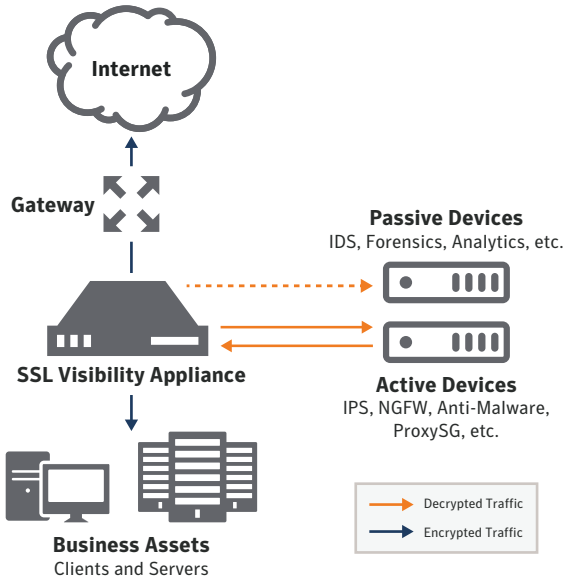
TECHNICAL BRIEF

TLS Vulnerabilities SSLV 4.x Mitigation and Protection

Authored by Roelof duToit

Transportation Layer Security (TLS) and its predecessor Secure Sockets Layer (SSL) are cryptographic protocols that enable data protection and user privacy through strong encryption. Over the past two decades, the protocol has continued to evolve to stay ahead of known security vulnerabilities and cybercriminals who look to exploit them. Each new version of SSL/TLS has been designed to close the gap of security vulnerabilities discovered in previous versions.

Unfortunately, owners of network content and infrastructure are often slow to adopt the newer TLS standards and discontinue the support of previous iterations. Updates can be time-consuming and expensive. Discontinuing support could also lead to service interruptions. When conducting a risk/reward analysis for making the change, the cost varies by organization and implementation, but a rough estimate can be quantified.



What's often more difficult is estimating the downside risk of inaction. As IT and Network professionals make the business-critical infrastructure decisions for which new TLS standard to embrace and when to disallow use of older protocols across the network, it's essential that they understand the inherent risks associated with each version of SSL. Many may be unaware of the number of vulnerabilities they're exposed to while using weak encryption and outdated protocols.

This paper provides a historical inventory of well-known SSL/TLS vulnerabilities, how they expose users to security risk. There are also recommended fixes for each identified threat. In many cases, a simple discontinuation of specific TLS cipher-suites/versions/algorithms may remove risks.

Nonetheless eliminating older TLS attributes is not always an option. In these cases, you can deploy a middlebox solution with Encrypted Traffic Management capabilities for further mitigation. The paper will also demonstrate how Symantec SSL Visibility Appliance (SSLV) can offer a higher level of TLS vulnerability protection and address each of the specific vulnerabilities users may face.

Contents

Vulnerabilities Timeline

1.1 – Marker: 1995 → SSL2

1.2 – Marker: 1996 → SSL3 (RFC 6101; historical)

1.3 – 1998: Chosen Ciphertext Attack (Bleichenbacher)

- http://archiv.infsec.ethz.ch/education/fs08/secsem/Bleichenbacher98.pdf
- https://crypto.stackexchange.com/questions/12688/can-you-explain-bleichenbachers-cca-attack-on-pkcs1-v1-5
- Chosen Ciphertext attack against RSA PKCS#1 version 1.5, specifically to attack the padding prior to RSA operation. Relies on attacker knowing that the TLS server thinks a chosen random sequence has proper padding after RSA decrypt. Attacker can extract pre-master-secret from TLS sessions using RSA key exchange.

FIX: TLS stack continues with invalid pre-master-secret, making it impossible to distinguish invalid RSA PKCS#1 padding from invalid pre-master-secret -- TLS 1.0 recommends this behavior. Alternative: do not use RSA key exchange.

1.4 – Marker: January 1999 → TLS 1.0 (RFC 2246)

1.5 – 2003: Attacking RSA Key Exchange (Klima et al)

- https://eprint.iacr.org/2003/052.pdf
- Variant of Bleichenbacher attack where TLS server fails early on ClientKeyExchange *client_version* check.

FIX: A TLS server that checks the ClientKeyExchange *client_version* field should not fail immediately, but rather randomize the PreMasterSecret in case of error.

1.6 – 2003: Remote Timing Attacks on RSA

- CVE-2003-0147
- http://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf
- Attacker extracts RSA private key remotely by analyzing the time a TLS server takes to respond to a decryption query in the ClientKeyExchange message.

FIX: RSA blinding or other anti-timing techniques. Avoid the use of RSA key exchange.

1.7 – 2003: Password Interception in SSL/TLS Channel / Padding Oracle Attack

- https://www.iacr.org/archive/crypto2003/27290581/27290581.pdf
- https://www.iacr.org/cryptodb/archive/2002/EUROCRYPT/2850/2850.pdf
- Error timing analysis while manipulating CBC-mode TLS sessions that has sensitive data repeated at predictable positions.

FIX: TLS stack time-invariant error response by running MAC verification calculation even if there is a padding error. Avoid CBC mode cipher-suites (use AEAD cipher-suites). Use Encrypt-then-MAC TLS extension.

1.8 – Marker: April 2006 → TLS 1.1 (RFC 4346)

1.9 – Marker: August 2008 → TLS 1.2 (RFC 5246)

1.10 – February 2009: SSL Stripping

- http://www.thoughtcrime.org/software/sslstrip
- Application layer modifications to prevent use of TLS.

FIX: HSTS

1.11 – November 2009: Renegotiation Attack

- CVE-2009-3555
- https://tools.ietf.org/html/rfc5746
- No cryptographic binding between original TLS handshake and renegotiation TLS handshake. Attacker can inject traffic and attack protocols that send unsolicited authentication information from the client to the server.

FIX: Avoid renegotiation, or cryptographically bind original and renegotiation TLS handshake with the secure renegotiation extension.

1.12 – October 2011: BEAST

- CVE-2011-3389
- Browser Exploit Against SSL/TLS
- https://security.stackexchange.com/questions/17080/is-there-a-way-to-mitigate-beast-without-disabling-aes-completely
- Predictable CBC mode IV results in detection of repeated plaintext patterns.

FIX: Use TLS 1.1+; Avoid CBC mode cipher-suites (use AEAD cipher-suites); 1/n-1 CBC record split for TLS 1.0.

1.13 – November 2011: STARTTLS Command Injection

- CVE-2011-0411
- Flavor#1: Pre-STARTTLS commands buffered and sent after TLS handshake.

FIX: pre-TLS buffer must be empty before starting TLS handshake.

- Flavor#2: STARTTLS stripped.

FIX: HSTS-like solution for protocols that use STARTTLS.

1.14 – September 2012: CRIME

- CVE-2012-4929
- Compression Ratio Info-leak Made Easy.
- https://security.stackexchange.com/questions/19911/crime-how-to-beat-the-beast-successor
- Brute force plaintext injection on TLS session that negotiates use of compression; if ciphertext length does not show an expected increase then plaintext has been identified.

FIX: Disable SSL3/TLS compression.

1.15 – October 2012: Cross-Protocol Attack on TLS (DHE+ECDHE)

- https://www.esat.kuleuven.be/cosic/publications/article-2216.pdf
- A MITM attacker intercepts a DHE key exchange and returns ECDHE parameters to the TLS client, which then interprets the blob as DHE parameters; the attacker obtains the ECDHE parameters from the real TLS server by running TLS sessions until the ServerKeyExchange response contains ECDHE parameters that could be mistaken for DHE parameters. The MITM can continue the handshake with the TLS client without being detected. Pre-conditions for the attack are: (a) client TLS stack must not check DH parameters, (b) server TLS stack must support explicit_prime EC curve type.

FIX: Use pre-defined FF-DHE groups (https://tools.ietf.org/html/rfc7919) instead of arbitrary DHE parameters.

1.16 – December 2012: Lucky Thirteen

- CVE-2013-0169
- http://www.ieee-security.org/TC/SP2013/papers/4977a526.pdf
- https://www.imperialviolet.org/2013/02/04/luckythirteen.html
- Padding Oracle attack on CBC mode due to MAC-then-Encrypt mechanism. Uses timing side-channel attack to decrypt arbitrary ciphertext.

FIX: Avoid CBC mode cipher-suites (use AEAD cipher-suites); Use Encrypt-then-MAC TLS extension; Time-invariant MAC calculation.

1.17 – March 2013: TIME

- https://media.blackhat.com/eu-13/briefings/Beery/bh-eu-13-a-perfect-crime-beery-wp.pdf
- Variant of CRIME where timing analysis indicates whether length of ciphertext increased, i.e., no need for access to network.

FIX: Disable SSL3/TLS compression.

1.18 – July 2013: DHE/ECDHE Parameter Checks

- https://tools.ietf.org/html/rfc6989 (refers to IKE, but checks also apply to TLS using DHE/ECDHE)
- http://cacr.uwaterloo.ca/techreports/2008/cacr2008-24.pdf
- TLS implementations are vulnerable to timing attacks when reusing DHE/ECDHE secret values.

FIX: TLS implementation must check received DHE/ECDHE parameters.

1.19 – August 2013: BREACH

- CVE-2013-3587 (unofficial)
- Browser Reconnaissance & Exfiltration via Adaptive Compression of Hypertext.
- http://breachattack.com
- Similar to CRIME, but attacks use of HTTP compression. **Not a TLS vulnerability.**

FIX: Disable HTTP compression. Randomize and mask secrets. Obfuscate length of HTTP.

1.20 – April 2014: HEARTBLEED

- CVE-2014-0160
- http://heartbleed.com ; https://tools.ietf.org/html/rfc6520
- TLS stack implementations not validating length inside heartbeat request; overflow leaks sensitive data in heartbeat response.

FIX: Proper TLS stack implementation; Fatal alert if heartbeat request received before it is negotiated.

1.21 – April 2014: Triple Handshake Attack

- CVE-2014-1295
- http://ieeexplore.ieee.org/document/6956559/
- http://antoine.delignat-lavaud.fr/doc/oakland14_slides.pdf
- The TLS master secret is not cryptographically tied to the TLS session parameters (e.g., server certificate). The attacker can set up independent TLS sessions with a client and server, and ensure that the master secret is the same on those sessions, after which the attacker can just forward messages between the client and server. This is a variant of the renegotiation attack where the attacker can inject data as if it is authenticated by the client.

FIX: Support TLS Extended Master Secret extension (https://tools.ietf.org/html/rfc7627)

1.22 – May 2014: Frankencerts

- https://www.cs.utexas.edu/~shmat/shmat_oak14.pdf
- Certificate fuzzing discovered X.509 validation vulnerabilities in various TLS implementations.

FIX: Patch TLS implementation, specifically the module that runs X.509 validation.

1.23 – October 2014: POODLE-SSL3

- CVE-2014-3566
- Padding Oracle On Downgraded Legacy Encryption.
- https://www.imperialviolet.org/2014/10/14/poodle.html
- SSL3 specification does not require validation of CBC record padding.

FIX: Do not use SSL3. Prevent SSL3 downgrade using TLS_FALLBACK_SCSV.

1.24 – December 2014: POODLE-TLS

- CVE-2014-8730
- https://www.imperialviolet.org/2014/12/08/poodleagain.html
- Similar to POODLE-SSL3. TLS implementation specific. Certain TLS stack implementations did not validate CBC padding according to the TLS specification.

FIX: Check CBC padding in TLS stack. Patch vulnerable TLS implementations.

1.25 – February 2015: RC4 NOMORE

- CVE-2015-2808
- Numerous Occurrence MOnitoring & Recovery Exploit.
- https://www.rc4nomore.com
- https://www.usenix.org/conference/usenixsecurity13/security-rc4-tls
- https://www.blackhat.com/docs/asia-15/materials/asia-15-Mantin-Bar-Mitzvah-Attack-Breaking-SSL-With-13-Year-Old-RC4-Weakness-wp.pdf
- RC4 vulnerable to statistical analysis due to statistical biases in keystream.

FIX: Do not advertise or negotiate RC4 (https://tools.ietf.org/html/rfc7465)

1.26 – Marker: February 2015 → Summary of Known Attacks (RFC 7457)

1.27 – March 2015: FREAK

- CVE-2015-0204, CVE-2015-1067, CVE-2015-1637
- Factoring RSA Keys.
- https://mitls.org/pages/attacks/SMACK#freak
- https://blog.cryptographyengineering.com/2015/03/03/attack-of-week-freak-or-factoring-nsa
- Downgrade attack.
- EXPORT cipher-suites negotiated (or EXPORT message sequence accepted even if not negotiated), forcing RSA key exchange with a key size of less than 512 bits. The weak RSA key can be factored at fairly low cost with enough CPU power.

FIX: Do not support EXPORT cipher-suites. Patch vulnerable TLS implementations.

1.28 – October 2015: Logjam

- CVE-2015-4000
- https://weakdh.org
- https://security.stackexchange.com/questions/89689/what-is-logjam-and-how-do-i-prevent-it
- Downgrade attack.
- Similar to FREAK, but attacks DHE instead of RSA. Mostly due to TLS endpoint support for DHE_EXPORT cipher-suites, i.e. MITM downgrades TLS session to use 512-bit export-grade cryptography, but even 1024-bit DH prime is now considered vulnerable to a nation-state attack.

FIX: Detect weak DH parameters in TLS implementation, i.e. DH parameters less than 1024-bit. Configure TLS server to use 2048-bit (or larger) DH parameters, or use ECDHE.

1.29 – March 2016: DROWN

- CVE-2016-0800 (also see OpenSSL issue: CVE-2015-3197)
- Decrypting RSA with Obsolete and Weakened eNcryption.
- https://drownattack.com/drown-attack-paper.pdf
- Vulnerable server supports SSL2 + EXPORT cipher-suites. If a non-vulnerable server shares RSA keys with the vulnerable server then its traffic can be decrypted.

FIX: Do not support SSL2 and/or EXPORT cipher-suites.

1.30 – August 2016: SWEET32

- CVE-2016-2183
- Pun based on “sweet sixteen birthday”.
- https://sweet32.info
- Practical birthday (collision) attack on 64-bit block ciphers (e.g. 3DES, Blowfish). Need less than 1TB of ciphertext.

FIX: Do not support or negotiate 3DES cipher-suites. At a minimum, AES should be preferred over 3DES. Limit length of TLS session.

1.31 – December 2017: ROBOT

- Many CVEs specific to toolkits and vendors.
- Return Of Bleichenbacher’s Oracle Threat
- https://robotattack.org
- Variant of Bleichenbacher attack that uses more advanced oracle detection techniques, e.g. timeouts and connection resets.

FIX: Countermeasures in TLS implementation to prevent oracle. Disable RSA key exchange.

1.32 – Marker: March 2018 → TLS 1.3 draft 28

1.33 – Marker: August 2018 → TLS 1.3 (RFC 8446)

SSLV 4.x Mitigation and Protection

Implementing the identified fixes across an entire network can present several challenges, and one should consider deploying a middlebox for Encrypted Traffic Management, a solution that's capable of identifying and mitigating risk across the entire network stack. Examining each identified threat will show where and how the Symantec SSL Visibility Appliance (SSLV) can counter these known TLS vulnerabilities.

2.1 – Inspected TLS Sessions

Inspected TLS sessions are terminated on the SSLV for the duration of the session, which enables SSLV to detect, report, mitigate, or prevent specific attacks. The columns used in the table below have specific meanings:

- "**n/a**" (Not Applicable): The attack is on the **application** layer and not on the TLS layer, even though the specific application protocol uses TLS. SSLV is **protocol-agnostic** and would not detect or prevent the attack. SSLV feeds the decrypted payload to attached security appliances, which would have the opportunity to detect and prevent application layer attacks.
- "**Report**": SSLV directly or indirectly detects the attack and records it in the TLS session log. An example of "indirect reporting" is the Renegotiation Attack, where SSLV would prevent the renegotiation and log a specific status code to indicate that renegotiation was not allowed. Not all renegotiations are an attack, but all Renegotiation Attacks use renegotiation.
- "**Mitigate**": Countermeasures are implemented to avoid a specific attack, or to make the attack impractical. The countermeasures are not a guarantee but severely narrows the attack vector. An example is the Lucky Thirteen attack, where SSLV performs the TLS record MAC calculation in a time-invariant manner.
- "**Prevent**": Active measures implemented or policy controls available to guarantee that the attack would not be successful. Example#1: SSLV strips all the elements needed to run the DROWN attack, i.e., EXPORT cipher-suites and SSL2. Example#2: If the endpoints support the extended-master-secret TLS extension then SSLV (by supporting that extension end-to-end) prevents attacks like the Triple Handshake Attack. Example#3: Customers have the option to configure a policy rule that would reject all TLS sessions that negotiate SSL3, which would prevent the POODLE-SSL3 attack.

VULNERABILITY	N/A	REPORT	MITIGATE	PREVENT	NOTES
Chosen Ciphertext Attack (Bleichenbacher)			●		Active countermeasures implemented per RFC 5246.
Attacking RSA Key Exchange (Klima et al.)			●		Active countermeasures implemented per RFC 5246.
Remote Timing Attacks on RSA			●		Blinding in RSA implementation.
Password Interception in SSL/TLS Channel			●	●	Active countermeasures implemented: time-invariant CBC MAC verify. SSLV also supports the encrypt-then-mac TLS extension.
SSL Stripping	●				Application layer should use HSTS.
Renegotiation Attack		●		●	Renegotiation not allowed. Error reported.
BEAST			●		Applies 1/n-1 TLS record split for TLS 1.0.

VULNERABILITY	N/A	REPORT	MITIGATE	PREVENT	NOTES
STARTTLS Command Injection	●				Application layer should prevent attack.
CRIME		●		●	Compression not allowed. Error reported.
Cross-Protocol Attack on TLS (DHE+ECDHE)				●	SSLV checks DH parameters and does not advertise support for explicit EC curve type.
Lucky Thirteen			●	●	"Always MAC" countermeasure implemented. SSLV also supports the encrypt-then-mac TLS extension as well as AEAD cipher-suites.
TIME		●		●	Compression not allowed. Error reported.
DHE/ECDHE Parameter Checks		●		●	DH parameter checks implemented. Error reported.
BREACH	●				HTTP compression vulnerability. SSLV is protocol agnostic.
HEARTBLEED		●		●	Detection and reporting, including detection for CUT sessions up to the actual cut point.
Triple Handshake Attack		●		●	Renegotiation not allowed. Error reported. SSLV supports extended-master-secret TLS extension.
Frankencerts				●	X.509 validation implementation not vulnerable.
POODLE-SSL3				●	SSL3 protocol vulnerability. Block SSL3 with SSLV policy. SSLV also implements TLS_FALLBACK_SCSV mechanism.
POODLE-TLS				●	TLS CBC record padding check implemented.
RC4 NOMORE				●	Algorithm vulnerability. Block RC4 with SSLV policy.
FREAK				●	SSLV strips EXPORT cipher-suites and actively prevents the use of weak RSA keys (512-bit or below). Customer could also block use of EXPORT cipher-suites with policy.
Logjam			●		Active checks to prevent use of weak DH parameters.
DROWN				●	SSLV strips SSL2 and EXPORT cipher-suites. Customer could also block use of SSL2 and EXPORT cipher-suites with policy.
SWEET32				●	Algorithm vulnerability. Block DES/3DES with SSLV policy.
ROBOT			●		Active countermeasures implemented per RFC 5246 and robotattack.org.

"In many cases, a simple discontinuation of specific TLS cipher-suites/versions/algorithms may remove risks. Nonetheless eliminating older TLS attributes is not always an option."

2.2 – Non-Inspected TLS Sessions

Non-inspected TLS sessions are either cut through early during the TLS handshake or actively/passively rejected. The cut/reject decision is made after receiving the first TLS handshake flight from the server – this is called the **server policy point**. The table below only lists attacks that can manifest **up to** the server policy point.

The SSLV acts as a **relay** for TLS sessions that are cut-through at the server policy point, and attacks that manifest after this point are not mitigated unless the attached security appliance detects them.

VULNERABILITY	REPORT	MITIGATE	PREVENT	NOTES
CRIME	●			Use of compression reported in session log.
TIME	●			Use of compression reported in session log.
DHE/ECDHE Parameter Checks	●		●	DH parameter checks implemented. Error reported.
HEARTBLEED	●		●	Detection and reporting, including detection for CUT sessions up to the actual cut point.
Frankencerts			●	X.509 validation implementation not vulnerable.
POODLE-SSL3			●	SSL3 protocol vulnerability. Block SSL3 with SSLV policy. SSLV also implements TLS_FALLBACK_SCSV mechanism.
RC4 NOMORE			●	Algorithm vulnerability. Block RC4 with SSLV policy.
FREAK			●	SSLV actively prevents the use of weak RSA keys (512-bit or below). Block use of EXPORT cipher-suites with policy.
Logjam		●		Active checks to prevent use of weak DH parameters.
DROWN			●	Block use of SSL2 and EXPORT cipher-suites with policy.
SWEET32			●	Algorithm vulnerability. Block DES/3DES with SSLV policy.

Conclusion

TLS is a complex protocol, with an ever-growing list of extensions. One should not assume that outdated and weaker versions are offering adequate protection. The protocol has continually evolved to protect users from known vulnerabilities inherent in earlier versions. Unfortunately, protecting against the full set of vulnerabilities can be difficult as it requires continual and complete updating across all impacted servers. Choosing to protect against TLS vulnerabilities on endpoints may be an option but it is also resource intensive and time-consuming. As an alternative, an Encrypted Traffic Management middlebox is a cost-effective means to help detect and mitigate TLS vulnerabilities. As such, it should be a strong consideration as a risk mitigating solution.

About Symantec

Symantec Corporation (NASDAQ: SYMC), the world's leading cyber security company, helps organizations, governments and people secure their most important data wherever it lives. Organizations across the world look to Symantec for strategic, integrated solutions to defend against sophisticated attacks across endpoints, cloud and infrastructure. Likewise, a global community of more than 50 million people and families rely on Symantec's Norton and LifeLock product suites to protect their digital lives at home and across their devices. Symantec operates one of the world's largest civilian cyber intelligence networks, allowing it to see and protect against the most advanced threats. For additional information, please visit www.symantec.com or connect with us on Facebook, Twitter, and LinkedIn.

Symantec™ 350 Ellis St., Mountain View, CA 94043 USA | +1 (650) 527 8000 | 1 (800) 721 3934 | www.symantec.com

19B182567_tb_TLS-Vulnerabilities-SSLV_EN