CTF Challenge: CRIME Attack Implementation

Ohad Agadi

November 2, 2024

1 Challenge Description

This CTF challenge focuses on implementing the CRIME (Compression Ratio Info-leak Made Easy) attack. The challenge demonstrates how compression leaks information about encrypted data.

2 Setup

- A vulnerable client browser, using TLS 1.2.
- An attacker with the ability to sniff outbound networking of the client, and an XSS abilty (can force the client to send chosen data to chosen destination).
- Encryption type - with SSL all communication is encrypted, for out purposes the algorithm is either RC4 or CBS-AES. That induces two setup modes, and two difficulty levels for the challenge.

3 Challenge Modes

3.1 Normal Mode

In the normal mode, the server uses RC4 stream cipher for encryption. This mode is more straightforward because:

- RC4 doesn't use padding
- The ciphertext length exactly matches the plaintext length

3.2 Hard Mode

The hard mode uses AES in CBC mode, which introduces additional complexity:

- Requires padding, ciphertext length is rounded to the next 16-byte block.
- Challengers must account for padding.

4 Challenge Components

4.1 organizer.py

The organizer script serves as an encryption oracle over HTTP. Key features:

- Listens on URL:443
- Accepts POST requests with plaintext data
- Compresses the data using zlib
- Encrypts using either RC4 or AES-CBC based on mode
- Returns the encrypted data.

4.2 oracle_demonstration.py

This script provides examples of interacting with the encryption oracle:

- Demonstrates proper HTTP request formatting
- Shows how to interpret the server responses
- Includes examples of payload construction

Example interaction:

```
def demonstrate_oracle():
    payload = "secret_flag: CTF_FLAG{b"
    encrypted_length = self.get_response_length(payload)
```

4.3 exploit.py

Template for challengers to implement their solution:

- Provides basic structure for the attack

5 Attack Implementation

5.1 Core Concept

The CRIME attack exploits the fact that when two strings share a common substring, the compressed size will be smaller than if they were completely different. This allows an attacker to guess secret values one byte at a time by observing changes in the encrypted data length.

5.2 Implementation Steps

To successfully complete the challenge, participants should:

1. Implement the main attack loop:
   - Generate candidate guesses
   - Send payloads to the oracle
   - Compare encrypted lengths
   - Select the best candidate based on compression ratio
2. Handle different encryption modes:
   - Handle block alignment in AES-CBC mode

6 References

1. CRIME-poc `https://github.com/mpgn/CRIME-poc/blob/master/CRIME-rc4-poc.py#L65`
2. Implementing the CRIME attack `https://shainer.github.io/crypto/2017/01/02/crime-attack.html`