TLS 취약점 분석과 해결 방법

지도교수 : 이 강 호
연 구 자 : 이 영 화

< 목 차 >

요 약

웹 사용 초창기부터 통신의 도청이나 유출을 막기 위해 1995년 미국의 넷스케이프 커뮤니케이션스사가 암호화와 통신 보안을 제공하여 인터넷 상거래를 가능하게 하는SSL(Secure Sockets Layer) 암호화 프로토콜을 개발되었고 점점 더 정교하면서 지능화되어 가는 사이버 공격 양상에 맞춰 지속적으로 업데이트되어 1999년에 국제 인터넷 표준화 기구(IETF)에 의해 차세대 버전인 TLSv1.0 (Transport Layer Security v1.0)으로 표준화되어 안정성이 보장되었다고는 하나 어느 보안이라도 취약점이 있기 마련이다. 따라서 TLS의 취약점에 알려진 사례를 제시하고 그에 대한 해결책을 검토해 보았다.

주요어 : TLS(Transport Layer Security), 취약점, 암호화 프로토콜

1. TLS(Transport Layer Security)

1.1 TLS란

웹과 인터넷 통신이 점차 발전해 가면서 통신 도중에 정보를 가로채어 유출되는 사건이 많이 발생하기 시작하자 통신 간의 기밀성을 보장하기 위해 미국의 넷스케이프 커뮤니케이션스사가 1995년에 SSL(Secuer Sockets Layer) 프로토콜을 개발하였고 SSL이 시간이 지날수록 정교해져 가는 사이버 공격과 여러 취약점에 노출되어 대부분의 취약점을 개선하여 1999년에 국제 인터넷 표준화 기구(IETF)에 의해 TLS v1.0으로 표준화 되었다.

TLS(Transport Layer Security)는 공개키 암호화 방식과 대칭키 암호화 방식을 혼용하여 통신을 하는 방식으로 공개키 암호화는 비대칭 암호화라고도 불리며 자신만 가지고 있으며 복호화에 사용되는 개인키와 암호화에 사용되며 타인에게 공개되는 공개키로 나눠지며 공개키가 유출되더라도 개인키를 모르면 복호화가 불가능하다.

대칭키 암호화 방식은 암호화와 복호화가 같은 키를 사용하며 키는 매번 랜덤으로 생성이 된다. 공개키 암호화 방식은 컴퓨터의 리소스를 많이 필요로 하여 통신상의 메시지를 전부 암호화 할 경우 버텨내지 못하여 두 암호화 방식을 혼용하는 것을 채택 하였는데 대칭키 암호화 방식의 키를 공개키 방식으로 암호화 하여 전달 한 후에 공유된 세션 키로 통신을 하게 된다.

암호화에 사용되는 세션 키를 합의하는 과정이 있는데 이를 핸드셰이크라고 부른다. 핸드셰이크는 TLS의 과정 중 가장 복잡한 과정으로 크게 4단계로 구분이 된다.

(1) 초기 협상 단계로 클라이언트가 서버에게 보안 연결 요청 메시지를 전송 후 클라이언트는 사용 가능한 암호 모음, 알고리즘 리스트, 버전 등을 서버에 전달하여 서버는 전달받은 모음 중 하나를 선택하여 클라이언트에 전송한다.
(2) 서버 인증단계로 서버가 인증서를 클라이언트에 제공하며 클라이언트는 인증서를 받아 진위 여부를 확인한다.
(3) 클라이언트 인증단계로 인증서 확인정보와 공개키로 암호화된 세션 키 그리고 클라이언트 인증서를 전달한다.
(4) 종료 단계로 다음부터 협상된 암호를 이용할 것을 알리며 핸드셰이크는 종료되고 데이터를 주고받게 된다.

핸드셰이크 과정 중 설명된 인증서는 TLS 프로토콜의 핵심으로 안전한 연결을 시작하는데 필요한 공개 암호화키를 클라이언트에 제공한다. 인증서의 키가 제공됨으로 인해 서버는 클라이언트에게 키가 제공되는 인증기관에 관련되어있는 신뢰성 있는 서버임을 증명할 수 있다. 인증서를 발급하는 기관은 CA(Certificate Authorities)라고 불리며 씬 암호화를 하기 위해서는 해당 기관으로부터 인증서를 구매해야 한다. 기관 중에서도 신뢰할 수 있는 CA라는 것이 있으며 해당 기관 들은 소프트웨어 제조업체들이식별을 하여 신뢰 가능한 CA 목록을 제공하고 있다. 인증서를 통해 인증된 사이트와 통신을 할 경우, [그림 1]처럼 URL 맨 앞에 자물쇠 모양이 생기며 HTTP가 아닌 HTTPS 연결이 된다.

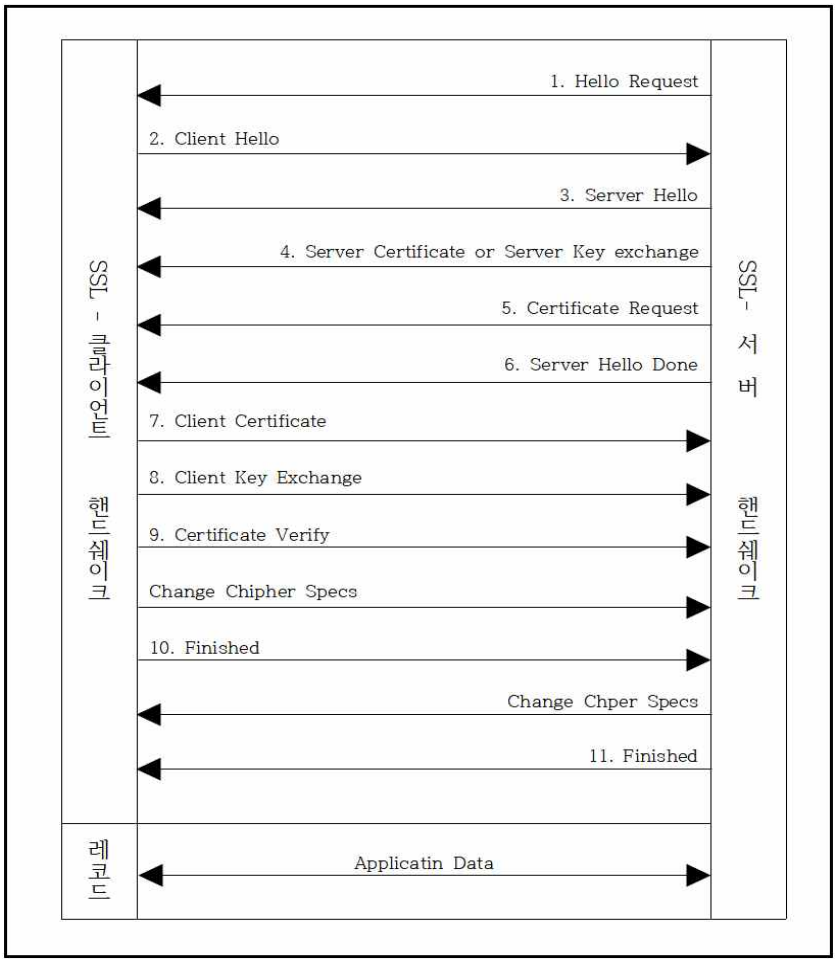
[사진 1] TLS를 통한 HTTPS 통신

1.2 TLS의 구조

핸드셰이크 프로토콜	암호명세 변경 프로토콜	경고 프로토콜	HTTP	하트비트 프로토콜
레코드 프로토콜				

[사진 2] TLS 구조

TLS 프로토콜은 TCP/IP 계층의 응용계층에 추가되어 보안기능을 제공하며 구성 프로토콜은 핸드셰이크 프로토콜, 암호명세 변경 프로토콜, 경고 프로토콜, 하트비트 프로토콜, 레코드 프로토콜, HTTP로 구성 된다.



[사진 3] 핸드셰이크 프로토콜 성립 과정

첫 번째로 소개할 프토토콜은 위에서도 간단하게 다룬 핸드셰이크 프로토콜로 TLS의 프로토콜 중 가장 복잡한 부분이며 서버와 클라이언트가 서로를 인증하고 암호화 MAC 알고리즘, 키 교환 알고리즘 그리고 TLS 레코드 안에 보낸 데이터를 보호하는 데 사용할 암호 키를 협상하는 역할을 한다. 핸드셰이크 프로토콜은 모든 응용 데이터를 전송하기 전에 사용하는데 각 메시지는 유형, 길이, 내용이라는 3개의 필드로 구성된다.

핸드셰이크 프로토콜은 4개의 단계를 통해서 협상을 한다.

(1) 초기 협상단계로 클라이언트가 서버에게 처음 연결하는 것으로 시작하며 Client Hello 메시지를 통해 세션 식별자, 각종 알고리즘을 포함한 CipherSuite 리스트, 압축 알고리즘 리스트 등을 보내고 서버가 이에 응할 경우 클라이언트가 메시지에 담아 보낸 각 리스트들 중 하나를 선택해 Server Hello 메시지에 담아 전송한다.

(2) 서버 인증단계로 서버가 ServerHello 뒤에 바로 Ciper suite의 알고리즘 타입에 맞는 인증서가 포함된 Certificate 메시지를 보내어 클라이언트가 대한 인증서에 대한 진위를 확인하며 그 후 부가적으로 클라이언트에게 인증서를 요청하는 등에 대한 메시지를 보내거나 ServerHelloDone 메시지를 통해 다 보냈음을 통보한다.

(3) 클라이언트 인증 단계로 서버가 클라이언트에게 인증을 요구 했을 경우 ClientCertificate 메시지를 보내며 그 후 클라이언트는 Client Key Exchange 메시지를 통해 세션 키를 생성하기 위한 임의의 비밀정보인 사전 마스터 비밀을 생성하여 공개키 알고리즘을 통해 암호화하여 서버와 공유한다.

(4) 암호명세변경 프로토콜인 ChangeChipherSpecs 메시지를 보내며 이후 전송되는 메시지는 협상된 알고리즘과 키를 이용할 것임을 통보한다. 그 후 Finished 메시지가 전송되며 이 메시지부터 협상된 알고리즘과 키가 적용된다. 서버도 이와 같이 두 개의 메시지를 전송하며 핸드셰이크 프로토콜을 마치게 되고 데이터 전송이 시작된다.

두 번째로 소개할 프로토콜은 암호명세변경(Change Chiper Spec) 프로토콜로 1값을 갖는 1바이트로 구성되며 앞서 소개 했듯이 핸드셰이크 프로토콜 과정 중 하나에 속하며 핸드셰이크 과정에서 서버와 클라이언트 간에 결정된 암호화 알고리즘이 다음 데이터부터 적용 또는 변경된다는 걸 클라이언트에 알리는 역할을 한다.

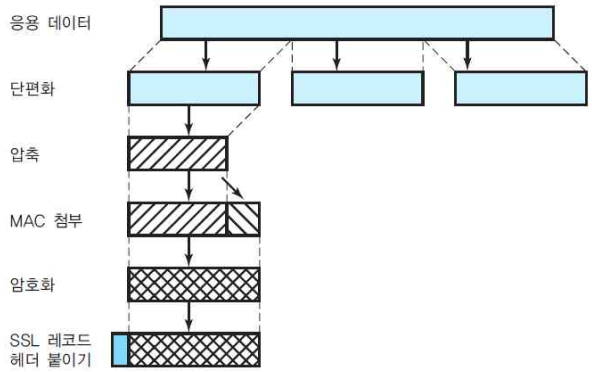
세 번째로 소개할 프로토콜은 경고(Alert) 프로토콜로 TLS 통신 과정에서 발생하는 오류를 통보하기 위해 경고 할 때 사용하며 경고메시지와 경고에 대한 상세한 정보를 전달한다. 치명적인 레벨의 경고 메시지인 경우 연결을 즉시 단절한다. 경고 프로토콜은 각 메시지가 2바이트로 구성되는데 첫 번째 바이트는 경고 혹은 심각이라는 두 가지 값을 가지며 두 번째 바이트에는 특정 경고를 나타내는 코드가 들어있다. 경고 프로토콜의 룰은 [표 1]이다.

Value	Description	Meaning
0	CloseNotify	Sender will not send any mor messages.
10	UnexpecteMessage	An inappropriate.
20	BadRecordMAC	An incorrect MAC received.
21	DecryptionFailed	Decrypted message is invalid.
22	RecordOverflow	Message size is more than 2^{14}+ 2014.
30	DecompressionFailure	Unable to deccompress appropriately.
40	HandshakeFailure	Sender unable to finalize the handshake.
42	BadCertificate	Received certificate corrupted.
43	UnsupportedCertificate	Type of received certificate is not supported
44	CertificateRevoked	Singer has revoked the certificate.
45	CertificateExpired	Certificate has expired
46	CertificateUnknown	Certificate unknown.
47	IllegalParameter	A field out of range or inconsistent with others.
48	UnknownCA	CA could not be identified.

[표 1] 경고 프로토콜의 경고

네 번째로 소개할 프로토콜은 하트비트 프로토콜로 현재 연결이 정상적으로 동작하는 걸

나타내기 위한 주기적 신호를 말하며 프로토콜 개체의 가용성을 모니터링 할 때 사용되며 TLS 프로토콜에서 매번 연결을 재협상 하지 않아도 신호를 통해 상호간에 통상 연결을 유지하게 해준다.



[사진 4] 레코드 프로토콜

마지막으로 소개할 프로토콜은 실질적인 보안을 제공하고 있는 [그림 4]의 레코드 프로토콜로 핸드셰이킹 프로토콜에서 협상된 알고리즘들을 통해 5단계를 거친다.

(1) 2^{14}바이트보다 작거나 같은 크기 블록으로 어플리케이션 데이터를 단편화한다.
(2) 각 단편화된 데이터들을 압축한다.
(3) 압축된 데이터에 MAC(Message Authentication Code)을 생성하여 적용한다.
(4) MAC이 추가된 블록을 암호화 한다.
(5) 암호화된 데이터 블록에 콘텐츠 유형, 주버전, 압축된 길이 등의 정보가 담긴 레코드 프로토콜 헤더를 추가한다.

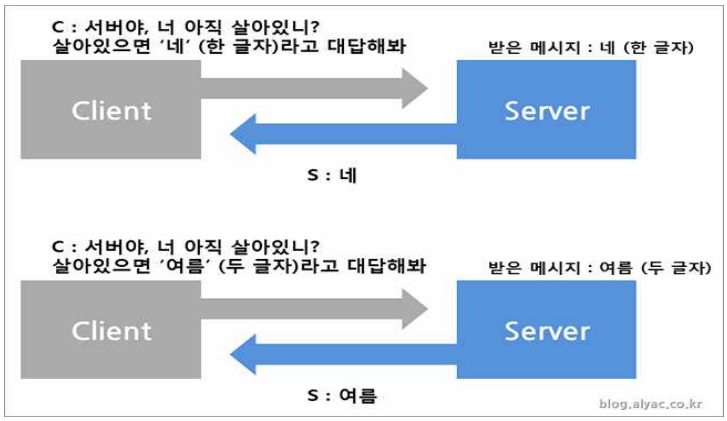
위의 단계를 거친 데이터 블록은 TCP/IP계층인 전송계층, 인터넷계층, 네트워크 액세스 계층을 거쳐 포트번호, IP, MAC주소 정보가 담긴 헤더가 각 계층마다 붙여지게 되고 수신자에게 전송된다. 수신된 데이터는 각 계층별로 헤더의 정보를 확인하는 절차를 걸치고 수신자 맞을 경우 헤더를 전부 제거한 후에 위의 단계를 역으로 진행을 하게 되며 블록의 복호화를 거친 후 압축을 해제하며 단편화된 각 데이터 들을 재조립하여 사용자에게 전달된다. 레코드 프로토콜은 TLS 연결을 위해 2가지 서비스를 제공하는데 기밀성과 메시지 무결성이다. 기밀성은 TLS 페이로드를 암호화 하는데 사용할 비밀 키를 정의하며 메시지 무결성은 MAC을 생성하는데 사용할 공유 비밀 키를 정의한다.

2. TLS 취약점

취약점으로 대표적인 것은 하트 블리드, 프리크, 로그잼, 푸들 등이 있으며 대다수의 취약점이 TLS버전 등을 협상하는 과정 중 중간자 공격으로부터 시작하였으며 예외적으로 하트 블리드가 확장 모듈의 취약점으로 인해서 발생하였다.

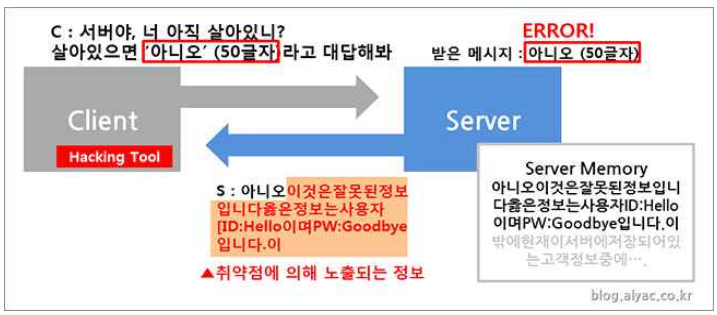
2.1 하트 블리드(HeartBleed)

하트 블리드 취약점은 2014년에 발견 되었으며 암호 통신 라이브러리 OpenSSL의 확장 모듈 하트 비트의 취약점으로 하트 비트란 TLS에서 매번 연결을 재협상하지 않아도 상호 간 연결 지속 신호를 주고받아 통신 연결을 유지해주는 기능으로 [그림 5]와 같이 클라이언트가 하트 비트를 요청하여 데이터와 길이를 보내면 서버도 응답으로 데이터의 길이만큼 복사하여 보내준다.



[사진 5] 정상적인 하트비트

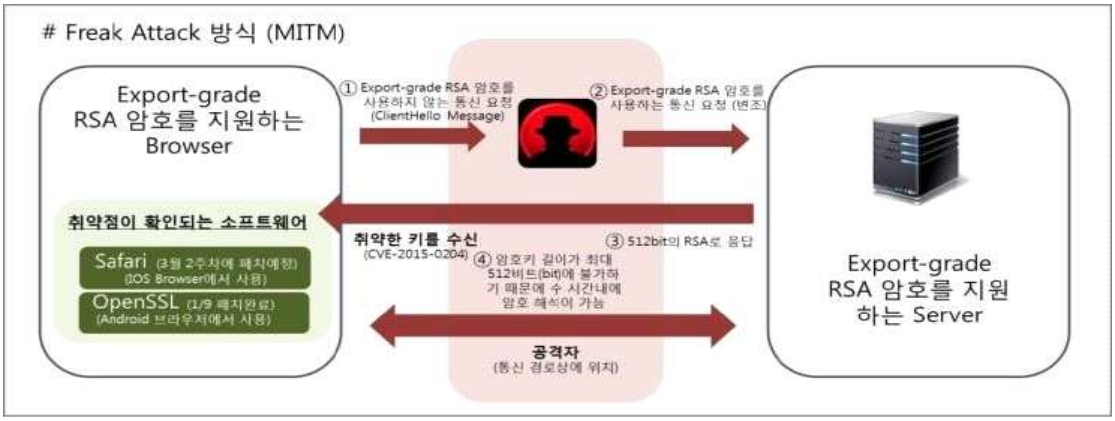
이 때, 클라이언트로부터 전달받은 정보와 길이가 일치하지 않는다면 클라이언트의 요청에 응답을 하지 않는 것이 정상인데 하트 블리드 취약점은 서버가 클라이언트로부터 전달받은 정보의 내용과 길이의 일치 여부를 검증하지 않고 다시 보낸다는 문제가 발생한다. 공격자가 서버에게 내용과 길이가 불일치하는 요청을 전송했을 경우 [그림 6]과 같이 서버는 이에 대한 검증을 하지 않고 요청한 것과는 거리가 먼 중요 데이터를 보내버리는 취약점이 발생한다.



[사진 6] 하트 블리드 취약점

2.2 프리크(FREAK)

프리크 취약점은 2015년에 발견된 취약점으로 과거 미국이 암호화 시스템의 해외 수출을 제한하여 해외로 수출하는 암호시스템을 의도적으로 약하게 만들었으며 수출용 RSA 암호의 경우 키의 길이를 최대 512 비트로 제한하였고 이후 수출제한을 없앴지만, 여전히 잔재가 여전히 남아있어 발생한 문제다.



[사진 7] 프리크 취약점 공격

[사진 7]과 같이 공격자가 클라이언트와 서버 사이에 위치하여 중간자 공격을 실행하며 핸드셰이크 프로토콜 성립과정 중 ClinetHello 메시지를 변조하여 수출용 RSA를 요구하는 것으로 보내고 서버가 512비트의 RSA키로 응답하게 되며 클라이언트는 해당키를 그대로 받아들인다. 키의 길이가 512비트에 불과하여 수 시간 내에 암호 해석이 가능하며 서버와 클라이언트가 주고받는 정보를 유출 시키는 취약점이라 할 수 있다.

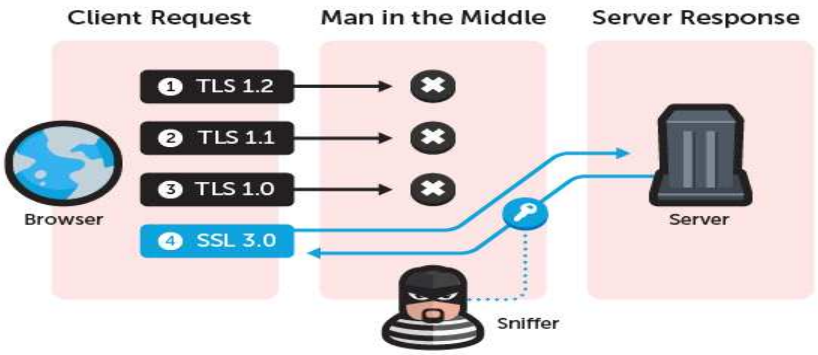
2.3 로그잼(LogJam)

로그잼 취약점은 2015년에 발견되었으며 디피-헬만 키 교환 알고리즘을 사용하는 TLS연결의 취약점으로 중간자 공격을 통해 취약한 암호화 기법으로 사용자와 웹, 또는 서버 간의 암호화 통신을 512비트 수출등급 암호화 연결로 다운그레이드 시킨다.

프리크와 유사하나 프리크는 실행과 관련된 취약점이고 로그잼은 TLS 프로토콜 자체의 기본 설계상의 취약점이라는 점에서 차이를 보인다. 따라서 수입용 키를 지원하는 TLS를 사용하는 웹브라우저와 메일서버들은 모두 이 취약점에 노출될 수 있다. 로그잼 취약점에 공격 받은 웹브라우저는 수입용 키가 아닌 일반 키를 사용한다고 믿게 만들고 공격자는 해당 연결을 통과하는 모든 데이터를 읽고 수정이 가능한 위험한 취약점이다.

2.4 푸들(POODLE)

푸들 취약점은 TLS의 이전버전인 SSL v3.0의 취약점으로 [그림 8]과 같이 TLS v1.2 이하를 지원하는 웹브라우저에서 발생한다.

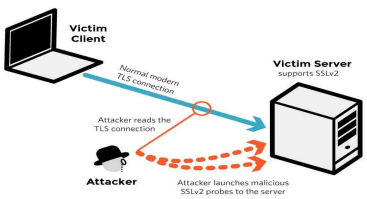


[사진 8] POODLE 취약점 공격

공격자가 중간자 공격을 통해 중간에서 코드를 삽입하여 제어하며 TLS에 해당하는 버전들의 요청을 전부 거부시켜 버전을 낮추도록 유도시키고 푸들 취약점이 존재하는 SSL v3.0버전으로 연결을 요청하도록 유도한다. SSL v3.0은 CBC 모드를 사용하여 암호화된 텍스트를 복호화하는 경우 패딩 바이트를 처리하는데 결함이 있어 이 결함으로 인해 공격자가 바이트 당 256개의 요청만을 사용하여 복호화한다. 푸들 취약점은 한번 바로 잡았다가 다시 발생한 취약점인데 두 번째로 발생하였을 때는 다운그레이드 필요 없이 TLS v1.2의 전송계층의 암호 매커니즘을 우회하여 사용자의 정보를 탈취하는 방식의 취약점이다.

2.5 드라운(DROWN)

드라운 취약점은 2016년도에 발견되었으며 취약한 구식 암호화 기법을 통한 RSA 복호화라는 뜻으로 현대에도 SSL v2.0 연결을 허용하는 서버를 대상으로 발생하였다. 위에서 설명했던 취약점들과 같게 해당 취약점도 [그림 9]처럼 중간자 공격을 통해 발생하며 중간자 공격을 통해 SSL v2.0 연결을 만들어내고 RSA의 키교환 과정인 PMS를 스니핑하여 서버에게 유사한 연관값들을 매우 많이 보내어 PMS에 대해 복호화를 시도하고 성공할 경우 키값을 획득할 수 있고 이 키를 통해 SSL을 복호화하여 정보를 빼내는 취약점이다.



[사진 9] DROWN 공격

3. 결론

위에 나온 취약점들의 공통적인 해결방안은 TLS의 버전을 업그레이드하는 것과 취약점이 있는 버전을 사용하지 않는 것에 있다. 개별적인 방안은 하트 블리드의 경우 [그림 10]의 snort 탐지 룰을 참고하여 침입탐지시스템 및 침입차단시스템에 패턴 업데이트를 하는 방법이 있다.

```
[OpenSSL HeartBeat 취약점 탐지 Snort Rule]
- SSL 서비스 포트에 대해 공격 요청시 전송되는 |18 03 ??| 탐지 패턴
alert tcp any any < > any
[443,465,563,636,695,898,989,990,992,993,994,995,2083,2087,2096,2484,8443,8883,9091]
(content:"|18 03 00|"; depth: 3; content:"|01|"; distance: 2; within: 1;
content:!"|00|"; within: 1; msg: "SSLv3 Malicious Heartbleed Request V2";
sid: 1;)
alert tcp any any < > any
[443,465,563,636,695,898,989,990,992,993,994,995,2083,2087,2096,2484,8443,8883,9091]
(content:"|18 03 01|"; depth: 3; content:"|01|"; distance: 2; within: 1;
content:!"|00|"; within: 1; msg: "TLSv1 Malicious Heartbleed Request V2";
sid: 2;)
alert tcp any any < > any
[443,465,563,636,695,898,989,990,992,993,994,995,2083,2087,2096,2484,8443,8883,9091]
(content:"|18 03 02|"; depth: 3; content:"|01|"; distance: 2; within: 1;
content:!"|00|"; within: 1; msg: "TLSv1.1 Malicious Heartbleed Request V2";
sid: 3;)
                                                          blog.alyac.co.kr
```

[사진 10] 하트 블리드관련 스노트 탐지 룰

프리크 취약점의 경우 업그레이드가 불가능 할 경우 “RSA_EXPORT” 암호화 알고리즘을 비활성화 한다. 클라이언트의 경우 취약점에 영향을 받는 OS 및 브라우저를 업그레이드하거나 취약점이 존재하지 않는 브라우저를 사용한다.

로그잼 취약점의 대응방안으로는 수출용 등급의 암호화 알고리즘을 비활성화하고 최신 브라우저를 사용하거나 서버 측에서 해킹이 불가능한 2048비트의 디피-헬만 그룹을 생성한다.

푸들 취약점은 공격의 성공률이 매우 높으며 대응방안으로는 웹브라우저나 서버에 SSL v3.0을 사용하지 않도록 설정을 하는 것이 이 취약점의 효과적인 대응방안이다. 두 번째로 발생한 취약점은 브라우저를 최신으로 업데이트를 하고 사용자들의 각별한 주의를 요한다.

개발사	브라우저	중단시행일
Microsoft	Microsoft Edge	2020년 상반기 예정
	Internet Explorer 11	
Google	Chrome	2020년 1월 예정
Mozila	Firefox	2020년 3월 예정
Apple	Safari	2020년 3월 예정

[표 2] TLS v1.1이하 버전 지원중단 시행일

위처럼 다양한 취약점들이 존재하였으나, 2020년 현재 [표 2]에 나왔듯이 TLS v1.1이하 버전이 주요 웹브라우저들에서 지원을 중단할 예정이라 몇몇 취약점들이 막히게 되었지만 TLS v1.2는 중간자 공격에 날이 갈수록 취약해지고 여러 취약점에 노출되어 있으며 TLS v1.3는 레거시 암호화 시스템에 대한 불필요한 지원을 모두 제거함으로써 상당 부분의 취약점이 해소된다. 만약 TLS v1.3이 지원 하지 않는 암호화 시스템이 아닌 경우 버전이 점점 다운그레이드 되는 역 호환성이 있다고 할 수 있으나 공격자가 그런 행위를 시도할 경우 프로토콜은 이를 감지하고

연결을 끊어 버림으로써 푸들이나 드라운 공격 등의 취약점을 막아낼 수 있다.

현재 브라우저들이 구 버전의 프로토콜의 지원을 중단 하더라도 다수의 서버들이 구 버전을 사용하는 서버들이 존재한다. 심한 경우 SSL 프로토콜을 사용하는 경우도 있으며 해당 서버들의 경우 문제가 되지 않도록 되도록이면 TLS v1.3으로 업그레이드 하는 것을 추천한다. 사용자들도 브라우저의 최신 버전을 유지하도록 권장하며 구 버전을 지원하는 브라우저이거나 업그레이드를 하지 못할 경우 브라우저 설정에서 구 버전의 사용을 하지 않도록 설정한다.

개발자들은 완벽한 보안이라 자부하지만 어떠한 보안이든 취약점이 존재하기 마련이며 그들의 말을 너무 신임하지 말아야 한다. 사용자 본인도 경각심을 가져야 하며 기본적으로 사용하는 보안 프로그램과 브라우저의 보안 경고를 무시하지 말아야 하며 주기적으로 업데이트를 하는 것이 중요하다. 또한, 개인이 지켜야 할 보안수칙을 우선적으로 지키는 것을 권장한다.

참고문헌

[1] 네이버지식백과 SSL - https://terms.naver.com/entry.nhn?docId=862837&cid=42346&categoryId=42346
[2] SSL 연결과정 - http://blog.daum.net/tlos6733/57
[3] OpenSSL 취약점 하트블리드(HeartBleed), 왜 위험한가? - https://blog.alyac.co.kr/76
[4] 프리크 분석 보고서 - https://blog.naver.com/hilineisp/220290760573
[5] 새로운 암호화 취약점 Logjam,인터넷 사용자들을 위험에 빠트려 - https://blog.alyac.co.kr/330
[6] POODLE 취약점 또 다시 발생 주의! - https://blog.alyac.co.kr/218
[7] SSL/TLS의 이해와 TLS 1.3으로 업그레이드해야 하는 이유
http://www.itworld.co.kr/news/113007