**hulk writeup**

This challenge is focused on the cryptography behind the BEAST (Browser Exploit Against SSL/TLS) attack presented by Thai Duong and Juliano Rizzo on September 23, 2011. This a [chosen-plaintext attack](https://en.wikipedia.org/wiki/Chosen-plaintext_attack) and this allow you to retrieve sensitives information if the Transport Layer Security used is TLS1.0 or SSLv3. The original proof of concept can be found here : [Here come the Ninjas](http://netifera.com/research/beast/beast_DRAFT_0621.pdf)

**Note**: This is also an implementation of the vulnerability originally discovered by [Phillip Rogaway](https://en.wikipedia.org/wiki/Phillip_Rogaway). Discovered in 2002, there was no exploit released until BEAST in 2011.The CBC IV for each record except the first is the previous records' last ciphertext block. Thus the encryption is not secure against adversaries who can adaptively choose plaintexts.

**Be the BEAST**

| **Encryption** | **Decryption** |
| --- | --- |
| Ci = Ek(Pi ⊕ Ci-1), and C0 = IV | Pi = Dk(Ci) ⊕ Ci-1, and C0 = IV |

The plaintext is divided into block regarding the encryption alogithm (AES,DES, 3DES) and the length is a mulitple of 8 or 16. If the plaintext don't fill the length, a [padding](https://en.wikipedia.org/wiki/Padding_(cryptography)#PKCS7) is added at the end to complete the missing space.

**Cryptology**

When we use the CBC we need a vector initialisation call IV. This IV is random (or fixed) but in any case it should not be predictable from anyone. In this challenge the first IV of the request is random, fine. But to gain some time and not generate a new random IV every time, the last block of the previous cipher text has an IV. In other words, the IV is now guessable. The length of each block will be 8 (DES) and the attacker have a MiTM to retrieve all the cipher.

Example :

C0 | C... | Ci-1 | Ci | Ci+1 |Cn

Now the interesting part, this is the different cryptographic steps of the attack to retrieve one byte :

* first we send a request call C² to get the last block of the cipher meaning the next IV of the second request
* this is a chosen plaintext attack, so the attacker can send this message bbbbbbbflagthrough the victim.

You can notice the seven b before the secret flag. If the length of a block is 8 we need to push 7 know bytes. This information is very important, the attacker know the 7 first bytes of the first block.

But why ? This allow us to have only 256 possibilty to find one byte and not 256^8 to find 8 bytes !

Now the victim send the request and it will be encrypted like this :

C0 | C1 | C2 | C3 | C4

Where C2 = Ek(C1 ⊕ bbbbbbbT)

the attacker want to retrieve the information in the block C3, C4 **he always need the previous block**. Since this is a chosen plaintext attack, the attacker can construct a block P'0 like this :

P'0 = C2  ⊕ C4 ⊕ bbbbbbbX

The only unknow element is X, there is 256 possibilities so he will try max 256 char. The request is sent and encrypted like this :

C'0 = Ek(P'0 ⊕ IV')   
 = Ek(C2  ⊕ C4 ⊕ bbbbbbbX ⊕ IV') or C4 ⊕ IV' = 0   
 = Ek(C2 ⊕ bbbbbbbX)

Now he compares : C'0 and C0, if they are equal, then he just found the byte X in position 8. If it doesn't match, he retries with another char and compare again etc.

Now we have one byte we can get another one by shift the previous request by one on the left : bbbbbbflag. He now have six b and we also now the T, so we have one char unknown. We build a new P'0 = C2  ⊕ C4 ⊕ bbbbbbTX etc....