Information Security

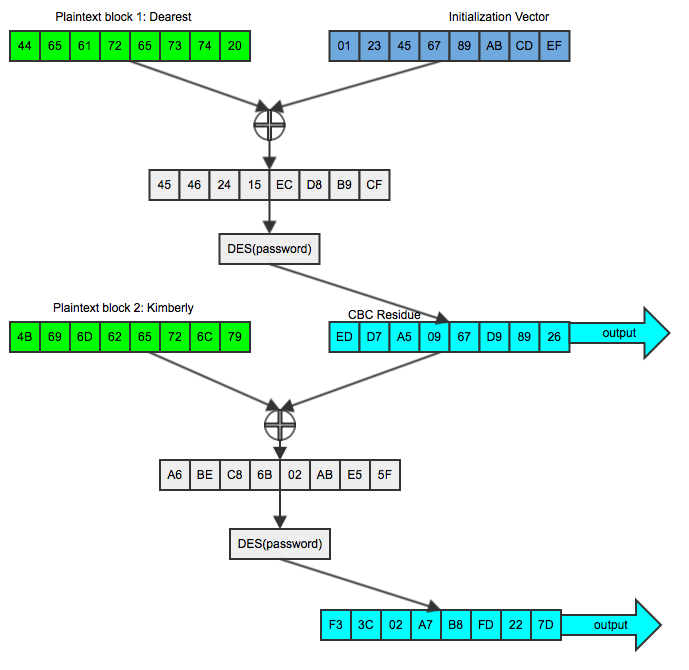
ELED 2013190250

Donghyun Lee

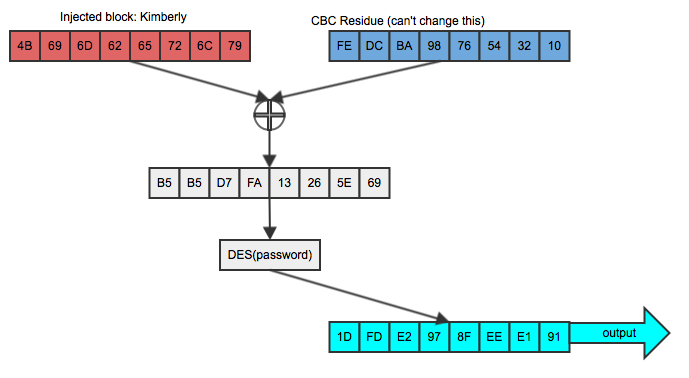
1. **The detailed process of BEAST Attack**

BEAST Attack은 TLS 1.0에서 초기화 벡터에 기인한 취약점을 바탕으로 한 공격이다. 원래 블록 암호는 키가 달라지면 동일한 평문이라도 암호화된 암호문이 달라진다는 특징이 있지만, 반대로 키가 같다면 무수히 많은 시행을 반복하더라도 평문에 대응하는 암호문이 변하지 않는다는 특징이 있다. 이와 같은 블록 암호의 취약점을 해결하기 위해 CBC라는 방어법이 나왔는데 이는 이전 블록과 XOR 연산을 하는 것이다. 이 때 초기 블록에는 XOR 연산을 수행할 이전 블록이 없기 때문에 초기화 벡터가 필요하다. 이 초기화 벡터가 고정됨으로 인해 발생할 수 있는 BEAST Attack에 대해 알아보자.

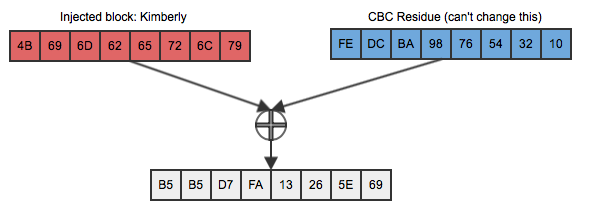
우선 그 과정을 알기 위해 송신자와 수신자, 그리고 공격를 가정하자. 이 때 송신자는 Dearest Kimberly라는 단어를 쓴다는 것을 공격자는 알고있다. 이 때 공격자는 암호와 키를 예측할 수 없다고 가정한다. 무작위로 123456789ABCDEF의 초기화 벡터를 생산한다.



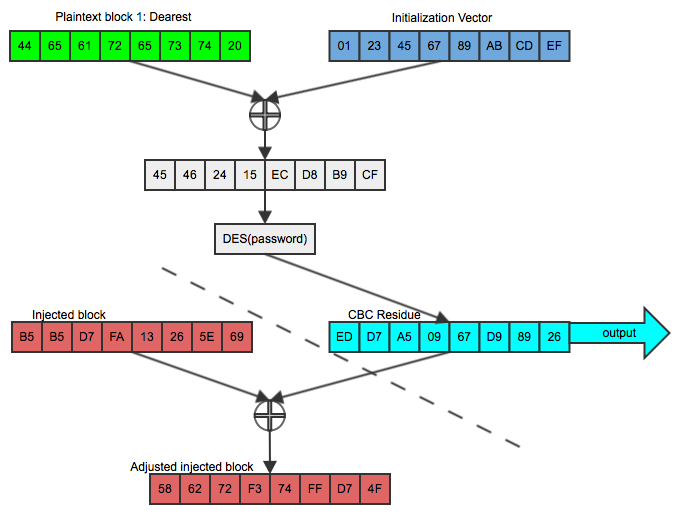
이때 Dearest 는 IV와 함께 XOR된다. 이때 공격자가 송신자의 트래픽을 본다면 이는 ed d7 a5 09 67 d9 89 26 f3 3c 02 a7 b8 fd 22 7d이다. 이제 공격자는 송신자를 속여 Kimberly 라는 단어를 보내 f3 3c 02 a7 b8 fd 22 7d와 비교하도록 패킷 인젝션을 시도한다.



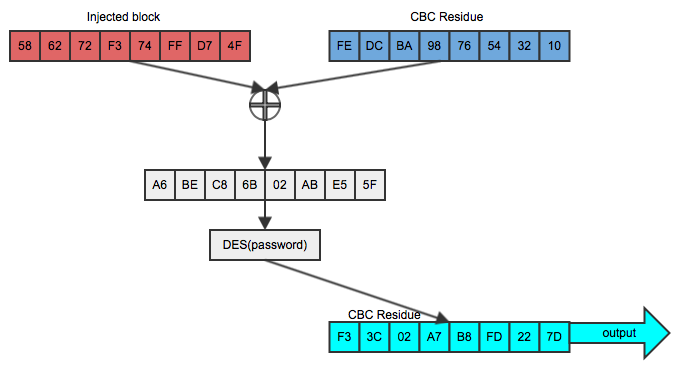
하나 추가로 가정하여 이전 블록의 값이 fe dc ba 98 76 54 32 10이라 가정하자. 이 때 이후 Kimberly라는 단어와 함께 XOR 연산을 할 것을 예측하고 미리 Kimberly라는 단어 즉, 4B 69 6D 62 65 72 6C 79와 XOR 연산을 한다.



그 후 첫번째 블록의 출력 값인 두번째 블록과 다시 XOR한다.



그 후 송신자를 속여 동일한 CBC 블록으로 암호화를 할 수 있다면 이에 따른 결과값은 원래의 두번 째 블록과 동일해진다.



공격자는 암호화 알고리즘의 키를 알거나 알고리즘을 깨뜨리는 것으로 결과를 동일하게 만든 것이 아니라 CBC Residue가 알려진 패킷 주입 공격을 통해서 원하는 결과를 만들어 내었다.

이러한 BEAST Attack은 다음과 같은 방법으로 구체화된다.

HTTP 중 알기 쉬운 텍스트를 포함하는 다음 Request를 보자.

|  |
| --- |
| GET /index.html HTTP/1.1  Host : mysite.com  Cookie: Session = 12345678  Accept-Encoding: text/html  Accept-Charset:utf-8 |

이들 중 분명한 부분은 굵게, 추측의 과정을 거쳐야하는 부분은 기울임을 통해 나타내겠다.

|  |
| --- |
| **GET /index.html HTTP/1.1**  **Host :** *mysite.com*  **Cookie: Session =** *12345678*  *Accept-Encoding: text/html*  *Accept-Charset:utf-8* |

이들 중 유일한 변수는 페이지와 세션 쿠키 값이다. 이때 CBC 모드에서 78바이트가 다음과 같다고 가정해보자.

|  |
| --- |
| GET/index.html HTTP/1.1\r\nHost:www.somesite.com\r\nCookie: Session=12345678 |

password키와 0x0123456789abcdef IV 값으로 DES CBC 모드 암호화한 결과는 다음과 같다.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GET /ind | ex.html | HTTP/1.1 | \r\nHost | : mysite | .com\r\n | Cookie: | Session= | 12345678 |
| a7d25abbd67b2dbf | e2ade7246ea5ed5 | e99063ebe430b75b | 746ae5eca36e2bc3 | f6f62f99a076056f | 6b14704973a779ae | 7fa9300d4e490cba | b6040b9542a59ad5 | f9bb888ac3763722b |

이때 공격자는 마지막블록인 f9bb888ac3763722b에 관심을 가질 것이다. 이에 대해 전수조사를 할 수도 있지만 이는 무수한 연산을 필요로한다. 하지만 패킷 주입이 가능하다면 다음과 같이 요청을 수정할 수도 있다.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GET /ind | ex.jsp H | TTP/1.1\r | \nHost | : mysite. | com\r\nC | ookie: S | ession=1 | 2345678 |

이때 공격자는 ‘ession =1’ 이후의 값만 추정하면되기 때문에 값을 추정하기 위한 연산이 줄어들게 된다. 따라서 패킷 주입 과정이 반복된다면 결과적으로 암호화된 세션 값이 노출될 수 있게된다.

**2. The countermeasure fixed in TLS 1.1**

TLS1.1에서는 TLS 1.0에서와 다르게 초기화 벡터를 패킷마다 다르고 유일하게 만들었다. 이는 하나의 평문에 하나의 암호문이 대응되는 문제점을 해소시켰다. 즉, 단순 패킷 주입을 통해 동일한 결과값이 지속적으로 나올 수 없게 된 것이다. 이는 모든 평문들을 예측 불가능하게 암호화할 수 있다는 점에서 BEAST Attack이 문제화한 취약점에 대한 해결책이 될 수 있다.