CS 427, Final Project
POODLE
Padding Oracle on Downgraded Legacy Encryption

Cody Malick
malickc@oregonstate.edu

March 15, 2017

Abstract

This paper outlines the POODLE exploit. It gives details on how it allows an adversary to repeatedly query a server using SSL version 3.0, and eventually decrypt a block one byte at a time. The exploit was first published by the Google Security Team on October 14, 2014 by Bodo Moller, Thai Duong, and Kryzsztof Kotowicz. While SSL version 3.0 is quite old, it is still in use today for browser backward compatibility. Later, an updated version of the POODLE exploit was published showing successful attacks against TLS.

Introduction

POODLE is an attack on Secure Socket Layer version 3.0 that was made public in 2014. It was announced on Google's Security Blog, co-published by Bodo Moller, Thai Duong, and Kryzsztof Kotowicz. POODLE allows an attacker targeting SSL 3.0 to decrypt one byte of an encrypted SSL block one out of two-hundred fifty-six attempts. While this is an average, being able to decrypt a byte of an encrypted block that quickly is quite a security flaw.[1] It was later found that the vulnerability extended to TLS version 1.0. It is more of an implementation problem. Some implementations of TLS in popular software did not correctly check the padding after decryption. [2]

The goal of SSLv3 was to encrypt and keep secret communication between a web browser, and a web server. This is accomplished through an initial secret-sharing hand shake. It then computes a MAC from the shared secret. Once message exchange begins, it uses AES CBC-Mode to encrypt and decrypt messages.[3] The goal of the protocol was to provided secrecy, authenticity, and integrity.

POODLE is executed using a padding oracle attack. A padding oracle attack, simply described, is abusing the fact that a web server will through an error on a valid ciphertext if the padding is bad. This allows an attacker to verify whether or not an arbitrary ciphertext has valid padding. Furthermore, with this information, and adversary can decrypt the arbitrary ciphertexts![4]

Context

To completely understand the attack, the context should first be made clear. SSL version 3.0 is quite dated, and should for all intents and purposes be completely retired. Modern browsers, however, use SSLv3 as a fall back for compatibility purposes. The browser will first try to connect to the web server using TLS version 1.2, the de facto standard for web communication. If that fails, it is default behavior in browsers to fall back to earlier standards, such as SSLv3.[1]

While browsers continue to support SSLv3, this attack will continue to be an issue. A savvy attacker with the proper resources could catch web requests with a man in the middle attack, and prevent all non-SSLv3 requests from going through. This would force the use of SSLv3 if it is supported by the web server.[5]

Padding

The primary point of vulnerability that the attack exploits is in the padding of the encrypted message. Padding in SSLv3 is accomplished by measuring the difference between the last block size, and the required size of the block. In this case, the block size must be a multiple of sixteen. In this case, the AES CBC mode block size is sixteen bytes, sixty-four bits. The SSLv3 implementation then measures the difference d, and fills the space $d - 1$ with random byte values. The last byte is then filled with the size of the padding. The padding can be zero, or a maximum of sixty-three bytes.[3] SSLv3 does not check

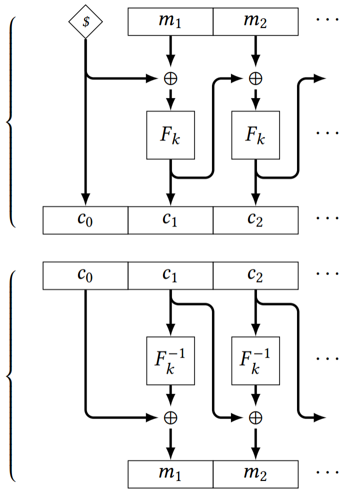
When attempting to decrypt, the algorithm checks the last byte for the size of the padding. The major issue with this becomes apparent here. Because the padding of SSLv3 is nondeterministic and the padding is not covered by the message authentication code provided, any value can be provided in the place of the padding byte.[1] More on this in the 'exploit' section.

AES CBC Mode

Cipherblock Chaining mode is one of the most popular modes for AES operation. It provides chosen plaintext attack security, and non-deterministic encryption using an initialization vector. Here is a diagram from the class textbook that illustrates how this process works:[4]

Construction 9.2
(CBC Mode)

```
Enc(k, m_1 ⋯ m_ℓ):
  $c_0 \leftarrow \{0,1\}^{blen}$;
  for $i = 1$ to $\ell$:
    $c_i := F(k, m_i \oplus c_{i-1})$
  return $c_0 c_1 \cdots c_\ell$
```



```
Dec(k, c_0 ⋯ c_ℓ):
  for $i = 1$ to $\ell$:
    $m_i := F^{-1}(k, c_i) \oplus c_{i-1}$
  return $m_1 \cdots m_\ell$
```

While understanding the underlying encryption algorithm is important to understanding the exploit, the primary point of vulnerability is the padding.

The Exploit

The exploit is accomplished by executing the following steps:

SSL Version 3.0 Fallback

While not required, the vast majority of modern web browsers and web servers use modern versions of TLS. Per Mozilla, which has since completely disabled SSLv3 in the browser, only about 0.3% of all their requests used SSLv3.[5] With that in mind, it is likely that an adversary wishing to utilize this attack must first force the web browser to fall back to SSLv3. This would typically be done during the initial connection between the web browser and web server when they first try to agree on a protocol to use.

An adversary, with a man-in-the-middle setup, could disallow any connections that were not SSLv3. If both the web browser and web server support SSLv3, they would eventually fall back on using the protocol.[1] Once the connection has been established, the stage is set for the attack.

Replacing the Padding Block

Because SSLv3 does not verify the integrity of the padding, or the padding byte due to its nondeterministic nature, this is the point of attack. An adversary who is able to intercept and alter traffic (our man-in-the-middle setup) can replace the padding block with any other block.

More formally, given ciphertext blocks $C_1 ... C_n$, an initialization vector C_0, and the block-cipher decryption D_k using key K, the recipient of the message would use the following process to decrypt:[1]

$$P_1 ... P_n, P_i \ = \ D_K(C_i) \ \oplus \ C_{i-1}$$

The receiver would then check the padding on the end (C_n), and verify the MAC. The attacker could replace C_n with any other block of encrypted ciphertext C_i from the same message. Then the ciphertext will still be accepted if the decryption of the last byte of block C_i is equal to the value of the padding byte. This is the padding oracle attack. [1].

This means that the attacker knows that the last byte of the decrypted message was in-fact equal to the value of the padding byte. The probability of this attack being successful turns out to be extremely probable. With only two-hundred fifty-six possibilities for any given byte, the probability of success simply $\frac{1}{256}$. With such a reliable attack, the attacker can easily iterate through the ciphertext blocks, shifting the bytes when a new byte has been decrypted. This allows for full decryption of entire blocks given enough time. Being able to make several hundred or thousand attacks in a few seconds means that an adversary can reliably decrypt one byte of a message at a time.

While this is fairly devastating, it should again be pointed out that an attacker must already have a fairly compromised system in order to execute this method. The attacker must have two requirements in place: the attacker must control some part of the client side connection, and the attacker must have visibility of the resulting ciphertext. These two prerequisites require a fairly compromised system to begin with.[6]

TLS 1.0 Vulnerability

It was discovered a few months after the initial POODLE announcement was made that certain implementations of TLS were vulnerable to the attack. TLS, unlike SSLv3, is very strict in the formatting of its padding. Certain implementations, however, did not check the padding structure after decryption. This leads to the same vulnerability to padding oracle attacks that SSLv3 had, minus the need to downgrade the security protocol being used.[2]

The Fix

There is a patch released for SSLv3 to prevent the POODLE vulnerability. The patch only works, however, if both the client and the server are up to date. The real danger lies in the fact that SSLv3 is a deprecated protocol that should no longer be used.[7] It's hard if not impossible to ensure that every server or client is patched and up to date with the latest SSLv3 versions. More importantly, many websites and modern browsers have completely disabled SSLv3 to prevent POODLE from being an issue in the first place. Modern browsers enforce the use of TLS 1.2, and will fail to connect to older servers that do not support modern encryption standards.

References

[1] K. K. Bodo Moller, Thai Duong. (2014, October) This poodle bites: Exploiting the ssl 3.0 fallback. [Online]. Available: https://www.openssl.org/ bodo/ssl-poodle.pdf

[2] I. Ristic. (2014, December) Poodle bites tls. [Online]. Available: https://blog.qualys.com/ssllabs/2014/12/08/poodle-bites-tls

[3] P. K. A. Freier, P. Karlton. (2011, August) The secure sockets layer (ssl) protocol version 3.0. [Online]. Available: https://tools.ietf.org/html/rfc6101

[4] M. Rosulek, *The Joy of Cryptography.* School of Electrical Engineering and Computer Science, Oregon State University, 2017, vol. Draft, January 3, 2017.

[5] R. Barnes. (2014, October) The poodle attack and the end of ssl 3.0. [Online]. Available: https://blog.mozilla.org/security/2014/10/14/the-poodle-attack-and-the-end-of-ssl-3-0/

[6] U. S. C. E. R. Team. (2014, October) Ssl 3.0 protocol vulnerability and poodle attack. [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA14-290A

[7] M. Spagnuolo. (2015, January) How to disable sslv3. [Online]. Available: http://disablessl3.com