
https:\/\/www.viainternet.foreningssparbanken.se\/§§BREAK
request:^[^ ]* \/§§^([^ ]*) \/¤$1
https:\/\/www.viainternet.foreningssparbanken.se\/§§
request:clearingnummer=(.*)&nbkonto=(.*)&kontonummer=(.*)&namn=(.*)
&laggtill=Till\+signera§clearingnummer,nbkonto,kontonummer,orignamn§c
learingnummer=(.*)&nbkonto=(.*)&kontonummer=(.*)&namn=(.*)
&laggtill¤clearingnummer=$evilclearing&nbkonto=$evilnbkonto&kontonumm
er=$evilkontonummer&namn=$evilnamn&laggtill§evilclearing,evilnbkonto,
evilkontonummer,evilnamn§§
body:710415-8592§§710415-8592¤$kontonummer§kontonummer§§
body:Olle Karlsson§§Olle Karlsson¤Evil$orignamnEvil§orignamn§§
body:Evil.*Evil§§\+¤ §§
body:Evil.*Evil§§Evil(.*)Evil¤$1§§
```