

## Factsheet FS 2011-09

# Vulnerability in SSL and TLS

In September 2011 at the 'Ekoparty Security Conference'<sup>1</sup> in Buenos Aires Juliana Rizzo and Thai Duong demonstrated a vulnerability in the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocol. During the demonstration the researchers acquired and decrypted a cookie that was sent as part of an encrypted message. As a result they gained access to a secure website.

The toolset they used was called the Browser Exploit Against SSL/TLS (BEAST). The attack directly exploits weaknesses in the encryption mechanism. Since the attacker has to fulfil a set of requirements that are hard to accomplish in practice, the chance that you will fall victim to this exploit at this moment is small.

### A high-level description of the vulnerability

The researchers exploited a vulnerability in the Cipher Block Chaining (CBC) encryption mode that is used by SSL and TLS v1.0 (see the frame on the last page for more details). The theory behind this vulnerability has already been known since 2002<sup>2</sup>. The BEAST-attack demonstrates an actual exploit now exists<sup>3</sup>. The BEAST toolset is capable of intercepting encrypted messages and decrypting part of the information.

The newer versions of TLS (v1.1 and v1.2) do not contain this vulnerability. However, these newer versions are hardly used on the internet. Therefore, at this moment, the attack will work on the majority of the websites that are secured with SSL/TLS.

For a successful attack, a number of conditions must be met:

- The attacker must obtain a 'man in the middle'-position and be able to intercept the network traffic.
- The attacker must gain control of the browser of the victim and be able to install a plug-in. In order to accomplish this installation, the end user must use HTML5-sockets or plug-ins like Java-applets or Silverlight.
- The connection between the browser process and the secured website must remain active for ten to thirty minutes in order to collect enough data.

When these conditions are met, the following steps will be executed.

- The toolset looks for SSL/TLS-sessions.
- After the discovery of such a session the toolset keeps this session open and forces the browser process to conduct a sequence of web requests. The piece of information that must be decrypted in these web requests changes slightly at each request.
- The toolset intercepts the web requests and decrypts part of the encrypted information.

Until recently, the exploit of the CBC vulnerabilities in SSL and TLS was considered practically impossible. However, in a demonstration setting it was proven that the exploit can be successful within a reasonable time. Thirty minutes of calculation time proved to be enough for a successful attack. Rizzo and Duong claim to have diminished this to ten minutes.

### The main facts at a glance:

- > TLS v1.0 and SSL contain a vulnerability that can undermine the confidentiality of information in encrypted messages.
- > The more recent TLS v1.1 and v1.2 do not contain this vulnerability. However, less than one percent of the secured websites support these recent versions.
- > We estimate that at this moment there is only a small chance this vulnerability can be abused, since the attack has to fulfill a complicated set of conditions that is hard to accomplish in practice.
- > Software suppliers are working on solutions for their products.
- > You can implement a number of counter measures to reduce the chance that this vulnerability will be exploited when you visit secured websites.

<sup>1</sup> See: <http://www.ekoparty.org/2011/juliano-rizzo.php>

<sup>2</sup> See e.g.: <http://eprint.iacr.org/2004/111.pdf> and <http://www.mail-archive.com/openssl-dev@openssl.org/msg10664.html>

<sup>3</sup> See: [http://www.securityvibes.com/servlet/JiveServlet/previewBody/1407-102-1-1425/ssl\\_jun21.pdf](http://www.securityvibes.com/servlet/JiveServlet/previewBody/1407-102-1-1425/ssl_jun21.pdf)

### Should you be worried?

At this moment there is no reason to be concerned. The attack was successful in a demonstration setting, but the requirements for success are hard to accomplish in a practical situation. Moreover, a complicated toolset is necessary to exploit the vulnerability. As far as we know, the BEAST-toolset was not released by the researchers.

History learns that attack methods are developing rapidly and that demonstrated techniques are practically applicable within a number of years. Therefore, we recommend to implement proper countermeasures as soon as possible, in order to reduce the chance that these vulnerabilities are exploited.

### What can you do as a visitor of secured websites?

- Do not accept Java-applets or other plug-ins without a valid certificate. This action reduces the chance that malicious software will be installed.
- Configure your browser to be secure. We advise to switch off SSL-support and continue support of TLS v1.0. See the section 'SSL and TLS' of this factsheet for an overview of support of SSL- and TLS-versions by the main browsers. Although continuing TLS v1.0 does not resolve the 'BEAST'-vulnerability, switching off SSL does resolve other vulnerabilities. Before you implement this advice, test whether your browser is still able to log on to the secured websites that are crucial for your business functions.
- Keep an eye on solutions provided by software-suppliers:
  - Microsoft released an advisory on how to use other (versions of) encryption modes<sup>4</sup>. An example is the use of RC4, a stream cipher. Please be aware that RC4 is not necessarily safer, since it contains other vulnerabilities. Microsoft is considering patches in the near future.
  - Google Chrome tests workarounds in a current beta-release<sup>5</sup>. As soon as this release is stable, it will be released as a public version.
  - We do not have information on solutions provided by other browsers, other than the information in the table in the section 'SSL and TLS'.
- We advise to close all browser windows and browser tabs and start with a fresh browser window before visiting a secured website. Close your browser window after your visit to a secure website and start with a fresh window again for follow-on web visits. These two habits reduce the chance that a background process is active that exploits the described vulnerability.

### What can you do as an owner of a website?

- Monitor error messages on your (web)server. Consider it as a serious warning if, for example, many *404-errors* (page not found) are generated on HTTPS requests with similar patterns. These errors might indicate that your website is under attack.
- Switch off support of SSL in your webserver. Test if the browser(version)s of the visitors of your website are still able to reach your website. The logfiles of your webserver often indicate what browser(version)s are used by your visitors.

On the long term, a migration of your webserver to TLS v1.2 will be the best solution. Since this requires TLS v1.2 support by all browsers as well, this migration is not a realistic option in the near future. We advise to investigate the opportunities to offer TLS v1.1 or v1.2 as an *additional* option on your webserver. See also the next section on SSL and TLS.

### SSL and TLS

SSL was developed in 1994 by Netscape as a response to the growing concern on the security (in particular confidentiality and authorisation) of internet traffic. SSL v2.0 was released in 1995, quickly followed by v3.0 in 1996 to solve a number of security issues. The protocol was further improved with

<sup>4</sup> See: <http://technet.microsoft.com/en-us/security/advisory/2588513>

<sup>5</sup> See: <http://www.imperialviolet.org/2011/09/23/chromeandbeast.html>

the upgrade to TLS. TLS uses stronger encryption modes and is able to use several ports. TLS Versions 1.0, 1.1 and 1.2 were released in 1999, 2006 and 2008.

The latest version of all browsers and web servers support TLS 1.0. Internet Explorer and Opera also support TLS versions 1.1 and 1.2.

The popular OpenSSL mode (mod\_ssl) of Apache only supports TLS 1.0. The GNUTLS mode (mod\_gnutls) of Apache and Microsoft Internet Information Services (IIS) support all TLS versions.

The two tables below list the support of TLS by the most important browsers and web servers.

Based on an analysis of 300.000 websites, it was shown that less than one percent of the investigated websites support TLS v.1.1 or v1.2<sup>6</sup>. Therefore for the near future GOVCERT.NL advises to continue support of TLS v1.0.

Web browser	Support
Internet Explorer	All TLS versions
Mozilla Firefox	Only TLS 1.0
Google Chrome	Only TLS 1.0
Opera	All TLS versions
Safari	Only TLS 1.0

Webserver	Support
Apache OpenSSL	Only TLS 1.0
Apache GNUTLS	All TLS versions
IIS	All TLS versions

<sup>6</sup> See: <http://blog.ivanristic.com/2011/09/ssl-survey-protocol-support.html>

## De vulnerability in detail

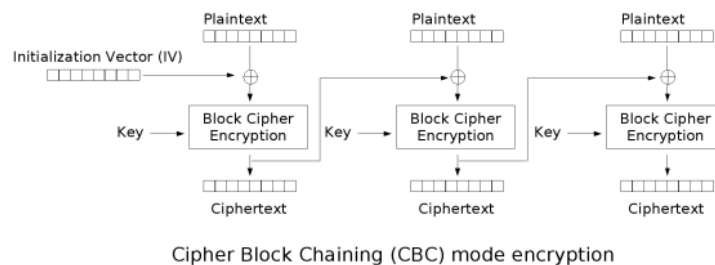
Cryptography offers two main cipher modes:

- Stream cipher: this mode encrypts a stream bit by bit. An example is RC4.
- Block cipher: this mode encrypts a stream Block by block. Cipher Block Chaining (CBC) is one of the modes used by SSL and TLS. The attackers used a vulnerability in the CBC mode.

CBC (see figure 1) divides a message into blocks with a certain maximum length<sup>1</sup>. Each block will be encrypted through a chain mechanism that works as follows:

- *Step 1:* the initial plaintext is encrypted by applying an 'X-OR'-operation on this Block and an Initialization Vector (IV). The IV is a random factor.
- *Step 2:* The result of the First step is encrypted with the certificate key, resulting in the first cipher block.
- *Step 3 and following steps:* an X-OR is applied to each next plaintext block and the current ciphertext block. Next, the result of this X-OR will be encrypted with the certificate key, resulting in the next ciphertext block. So, the current ciphertext block serves as the IV for the next block.

The standard version of CBC uses a new random IV for each new message. However, SSL and TLS use a weaker variant, whereby new messages are chained to the current message. The last ciphertext block of the current message is the IV for the first block of the next message. In this way, for a sequence of messages (for instance HTTPS web requests) a long sequence of blocks is created for which all IVs, except the first one, are predictable.



**Figure 1: the CBC encryption mode (source wikipedia)**

The attack conducted by Rizzo en Duong, is known as the '*blockwise plaintext attack*'. Based on the available information, this attack seems to work as follows<sup>2</sup>:

- The attackers generate a series of HTTPS web requests. They have an easy way to influence the first bytes due to knowledge of the HTTP format. As a result, they can easily accomplish that only one byte of a certain block is unknown. The error messages of the webserver as a reply to these 'fake requests' are irrelevant.
- The Initialization Vector of block two and all next blocks of the chain of HTTPS web requests can easily be obtained.
- The attackers attach an additional block to each HTTPS web request. This block has the same unknown byte as the block mentioned in the first bullet.
- Since only one byte per block is unknown, the IV is known and X-OR is a symmetric operation, Duong en Rizzo could determine the content of the unknown byte within a maximum of 256 web requests. On average it took only 128.
- After an unknown byte was determined, the URL would be extended by one byte. As a result Duong and Rizzo could focus on the next unknown byte. By doing this a number of times the content of a Block could be decrypted byte by byte. As a result they were able to decrypt the content of a cookie, which enabled access to a secure website.

<sup>1</sup> Zie o.a.: [http://en.wikipedia.org/wiki/Cipher\\_Block\\_Chaining#Cipher-block\\_chaining\\_.28CBC.29](http://en.wikipedia.org/wiki/Cipher_Block_Chaining#Cipher-block_chaining_.28CBC.29)

<sup>2</sup> Zie o.a.: <http://isc.sans.edu/diary.html?storyid=11722&rss> for a good description