Attacks on TLS [Graded-Optional, 4/9-14/9]

BEAST Attack

⇥ Remote Timing Attacks are Practical

Wild at Heart: OpenSSL's Heartbleed ⇤

Display replies flat, with oldest first

Settings

The cut-off date for posting to this forum is reached so you can no longer post to it.

BEAST Attack
by علیرضا میرزائی - Thursday, 9 Azar 1402, 11:40 AM

Title: Demystifying the BEAST Attack: Decrypting Data Through Chosen Ciphertext Exploitation

1. Exploiting the Vulnerability:

The BEAST attack capitalizes on a vulnerability in the CBC mode, which allows an adversary to decrypt portions of the encrypted traffic.

To perform the BEAST attack, the attacker needs to find a way to manipulate the IV and observe the resulting ciphertext. In older versions of TLS and SSL, the IV was often predictable, which aided in launching this attack.

First step in executing the BEAST attack, is injecting malicious JavaScript into the victim's browser, often done using compromised website or a man-in-the-middle attack.

Once the malicious JavaScript is executed, the attacker initiates a series of chosen plaintexts. By manipulating the plaintext, the attacker controls the content flow of the corresponding ciphertext blocks. The chosen plaintext in this step, decreases the size of the size of brute-force possibilities from 2^ (64) for 8 bytes, to only one byte meaning 2^8 = 256 possible keys.

Using the plaintext-ciphertext pairs, the attacker performs a chosen ciphertext attack, decrypting the targeted data byte-by-byte. The process is brute-force, gradually revealing the desired information, such as session cookies, which can give the attacker the ability to hijack the session and go on from there.

During each iteration, the attacker XORs the target ciphertext block with the manipulated plaintext to retrieve the intermediate value. By observing the resulting plaintext, the attacker gains insight into the encrypted data.

A POC (proof of concept) of beast includes a good animation describing this step in this link.

2. Countermeasures and Mitigation:

a. Implementing TLS 1.1 or Higher: Newer versions of TLS, such as TLS 1.1 and TLS 1.2, address the BEAST vulnerability by using countermeasures like random IVs and encrypting the MAC along with the ciphertext.

Even better, TLS 1.3 which is the latest version and provides enhanced security against such attacks, should be prioritized in every website.

Citations:

Github - BEAST PoC

Wikipedia - BEAST

Permalink

در پاسخ به: BEAST Attack
by رسول کامکار - Friday, 10 Azar 1402, 10:27 PM

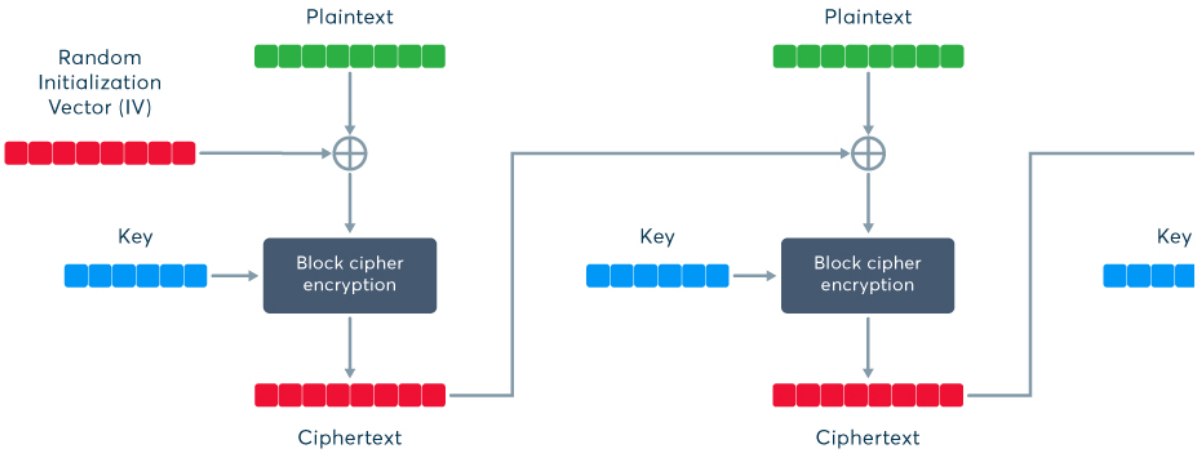
the history of BEAST

Browser Exploit Against SSL/TLS is alluded to as BEAST. It is an organization weakness assault against TLS 1.0 and prior SSL conventions. The assault was first completed in 2011 by security scientists Thai Duong and Juliano Rizzo. However, Phillip Rogaway previously recognized the possible weakness in 2002.

For what reason do we have to examine such an obsolete assault system? Research revealed that 30.7% of examined web servers had powerless TLS 1.0 empowered, making them defenseless against the BEAST assault.

Block Ciphers and Initialization Vectors

In CBC mode, the first block is combined with an initialization vector (IV) – a random block of data that makes each message unique. The security of any block cipher in CBC mode depends entirely on the randomness of initialization vectors. And here's the problem: in TLS 1.0, initialization vectors were not randomly generated. Instead of generating a new IV for each message, the protocol uses the last block of ciphertext from the previous message as the new IV. This opens up a serious vulnerability because anyone who intercepts the encrypted data also gets the initialization vectors. Blocks are combined using XOR, which is a reversible operation, so knowing the initialization vectors could allow an attacker to discover information from encrypted messages.



Recovering Information Without Decrypting It

Let's say we have a man-in-the-middle attacker who is sniffing TLS 1.0 traffic and can inject data into it. If the attacker knows what kind of data is being sent and where it is in the message, they can inject a specially crafted data block and check if the resulting encrypted block is the same as the corresponding block in the actual message stream. If so, the injected guess was correct and the attacker has discovered the plaintext block. If not, they can try again and again with different likely values. This is called a record splitting attack.

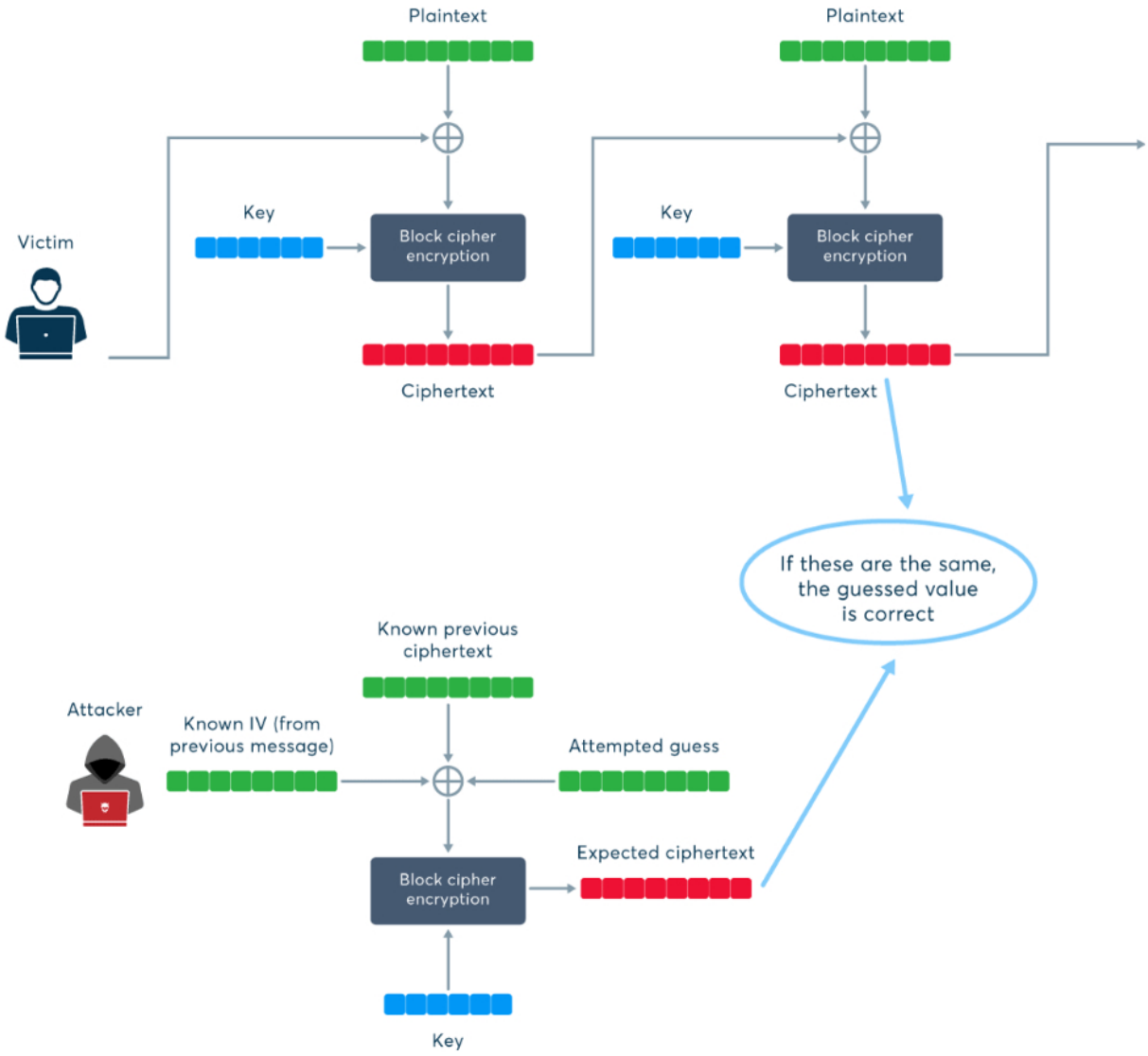
the problem is that, the attacker can only check if the entire block was guessed correctly, which we know could grow exponentially and require an impractically large number of attempts.

The Exploit: Record Splitting with a Chosen Boundary Attack

In 2011, security researchers managed to drastically cut the number of attempts required to guess a value by shifting cipher block boundaries to isolate just one byte of a block. This greatly reduced the complexity of the attack – instead of guessing the whole block, the attacker is brute-forcing one byte at a time, so guessing a 10-digit number would require just 10 guesses for each digit and no more than 100 attempts for the whole number (50 on average).

The approach described by Duong and Rizzo relies on the rigid structure and predictable content of HTTP packets, especially containing equally predictable HTML code. By carefully crafting HTTP requests, it is possible to control the location of cipher block boundaries (hence the name Blockwise Chosen Boundary Attack) and create a message where all bytes are known except the targeted data – typically the session cookie. The block boundaries are then moved to obtain a block where exactly one byte is unknown. Now the actual TLS vulnerability can be exploited to check likely values. After each successful check, the block boundaries

are shifted by one byte and the process repeats until all unknown bytes have been discovered.



references

https://www.wallarm.com/what/what-is-a-beast-attack

https://www.invicti.com/blog/web-security/how-the-beast-attack-works/

Sum of ratings: 35 (1)

Permalink Show parent

⇥ Remote Timing Attacks are Practical

Wild at Heart: OpenSSL's Heartbleed ⇤