BEAST TITLE TO-FILL

Yang Shichu
Huazhong University of Science and Technology
TO-FILL

Shu Yi
Huazhong University of Science and Technology
TO-FILL

Su Haochen
Sichuan University
TO-FILL

Li Yucong
Shandong University
TO-FILL

Liao Haicheng
University of Electronic Science and Technology of China
TO-FILL

ABSTRACT
Transport Layer Security (TLS) is an protocol that provides communication security over networks. However, there is a flaw in TLS 1.0 where the initial vectors for block ciphers are predictable. The BEAST attack, with some prerequisites and efforts, allows attackers in the middle to decrypt those encrypted messages without knowing the key. This paper will demonstrate the procedures of the BEAST attack, and propose methods in simulation and vulnerability detection.

Keywords
BEAST attack, TLS flaws, CBC exploits, vulnerability detection

1. INTRODUCTION
Transport Layer Security (TLS) has several versions. The specification for TLS 1.0 is RFC 2246[1].

2. BACKGROUND
2.1 A glance at TLS
TLS protocols

2.2 CBC in block ciphers
CBC is one of the modes of operation used in block ciphers.

Supposing that $P_1, P_2, \cdots P_n$ are the plaintext blocks, with a initial vector IV, we have:

$$C_1 = E_k(P_1 \oplus IV)$$
$$C_i = E_k(P_i \oplus C_{i-1})(i \geq 2)$$

to obtain ciphertext blocks $C_1, C_2, \cdots, C_n$.

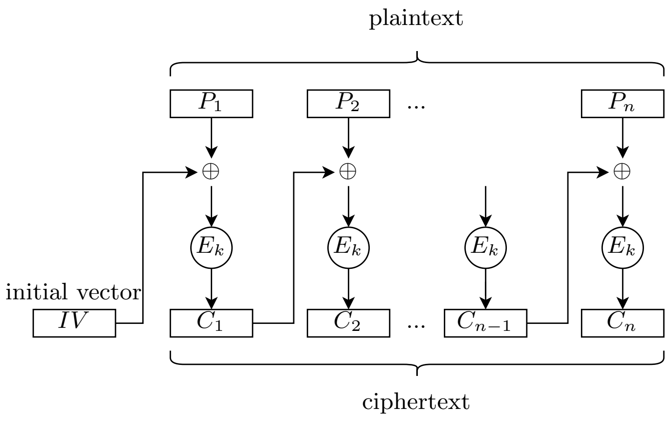


Figure 1: CBC encryptor

3. THREAT MODEL
3.1 Prerequisites for attackers
- TO-DO

4.

5. DEMONSTRATION

6. FEASIBILITY AND DEFENSE
6.1 Feasibility
While BEAST attacks are theoretically feasible, with the enhancement of security features of browsers and other clients, BEAST attacks are less and less practical for an attacker to exploit.

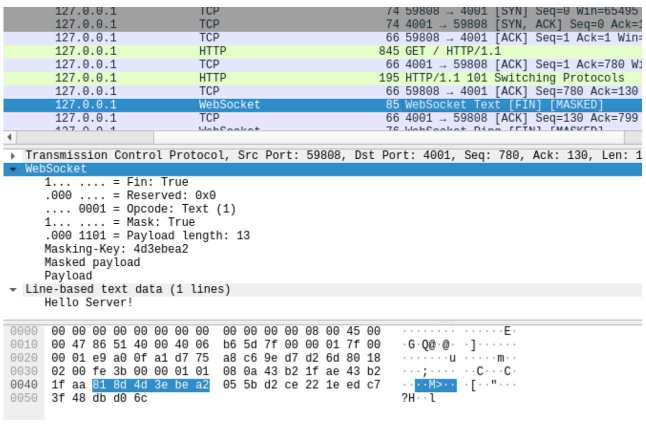
6.1.1 CORS
CORS is a group of policies to regulate the contents of cross-origin requests. An attacker cannot make a request to other sites using JavaScript. If an attacker wants to send some requests to Facebook, they will be rejected by browser's policies.

That is to say, when attackers want the clients to forge a request to websites with credentials. These requests will not be sent. Thus, BEAST attacks will not work any longer.

6.1.2 *WebSocket mask*

WebSocket is not subject to CORS policies, and it can be also wrapped by TLS. It seems that WebSocket will be a good choice to implement BEAST attack.

However, in modern clients, WebSocket payloads are masked by a value shown in 2.



```
127.0.0.1        TCP              74 59808 → 4001 [SYN] Seq=0 Win=65495
127.0.0.1        TCP              74 4001 → 59808 [SYN, ACK] Seq=0 Ack=1
127.0.0.1        TCP              66 59808 → 4001 [ACK] Seq=1 Ack=1 Win=
127.0.0.1        HTTP            845 GET / HTTP/1.1
127.0.0.1        TCP              66 4001 → 59808 [ACK] Seq=1 Ack=780 Wi
127.0.0.1        HTTP            195 HTTP/1.1 101 Switching Protocols
127.0.0.1        TCP              66 59808 → 4001 [ACK] Seq=780 Ack=130
127.0.0.1        WebSocket        85 WebSocket Text [FIN] [MASKED]
127.0.0.1        TCP              66 4001 → 59808 [ACK] Seq=130 Ack=799
127.0.0.1        WebSocket        76 WebSocket Ping [FIN] [MASKED]

▸ Transmission Control Protocol, Src Port: 59808, Dst Port: 4001, Seq: 780, Ack: 130, Len: 1
▾ WebSocket
    1... .... = Fin: True
    .000 .... = Reserved: 0x0
    .... 0001 = Opcode: Text (1)
    1... .... = Mask: True
    .000 1101 = Payload length: 13
    Masking-Key: 4d3ebea2
    Masked payload
    Payload
▾ Line-based text data (1 lines)
    Hello Server!

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 47 86 51 40 00 40 06  b6 5d 7f 00 00 01 7f 00   ·G·Q@·@· ·]······
0020  00 01 e9 a0 0f a1 d7 75  a8 c6 9e d7 d2 6d 80 18   ·······u ·····m··
0030  02 00 fe 3b 00 00 01 01  08 0a 43 b2 1f ae 43 b2   ···;···· ··C···C·
0040  1f aa 81 8d 4d 3e be a2  05 5b d2 ce 22 1e ed c7   ····M>·· ·[··"···
0050  3f 48 db d0 6c                                     ?H··l
```

Figure 2: WebSocket mask

Besides, WebSocket prepends extra bits before real payload. It is still hard to control the block boundary.

The mask makes it hard for attackers to do BEAST attacks, since the attack requires the first sent block mostly controllable by attackers.

6.2 Defense

BEAST attacks make use of a flaw in the specification of TLS 1.0, and the attack only works for block ciphers. That is to say, stream ciphers with TLS 1.0 are not vulnerable to BEAST attacks.

However, TLS 1.0 is still vulnerable to other attacks when using stream ciphers (e.g. RC4). Therefore, a much more direct way is just to abandon TLS 1.0, and update to later TLS versions.

Many modern browsers and clients have also limited users to browse those sites with TLS 1.0 enabled alone. This kind of action will boost organizations to update their websites TLS versions. Today there are only few sites supporting TLS 1.0.

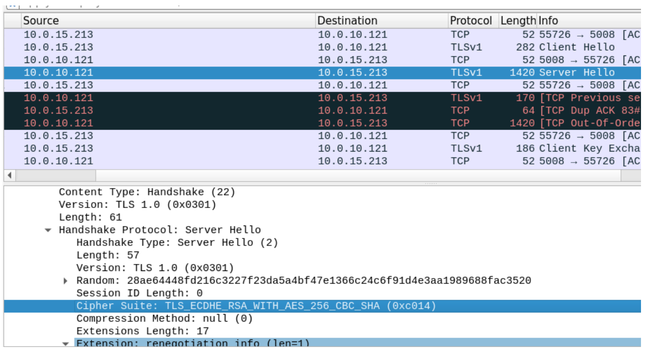
7. DETECTION

We will propose a method to detect BEAST vulnerability of a server, together with a Python script which is easy to use.

At the stage of TLS handshake, a cipher suite will be selected through steps:

1. (Client Hello) Client sent a list of accepted cipher suites.
2. (Server Hello) Server chose a best accepted cipher suite, or a handshake failure occured.

Based on this, a scanner could change the list of cipher suites to enumerate all cipher suites that the server will accept.



```
Source                          Destination        Protocol Length Info
10.0.15.213                     10.0.10.121        TCP        52 55726 → 5008 [AC
10.0.15.213                     10.0.10.121        TLSv1     282 Client Hello
10.0.10.121                     10.0.15.213        TCP        52 5008 → 55726 [AC
10.0.10.121                     10.0.15.213        TLSv1    1420 Server Hello
10.0.15.213                     10.0.10.121        TCP        52 55726 → 5008 [AC
10.0.10.121                     10.0.15.213        TLSv1     170 [TCP Previous se
10.0.15.213                     10.0.10.121        TCP        64 [TCP Dup ACK 83#
10.0.10.121                     10.0.15.213        TCP      1420 [TCP Out-Of-Orde
10.0.15.213                     10.0.10.121        TCP        52 55726 → 5008 [AC
10.0.15.213                     10.0.10.121        TLSv1     186 Client Key Excha
10.0.10.121                     10.0.15.213        TCP        52 5008 → 55726 [AC

      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 61
    ▾ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 57
        Version: TLS 1.0 (0x0301)
      ▸ Random: 28ae64448fd216c3227f23da5a4bf47e1366c24c6f91d4e3aa1989688fac3520
        Session ID Length: 0
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
        Compression Method: null (0)
        Extensions Length: 17
      ▾ Extension: renegotiation_info (len=1)
```

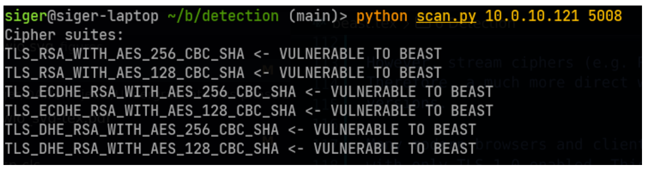
Figure 3: Negotiation on the cipher suite

The server is vulnerable to BEAST attacks if it accepts TLS 1.0 handshake and support cipher suites with CBC modes.

```
python scan.py host port
```

The `openssl` utility is able to start a TLS server with many options.

```
openssl s_server \
    -CAfile ca_cert.pem \
    -cert server_cert.pem \
    -key server_key.pem \
    -HTTP -port 5008 -tls1
```



```
siger@siger-laptop ~/b/detection (main)> python scan.py 10.0.10.121 5008
Cipher suites:
TLS_RSA_WITH_AES_256_CBC_SHA <- VULNERABLE TO BEAST
TLS_RSA_WITH_AES_128_CBC_SHA <- VULNERABLE TO BEAST
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA <- VULNERABLE TO BEAST
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA <- VULNERABLE TO BEAST
TLS_DHE_RSA_WITH_AES_256_CBC_SHA <- VULNERABLE TO BEAST
TLS_DHE_RSA_WITH_AES_128_CBC_SHA <- VULNERABLE TO BEAST
```

Figure 4: Detection output on a TLS 1.0 server

8. REFERENCES

[1] C. Allen and T. Dierks. The TLS Protocol Version 1.0. RFC 2246, Jan. 1999.

APPENDIX