BEAST - Browser Exploit Against SSL/TLS

Yuval Lihod
Tel Aviv University
lihodyuval@gmail.com

Abstract

This report provides an overview of the BEAST attack against TLS/SSL protocols and describes the exploit with a complete End-to-End (E2E) attack demonstration. The E2E attack simulates a real-world scenario where a victim accesses malicious website which inject his browser Java Applet (or javascript), enable the attacker to obtain secret cookies of the victim. This report covers the attack implementation, the CTF (Capture The Flag) design and instructions.

1 Introduction

The **BEAST** (CVE-2011-3389) is an attack that allow a MITM attacker to uncover information from an encrypted SSL/TLS 1.0 session by exploiting a known theoretical vulnerability in CBC mode, discovered in 2002. It could be exploited regardless of the type and strength of the block cipher.

To understand the BEAST vulnerability, several key concepts are required:

- **AES**: Block cipher - a deterministic algorithm that operates on fixed-length groups of bits, called blocks. Block size of AES is 16.
- **Initialization Vector(IV)**: An input to a cryptographic primitive being used to provide the initial state. The IV is typically required to be random or pseudo-random, but sometimes an IV only needs to be unpredictable.

  The BEAST attack utilize the fact that it IV is predictable in TLS 1.0 and SSL versions.
- **Mode of operation**: An algorithm that designed to allow securely transform amounts of data larger than a block, using block cipher.
- **Cipher Block Chaining(CBC)**: Mode of operation. In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted. To make each message unique, an IV must be used in the first block.

The SSL standard mandates the use of the CBC mode encryption with chained IVs; i.e., the IV used when encrypting a message should be the last block of the previous ciphertext. That leads to a vulnerability exploit by the BEAST attack - Combined with clever manipulation of block boundaries, the flaw allowed a MITM attacker sniffing encrypted traffic to discover small amounts of information without performing any decryption.
According to 'Qualys labs' as of January 2014, only 25.7% of websites support TLS 1.1 or 1.2, while the attack was discovered in 2011. [3]

2 Related Work

Several sources provide a detailed analysis of the attack:

- **T.Duong and J.Rizzo(2011)**: *Here Come The Ninjas* [2] is the original attack paper. It provides a comprehensive analysis of cryptographic aspect of the attack, the threat model and a broswer exploit implementations.
- **P.Rogaway (2002)**: *Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures.* [4], The vulnerability originally discovered by Phillip Rogaway in 2002.
- **W.Dai (2002)**: *An Attack Against SSH2 Protocol.* [1] Extension to Rogaway's attack to SSH protocol.

3 Attack Description

The E2E attack demonstrates the BEAST attack by setting up a TLS 1.0 bank server, malicious web, client and an attacker.

3.1 Server Setup

The server is configured with **TLS 1.0** to simulate the vulnerability. The server accepts TLS connections and handle HTTP requests.

3.2 Client Operations

The client establishes a connection to malicious website, controled by the attacker.

3.3 Malicious-Website Setup

When client accesses the web, it injects him Java Applet, which make the client open connections to his Bank website (the 'Server') and make requests that include the client's cookies.

3.4 Attacker Operations

The attacker perform MITM attack. He eavesdrop the communication between the victim (the 'Client') and the Bank, captures the cookie bearing requests. Then he perform the cryptographic part of the attack, and decide what will be the next path of the following request (that is how he controls the block boundaries).

3.5 Demonstration Steps

(1) Set up the bank server by running:

```
docker docker build -t
    server_image .
docker run --rm -t --network=
    host server_image
```

(2) Set up the malicious website by running:

```
docker build -t
    mal_server_image .
docker run --rm -t --network=
    host mal_server_image
```

(3) Run the attacker by executing:

```
docker build -t attacker_image
    .
docker run --rm -t --network=
    host attacker_image
```

(4) Run the client by executing:

```
docker build -t client_image .
docker run --rm -t --network=
    host client_image
```

4 Design

There are 2 CTF challenges: **Beginner** and **Advanced**. Each challenge created by removing some code parts of the E2E 'Attacker' implementation, where the Advanced challenge created by removing more parts than the Beginner.
Thus, you should start with the Advanced challenge, and only if you can't solve it try the esaier 'Beginner' challenge.

As the CTF created from the E2E attack, it has the same design: The CTF project is designed using Docker to containerize the bank server, malicious web server, attacker and client, ensuring isolation and reproducibility.
All of them, but the client, are implemented in Python. The client is consists of nothing but its Dockerfile.
Docker simplifies the deployment and setup for participants, allowing them to focus only on the implementation of the cryptographic part of the attack in the CTF.

The tools and libraries used include:

- **Docker**: To create isolated containers for the servers, attacker and client, ensuring the environment is consistent across different systems.
- **Java Development Kit (JDK)**: The client's browser uses JDK 7 to run the malicious website script, written in Java.
- **Firefox**: The client uses Firefox 7 which support the unpatched vulnerable version of TLS 1.0.
- **Python**: The attacker uses the following libraries: -`scapy` library to eavesdrop the communications between the client and the bank server (Imitate the MITM attack).
  -`socket` library to communicate with the client, make him send crafted requests.
  -`threading` library to run 2 threads: the first accounts for the cryptographic aspect and the second for the eavesdropping.
  Both servers use the `http` library. The bank's server also use the `ssl` library to support secure connections, using SSL or TLS.

5 CTF Instructions

You should complete the missing code parts denoted in '?'. Notice that each '?' might be more than 1 line of code. Any clues, if exist, will appear in a remark adjacent to the relevant missing code part.
In order to check your solution you should act in accordance with the instructions detailed in section 3.5, **but instead of** running the original attacker image (step 3) you need to run the following commands from inside the folder of the relevant CTF challenge:

```
docker build -t attacker_ctf_image .
docker run --rm -t --network=host
    attacker_ctf_image
```

If you were right then the attacker will print the client's cookie.

6 Conclusion

The BEAST attack demonstrates how cryptographic imperfections that were viewed as hypothetical or unreasonable could utilized to breach communications security.

When the direct assault was delivered in 2011, the TLS 1.1, which was published in 2006, particular had proactively been tended to for the issue with the CBC mode, yet it was not widely used. The moral of the story is that you shouldn't linger behind the latest security conventions.

By participating in this CTF, participants learn how to exploit the cryptographic imperfection in CBC mode in order to breach SSL/TLS communications security and decrypt confidential data.

References

[1] Wei Dai. 2002. An Attack Against SSH2 Protocol. (2002). http://www.weidai.com/ssh2-attack.txt

[2] Thai Duong and Juliano Rizzo. 2011. Here Come the Ninjas. (2011). https://nerdoholic.org/uploads/dergln/beast_part2/ssl_jun21.pdf

[3] Qualys labs. 2014. *Dashboard monitoring SSL/TLS support.* Technical Report. https://www.ssllabs.com/ssl-pulse/

[4] Phillip Rogaway. 2002. Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures. (2002). https://web.archive.org/web/20120630143111/http://www.openssl.org/~bodo/tls-cbc.txt