

scanit

The security company

Bld. du Roi Albert II, 27, B-1030 Brussels

Tel. +32 2 203 82 82

Fax +32 2 203 82 87

www.scanit.be

SSL Security in the .be TLD

Daniel Lucq
March 11th, 2008



Table of Contents

1	Introduction	3
2	SSL Overview	4
3	Attacks on SSL	5
3.1	SSL 2.0 Cipher Down-grade Attack	5
3.2	SSL Protocol Down-grade Attack	6
3.3	Certificate-based Man-in-the-middle Attack	7
4	Survey Results	8
5	Practicality of the Attacks	9
5.1	Network-level Man-in-the-middle Attack	9
5.2	SSL 2.0 Cipher Suite Down-grade Attack	10
5.3	SSL Protocol Down-grade Attack	12
5.4	Certificate-based Man-in-the-middle Attack	13
5.5	Attack Impact Examples	14
5.5.1	Internet Banking	14
5.5.2	Web Mail	14
5.5.3	E-Commerce Server	15
5.5.4	Payroll Management	15
5.6	Attack Practicality Summary	16
6	Mitigation	17
7	Conclusion	17
8	References	18
9	About Scanit	19



1 Introduction

The Internet is increasingly being used to transmit sensitive information. This includes data such as confidential medical information, financial information and transactions, proprietary corporate know-how, etc. The SSL (Secure Sockets Layer) protocol was developed in recognition of the fact that appropriate security controls are needed to safeguard the confidentiality and integrity of such data in view of its transmission over an essentially “open” and untrusted network such as the Internet.

This protocol (or rather, protocol family) offers such protection by providing a secure “tunnel” through which this sensitive data can be exchanged. The best known example of the use of this protocol family are without a doubt the so-called “https” URLs. These are sensitive web sites and web applications such as on-line banking applications that are accessed through such an SSL tunnel.

Over time, the status of SSL as a security measure seems to have evolved to near-mythical proportions, to the point where several organisations seem to consider the presence of SSL alone a proof of a site's security and trustworthiness. Indeed, the mantra “We're secure because we use 128-bit SSL” has been observed on numerous web sites.

Apart from the fact that this is a fallacy – the presence of an SSL tunnel of course provides no indication regarding the level of security of the web application itself, since it is just a network transport-level security measure – the SSL protection itself is often observed to suffer from configuration problems, which lead to a significant reduction in effectiveness of the protection offered by this protocol.

This paper sets out to conduct a small survey of HTTPS web sites in the “.be” top-level domain, in order to determine how wide-spread such SSL configuration issues really are. A sample set of 432 HTTPS web sites in this top-level domain (collected via Google, Yahoo, etc.) was tested on the level of SSL for a number of configuration and implementation issues. 40% of the tested servers was found to suffer from at least one of these problems. All of these affected servers are to at least some extent vulnerable to man-in-the-middle attacks, whereby an attacker interposes himself between the client (web browser) and the server. If successfully executed, the attacker will be able to gain access to the (presumably sensitive) information exchanged via the SSL tunnel. In certain cases, this type of attack can even be executed without either the server or the victim user being able to notice at all that a man-in-the-middle attack is occurring.

The set of affected (vulnerable) servers consists not only of “hobby” sites, but also of “high-profile” sites, such as sites related to major companies, governmental sites, and sites related to financial institutions.



2 SSL Overview

SSL is designed to operate as a wrapper (or tunnel) around a normal network connection, offering amongst others the following security protections.

- Traffic confidentiality, by encrypting the data which is tunnelled through the SSL connection.
- Integrity and replay protection, by adding Message Authentication Codes (MAC) to transmitted data.
- Server authentication, through an X.509 certificate presented by the server to the client and an associated public/private key pair.
- Optionally, client authentication (again through an X.509 certificate and an associated public/private key pair).

The most-often cited protections are the traffic encryption and server authentication. The former provides a safeguard against so-called “sniffing” attacks whereby an attacker observes network traffic and gains knowledge of its contents. The latter offers protection against man-in-the-middle attacks from the point-of-view of the client (i.e. preventing an attacker from interposing himself between the client and the server).

The first incarnation of the SSL protocol was developed by Netscape. Since then, the protocol family has expanded to include three protocols which are in effective use: SSL version 2.0 ([SSL2]), SSL version 3.0 ([SSL3]) and TLS (Transport Layer Security) 1.0 ([TLS1]). The latter of these (TLS) is an Internet standard (IETF RFC 2246).

All of these protocols operate by first performing a client-initiated hand-shake with the server. This hand-shake has a number of goals.

- Come to an agreement on the particular flavour (version) of the SSL (or TLS) protocol which is to be used. In principle the client and server will select the highest version of the protocol supported by both. To allow for this protocol selection, the client sends – as first packet of the set-up hand-shake – an initiation (“Client Hello”) packet which is compatible with the lowest version of the protocol which it supports, but indicates in it the highest supported protocol version.
- Come to an agreement on the cryptographic algorithms which will be used for the rest of the connection. In effect, all versions of the SSL (and TLS) protocol support multiple cipher suites. Each of these suites defines a bundle of cryptographic algorithms, and the client and server can select a particular set of algorithms to use based upon the algorithm's cryptographic strength, key length, and upon the set of algorithms supported by either peer. To support this choice, the client includes in its first set-up packet the list of all cipher suites which it supports.
- Allow the client to authenticate the server – the server sends its X.509 certificate as part of its first set-up packet, and a subsequent part of the set-up exchange authenticates the server using the public/private key pair associated with this certificate.
- Determine the cryptographic key material which will be used to secure the

A large white iceberg floats in a clear blue sky. The iceberg is jagged and has a prominent peak. The sky is a solid, bright blue.

transmitted data.

As we will see, all of the considered attacks attempt to tamper in some way with the packets exchanged during this hand-shake.

3 Attacks on SSL

Over the years, a number of vulnerabilities were discovered in the SSL protocols and in particular implementations. This section briefly describes the attacks on SSL-protected connections which were considered for the survey (these attacks have already been described elsewhere in more detail, e.g. see [Sch1] and [SSLmitm]). We only considered attacks which we deem to be “within reach” of an ordinary hacker, without access to particularly powerful computational resources. This rules out e.g. attacks against the public/private key pairs – cracking even a 512-bit RSA key, even though which is considered to be too weak by current best practices is currently not considered to be feasible without significant resources.

Also, the survey did not consider implementation-specific attacks such as buffer overflows, or e.g. the OpenSSL private key leaking attack (see e.g. [OSSLRSA]). Rather, we limited ourselves to issues which can be related to server-support for SSL version 2.0, server-side support for weaker cipher suites, and configuration issues related to the server X.509 certificate.

3.1 SSL 2.0 Cipher Down-grade Attack

As explained in a previous section, all members of the SSL protocol family use a set-up hand-shake to agree upon a number of parameters. This includes agreeing upon the encryption algorithms (cipher suite) which will be used to secure the transmitted traffic. As a part of this, the client includes a list of the cipher suites which it supports in the first set-up packet which it sends.

In the case of version 2.0 of the SSL protocol, the problem is that the packets transmitted by client and server during the set-up hand-shake do not have integrity protection. This means that an attacker can perform a man-in-the-middle attack and modify these packets. To do this, the attacker has to ensure that all client/server traffic passes through a host controlled by the attacker – a brief discussion on this will be included the section treating the practicality of the described attacks.

For the cipher down-grade attack, the attacker would modify the list of supported cipher suites in the client's initial set-up packet to only include the weakest cipher suite supported by the client. Provided that the server also supports this weaker cipher suite, it would subsequently be chosen to secure the traffic transmitted through the SSL tunnel (i.e. in effect “down-grading” the strength of the cipher suite used to protect the traffic).

If this cipher suite is sufficiently weak, the attacker could observe the transmitted



(encrypted) network traffic, and have a realistic chance of “guessing” the encryption key and thereby crack the encryption. This is especially the case for the so-called “export-grade” cipher suites, and to a somewhat lesser extent for the so-called “low-grade” cipher suites.

The former (“export-grade”) are cipher suites which were introduced in response to the export restrictions on strong cryptographic algorithms by the US government. These are cipher suites where the length (“strength”) of the cryptographic key has been drastically reduced (to 40 bits). This reduction in key length makes it feasible for an attacker with modern hardware to “guess” the correct encryption key by simply trying all possibilities.

3.2 SSL Protocol Down-grade Attack

The cipher down-grade attack described above has the prerequisite that version 2.0 of the SSL protocol has to be used for the connection. Subsequent protocol versions do introduce integrity protection of the set-up hand-shake, making it impossible to tamper with these set-up packets. This impossibility is equivalent to cracking a public/private key pair – an attack involving selection of an ephemeral RSA key might make this more feasible for an attacker, but this is out-of-scope for the current paper – e.g. see [SSLmitm].

In practice, it is – nowadays – unlikely that SSL version 2.0 will be selected, since all modern clients and servers also support the more recent versions of this protocol family.

However, if both the client and the server are willing to accept SSL version 2.0, a straight-forward protocol down-grade attack may exist. As previously stated in the SSL overview section, a client who is willing to accept SSL version 2.0 will send out an initial set-up hand-shake packet that is compatible with SSL version 2.0. In this initial packet, the client will set the client version field to the highest protocol version that the client is willing to accept (“3.0” for SSL version 3.0 and “3.1” for TLS version 1.0).

If an attacker is capable of carrying out a man-in-the-middle attack, he can then alter this initial set-up packet and change the indicated client version to “2.0”. At the same time the attacker would carry out a cipher down-grade attack as explained previously, by editing the included list of supported cipher suites. The modified packet will look like a packet from a client who only supports SSL version 2.0. If the server is willing to speak SSL 2.0, this would result in an SSL connection using protocol version 2.0 with a down-graded cipher (as explained previously). The main objective of this protocol down-grade attack is indeed to be able to carry out a simultaneous cipher suite down-grade attack, in order to have a realistic chance to crack the encryption of subsequent network traffic through this SSL tunnel.

An opportunistic server-side check for this type of attack was developed. This check relies upon the padding used for the RSA-encrypted master key sent by the client to the server (further technical details for this check are out-of-scope for this document, e.g. see [SSLmitm]). However, some servers do not implement this check, and in some cases the check can be disabled (for compatibility reasons, since the opportunistic check does not



work with some older SSL client implementations; e.g. see [OSSLneg] in this context).

3.3 Certificate-based Man-in-the-middle Attack

The man-in-the-middle protection (from the point-of-view of the client) offered by the SSL protocols relies upon an X.509 certificate presented by the server, and upon the server's associated public and private key. In this certificate, the server's identity – being the server's host name – is bound to the server's public key. This binding is performed by means of a digital signature on both components, by a trusted party (the Certificate Authority or CA).

An SSL client verifies a server's identity through the following checks.

- The validity of the certificate's digital signature is checked. The client validates this signature using the public key contained in the Certificate Authority's "root" X.509 certificate; this certificate should be "trusted" by the client (in practice this means that the client should have imported this certificate into the local certificate store).
- The status of the certificate is checked. This includes checking if the certificate is not expired (each X.509 certificate is issued for a definite time period), and possibly if it is not revoked (using a Certificate Revocation List or some on-line check).
- The identify (Common Name or CN) contained in the certificate is compared with the server's host name which the client has used to contact the server. These should match.
- At this point, the client needs to verify that the certificate indeed "belongs" to the server. This is done by sending a random value to the server, encrypted with the public key contained in the server's certificate. If the server is capable of decrypting this and repeating the random value back to the client, this proves that the server "knows" the associated private key. This is taken as of proof of ownership (authentication).

For the purpose of this survey, three common problems with the server certificate are considered.

- The certificate is not signed by some trusted party (Certificate Authority, such as e.g. VeriSign), but rather with the private key of the server itself. This is a so-called self-signed certificate.
- The certificate references an incorrect identity (Common Name), i.e. one which does not match the server's host name.
- Sometimes the server certificate is not renewed in time, and the server presents a certificate which is in effect expired.

Either of these problems results in the client being unable to conclusively verify the server's identity. In case of an SSL-protected web application, this normally results in the web browser popping up a window to signal this problem, offering the user the option to continue ("OK") or stop interacting with the offending server ("Cancel").



If a server presents such a problematic certificate, an attacker could craft a spoofed certificate, and perform a man-in-the-middle attack. The browser will signal the problem to the user, but since this happens anyway for the real server, the user is likely to accept this. Typically only detailed scrutiny of the certificate may reveal the problem, but the average user is unlikely to do this. Note that experience indicates that even for servers with a “correct” certificate, many users are unfortunately still likely to accept spoofed certificates.

Since this attack requires a user to actively accept the attacker's spoofed certificate, the attack effectively involves a social engineering component. If an attacker succeeds in having the victim user accept the spoofed certificate, the attacker will thereafter have access to network traffic tunnelled through the SSL tunnel, without even having to crack the encryption.

4 Survey Results

To conduct the survey, a list of 432 SSL-protected web sites was identified (i.e. web sites or applications accessed through a “https://” URL), having a host name in the “.be” top-level domain. This list of targets was identified through use of Google and Yahoo Search and by examining a number of public web sites.

All targets were found as “https://” URLs, which implies that they are indeed intended to be accessed using the recorded host name (this is important for the server authentication verification, as explained previously).

Due to the fairly large number of host names, and due to the composition of this target sample (consisting of high-profile and lesser-known sites, governmental sites, corporate sites, etc.), we believe that this survey indeed provides a representative overview of the state of HTTPS usage in Belgium.

The following results were obtained.

Test	Affected hosts	Fraction of all hosts	
Vulnerability to SSL 2.0 protocol down-grade and cipher suite down-grade attacks with export-grade ciphers	61	14.1%	
Vulnerability to certificate-based man-in-the-middle attack	122	28.2%	
	Expired certificate	23	5.3%
	Certificate with incorrect Common Name	89	20.6%
	Self-signed certificate	43	10.0%
Vulnerability to either attack	173	40.0%	

Table 1: The quantitative results of the SSL security survey for a sample of HTTPS servers in the “.be” top-level domain.



In this context, a target is assumed to be vulnerable to a certificate-based man-in-the-middle attack if the presented server certificate is either expired, self-signed, or if the certificate has a Common Name which does not match the host name.

The data listed above means that a shockingly large percentage (40.0%) of all SSL-protected web sites and applications in the “.be” top-level domain can in some way be considered to be vulnerable to a man-in-the-middle attack – either through an SSL protocol down-grade attack or through a certificate-based man-in-the-middle attack – potentially leading to disclosure of the network traffic. For 14.1% of the servers, this does not even involve a social engineering component, although it does involve more effort on the part of the attacker as explained in a subsequent section.

It should be noted at this point that the vulnerable hosts included several sites in “high-profile” domains, including web mail, hosting management, and sites from the governmental and financial sectors.

5 Practicality of the Attacks

From the results of the survey which were presented in the previous section, it appears that – surprisingly enough – a significant portion of SSL-enabled web servers suffer from SSL configuration or implementation issues. The next question now is how easy it is in practice for an attacker to actually exploit these issues, and actually gain access to the tunnelled data.

5.1 Network-level Man-in-the-middle Attack

All of the attacks on the SSL protocol described further actually require the attacker to be able to perform a man-in-the-middle attack against the network traffic between the victim client and server. This means that the attacker has to be able to insert a system under his control in this network traffic flow, so that data sent by the client to the server is actually received by the attacker's system (who then forwards this – modified or unmodified – to the actual server), and vice-versa.

Some typical techniques which could be used by an attacker to perform such a man-in-the-middle attack include the following.

- ARP spoofing. The attacker may be able to trick the client machine into sending network traffic to the attacker's system through manipulation of the lower-level Address Resolution Protocol (ARP) (e.g. see [ARPspooft]).
- Hacking network components such as routers or switches. An attacker might be able to compromise network devices, and modify their configuration to redirect all network traffic to or from the victim via the attacker's host (e.g. see [BBhack]).
- Attacks targeting routing protocols such as BGP or OSPF. On a more complicated level, an attacker might be able to influence core network routing protocols, to again



cause all traffic to or from the victim (or to or from the server) to be transmitted via the attacker's machine (again, see e.g. [BBhack]).

- Setting up a spoofed wireless access point (in case of a wireless network). If the victim is using a wireless network, an attacker might try to set up a malicious wireless network access point, and try to get the victim's host to connect to this malicious access point instead – this would cause all traffic between the victim and the server to transit via the attacker's system (the spoofed access point; e.g. see [WiFiAP]).
- DNS cache poisoning, DNS reply spoofing (e.g. see [MSDNS]), and other “pharming” techniques. If an attacker can manipulate the underlying Domain Name System (DNS) which translates host names to actual network addresses, he may be able to cause the host name of a sensitive web application to be translated into the network address of the attacker's system (from the point-of-view of the victim), which would again cause the victim to send his network traffic to the attacker instead of directly to the server.

This document will not go into further details concerning these attacks. Suffice it to say that many proven techniques exist (often with freely available associated tools), and that this type of man-in-the-middle attack against network traffic is certainly very feasible.

5.2 SSL 2.0 Cipher Suite Down-grade Attack

In order for an attacker to be able to perform this attack, there are essentially two prerequisites.

1. Both the victim client and the server need to support SSL 2.0, and at least one of both needs to support *only* SSL 2.0, so that this protocol will be automatically selected when both parties communicate (or they need to be otherwise coerced to use this protocol – see the next section).
2. The attacker needs to be able to perform a traffic-level man-in-the-middle attack against the network traffic between the victim client and the server. The previous section outlined some ways in which this could be accomplished. Obviously, the attacker also needs to execute the attack in a timely manner, i.e. when the victim client accesses the SSL-enabled server.

Once these preconditions are met, the attack itself is easy to carry out, and involves the following steps.

1. The attacker intercepts the initial hand-shake packet by the client, removes all cipher suites listed as supported by the client, except for the weakest one, and forwards this “edited” packet to the real server.
2. After the initial hand-shake packet, the attacker simply forwards all data from the victim client to the server and vice-versa.
3. If the attack succeeded, victim client and server are now communicating using the weakest common supported SSL cipher suite. The attacker uses a network traffic sniffer (such as e.g. tcpdump) to record all traffic exchanged between the victim



client and server.

4. The attacker now takes the recorded traffic, and attempts to guess the session encryption key (which is used to encrypt the actual exchanged data). If the attacker succeeds, the guessed session encryption key can subsequently be used to decrypt all the recorded network traffic.

The main effort for the attacker lies in step 4, i.e. the guessing of the encryption key. If an attacker can force the victim client and server to use a so-called “export-grade” cipher suite, the key space size is forty bits (this is on the order of 1,000,000,000,000 possible encryption keys). Although this may still seem like a very large key space, guessing the encryption key by trying every possible key is absolutely feasible in this case.

Taking the example of the “EXP-RC4-MD5” cipher suite, which uses a forty-bit key space and the RC4 encryption algorithm, an informal test was performed. On a single core of the 2.33GHz Intel Xeon processor, an unoptimized program was found to take approximately twenty days to try the entire key space – i.e., after at most twenty days, the attacker would have succeeded in decrypting the collected network traffic.

Again, twenty days may seem like a long time. However, this part of the attack is entirely automated (i.e. the attacker can simply leave the program running without further manual intervention or effort). Furthermore, the nature of many information sent via SSL is such (e.g. financial data, user names and passwords, etc.) that it may indeed be worthwhile for an attacker to invest this amount of time. Also, we expect that careful optimization of the guessing program could result in further speed improvements, and – most importantly – the encryption key guessing step of the attack can be perfectly distributed across multiple machines, multiple CPUs, or multiple CPU cores. As such, on the test CPU, which is actually a quad-core CPU, the maximum required time in this respect would have been five days instead of twenty!

The situation becomes even more critical when the assumption is made that the attacker has access to significant computing resources, e.g. under the form of (part of) a “botnet”. Taking the example data above, even a moderately sized “botnet” of 5,000 hosts of moderate computing power (e.g. with single-core CPUs half as powerful as our test CPU) would take at most ten minutes to find the correct encryption key. This is more or less on the order of “pseudo-realtime” in the context of web applications, where an attacker can crack the SSL encryption fast enough so that even relatively short-lived data such as session IDs are still valid.

Lastly, it should be noted that this type of attack does not scale very nicely from the perspective of the attacker, since the encryption key guessing has to be performed for every single SSL connection which is to be cracked.

When considering the likelihood of a pure SSL 2.0 cipher suite down-grade attack, and leaving coercion aside for now (this will be discussed in the next section), the attack implies that either the victim client or the server needs to *only* support SSL 2.0. For client software, this is rather unlikely, given the age of the SSL 3.0 protocol specification. This protocol specification was first published by Netscape in 1996 ([SSL3]), which means that clients not supporting this protocol would have to date from around this time. Most likely all



of such older client software has been phased out by now (but note that no study on this was conducted in the context of the current document).

No servers supporting only SSL 2.0 were found in our target sample in the “.be” top-level domain. Do note that such servers do exist on the Internet, and often even provide instructions on how to re-enable SSL 2.0 support in web browsers (Firefox 2.0, Microsoft Internet Explorer 7.0) where this is disabled by default (see e.g. [SSLfix]).

5.3 SSL Protocol Down-grade Attack

A pure SSL 2.0 cipher suite down-grade attack is considered less likely mostly because SSL 2.0-only servers and clients are rare. However, if the attacker can also perform a protocol down-grade attack to coerce both parties to use the SSL 2.0 protocol, this picture changes.

The preconditions for successful execution of a combined SSL protocol down-grade attack and SSL cipher suite down-grade attack are as follows.

1. Both the victim client and the server support the SSL 2.0 protocol, but can support other protocol version as well (SSL 3.0 and TLS 1.0).
2. The server should not prevent the SSL protocol down-grade attack.
3. The attacker needs to be able to perform a network-level man-in-the-middle attack against the network traffic between the victim client and the server. The previous section outlined some ways in which this could be accomplished.

Server-side support for SSL 2.0 is still wide-spread, even though this is an older protocol (1994 – see e.g. [SSL2]). Of the tested hosts in the “.be” top-level domain, 322 (78.0%) were found to accept SSL version 2.0.

On the client-side, recently there has been a security-conscious attitude shift, whereby web browser vendors start to ship newer versions of their browsers with support for SSL 2.0 disabled by default. This includes Mozilla Firefox 2.0 and Microsoft Internet Explorer 7.0. Nonetheless, a sufficient number of browsers are still around which do accept this protocol; e.g. [browser] mentions that Firefox 2.0 and Internet Explorer 7.0 now make up approximately 45.56% of all browsers, implying that 54.44% of the current web browsers normally still support SSL 2.0.

The second precondition was tested during our survey. Specifically, 14.12% of all tested SSL servers were found not to detect or prevent a straight-forward SSL protocol down-grade attack.

Once these preconditions are met, the attack is as simple as the cipher suite down-grade attack described previously. In fact, both can be easily combined, since the protocol down-grade attack is essentially the same as the cipher suite down-grade attack, with the difference that the “support version” field of the client's initial hand-shake packet is also changed by the attacker before being sent to the server. The performance figures outlined in the previous section are fully applicable here as well.



5.4 Certificate-based Man-in-the-middle Attack

This type of attack is different from the previous two in the sense that this attack usually involves a social-engineering component. Specifically, the victim user has to be enticed to accept a server certificate which was faked by the attacker. A possible exception is the case of a self-signed certificate which is not “trusted” by the client – in this case the client can be assumed to be entirely incapable of distinguishing the spoofed certificate from the real one. The client will still have to explicitly “accept” the spoofed certificate, but he has to do this anyway for the “real” server. The attack has the following preconditions.

1. The victim user needs to be enticed to “accept” the faked server certificate.
2. The attacker needs to be able to perform a network-level man-in-the-middle attack against the network traffic between the victim client and the server. The previous section outlined some ways in which this could be accomplished.

The list of preconditions is shorter than that for the previous two attacks. Indeed, this type of attack essentially applies to any web browser and any SSL-enabled server. However, the important difference here is that the success of the attack is not guaranteed – if the user chooses not to accept the server certificate, the attack fails.

The attack itself is again rather simply. When the attacker sees a new connection by a victim client, he generates a fake server certificate (this can be done beforehand, or by examining the certificate of the “real” server and generating a similar-but-not-quite certificate on-the-fly) and presents this to the client. If the client accepts this certificate, an SSL tunnel is subsequently established between the client and the attacker. The attacker then establishes a tunnel to the “real” server, and relays traffic between the victim client and the server.

In this scenario, the attacker will have immediate access to the sensitive information without any further key guessing and cracking. The main difficulty is convincing the victim user to accept the faked certificate. Although it is known that many users will accept such a certificate anyway, the task becomes even more easy if the “real” server's certificate is incorrect in some way, so that users always have to accept a faulty certificate when they access the server (even without an on-going attack). In this case, they are fairly unlikely to see the difference between the real server's incorrect certificate, and the attacker's faked certificate.

In our survey, 28.24% of the tested SSL servers were found to have some kind of problem in their certificate, which would result in the user having to always explicitly accept the certificate (resulting in a higher likelihood of him accepting an attacker's certificate as well).

Note that tools are readily available to perform this type of attack (notably tools included in the “dsniff” suite – see [dsniff]).



5.5 Attack Impact Examples

As stated before, the impact of this type of attack – if successful – consists in disclosure of the data exchanged through the SSL tunnel which was compromised.

In practice, the following short scenario's describe what information might leak, based upon the function of servers in our sample set which we have found to be vulnerable to the combined SSL protocol and cipher suite down-grade attack.

5.5.1 Internet Banking

A residential customer has deployed a wireless router to connect to his broad-band Internet connection. Unfortunately, since he has only configured WEP security on this wireless network (e.g. see [WEP]), a malicious neighbour can get access to this network without too much effort.

Having gotten network access, the attacker has observed that the victim uses Internet banking. The next time the victim prepares to surf to his Internet banking server, the attacker (using e.g. ARP spoofing) observes the resulting DNS requests and spoofs the replies to the victim (e.g. see [WinDNS]), pointing the banking server's URL to the attacker's IP address instead.

Following a successful protocol and cipher suite down-grade attack, and after cracking the SSL connection's session key, the attacker can view sensitive information which the victim exchanged with his banking server. This includes the following.

- Bank account numbers, balances, and histories.
- Bank card numbers.
- Credit card numbers.
- Investment portfolio contents.
- The victim's user name.

Depending on the authentication mechanism used by the banking server, the attacker might at this point even have sufficient information (access credentials) to gain full access to the banking server in place of the victim. If the attacker would use the victim's wireless network for this, this fraudulent access would be very difficult to detect and prove.

5.5.2 Web Mail

While a corporate user is waiting to catch his flight in the airport, he decides to have a final look at his e-mail via the local wireless hot-spot.

Unfortunately, another passenger has configured his laptop to act as a fake, malicious access point for this hot-spot system (e.g. see [hostap]). After the victim's laptop has connected to this spoofed access point, all traffic to and from the victim is routed via the



attacker's system.

The attacker performs a certificate-based man-in-the-middle attack against the victim by impersonating the victim's company's web mail server. The victim, thinking that this is a consequence of using the hot-spot gateway, unwittingly accepts the spoofed SSL server certificate.

At this point, the attacker has successfully performed the attack, and can view all the traffic the victim exchanges with the web mail server. This includes the following.

- The victim's user name and password. Note that if the victim's organisation uses a single sign-on infrastructure, this user name and password might also grant access to other systems and web applications.
- The victim's e-mail, including possibly confidential information.

Note that if the hot-spot itself required access credentials (e.g. a user name and password), the attacker might also have learned these using a similar attack.

5.5.3 E-Commerce Server

The victim uses an always-on broad-band connection to the Internet. Unfortunately, the ISP-supplied router is not securely configured, and an attacker has succeeded in gaining access to this router via the Internet. The attacker then modified the router's configuration so that instead of using the victim's ISP DNS servers, the router now sends all domain name queries to a server under control of the attacker.

Due to this change, when the victim surfs to an e-commerce server, he ends up on an attacker-controlled server. The attacker then performs a certificate-based man-in-the-middle attack. Following acceptance of the spoofed SSL server certificate by the victim who – like many users – simply clicks “OK” when his web browser presents a warning, the attacker has full access to the traffic between the victim and the e-commerce server.

Information which is disclosed to the attacker in this way may include the victim's credit card number, expiration date and card verification value, if the victim user decides to pay for a purchase with his credit card.

5.5.4 Payroll Management

A company uses an external organisation to manage its payroll and to perform actual salary payments. An HR employee is responsible for uploading relevant data to the salary management web application, from where salary payments are then made.

A malicious network administrator has noticed this. The next time the HR employee prepares to upload the salaries information, this attacker reconfigures a network switch to route all traffic between the HR employee's computer and the external web application via a computer under control of the attacker.



The attacker then performs an SSL protocol and cipher suite down-grade attack. After cracking the SSL connection session key, the attacker has gained access to information such as the following.

- Confidential and privacy-sensitive information such as personal employee information and salary information.
- Possibly also to the access credentials of the HR employee, which may give the attacker full access to the salary management application.

5.6 Attack Practicality Summary

The following table summarises some of the characteristics of the described attacks.

	Requires social engineering	User detectability	Computational overhead	Scalability
Pure cipher suite down-grade attack	No	Low to very low	High	Low
Combined protocol and cipher suite down-grade attack	No	Low to very low	High	Low
Certificate-based man-in-the-middle attack	Yes	Medium to high	Negligible	High

Table 2: Summary of the characteristics of the considered man-in-the-middle attacks on SSL.

Although the effort required on the part of the attacker to get to the actual sensitive information is (much) higher in the case of the protocol and cipher suite down-grade attacks, the importance of these attacks lies in the fact that no social engineering is required and the attack is at best very difficult to detect by an average user (from a liability perspective, a victim user also cannot realistically be blamed if these attacks succeed, since he will have done nothing wrong).

A large white iceberg floats in a clear blue sky. The iceberg is jagged and has a prominent peak. The sky is a solid, bright blue.

6 Mitigation

The investigated issues can – and should be – remedied on the server-side. In general, we recommend the following.

- Use a proper SSL server certificate, with a correct Common Name (set to the host name through which the server is accessed), issued by a Certificate Authority which is “trusted” by all users of the service, associated with a sufficiently strong public/private key pair, and which is renewed in a timely manner.
- Disable server-side support for SSL version 2.0 entirely, or at the very least use a server which detects and prevents protocol down-grade attacks. Also, if you need to continue supporting SSL version 2.0, consider alerting your users to the fact that this protocol carries inherent security risks.
- Preferably disable support for weaker cipher suites (the “export-grade” and “low-grade” cipher suite categories).

Additionally, we recommend educating individual users to take the following precautions.

- Never agree to communicate with a supposed trusted server which causes your web browser to signal certificate problems. If you need to communicate with this server, be aware of the security risks, since the server’s identity cannot be reliably ascertained.
- If your web browser supports SSL version 2.0, consider disabling this support altogether.

7 Conclusion

This document set out to investigate in how far SSL-enabled web servers in the “.be” top-level domain are subject to common configuration and implementation issues. Specifically, a set of issues was investigated which may give rise to vulnerability to man-in-the-middle attacks, which might in turn lead to disclosure of the sensitive information exchanged between the SSL-enabled server and the victim client (such information typically involves e-mail, financial or medical information, access credentials such as user names and passwords, and so on). Although the survey was limited to host names in the “.be” top-level domain, there is currently no reason to assume that the results are not representative for a broader scope such as the entire Internet.

The survey found that a surprisingly large number of servers suffered from SSL configuration issues, including problems with their server certificates, and vulnerability to SSL protocol and cipher suite down-grade attacks. Although applicable to less servers, the protocol and cipher suite down-grade attacks are equally important as certificate-based attacks, since it is at best very difficult to detect the occurrence of these attacks (provided the server is vulnerable to them), and since the victim user does not need to make a mistake (in the sense that he agrees to communicate with a supposed secure server for



which the identity cannot be verified due to certificate problems).

Furthermore, the vulnerable server population was found to cover the entire range from smaller “hobby” sites, to governmental and corporate sites (including sites related to “high profile” organisations, and organisations of which one could assume that they should be sufficiently knowledgeable).

The large number of affected sample servers is also surprising because SSL in itself is a relatively straight-forward security measure. It can be added relatively independently to a service without (normally) interacting too much with it, and its configuration does not involve too many different knobs and dials. The fact that so many sites get it wrong might also be indicative of the wider problem of lack of understanding of security measures, and lack of a structured security process in many organisations (e.g. to keep up-to-date with new evolutions and to evolve implemented security measures accordingly).

Although this document only considered SSL-enabled web sites for the survey, it needs to be noted that SSL is often used to secure other types of services as well, including e-mail delivery (through the SMTP protocol) and e-mail retrieval (e.g. through the IMAP protocol). Also, many wireless hot-spot systems rely on an SSL-enabled web portal, where a user of the wireless network first has to provide authentication credentials before being allowed to use the network. Conducting a survey on the quality of the SSL configuration for such hot-spot systems could make an interesting follow-up to the current document (experience indicates that the situation might even be worse). In this context it also needs to be noted that wireless network users (whether via hot-spot networks, residential wireless networks, or otherwise) may be the prime target population for this type of attack, since it is often relatively easy for an outside attacker to gain access to such networks and the traffic they carry.

8 References

- [ARPspoo] <http://www.oxid.it/downloads/apr-intro.swf> – “Introduction to Arp Poison Routing”
- [BBhack] http://events.ccc.de/congress/2006/Fahrplan/attachments/1197-CCC_infrastructure_hacking_12_29_06.ppt – “Router and Infrastructure Hacking”
- [dsniff] <http://www.monkey.org/~dugsong/dsniff/> – “dsniff”
- [hostap] <http://hostap.epitest.fi/> – “Host AP Linux driver”
- [MSDNS] <http://www.scanit.be/advisory-2007-11-14.html> – “Predictable DNS Transaction IDs in Microsoft DNS Server”
- [OSSLneg] <http://www.securityfocus.com/bid/15071> – “OpenSSL Insecure Protocol Negotiation Weakness”
- [OSSLRSA] <http://www.securityfocus.com/bid/7101/> – “OpenSSL Timing Attack RSA Private Key Information Disclosure Vulnerability”



- [browser] <http://www.w3counter.com/globalstats.php> – “Global Web Stats”
- [Schn1] <http://www.schneier.com/paper-ssl-revised.pdf> – “Analysis of the SSL 3.0 Protocol”
- [SSL2] http://wp.netscape.com/eng/security/SSL_2.html – “SSL 2.0 Protocol Specification”
- [SSL3] <http://wp.netscape.com/eng/ssl3/> – “SSL 3.0 Specification”
- [SSLfix] <http://www.sog.co.nz/sslbrowserfix.htm> – “SSL 2.0 Browser Fix”
- [SSLmitm] <http://www.lore.ua.ac.be/Publications/pdf/Boeynaems2004.pdf> – “Man-in-the-middle aanval op het SSL protocol”
- [TLS1] <http://www.ietf.org/rfc/rfc2246.txt> – “RFC 2246 – The TLS Protocol Version 1.0”
- [WEP] http://www.theregister.co.uk/2007/04/04/wireless_code_cracking/ – “WEP wireless cracking made easy”
- [WiFiAP] <http://www.techworld.com/mobility/features/index.cfm?featureid=1275> – “Keep clear of spoofing at hot-spots”
- [WinDNS] <http://www.securiteam.com/tools/6X0041P5QW.html> – “WinDNSSpoof, A A Windows Based DNS Spoofer”

9 About **scanit**

scanit is an IT security company specializing in ethical hacking, penetration testing, vulnerability assessments and security configuration reviews.

scanit provides services ranging from high-tech penetration testing over application source code review, risk assessments and management-level security audits, to security courses.

scanit is located in Brussels, Belgium and operates in Belgium and abroad. We have customers in Belgium, Netherlands, Luxembourg, France, Austria, Lithuania and the Middle East.