Global Information Assurance Certification Paper

Clickbait: Owning SSL via Heartbleed, POODLE, and Superfish

GIAC (GCIA) Gold Certification

Author: Matthew Toussain, mtoussain@gmail.com
Advisor: Rob VandenBrink

Accepted: December 21, 2015

Abstract

In the twilight of SSL's effectiveness as a method of secure communication, demonstration of associated risk should be a vital portion of modern penetration testing. While SSL is broken, practical exploitation by security analysts is a confusing process. A holistic analysis of the Secure Socket Layer's attack surface can propel the development and adoption of practical strategies for vulnerability exploitation. Subsequent risk assessment, based on these processes, can drive enterprises to support higher levels of communications security within their organization. This paper will discuss tactics, techniques, and procedures targeted at leveraging SSL vulnerabilities within an information security assessment.

1. Introduction

SSL is dead. Security researchers have now broken nearly every method of implementing the Secure Socket Layer (SSL). Unfortunately, the Internet is struggling to catch up to the new world order. SSL version 3.0 is still supported by 31.5% of public web servers (Kario, 2015). As a result attackers can gain access to key confidential information.

SSLv3 was released by Netscape in 1996, and is still heavily used today. Transport Layer Security (TLS), the successor to SSL, is the most prevalent cryptographic implementation in the world (Sarkar & Fitzgerald, 2013). The heavy utilization SSL has seen for decades was driven by the explosive growth of the World Wide Web. As the primary mechanism for transmission of sensitive personal and business information the web must be secure. SSL/TLS lays the foundation for all web services requiring confidentiality.

In 2014 and 2015 the security community was showered with vulnerabilities affecting SSL. Due to their broad implications these attacks received record news attention. However while SSL may be broken, exploiting the protocol in an impactful way can be difficult for penetration testers to accomplish. From a security audit perspective launching vulnerability scanners and evaluating software packages to determine patch and configuration management is a simple solution, but during a penetration test demonstrating value to the host organization is the primary concern. Value is demonstrated through exploitation. This paper will discuss strategies to target SSL for exploitation. The paper will further analyze and demonstrate exploitation of the following vulnerabilities:

- Heartbleed
- POODLE
- Superfish

These recent attacks combined with earlier vulnerabilities targeting SSL render the protocol incapable of supporting secure communications.

Author Name, email@address

2. Attacking the Secure Socket Layer

Typically, when an attacker targets the secure socket layer for exploitation they are after the content of the encrypted communications. This data may be the final objective, or it could be an avenue leveraged to further an ongoing compromise. In either case a successful attack could cause endless harm to the victim organization.

2.1. SSL Attack Surface Analysis

The SSL/TLS trust model utilizes X.509 certificates and asymmetric cryptography to negotiate a symmetric key, which is then used to facilitate communication between client and server (IETF, 2008). Use of the X.509 system inherently requires the use of a trusted Certificate Authority (CA) to establish a chain of trust. Certificate authorities use root certificates to issue intermediate root certificates and digital certificates for use by end users (GlobalSign, 2015). This process engenders a trust hierarchy with an overarching vulnerable nexus. The SSL certificate market is overwhelmingly dominated three major CAs: Symantec /VeriSign, Comodo, and GoDaddy (Netcraft, 2015). As a result, a compromise of the root certificates of any one of these CAs invalidates the security of the entire chain. There were three publicly acknowledged CA compromises in 2011 alone including major names like: Comodo, DigiNotar, and GlobalSign (Blaich, 2015). More recently, Google Chrome removed CNNIC from its list of trusted CAs when certificates against several Google domains were issued incorrectly (Goodin, 2015). The Superfish malware, installed on default builds of Lenovo devices, was able to install its own certificate as a trusted root CA and decrypt all SSL by perverting the protocol's chain of trust (Masnick, 2015).

While attacks against certification authorities to facilitate operations against other targets may be within the scope of nation-state sponsored Advanced Persistent Threat (APT) actors, directly breaking SSL's trust model is a bridge too far for most attackers. In these cases attackers have three primary options:

- Subverting the trust model entirely
- Attacking protocol implementation
- Cracking the encryption directly

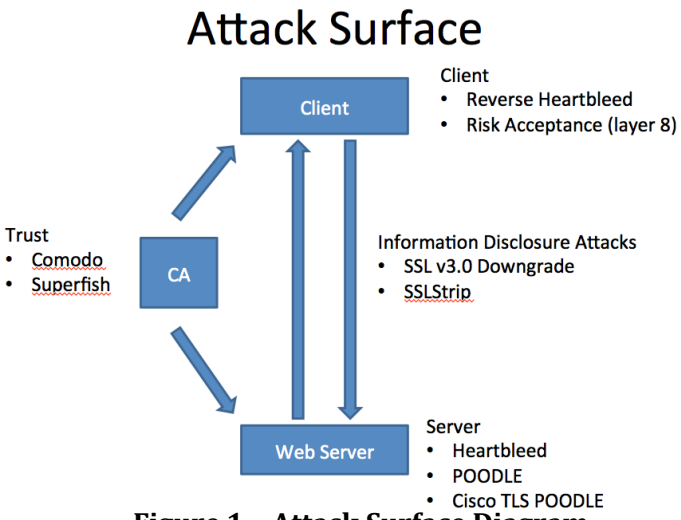

Figure 1 – Attack Surface Diagram

2.1.1. Gaining access to protocol data

Attacking SSL is well and good, but in order to acquire plaintext secrets the attacker must first gain access to the protocol data. An attacker acquires protocol data by creating a man-in-the-middle (MITM) condition and forcing victims to route traffic through the attacker's machine. Alternately, in a WiFi or hub network environment an attacker can often sniff encrypted data directly. In this case, however, the attacker lacks the ability to manipulate traffic and is unable to trigger certain vulnerabilities as a result.

Some of the most common MITM attack vectors are listed below:

- Address Resolution Protocol (ARP) Cache Poisoning or ARP Spoofing is a network based attack that allows the attacker to impersonate any device on a network, including the gateway. Impersonation of the gateway leads to a MITM position where other devices on the network transmit their information to the attacker.

- The Karma attack uses wireless sniffing to identify beaconing WiFi devices and discover their trusted networks. Next individual clients are targeted by creating rogue access points based on the beaconed SSID information.

- A Rogue Dynamic Host Configuration Protocol (DHCP) server attempts to respond to DHCP Discovery messages with false gateway information

corresponding to the attacker's machine. When legitimate DHCP services are present the attacker must respond to the request more quickly than the legitimate server engendering a race condition. When exploitation is successful the attacker gains MITM.

- A Domain Name System (DNS) Cache Poison attack attempts to associate domain name records with incorrect results. This can provide MITM by allowing attackers sit in-between connections to arbitrary domain names.

- Web Proxy Auto-Discovery (WPAD) Hijacking can allow attackers to MITM web connections by convincing hosts to use a malicious proxy to handle their HTTP/HTTPS communications. This attack can be instigated through DHCP, DNS, or NetBios manipulation.

2.1.2. Attacking trust

The foundation of the HTTPS trust model is rooted in the trust delegated to third-party certification authorities. Signed certificates are validated against a list of trusted third parties by the end user's browser. Currently, Comodo, Symantec, GoDaddy, and Global Sign combine to issue 91.1% of all X.509 Public Key Infrastructure certificates in use by websites today (W4Techs, 2015).

Certificate authorities maintain a close relationship with browser and device vendors in order to provide certificate revocation services, timestamping services, and signed root certificates (GlobalSign, 2015). Unfortunately, two major flaws exist within this methodology enabling resultant exploitation vectors.

- Attackers can apply for legitimate certificates and use them nefariously
- Compromise of a major root certificate invalidates all subordinate certificates

A digital certificate attaches a server identity or domain name to a public key signed by a trusted certificate authority (Dacosta, Ahamad, & Traynor, 2012). Ideally, this identity-to-certificate binding would prevent adversary usage of legitimate certificates pertaining to secondary identities. In order to support content from third-party domains web servers often return content, including advertisements that are validated by

multiple certificates. A survey of popular web sites determined that on average 12 certificates were encountered when making requests against individual domain names (Dacosta, Ahamad, & Traynor, 2012). The interconnected nature of web content provides adversaries a large attack surface for nefarious usage of legitimately acquired certificates. Further because, due to performance considerations, many web applications limit encryption usage to client authentication. Attackers with access to protocol data can embed external resources into the web traffic stream. This fine-grained control of web sessions via MITM and enables several classes of attack to include the POODLE vulnerability discussed in section 2.3.

Root certificates are not immune to compromise. An Iranian operation known as Black Tulip instigated an attack against the Dutch certificate authority DigiNotar. Attackers issued 531 rogue SSL certificates for many popular domains including Google (Leyden, 2011). Forged certificates were used in a DNS cache poisoning attack to intercept encrypted traffic for over 300, 000 victims over a period of 26 days (Hoogstraaten, 2012).

In addition to subverting SSL/TLS over the network, it is also possible for advanced attackers to perform covert post-compromise information gathering by installing a trusted root certificate authority to the system and locally intercepting targeted traffic. This attack is similar to Superfish in that the CA trust model is invalidated by locally adding nefarious entries into the hierarchy. Attackers can use this technique to recover sensitive information including web application login credentials.

Microsoft provides a set of .NET functions that can control how Windows uses cryptography. Using PowerShell new root certificates can be added. The [System.Security.Cryptography.X509Certificates.StoreName] enumeration (Microsoft, 2014) provides enhanced capabilities that can be leveraged maliciously for information gathering.

Once trust conditions have been established attackers can use publicly available tools to locally intercept sensitive traffic. Interceptor is a PowerShell-based HTTPS intercepting proxy written by Casey Smith and released at DerbyCon in 2014 (Smith,

2014). Interceptor is built on a set of standalone functions that can be individually actuated enabling attackers to trigger components of SSL MITM as necessary.

Functions:

- Receive-ClientHttpRequest
- Send-ServerHttpRequest
- Receive-ServerHttpResponse
- Invoke-CreateCertificate

Tool execution is straightforward:

```
Interceptor.ps1 -ProxyServer localhost -ProxyPort 8888
```

Figure 2 – Interceptor Execution

2.2. Heartbleed Bug

The Heartbleed Bug or CVE-2014-0160 refers to a vulnerability that existed in OpenSSL from December 2011 through March of 2014. This vulnerability garnered an unprecedented amount of public attention and was exploited pervasively in the wild.

Figure 3 – Heartbleed

While heartbleed is not a vulnerability of the Secure Socket Layer itself, it is a critical flaw in one of the most popular implementations. Heartbleed illustrates an important point when discussing attacks against a protocol. Often the available attack surface is not limited to a Request for Comments (RFC) or set of standards. Devastatingly successful attacks are just as likely to exist in the individual implementations of a protocol, as they are within the logic behind the standard itself. In the case of CVE-2014-0160 the OpenSSL implementation led to vulnerabilities within the vast majority of products using Secure Socket Layer encryption. Heartbleed is a buffer over-read vulnerability in the OpenSSL heartbeat extension. It effectively allows attackers to access up to 64 kilobytes of random memory that the web server is handling.

This information can include sensitive plaintext information such as user passwords, email addresses, or even social security numbers.

2.2.1. Heartbleed Attack Plan

Triggering the Heartbleed vulnerability is quite simple, but leveraging it to acquire useful data is more complex. The difficulty is twofold, 64 kilobytes is a relatively small segment of memory to work with and the vast majority of data returned is not useful. Overcoming these limitations involves a basic three step process:

1. Connect to the target and trigger heartbeat over-read to dump 64KB of data
2. Repeat many times to generate large dump file
3. Parse memory dump for readable and useful ASCII strings

Unlike many attacks targeting SSL or its implementations, attackers trigger Heartbleed against either side of the connection directly. This means that a MITM position is not required for exploitation.

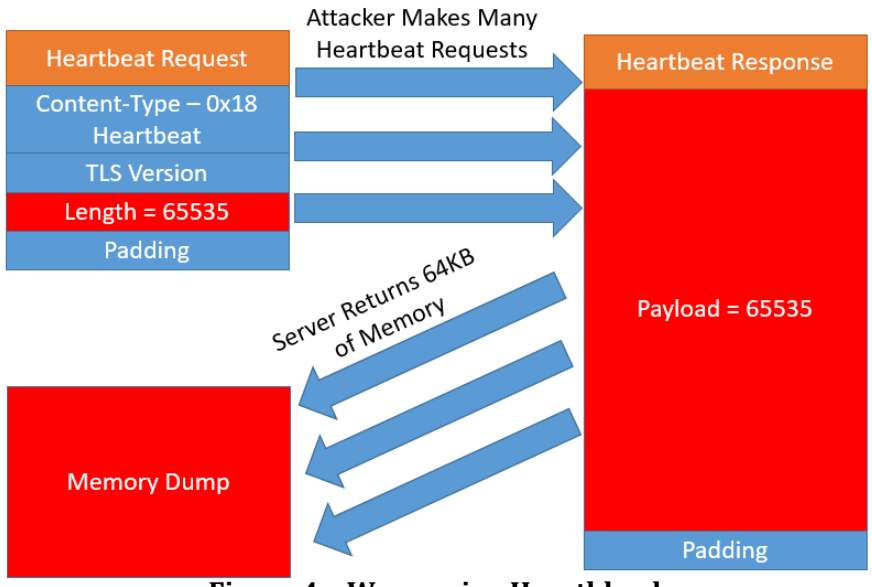

Figure 4 – Wargaming Heartbleed

2.2.2. Triggering the Vulnerability

Many security tools exist to trigger the Heartbleed vulnerability. In addition to a working metasploit module various python scripts exist to automatically trigger the vulnerability and dump sensitive data.

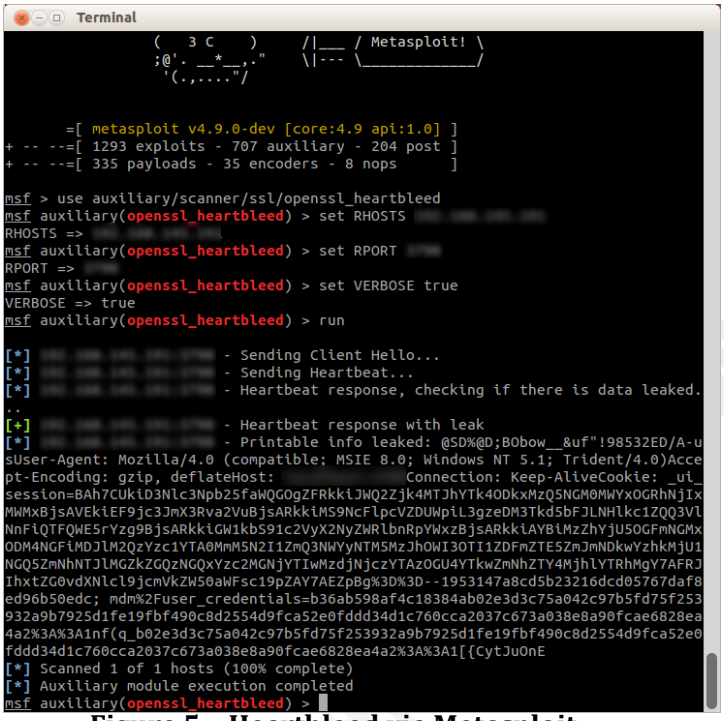

Figure 5 – Heartbleed via Metasploit

Exploit code capable of triggering heartbleed is readily available to the public. Following its announcement on 1, April 2014, attackers rapidly integrated the vulnerability into their toolkit and launched attack campaigns targeting vulnerable webservers. The value of information returned by Heartbleed attack attempts is often critical and can lead to compromise of user accounts as well as, in some cases, the compromise of the underlying webserver.

SensePost developed a comprehensive python-based test script based on the original PoC developed by Jared Stafford (White, 2014).

```
python heartbleed-poc.py –n 1000 localhost
```

The SensePost code allows looping of the vulnerability in order to trigger Heartbleed many times. The information received in heartbeat responses is stored in a dump.bin file.

2.2.3. Handling the Memory Dump

Information gathering, in order to demonstrate the danger and effectiveness of this attack, can be performed against the dump file. By chaining standard Unix commands an

attacker to quickly parse large quantities of webserver data in order to retrieve desired content including social security numbers as demonstrated below:

```
strings dump.bin | grep '[0-9]\{3\}-[0-9]\{2\}-[0-9]\{4\}'
```

2.2.4. Reverse Heartbleed

In addition to handling encryption for webservers, OpenSSL is the cryptographic implementation leveraged by many browsers. Reverse heartbleed is an attack targeting clientside OpenSSL employment. In these cases susceptible devices making connections to malicious servers, are vulnerable to exploitation. The server itself could bypass victim browser protections in order to read arbitrary information from the victim's system. Though exploitation of this condition has not been observed in the wild it does raise questions concerning the security of 50 million Android devices using OpenSSL. It further increases the life expectancy of Heartbleed attacks by introducing platforms that are much less likely to be patched on a regular basis.

Heartbleed remains a significant threat with over 200,000 publicly accessible devices alone remaining vulnerable according to the Shodan vulnerability search tool (The Register, 2015). In addition to publically available information systems, vast numbers of internally networked devices and SCADA systems connected across 2G/3G networks likely remain critically vulnerable to exploitation.

2.3. POODLE

The Padding Oracle On Downgraded Legacy Encryption or POODLE attack affects all implementations of SSL 3.0 with cipher-block chaining (CBC) mode ciphers (US-CERT, 2014). When coupled with techniques like SSLv3 downgrade this vulnerability poses a significant threat to the integrity of a diverse spectrum of information systems. The severity of this flaw has led to wide support of TLS_FALLBACK _SCSV and even mandatory enforcement of TLS by both clients and servers. Although the vast majority of fully updated clients have integrated solutions to

protect against the risk now inherent to SSL utilization, many even moderately out of date systems fail to have any protections built in.

Browser	SSL 3.0 Support	POODLE Status
Chrome	Removed	Not Effected
Android OS Browser	Removed	Not Effected
Firefox (Mobile)	Removed	Not Effected
Internet Explorer	Disabled	Mitigated (IE 11 only)
Internet Explorer (Mobile)	Default	Vulnerable
Safari	Removed	Not Effected
Safari (Mobile)	Removed	Not Effected

Table 1 – Browser POODLE vulnerability status

2.3.1. POODLE Attack Plan

In order to successfully implement the POODLE attack an adversary must overcome several hurdles:

1. Gain access to protocol data
2. Ensure SSLv3 encryption is utilized
3. Perform POODLE attack to decode arbitrary data

In order to perform the POODLE attack the aggressor will generally require a MITM position to their target. As discussed in section 2.2.1 access to protocol data can be acquired through various means. Once MITM access has been achieved an attacker must ensure that the target data is encrypted with SSLv3 in a typical connection between a webserver and browser this can be enforced by attacking secure channel negotiation to effect a SSLv3 downgrade attack. Finally, the attacker can now leverage the POODLE vulnerability to decrypt arbitrary bytes of information.

2.3.2. Attacking Secure Channel Negotiation

Transport Layer Security secure channel negotiation begins with a client-server handshake. If an attacker with MITM access intercepts and drops the initiating ClientHello packet and terminates the connection, the target browser will re-attempt establishment of a secure channel using a weaker cipher. Attackers can implement this

attack very easily by using readily available open source tools. The Subterfuge Framework pre-proxy enables straightforward manipulation of intercepted packets through the Scapy python packet-crafting library.

Regular expressions can be used to identify and drop TLS instantiation packets:

```
if re.search('\x16\x03\x01.{2}\x01',orig_load,flags=0):
        p.drop()
```

Dropping the current connection should prompt connection reattempts at lower security levels:

```
new_packet = IP(dst=pkt[IP].dst, src=pkt[IP].src)/TCP()
new_packet[TCP].sport = pkt[TCP].sport
new_packet[TCP].dport = pkt[TCP].dport
new_packet[TCP].seq = pkt[TCP].seq
new_packet[TCP].ack = pkt[TCP].ack
new_packet[TCP].flags = 'FA'
send(new_packet)
```

(Blauzvern, 2014)

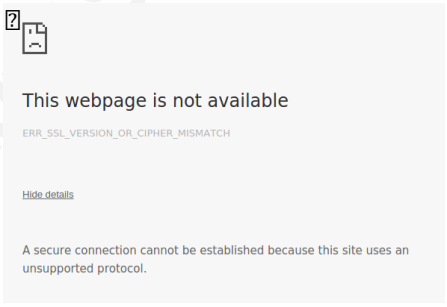

Figure 6 – Unsupported Protocol

Fortunately, most fully updated browsers are no longer susceptible to this attack. Google Chrome versions 44 and newer return unsupported protocol responses to SSLv3 connection attempts. However, Internet Explorer versions 6-10 are not only susceptible to this attack, but SSLv3 is enabled by default. Furthermore, prior to IE 11 no default mitigations for the POODLE attack were available to users. According to current Internet

trends, 23.73% of web traffic as of this writing is still a result of IE versions 6-10 all of which are inherently vulnerable to SSLv3 downgrade attacks as demonstrated in the image below (Netmarketshare, 2015).

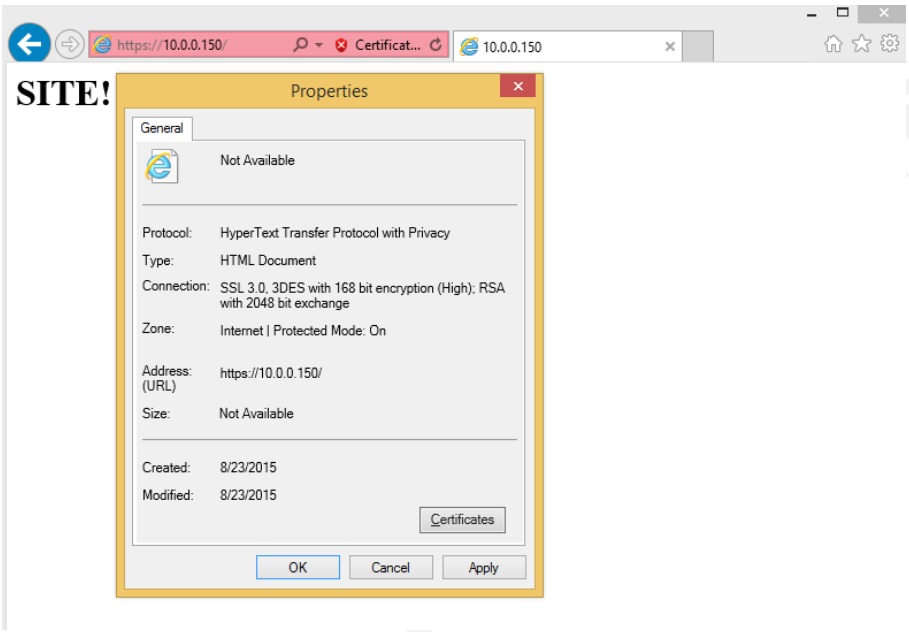

Figure 7 – SSLv3 Downgrade

2.3.3. Analysis of the POODLE Vulnerability

Reliable exploitation of the POODLE Vulnerability requires the ability to control the victim's browser. Hooking the browser with a snippet of JavaScript that initiates requests containing sensitive information, affords the attacker an opportunity to fiddle with session ciphertext indefinitely.

```
SSLv3    132  Client Hello
TCP       66  30001→41568 [ACK] Seq=1 Ack=67 Win=43776 Len=0 TSv
SSLv3    780  Server Hello, Certificate, Server Hello Done
TCP       66  41568→30001 [ACK] Seq=67 Ack=715 Win=45184 Len=0 T
SSLv3    780  Server Hello, Certificate, Server Hello Done
TCP       66  50698→30002 [ACK] Seq=67 Ack=715 Win=45184 Len=0 T
SSLv3    278  Client Key Exchange, Change Cipher Spec, Finished
SSLv3    203  Client Key Exchange
```

Figure 8 – SSLv3 Protocol Traffic

If an attacker modifies a portion of the ciphertext and invalidates the message the server will reject the record and close the connection; however, if the record is accepted the attacker has an opportunity to determine the plaintext of one byte of arbitrary

information. The weakness exists because the SSL CBC padding is nondeterministic and not guaranteed by the Message Authentication Code (MAC). This means that the integrity of the padded block cannot be assured (Moller, Duong, & Kotowicz, 2014).

The SSL ciphertext record contains three relevant sections: record content, MAC, and record padding. In SSL blocks are either 16 bytes (AES) or 8 (3DES) with the final block padded with garbage to enforce proper message length. If the record + 20 byte MAC are perfectly aligned to form complete blocks a final padding block will be strapped onto the message and contain only padding. The message is validated by taking the XOR of the decrypted final block with the previous block of ciphertext, the initialization vector (IV), where the result must equal seven.

CONTENT
MAC
PADDING

If an attacker controls the message length to force the last block to contain only padding, then replaces the final block with an earlier block of ciphertext the server will accept the record only if the decrypted block XORed with the previous block's ciphertext equals seven. This means that the final encrypted byte XORed with the number seven equals an unencrypted byte of sensitive information. In practice the attack is performed as follows:

Attacker uses the URL parameter to control message length and CBC block alignment:

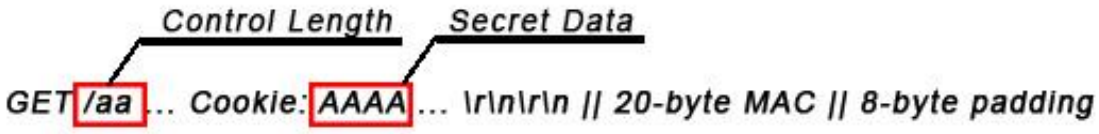

Figure 9 – POODLE Attack Message Structure

Next the attacker must fiddle with the blocks in the SSL stream in order to decrypt sensitive information like web cookies:

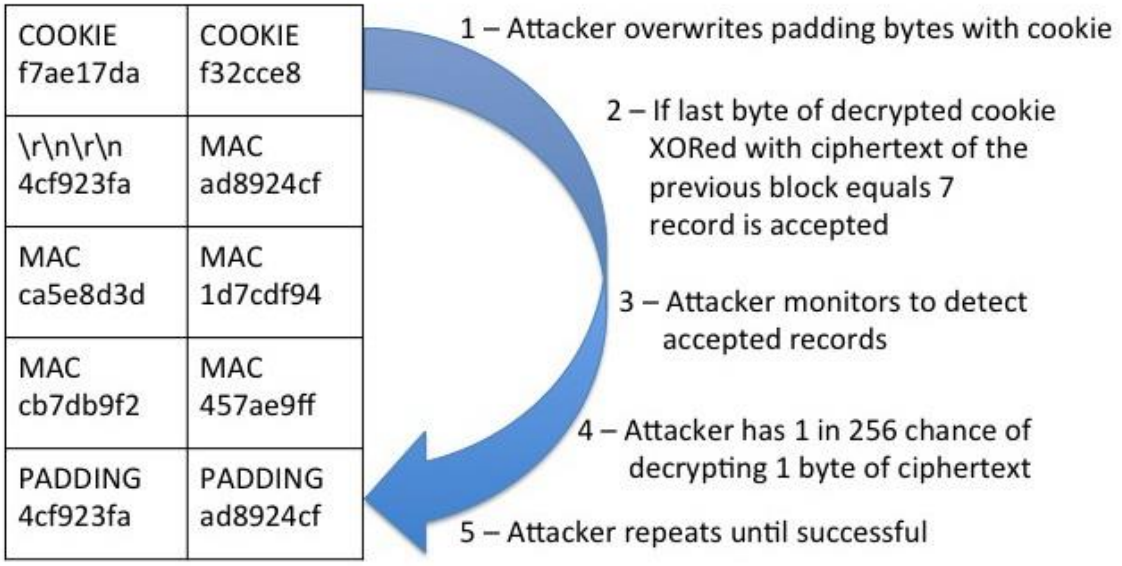

Figure 10 – POODLE Attack Procedures

When implementing attacks of this sort, Wireshark is a vital instrument in the security researcher's toolkit. It supports SSL decryption by loading of known keys. In order to corroborate appropriate message content and confirm successful decryption, researchers can view the stream with Wireshark and very easily identify successful protocol traffic as shown below.

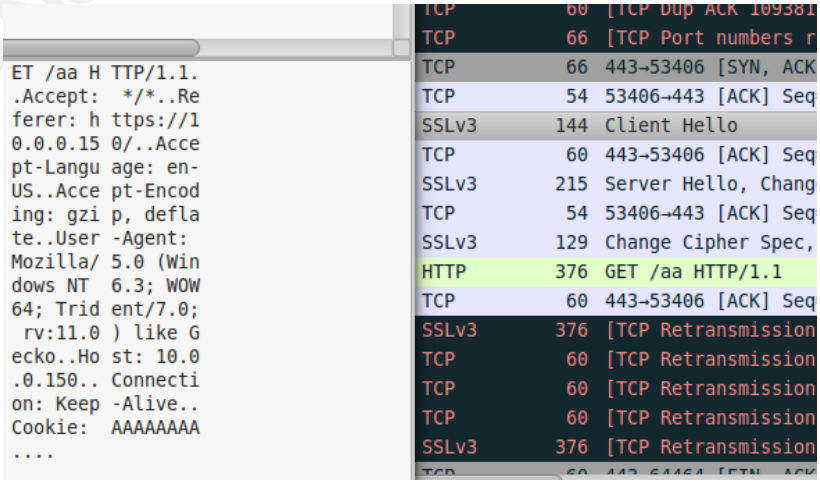

Figure 11 – POODLE Decryption via Wireshark

Author Name, email@address

2.4. Superfish

The Superfish scandal is not a vulnerability in SSL. It is an affront to the third party trust model. Superfish is an advertisement firm focused on generating image driven sales with their proprietary adware suite (Hoge, 2014). Beginning in September of 2014, Superfish adware was packaged with the default operating system build of most Lenovo computers. In addition to providing a degraded browsing experience to users, Superfish introduced a key vulnerability to the system that effectively rendered SSL irrelevant. In the wake of the news hurricane surrounding the "Lenovo fiasco" Superfish closed their doors, rebranding as JustVisual (Brinkman, 2015).

The integrity of HTTPS communications is seated in the certification authority trust model. In order to inject "targeted ads" into user browsing experiences the Superfish adware suite added a root certificate authority to the computer on install. In the case of Lenovo devices this occurred when the operating system image was loaded onto the machines at the factory.

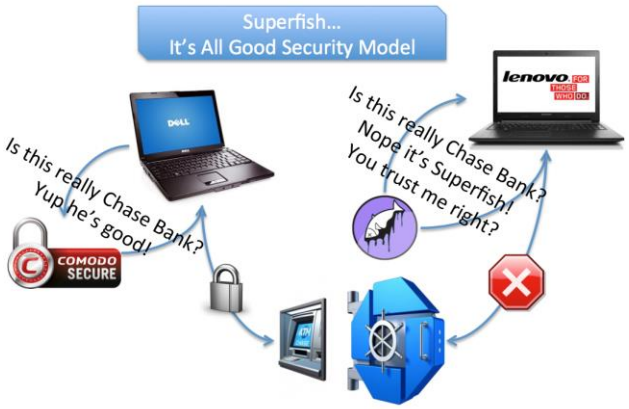

Figure 12 – Superfish CA Trust

HTTPS certificates for any domain that were signed by the Superfish root CA are transparently accepted as valid by effected systems. Security researchers quickly retrieved the key and certificate passphrase (komodia) from system memory using procdump (Graham, 2015).

The Superfish key was rapidly integrated into attack tools like the Subterfuge Framework, which supports automated interdiction of sensitive SSL traffic and uses the Superfish key by default.

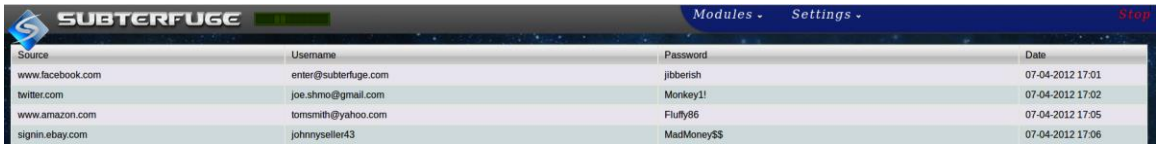

Figure 13 – Subterfuge SSL Password Harvesting with Superfish Key

2.4.1. Triggering the Vulnerability

Exploitation of effected systems generally requires MITM. The Subterfuge MITM Framework can implement an ARP Cache Poison, WPAD Hijack, Rogue AP, or Rogue DHCP attack in order to easily acquire this level of access to victim data. Attacker's seeking to leverage the Superfish root certificate can configure Subterfuge to SSL intercept. In this mode SSL/TLS will be preserved, while the framework attempts to access encrypted data by presenting the victim with a self-signed SSL certificate.

Figure 14 – Superfish Certificate

By default the Subterfuge Framework utilizes the Superfish private key and certificate in order to perform SSL intercept MITM attacks. Standard utilization of the toolkit is automatically transparent to users of Lenovo PCs.

3. Conclusion

SSL version 3.0 was developed by Netscape and has been in use since 1996; since its inception it has been the keystone of secure communication on the World Wide Web (Netscape, 1997). Decades of security research have unveiled flaws in nearly every aspect of the fabric of the protocol itself. Continued usage of the protocol endangers key information resources and degrades the perception of freedom and privacy on the Internet. SSL is dead and it is time to move on.

Attack surface analysis reveals that many of the strategies that are used to attack SSL are based on the trust model in its entirety. In TLS the trust model has

not changed. As a result attacks against the new system will maintain the general form seen in attacks against SSLv3:

- Subverting the trust model entirely
- Attacking protocol implementation
- Cracking the encryption directly

Fully entrusting one system of confidentiality with the security of an institution as foundational to the Internet as HTTPS may be the simplest alternative, but it is far from secure. The public attention that SSL has received this year alone highlights the protocol's importance, and demonstrates the need for diverse means of secure communication.

Author Name, email@address

4. References

Akamai. (2015, November 4). *Akamai*. Retrieved from Net Usage Index by Industry: http://www.akamai.com/html/technology/nui/industry/index.html

AlertLogic. (n.d.). *POODLE - The man-in-the-middle attack on SSLv3 .* Retrieved November 3, 2015, from Alert Logic: https://www.alertlogic.com/blog/poodle-the-man-in-the-middle-attack-on-sslv3/

Biondi, P. (2005, November 16). *Network packet forgery with Scapy.* Retrieved from SecDev: http://www.secdev.org/conf/scapy_pacsec05.handout.pdf

Blaich, A. (2015, April 23). *Questioning the chain of trust: investigations into the root certificates on mobile devices*. Retrieved from Bluebox: https://bluebox.com/technical/questioning-the-chain-of-trust-investigations-into-the-root-certificates-on-mobile-devices/

Blauzvern. (2014, August 19). *Man-in-the-Middle TLS Protocol Downgrade Attack*. Retrieved from Praetorian: https://www.praetorian.com/blog/man-in-the-middle-tls-ssl-protocol-downgrade-attack

Brinkmann, M. (2015, June 21). *Superfish closes shop, relaunches as JustVisual.* Retrieved from ghacks: http://www.ghacks.net/2015/06/21/superfish-closes-shop-relaunches-as-justvisual/

Dacosta, I., Ahamad, M., & Traynor, P. (2012). *Trust No One Else: Detecting MITM Attacks Against SSL/TLS Without Third-Parties.* Converging Infrastructure Security (CISEC) Laboratory, Georgia Tech Information Security Center (GTISC). Georgia Institute of Technology.

Duncan, R. (2015, May 13). *Counting SSL certificates*. Retrieved from http://news.netcraft.com/archives/2015/05/13/counting-ssl-certificates.html

ENISA. (n.d.). *Operation Black Tulip: Certificate authorities lose authority*. Retrieved November 4, 2015, from European Network and Infromation Security Agency: https://www.enisa.europa.eu/media/news-items/operation-black-tulip

GlobalSign. (2015, November 4). *Certificate Authorities & Trust Hierarchies*. Retrieved from GMO Internet Group: The Transport Layer Security (TLS) Protocol

Goodin, D. (2015, April 2015). *Google Chrome will banish Chinese certificate authority for breach of trust* . Retrieved from ArsTechnica: http://arstechnica.com/security/2015/04/google-chrome-will-banish-chinese-certificate-authority-for-breach-of-trust/

Author Name, email@address

Graham, R. (2014, October 14). *Some POODLE notes.* Retrieved from Errata Security: http://blog.erratasec.com/2014/10/some-poodle-notes.html#.Vdj1InUVhBd

Graham, R. (2015, February 19). *Extracting the SuperFish certificate.* Retrieved from Errata Security: http://blog.erratasec.com/2015/02/extracting-superfish-certificate.html#.VjpCzq6rRE4

Green, M. (2014, October 2014). *Attack of the week: POODLE.* Retrieved from Cryptography Engineering: http://blog.cryptographyengineering.com/2014/10/attack-of-week-poodle.html?m=1

Grigorik, I. (2013). *High Performance Browser Networking What every web developer should know about networking and web performance.* O'Reilly Media. Retrieved from http://chimera.labs.oreilly.com/books/1230000000545/ch04.html#_testing_and_verification

Hoge, P. (2014, October 21). *Superfish dives deep into visual search.* Retrieved from San Francisco Business Times: http://www.bizjournals.com/sanfrancisco/feature/fast-100-superfish-dives-deep-into-visual-search.html

Hoogstraaten, H. (2011). *Interim Report - DigiNotar Certificate Authority breach "Operation Black Tulip".* Delft: FoxIT.

Hoogstraaten, H. (2012). *Black Tulip - Report of the investigation into the DigiNotar Certificate Authority breach .* Delft: FoxIT.

Hubert, K. (n.d.). *Fedora Project · Alexa top 1 million HTTPS Scans*. Retrieved from Scans.io: https://scans.io/study/fedora-cipherscan

IETF. (2008, August). *RFC 5246 - The Transport Layer Security (TLS) Protocol*. Retrieved from IETF: http://tools.ietf.org/html/rfc5246

Kario, H. (2015, October 3). Retrieved from Security Pitfalls: https://securitypitfalls.wordpress.com/

Langley, A. (2014, October 14). *POODLE attacks on SSLv3.* Retrieved from ImperialViolet: https://www.imperialviolet.org/2014/10/14/poodle.html

Leyden, J. (2011, September 6). *Inside 'Operation Black Tulip': DigiNotar hack analysed*. Retrieved from The Register: http://www.theregister.co.uk/2011/09/06/diginotar_audit_damning_fail/

Masnick, M. (2015, February 23). *Thought Komodia/Superfish Bug Was Really, Really Bad? It's Much, Much Worse!* Retrieved from Techdirt: https://www.techdirt.com/articles/20150223/07363930113/thought-komodiasuperfish-bug-was-really-really-bad-its-much-much-worse.shtml

McKinley, H. (2003). *SSL and TLS: A Beginners Guide*. (SANS, Producer) Retrieved from SANS Reading Room: www.sans.org/reading-room/whitepapers/protocols/ssl-tls-beginners-guide-1029

Author Name, email@address

Microsoft. (2003, March 28). *What is TLS/SSL?* Retrieved from TechNet: https://technet.microsoft.com/en-us/library/Cc784450(v=WS.10).aspx

Microsoft. (2014, March 31). *PowerShell to add a Certificate to "Trusted Root Certification Authorities"*. Retrieved from TechNet: https://social.technet.microsoft.com/Forums/en-US/8e016573-9191-4152-8c4e-b74d739f5894/powershell-to-add-a-certificate-to-trusted-root-certification-authorities?forum=winserverpowershell

Moller, B., Duong, T., & Kotowicz, K. (2014, September). *This POODLE Bites: Exploiting The SSL 3.0 Fallback.* Retrieved from OpenSSL: https://www.openssl.org/~bodo/ssl-poodle.pdf

Moser, J. (2009, June 10). *The First Few Milliseconds of an HTTPS Connection*. Retrieved from Moserware: http://www.moserware.com/2009/06/first-few-milliseconds-of-https.html

Netscape. (1997, June 14). *The SSL Protocol*. Retrieved November 4, 2015, from Netscape: https://web.archive.org/web/19970614020952/http://home.netscape.com/newsref/std/SSL.html

Red Hat Product Security. (2014, October 15). *POODLE – An SSL 3.0 Vulnerability (CVE-2014-3566) .* Retrieved from RedHat - Security Blog: https://securityblog.redhat.com/2014/10/15/poodle-a-ssl3-vulnerability-cve-2014-3566/

Rorot. (2013, October 28). *Infosec Institute*. Retrieved from SSL Attacks: http://resources.infosecinstitute.com/ssl-attacks/

Sarkar, P. G., & Fitzgerald, S. (2013, August 15). *ATTACKS ON SSL A COMPREHENSIVE STUDY OF BEAST, CRIME, TIME, BREACH, LUCKY 13 & RC4 BIASES.* San Francisco: iSECparners. Retrieved from iSECpartners: https://www.isecpartners.com/media/106031/ssl_attacks_survey.pdf

Smith, C. (2014, September 27). *Interceptor: A PowerShell SSL MITM Script* . Retrieved from Irongeek: http://www.irongeek.com/i.php?page=videos/derbycon4/t112-interceptor-a-powershell-ssl-mitm-script-casey-smith

Smith, C. (n.d.). *Interceptor*. Retrieved November 4, 2015, from GitHub: https://github.com/subTee/Interceptor

The Register. (2015, September 2015). *Thought Heartbleed was dead? Nope – hundreds of thousands of things still vulnerable to attack*. Retrieved from The Register: http://www.theregister.co.uk/2015/09/15/still_200k_iot_heartbleed_vulns/

Toussain, M. (2015, March 2). *EXPLOITING SUPERFISH WITH SUBTERFUGE.* Retrieved from Kinozoa: http://kinozoa.com/blog/exploiting-superfish-subterfuge/

Author Name, email@address

Trustworthy Internet Movement. (2015, October 3). *SSL Pulse*. Retrieved from Trustworthy Internet: https://www.trustworthyinternet.org/ssl-pulse/

US-CERT. (2014, December 10). *SSL 3.0 Protocol Vulnerability and POODLE Attack.* Retrieved from United States Computer Emergency Readiness Team: https://www.us-cert.gov/ncas/alerts/TA14-290A

W3Techs. (2015, November 4). *Usage of SSL certificate authorities for websites*. Retrieved from W3Techs: http://w3techs.com/technologies/overview/ssl_certificate/all

White, D. (2014, July 10). *Heartbleed-PoC*. Retrieved from GitHub: https://github.com/sensepost/heartbleed-poc

Author Name, email@address