**<1>** **Describe BEAST attack scenario in detail, which is the chosen-plaintext attack on CBC mode in TLS1.0 (or prior SSL versions), to decrypt a message byte-by-byte**

1) Definition of BEAST attack and its assumption

BEAST attack은 Browser Exploit Against SSL/TLS의 약어로, SSL3.0과 TLS1.0에 대한 chosen-plaintext attack이다. 즉, BEAST attack에서 attacker는 plaintext를 선택할 수 있고, 그를 암호화한 ciphertext를 알 수 있다.

BEAST attack의 대상인 TLS1.0은 message(record)를 Block cipher인 CBC mode로 암호화한다. 각 message는 특정 길이의 plaintext block으로 나뉘어 암호화된다. 한 가지 특징은 첫 message를 제외한 각 message는, 이전 message의 마지막 ciphertext block을 IV로써 사용한다는 것이다. 이 특징이 BEAST attack 모델에서 사용된다. (이 때문에, stream cipher는 BEAST attack에 안전하다)

BEAST attack 모델의 두 가지 가정이 있다.

(가정1) attacker는 victim의 plaintext 앞에 자신이 원하는 문자열을 prepend할 수 있다.

(가정2) attacker는 victim이 server로 전송하는 ciphertext를 eavesdrop할 수 있다. (Sniffing)

앞선 문단에서 언급했듯이 TLS1.0은 이전 message의 마지막 ciphertext block의 값을 다음 message를 암호화하는 IV로 사용하기 때문에, attacker가 victim이 server로 전송하는 ciphertext를 eavesdrop할 수 있다는 것은, victim이 server로 다음 message를 전송할 때 사용할 IV를 attacker가 미리 안다는 것과 동일하다.

2) BEAST attack Scenario

Attack Scenario는 Chosen Boundary Phase, Block-wise Phase의 두 Phase로 나눌 수 있다.

<1st Phase: “Chosen Boundary Phase”>의 과정은 다음과 같다.

1) attacker는 특정 길이의 문자열 P을 생성하고 victim에게 이를 전송한다.

2) victim은 attacker가 전송한 문자열을 암호화하려고 하는 plaintext 앞에 이어 붙인 뒤, CBC mode로 해당 문자열을 암호화하고 그 결과인 C를 서버에 전송한다.

- Figure 1의 C\*[0 … n-1]이 C에 해당한다.

3) attacker는 victim이 server로 전송하는 ciphertext를 가로챈다.

- (가정2)에 따라 attacker는 이 값을 sniffing할 수 있다.

2nd Phase: “Block-wise Phase”의 과정은 다음과 같다.

1) attacker는 plaintext의 특정 block, ciphertext의 특정 block, ciphertext의 마지막 block을 XOR하여 victim에게 전송한다.

- Figure 1의 (r || i), C\*[0], C\*[n-1]이 각각에 해당한다.

- (r || i)의 r은 attacker가 모르는 값 한 바이트가 포함된 plaintext block에서 그 바이트를 제외한 나머지 문자열이고 i는 한 바이트에 저장될 수 있는 값인 0x0 ~ 0xFF에 해당한다.

- C\*[0]는, 좀 더 정확히 하자면, m[0], 즉 attacker가 모르는 값 한 바이트가 포함된 plaintext P\*[k]와 XOR 되는 ciphertext block C\*[k - 1]이다. (따라서, 꼭 C\*의 [0]가 아닐 수도 있음)

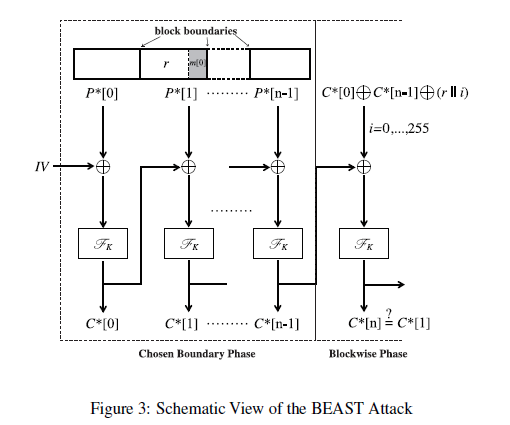
- C\*[n-1]은 victim이 이전에 전송한 ciphertext의 마지막 block 값이다. (즉, 이번 message의 IV)

2) victim은 attacker가 전송한 문자열을 암호화하고 그 결과를 서버에 전송한다.

- Figure 1의 C\*[n]이 이에 해당한다.

3) attacker는 특정한 조건이 만족될 때까지 위 과정을 반복한다.

- 여기서의 특정한 조건이란, 2.)의 C\*[n]이 C\*[1] (정확히는 C\*[k])과 동일한 경우를 말한다.



**<Figure 1: Kurokawa의 Paper에 명시된 BEAST Attack에 관한 도식>**

‘특정한 조건’에 관해 부연 설명을 하기 위해 각각에 대해 다시 정리해보자면,

1) P\*[1]: attacker가 아는 r과 attacker가 모르는 m[0]로 이루어진 plaintext

2) C\*[0]: P\*[1]이 F\_k 연산하기 전 XOR 하는 ciphertext

3) C\*[n-1]: 해당 message의 마지막 ciphertext block (다음 message의 IV)

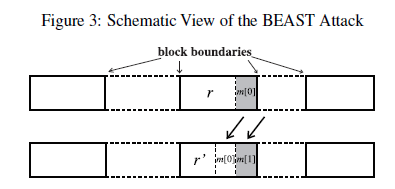
이다. (가정2)에 따라, attacker는 sniffing을 통해 C\*[0] ~ C\*[n] 까지의 값을 모두 알고 있다.

이 때, 2nd phase의 첫 단계에서 수행하는 XOR 연산은 결과적으로 다음과 같다.

1st phase에서 C\*[1] = F\_k(C\*[0] XOR P\*[1]) 이고 이 때 P\*[1] = r || m[0] 이다.

따라서 0x00 ~ 0xFF 사이의 값 중 C\*[n] = C\*[1]을 만족하는 값 i는 m[0]에 해당한다.

특정한 조건을 만족하는 i를 찾음으로써 한 byte(m[0])값을 알 수 있게 된다. 이 반복을 한 cycle이라고 할 때 attacker는 이후 각 cycle 마다 특정 plaintext block의 마지막 바이트에 attacker 가 모르는 값이 오도록 1st phase의 P의 길이를 조정해야 한다. (Figure 2) attacker는 자신이 알고자 하는 byte 수만큼 cycle을 돎으로써 plaintext 값을 복원할 수 있다. (decrypt a message byte-to-byte)



**<Figure 2: Kurokawa의 Paper에 있는 1st phase- 1. 에 관한 도식>**

--------------------------------------------------------------------------------------------------------------

- J. Rizzo, T.Duong, “Here come the XOR ninjas”

- G.V. Bard, “The vulnerability of SSL to chosen plaintext attack”

- G.V. Bard, “A challenging but feasible blockwise-adaptive chosen-plaintext attack on SSL”

- T. Kurokawa, R. Nojima, S. Moriai, “On the Security of CBC Mode in SSL3.0 and TLS1.0”

**<2> Find the countermeasure fixed in TLS1.1, and demonstrate why this is the case (how it can solve the vulnerability)**

1) Countermeasure fixed in TLS 1.1

TLS 1.1에서는 TLS 1.0과 달리, implicit IV 대신 explicit IV를 사용한다. (Figure 3)

Rfc4346에서는 세 notation을 정의하고 있으며 그 내용은 다음과 같다.

1. IV: GenericBlockCipher 구조 안의 IV field에 위치한 (전송된) 값

2. CBC residue: 이전 record(message)의 마지막 ciphertext block

3. mask: 첫 번째 plaintext block을 encrypt 하기 전에 plaintext에 XOR 하는 값

TLS 1.1 이전 버전에서는 IV를 따로 정의하지 않고 있으며, CBC residue와 mask는 동일한 값을 가진다. TLS 1.1에서는 record(message)별 IV를 생성하기 위해 아래의 두 알고리즘 중 하나를 사용해야한다.

(1) block 길이의, cryptographically 강력한 랜덤 문자열 R을 생성한다.

R을 IV 필드에 저장하고 R을 mask의 값으로 사용한다.

🡪 첫 번째 cipher block 은 E\_k(R XOR Data)의 값을 가진다 (where Data is first plaintext block)

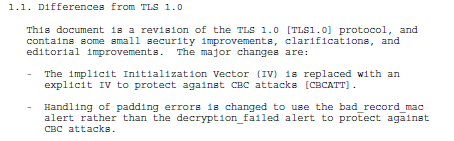
(2) block 길이의, cryptographically 강력한 랜덤 문자열 R을 생성한다.

암호화(E\_k) 전에 plaintext 앞에 R을 이어붙인다(prepend).

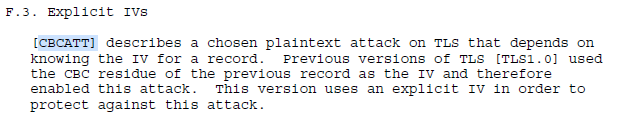
고정된 mask를 이용하거나 CBC residue를 mask로 사용한다.

🡪 첫 번째 cipher block은 E\_k(mask XOR R)의 값을 가지고 이 값을 IV 필드에 저장한다.

🡪 첫 번째 cipher block of “data”는 E\_k(IV XOR data)의 값을 가진다.



**<Figure 3 – rfc4346에 명시된 Differences from TLS1.0>**

****

**<Figure 4 – rfc4346에 명시된 Explicit IVs>**

2) How it works

TLS 1.0에 대한 BEAST attack의 원인은 이전 record(message)의 마지막 ciphertext block을 다음 record의 IV로 사용하기 때문에 attacker가 IV를 알 수 있기 때문이었다. TLS 1.1에서는 이 vulnerability를 해결하기 위해 cryptographically strong한 문자열 R을 추가적으로 사용하였다.

앞선 알고리즘 중 첫 번째 알고리즘은 각 record 마다 R 자체를 IV로 쓴다. 두 번째 알고리즘은 R을 plaintext 앞에 이어 붙인 뒤 고정된 mask를 사용하거나 CBC residue를 사용하는데 이 경우는 ciphertext의 첫 번째 block을 IV로써 사용하는 것과 동일하다. 따라서, 두 알고리즘에 따라 암호화하는 경우, CBC residue를 sniffing 했다고 하더라도 IV 값을 특정 지을 수 없게 되어 BEAST attack 이 불가능하게 된다.

--------------------------------------------------------------------------------------------------------------

- RFC 4346 (TLS 1.1): <https://www.ietf.org/rfc/rfc4346.txt>

================================================================