논문 19-44-12-23 The Journal of Korean Institute of Communications and Information Sciences '19-12 Vol.44 No.12
https://doi.org/10.7840/kics.2019.44.12.2339

클라이언트 환경에서 보안 소켓 레이어의 트래픽 복호화

박 성 욱◆, 박 민 호°

Decryption of Secure Socket Layer in Client Environment

Sung-wook Park◆, Min-ho Park°

요 약

보안 소켓 레이어는 네트워크상의 통신을 암호화하여 트랙의 내용을 알 수 없게 하고 이를 악용한 보안 위협이 존재한다. 이에 따라 보안 소켓 레이어를 사용한 위협의 통제방법으로 네트워크의 장비들은 프록시 서버와 가시화 장비를 활용하여 암호화된 트래픽을 복호화 하였다. 하지만 클라이언트에서는 동작하는 보안프로그램이 활용할 수 있는 암호화 트래픽 복호화 방법은 연구가 부족한 실정이다. 본 논문에서는 클라이언트 환경에서 암호화 트래픽을 복호화하는 서비스를 제안하였다. 제안 서비스는 API Hooking을 이용하여 보안 소켓 레이어의 연결을 감시하고, 클라이언트와 서버를 각각 보안 소켓 레이어로 연결하여 암호화된 트래픽을 복호화하였다. 클라이언트 환경에서 복호화 방법은 기존의 방법보다 대역폭 성능이 낮은 네트워크에 효과적이고, 장애 발생 시 가용성이 더 높았다. 따라서 본 논문에서 제안하는 서비스는 기존의 보안프로그램에 부족한 암호화 트래픽 통제 부분을 보완 할 수 있다.

Key Words : security socket layer, ssl, decryption, traffic monitoring, client security

ABSTRACT

A security socket layer encrypts packets over a network and makes the content of the traffic invisible, which may cause security threats that can be exploited by attackers. A hardware based approach to decrypt the traffic by using proxy servers or visualization devices is the most general counter measure. However, there is a rack of research on a software based approach where a security program running on a host decrypts the encrypted traffic. This paper propose an efficient software based approach that can monitor the connection of the secure socket layer using API hoocking and decrypt the encrypted traffic in a client side, therefore, the proposed method can reinforce the management function of the encrypted traffic in the existing security program.

Ⅰ. 서 론

보안 소켓 레이어(Secure Sockets Layer)는 클라이언트와 서버 사이에 암호화된 통신을 구현하는 보안 프로토콜로써, 클라이언트와 서버의 데이터 통신에서 평문 전송보다 안전한 전송을 제공한다. 따라서 온라인 금융 서비스나 웹사이트 로그인, 메일 전송 서비스와 같이 데이터의 보호와 신뢰가 필요한 인터넷 서비스에 활용도가 높아지고 있다.

하지만 기존의 보안제품들은 암호화된 통신의 내용을 확인할 수 없어 보안 기능이 무력화되는 문제가 있으며, 이 점은 인터넷 서비스 이용자의 정보 보호를 위협하는 요소가 되었다. 이러한 위협을 통제하기 위해 암호화 통신을 복호화하는 기술의 관심이 높아졌으며, 이에 2가지 방법으로 발전되었다.

첫 번째, 클라이언트와 서버 사이에 중계자 역할인

◆ First Author : Department of Information Sercurity, Soongsil University, barzell@soongsil.ac.kr, 정회원
° Corresponding Author : School of Electronic Engineering, Soongsil University, mhp@ssu.ac.kr, 종신회원
논문번호 : 201806-035-D-RE, Received May 28, 2018; Revised September 28 15, 2019; Accepted October 11, 2019

프록시 서버(Proxy Server)를 활용하여 암호화 통신을 복호화하는 방법이고 두 번째, 가시성 장비(Visibility Appliance)를 도입하여 네트워크망에서 암호화 통신을 복호화하는 방법이다.

이 방법들은 클라이언트가 암호화된 트래픽을 외부로 전송하면 네트워크상의 복호화 지점을 지날 때 복호화하는 공통점이 있다. 즉, 암호화된 트래픽은 복호화하는 구간을 반드시 거쳐야 트래픽 복호화가 가능하다. 따라서 방화벽, 침입탐지시스템, 침입방지시스템과 같이 네트워크 구간에 적용하는 보안 장치는 기존의 방법을 통해 트래픽 통제가 가능하지만, 백신이나 내부정보유출방지 제품과 같이 클라이언트에서 동작하는 보안프로그램은 트래픽을 외부로 전송하기 전에 통제하므로 프로그램 내부에서 복호화가 필요하다. 본 논문에서는 클라이언트 환경에서 동작하는 보안프로그램이 암호화된 트래픽을 복호화하는 방법을 수립하고자 한다.

Ⅱ. 관련연구

2.1 보안 소켓 레이어

보안 소켓 레이어는 클라이언트와 서버의 응용 프로그램이 네트워크로 통신하는 과정에서 두 지점 간의 인증(Authentication)과 데이터 보안성(Confidentiality), 무결성(Integrity)을 제공하는 암호 규약이다.

보안 소켓 레이어는 넷스케이프에서 발표하여 초기 SSL 1.0 거쳐, SSL 2.0과 SSL 3.0 순서로 발표하였고, IETF(Internet Engineering Task Force)는 마지막에 발표한 SSL 3.0을 기초로 하여 TLS(Transport Layer Security)를 표준 규약으로 정의하였다.

2.1.1 보안 소켓 레이어 구조

보안 소켓 레이어는 그림 1과 같이 TCP/IP 4계층을 기준으로 3계층인 전송 계층(Transport Layer)과 4계층인 애플리케이션 계층(Application Layer) 사이에 존재하며, 애플리케이션 계층의 HTTP, SMTP, FTP 등 여러 프로토콜에 암호화 통신을 적용할 수 있다.

보안 소켓 레이어는 2가지 계층으로 나누어지며 첫 번째 계층에는 핸드쉐이크, 암호 사양 변경, 경고 알림, 애플리케이션 데이터로 구성되어있고, 두 번째 계층은 레코드로 구성되어있다. 첫 번째 계층에서는 클라이언트와 서버의 연결에서 하나 또는 두 개의 지점을 각각 인증하고, 데이터 암호화와 암호를 해독을 위해 사용할 대칭 키를 설정한다. 두 번째 계층에서는

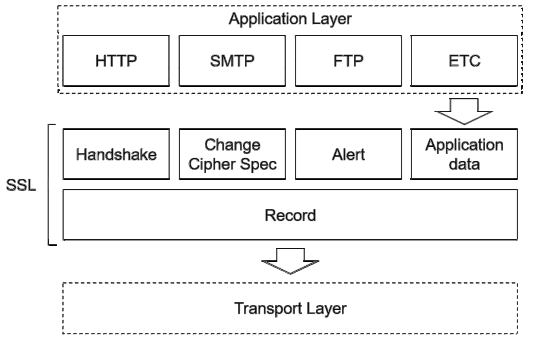

Fig. 1. TCP/IP over SSL

클라이언트와 서버의 연결에서 데이터 흐름을 제어한다[1].

2.1.2 보안 소켓 레이어 동작 과정

클라이언트는 그림 2와 같은 절차로 암호 협상을 진행하며, 이 과정에서 상대방 서버를 인증한다[2].

1 : 클라이언트는 서버에게 Client Hello를 전송한다.
2 : 서버는 클라이언트에게 Server Hello, SSL 인증서, Server Hello Done을 전송한다.
3 : 서버가 제공한 SSL 인증서를 검증하며 SSL 인증서를 신뢰할 수 있는지 확인한다.
4 : 클라이언트는 서버에게 Client Key Exchange와 Change Cipher Spec, Finished를 전송한다.
5 : 서버는 클라이언트에게 Change Cipher Spec, Finished를 전송한다.

서버 인증에서 SSL 인증서를 검증하는 항목은 인증서 발급자, 인증서 날짜, 인증서의 이름과 사이트의 이름이며 검증 항목을 모두 만족 못 하면 그림 3과 같은 보안 경고가 출력된다.

서버 인증이 통과하고 암호 협상이 종료되면, 클라

Fig. 2. SSL Flow

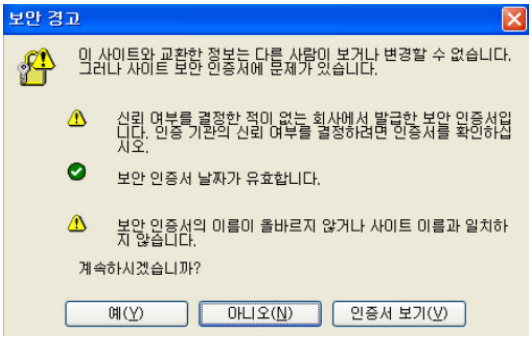

Fig. 3. Security Alert

이언트와 서버는 대칭 키를 생성하여 데이터를 암호화하여 주고받게 된다.

2.2 기존의 보안 소켓 레이어 복호화 방법

보안 소켓 레이어의 복호화 방법은 프록시 서버를 활용한 방법, 가시성 장비를 활용한 방법 2가지로 분류할 수 있다.

2.2.1 프록시 서버를 활용한 방법

암호화된 트래픽을 복호화하기 위해 초기에 등장한 방법으로 그림 4와 같은 형태로 구성을 한다.

클라이언트는 프록시 서버가 발급한 인증서를 신뢰할 수 있는 인증기관에 등록하고, 프록시 서버의 IP 주소를 시스템에 등록하여 암호화된 트래픽이 프록시 서버를 지나게 한다. 프록시 서버는 클라이언트와 서버에 각각 서버 인증과 암호 협상을 맺음으로써, 클라이언트의 암호화된 트래픽을 해독할 대칭 키와 서버의 암호화된 트래픽을 해독할 대칭 키를 얻게 된다. 두 대칭 키를 확보한 프록시 서버는 암호화 트래픽을 복호화하여 평문을 얻을 수 있다[3].

하지만 클라이언트에 프록시 서버 IP 주소를 시스템에 등록하여 트래픽을 복호화하는 방법은 다음과 같은 한계점 갖는다.

첫 번째, 클라이언트에서 동작하는 모든 프로그램이 프록시 서버 설정을 지원하지 않는다. 따라서 일부 프로그램은 프록시 서버를 거치지 않고 서버와 통신을 할 수 있다. 이 경우 암호화된 트래픽을 해독할 수 없다.

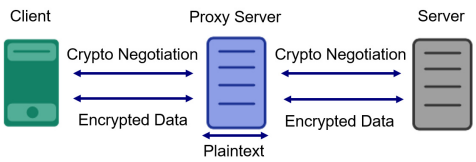

Fig. 4. Proxy Server Architecture

두 번째, 클라이언트에서 전송한 트래픽은 프록시 서버를 거치면 출발지 IP 주소가 프록시 서버의 IP 주소로 변경된다. 이 점으로 인해 프록시 서버를 거쳐간 클라이언트는 방화벽이나 침입탐지시스템, 침입방지시스템이 미리 설정한 보안규칙을 적용할 수 없다.

세 번째, 프록시 서버가 장애가 발생하여 중지된다면 연결된 모든 클라이언트는 서버로 통신할 수 없다.

2.2.2 가시성 장비를 활용한 방법

프록시 서버를 사용한 방법의 한계점을 개선한 방법으로 그림 5와 같은 형태로 구성을 한다.

클라이언트가 서버로 트래픽을 전송하는 구간에 가시성 장비를 적용한다. 클라이언트는 가시성 장비가 발급한 인증서를 신뢰할 수 있는 인증기관에 등록한다.

가시성 장비는 클라이언트의 암호화 트래픽 전송을 감지하고 트래픽을 해독하여 보안 장치로 전달한다. 보안 장치는 해독된 트래픽을 전달받아 통제를 수행한다[4].

가시성 장비를 도입하는 경우 다음과 같은 한계점을 고려해야 한다.

첫 번째, 가시성 장비의 트래픽 처리 성능보다 많은 트래픽이 유입되면 처리 성능이 저하된다.

두 번째, 가시성 장비를 지나가지 않는 클라이언트 통신 구간은 암호화 트래픽을 해독할 수 없다.

세 번째, 가시성 장비에 장애가 발생하는 경우 클라이언트의 트래픽을 통과시켜 서버와 통신을 할 수 있지만, 암호화 트래픽을 해독할 수 없다.

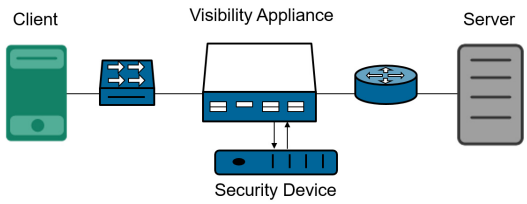

Fig. 5. Visibility Appliance Architecture

2.3 OpenSSL

OpenSSL은 보안 소켓 레이어를 구현한 오픈소스 라이브러리로써 암호화 라이브러리와 통신 라이브러리로 구성되어있다.

암호화 라이브러리에서는 공개 키 암호 알고리즘인 RSA 알고리즘을 지원하며, 이 알고리즘을 사용하여 CA 개인 키, CA 인증서, SSL 인증서 생성 할 수 있

Table 1. OpenSSL Function

Library	Function	Description
Crypto library	RSA_generate_key	Generate RSA
	EVP_PKEY_new	Generate private key
	X509_new	Generate certificate
SSL library	SSL_accept	Accept connection
	SSL_connect	Request connection
	SSL_read	Send encryption traffic
	SSL_write	Receive encryption traffic

다. 통신 라이브러리에서는 암호화 통신의 암호 협상 과정을 포함한 암호화 연결 수락 및 요청 기능과 트래픽간의 통신에서 암호화 및 복호화 기능을 제공한다 [5]. 아래 표 1은 OpenSSL에서 암호화 통신을 사용하기 위해 필요한 함수를 나열하였다.

Ⅲ. 클라이언트 보안 소켓 레이어 트래픽 복호화

본 논문에서의 클라이언트 환경이란 HTTP, FTP, SMTP와 같은 인터넷 서비스에 접근하는 개인 사용자 컴퓨터 환경을 의미한다. 이 환경에서 암호화된 트래픽을 복호화하는 로컬 프록시의 형태의 서비스를 제안할 것이며, 크게 5단계로 구성하였다.

1 : CA 개인 키 및 CA 인증서 생성
2 : 연결 감시 단계
3 : 대칭 키 확보 단계
4 : 요청 트래픽 복호화 단계
5 : 응답 트래픽 복호화 단계

첫 번째 과정에서는 OpenSSL의 암호화 라이브러리를 이용하여 CA 개인키와 CA 인증서를 생성하고, 두 번째 과정에서는 API Hooking을 이용하여 클라이언트 프로그램의 보안 소켓 레이어 연결을 감시한다. 세 번째 과정에서는 서버의 인증서를 복제하여 클라이언트에 제공함으로써 두 지점의 대칭 키를 확보하고, 네 번째 과정에서 클라이언트에서 서버로 향하는 암호화 트래픽을 복호화하고, 마지막 과정으로는 서버에서 클라이언트로 향하는 암호화 트래픽을 복호화한다.

3.1 서비스 구성

서비스는 총 2가지 모듈로 나눌 수 있다. 첫 번째, 연결을 감시하기 위한 감시 모듈(Watch.dll). 두 번째, CA 개인 키와 CA 인증서, SSL 인증서를 생성하고 암호화 트래픽을 해독하여 복호화 트래픽을 재전송을 하는 가시화 모듈(Visuallize.dll)이다. 모듈의 구성은 그림 6과 같다.

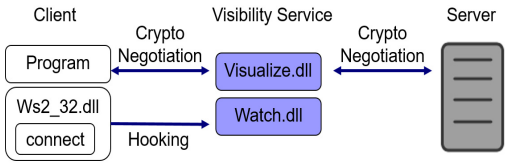

Fig. 6. Visibility Service Architecture

3.2 감시 모듈

감시 모듈은 클라이언트의 연결을 감시하여 목적지 정보를 수정하는 역할을 한다.

그림 7과 같이 프로그램은 네트워크에 연결하여 데이터를 주고받기 위하여 Ws2_32.dll에서 socket을 할당하고, connect 함수를 호출하고, send/recv 순서로 함수를 호출한다. 이때, 감시 모듈은 프로그램이 connect 함수를 호출하기 이전에 API Hooking 기술을 사용하여 connect 함수의 호출을 감시하고, connect 함수가 호출되면 감시 모듈의 제어 코드를 호출되게 한다.

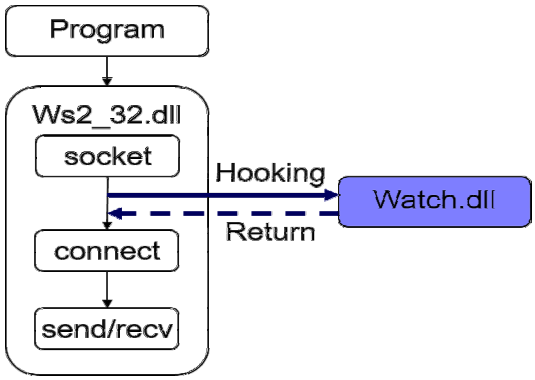

Fig. 7. API Hooking

3.2.1 연결 감시

제안 서비스의 감시 모듈은 그림 8과 같은 동작 절차를 갖는다.

1 : 클라이언트에서 프로그램이 실행된다.

2 : 감시 모듈은 프로그램이 사용하는 Ws2_32.dll의 connect 함수를 API Hooking 한다.
3 : 클라이언트에서 connect 함수를 호출한다.
4 : 감시 모듈에서 제어 코드가 호출된다.
5 : 클라이언트의 연결이 가시화 모듈로 연결되도록 목적지 정보를 수정한다.
6 : 클라이언트가 기존에 연결하려고 했던 목적지 정보를 가시화 모듈로 전달한다.

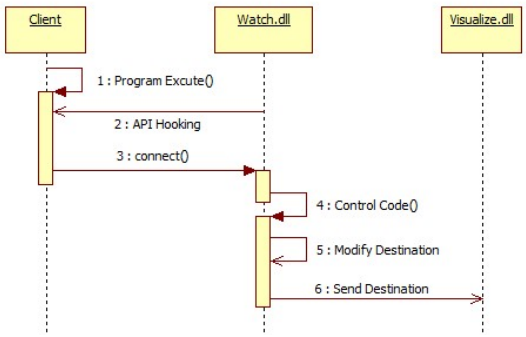

Fig. 8. Watch Module Diagram

3.3 가시화 모듈

가시화 모듈에서는 CA 개인 키와 CA 인증서를 만들어 클라이언트 시스템에 등록한다. 이 후, 감시 모듈에서 목적지 정보데이터를 전달받아 클라이언트와 서버에 각각 보안 소켓 레이어 암호 협상을 맺어 대칭 키를 생성하고, 생성한 대칭 키로 암호화 트래픽을 해독하여 복호화된 트래픽을 확보하는 역할을 한다.

3.3.1 CA 개인 키 및 CA 인증서 생성

가시화 모듈이 시작되면 OpenSSL의 암호화 라이브러리로 RSA 키 쌍 생성을 생성하여, CA 개인 키와 CA 인증서로 사용한다. 이 때, 생성 된 CA 인증서는 CertOpenStore, CryptUIwizImport 함수를 이용하여 신뢰할 수 있는 루트 인증 기관에 등록 한다. 이 과정은 그림 9와 같은 절차를 갖는다.

1 : RSA_generate_key 함수를 호출하여 2048 bit의 크기만큼의 키 쌍을 생성한다.
2 : EVP_PKEY_new 함수를 호출하여 CA 개인 키 영역을 할당하고 개인 키를 저장한다.
3 : X509_new 함수를 호출하여 CA 인증서 영역을 할당하고 인증서를 저장한다
4 : CertOpenStore 함수를 호출하여 저장한 CA 인증서를 메모리에 불러온다.

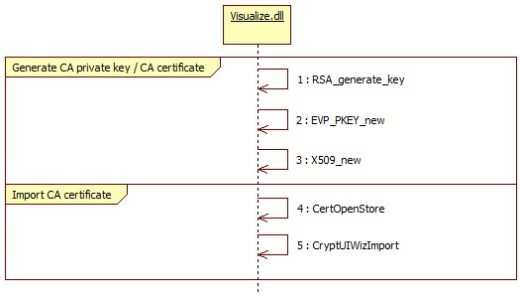

Fig. 9. Generate CA

5 : CryptUIWizImport 함수를 호출하여 CA 인증서를 시스템의 신뢰할 수 있는 루트 인증 기관에 등록한다.

3.3.2 대칭 키 확보

가시화 모듈이 클라이언트와 서버 사이에 각각 보안 소켓 레이어 암호 협상을 하는 과정에서 대칭 키를 생성하는 과정은 그림 10와 같은 절차를 갖는다.

1 : 클라이언트는 connect 함수를 호출한다. 이때, 감시 모듈로 인해 목적지가 가시화 모듈로 변경되어 연결된다.
2 : 가시화 모듈은 원래의 목적지 정보를 수신한다.
3 : 가시화 모듈은 connect 함수를 호출하여 서버에 연결한다.
4~8 : 가시화 모듈은 서버와 보안 소켓 레이어 암호 협상과 서버 인증을 수행한다. 이후 가시화 모듈과 서버의 통신에 사용할 대칭 키를 생성한다.
9 : 가시화 모듈은 암호 협상 과정에서 수신한 SSL 인증서에서 주체 이름(Common Name)과 주체의 대체 이름(Subject Alternative Name)을 얻

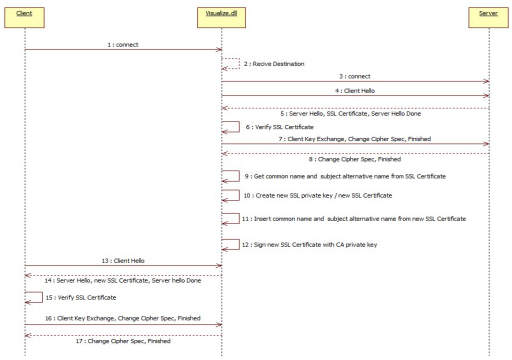

Fig. 10. Visualize Module Diagram

어온다.

10 : 가시화 모듈은 OpenSSL의 암호화 라이브러리로 RSA 키 쌍을 생성하여 새로운 SSL 개인 키와 SSL 인증서를 생성한다.

11 : 가시화 모듈은 새로운 SSL 인증서에 주체 이름과 주체 대체 이름을 입력한다.

12 : 가시화 모듈은 CA 개인 키로 새로운 SSL 인증서에 전자 서명을 한다.

13~17 : 클라이언트는 가시화 모듈과 보안 소켓 레이어 암호 협상을 수행하며, 수신한 SSL 인증서는 클라이언트에 등록된 신뢰할 수 있는 루트 인증기관의 전자 서명이 검증되어 서버 인증을 통과한다. 이후 클라이언트와 가시화 모듈의 통신에 사용할 대칭 키를 생성한다.

3.3.3 요청 트래픽 복호화

이 시점부터 클라이언트-가시화 모듈-서버는 보안 소켓 레이어로 연결되어 트래픽을 암호화하여 전송한다. 클라이언트의 요청 트래픽을 서버로 전송할 때, 암호화된 트래픽을 복호화하는 과정은 그림 11과 같은 절차를 갖는다.

1 : 클라이언트는 요청 트래픽을 암호화하여 가시화 모듈에 전송한다.

2 : 가시화 모듈은 암호화된 트래픽을 수신한다.

3 : 가시화 모듈은 클라이언트와 생성한 대칭 키를 이용하여 암호화된 요청 복호화한다.

4 : 가시화 모듈은 복호화된 평문 트래픽을 얻는다.

5 : 가시화 모듈은 서버와 생성한 대칭 키를 이용하여 평문 트래픽을 암호화한다.

6 : 가시화 모듈은 새로운 암호화 트래픽을 서버에 전송한다.

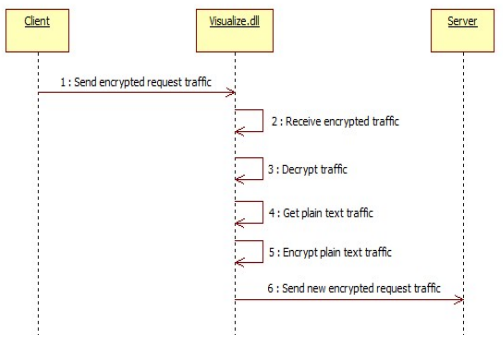

Fig. 11. Request Traffic Visualization

3.3.4 응답 트래픽 복호화

암호화된 트래픽을 수신한 서버는 응답 트래픽을 암호화하여 전송한다. 서버가 응답 트래픽을 클라이언트로 전송할 때, 암호화된 트래픽의 복호화되는 과정은 그림 12와 같은 절차를 갖는다.

1 : 서버는 응답 트래픽을 암호화하여 가시화 모듈에 전송한다.

2 : 가시화 모듈은 암호화된 트래픽을 수신한다.

3 : 가시화 모듈은 서버와 생성한 대칭 키를 이용하여 암호화된 응답 트래픽을 해독한다.

4 : 가시화 모듈은 복호화된 평문 트래픽을 얻는다.

5 : 가시화 모듈은 클라이언트와 생성한 대칭 키를 이용하여 평문 트래픽을 암호화한다.

6 : 가시화 모듈은 새로운 암호화 트래픽을 클라이언트에 전송한다.

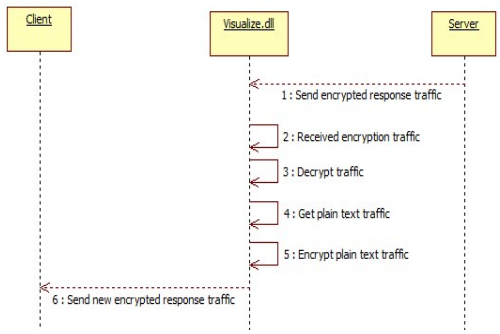

Fig. 12. Response Traffic Visualization

Ⅳ. 시험 및 성능 평가

4.1 시험 환경 구성

클라이언트 환경에서 시험 환경은 그림 13과 같이 구성하였다.

클라이언트의 운영체제는 Windows 10 Pro N을 사용하고, 네트워크는 1.0 Gbps로 연결하고, 모든 프로그램에 별도의 프록시 설정을 적용하지 않았다. 응용 프로토콜에 따른 프로그램 사용은 표 2와 같다.

제안 서비스의 상세정보는 표 3과 같다.

시험은 아래와 같은 순서로 진행한다.

1 : 클라이언트에 제안한 서비스를 실행하여 감시 모듈과 가시화 모듈을 적용한다.

2 : 클라이언트에서 각각 프로그램을 실행하여 보안 소켓 레이어가 적용된 서버에 접속한다.

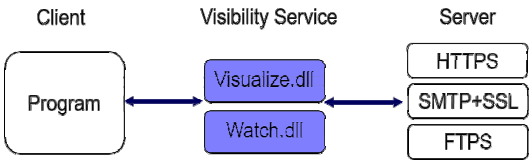

Fig. 13. Test Environment

Table 2. Test Protocol - Program

Protocol	Program
HTTPS	Internet Explorer
HTTPS	Chrome
SMTP+SSL	Outlook
FTPS	Filezilla

Table 3. Test Service Information

Name	Value
Process Name	VisibilityService.exe
Receive Port	TCP 10753
CA Name	Visibility

3 : 접속한 서버에서 메일을 발송하거나 로그인을 수행한다.

4 : 가시화 모듈에서 평문 트래픽을 파일로 저장한다.

5 : 저장된 파일을 메일의 내용을 확인하거나 로그인 과정의 패스워드를 확인한다.

4.2 시험 결과 및 성능 평가

운영체제에서 지원하는 리소스 모니터를 사용하여 그림 14와 같이 제안 서비스의 실행과 수신 포트를 확인할 수 있다.

인증서 관리 콘솔을 통하여 그림 15와 같이 가시화 모듈이 생성한 CA 인증서를 신뢰할 수 있는 루트 인증 기관에 등록한 것을 확인할 수 있다.

그림 16은 프록시를 설정하지 않은 Chrome에서 https://www.kics.or.kr에 연결하였을 때 감시 모듈에 의하여 목적지가 서비스의 수신 IP와 포트로 수정되고 가시화 모듈이 원래의 목적지로 연결한 것을 확인할 수 있다.

Chrome에 포함된 DevTools를 사용하여 HTTPS 연결에 사용된 인증서를 확인할 수 있다. 그림 17과 같이 https://www.kics.or.kr 연결에 사용된 인증서는 가시화 모듈이 생성한 CA에서 발급됨을 확인할 수

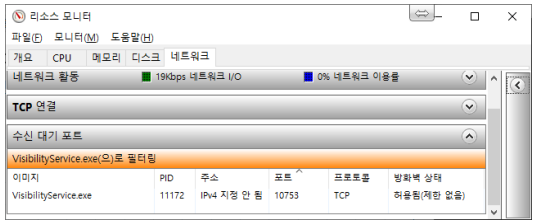

Fig. 14. Service - Process Name/Receive Port

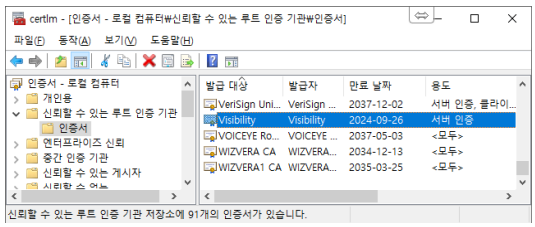

Fig. 15. Visualize.dll - CA Certificate

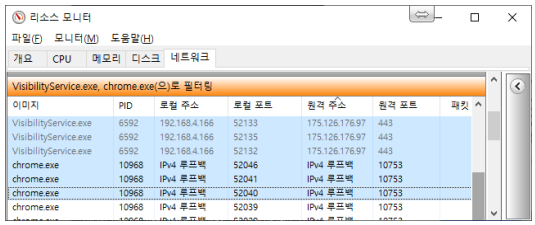

Fig. 16. Watch.dll - Connection

있다.

이후 가시화 모듈이 생성한 파일을 통해 클라이언트가 보안 소켓 레이어를 적용한 암호화 트래픽 전송에서 평문 트래픽을 확인할 수 있다.

그림 18은 Internet Explorer로 발송한 메일의 트래픽을 복호화하여 파일로 저장하였다. 파일을 통해 제목(subject), 본문(contents), 받는사람(toList), 참조(ccList), 숨은참조(bcclist), 보내는사람(from)의 내용이 JSON(JavaScript Object Notation) 형태로 전송됨을 확인할 수 있다.

그림 19는 Chrome으로 발송한 메일의 트래픽을 복호화하여 파일로 저장하였다. 파일을 통해 보내는이름

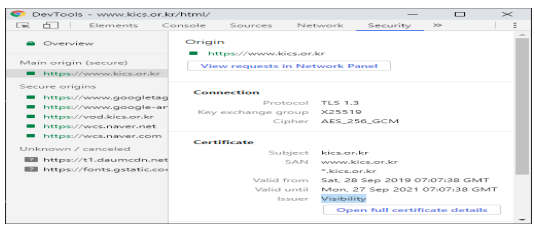

Fig. 17. HTTPS Certificate

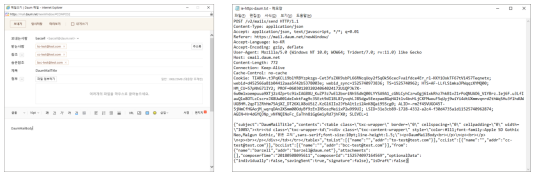

Fig. 18. Visualization - HTTPS 1

(senderName), 받는사람(to), 참조(cc), 숨은참조(bcc), 제목(subject), 본문(body)의 내용이 URL 인코딩(Uniform Resource Locators Encoding) 형태로 전송됨을 확인할 수 있다.

그림 20은 Oulook으로 발송한 메일의 트래픽을 복호화하여 파일로 저장하였다. 파일을 통해 받는 사람(To), 참조(Cc), 제목(Subject) 내용이 SMTP 헤더(Header)에 포함됨을 확인할 수 있다.

그림 21은 Filezilla의 FTPS 로그인 트래픽을 복호화하여 파일로 저장하였다. 파일을 통해 계정(USER)과 암호(PASS)를 확인할 수 있다.

제안한 서비스는 브라우저 외에도 프록시 설정을 지원하지 않는 프로그램의 암호화 트래픽을 복호화를 수행하여 기존 프록시 서버를 이용한 암호화 트래픽 복호화 방법의 한계점을 개선하였다. 또한, 제안 서비스는 클라이언트마다 독립적으로 동작하므로 분산 시스템의 특징을 갖는다. 따라서 중앙 집중 형식인 가시

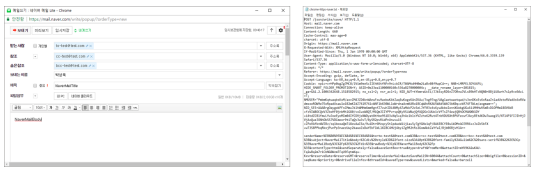

Fig. 19. Visualization - HTTPS 2

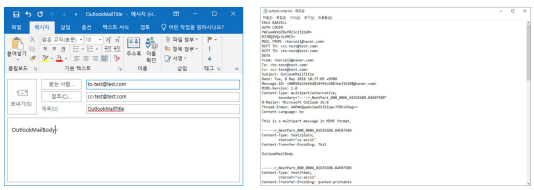

Fig. 20. Visualization - SMTP+SSL

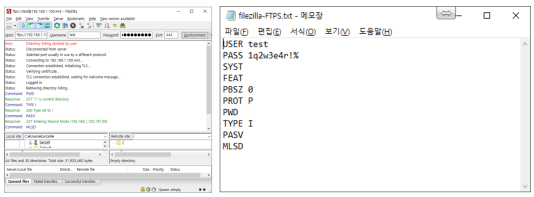

Fig. 21. Visualization - FTPS

표 4. Comparison

Type	Proxy	Visbility Solution Appliance	Visibility Service
Taget	Proxy support only	Network	Program
Structure	Centralized	Centralized	Distributed
Availability	Low	Middle	High

성 장비를 적용하기 어려운 네트워크 연결 속도가 느리거나, 여러 네트워크로 분산된 환경에 적용하기 용이하다. 그리고 서비스에 장애가 발생했을 때 다른 클라이언트에 동작하는 서비스에는 영향을 주지 않는다. 이 점은 기존의 암호화 트래픽 복호화 방법보다 가용성 측면에서 이점을 갖는다.

가시화 서비스는 복호화된 트래픽을 다시 암호화하여 재전송을 방법을 사용한다. 따라서 시간적 지연이 필연적으로 발생하였지만 이를 수치화하여 측정하기에는 네트워크 트래픽 전송량이 항상 일정하지 않고, 클라이언트에 프로세스 스케줄링도 항상 일정하지 않기에 측정 방법을 구체화하기에 어려움이 있었다.

V. 결 론

본 논문에서는 클라이언트 환경에서 보안 소켓 레이어의 암호화 트래픽을 복호화하는 서비스를 제안하였다. 새로운 클라이언트의 복호화 방법은 API Hooking을 사용한 감시 모듈과 OpenSSL 라이브러리를 활용한 2가지 모듈로 각각 구성했고, 모듈을 동작 과정을 구체적인 절차로 제안하였다.

제안한 서비스는 클라이언트에서 기존의 프록시 서버의 복호화 방법에서 지원하지 않는 다양한 실행 프로그램을 대상으로 암호화 트래픽을 복호화 하였으며, 이로인해 분산된 네트워크 환경에 적용이 용이한 특성이 발생하였다. 성능적인 측면에서는 서비스가 적용되지 않는 환경과 비교하였을 때 시간적 지연이 발생하였다.

향후 성능적인 측면에서 발생한 시간적 지연에 대한 측정 방법을 구체화하고 시간적 지연을 최소화하여 높은 성능으로 암호화 트래픽을 복호화할 수 있도록 연구가 필요하다.

본 논문의 제안은 기존의 보안프로그램의 미흡한 암호화 트래픽 통제 기능의 기반이 되는 방법을 제안함으로써 더욱 안전한 보안 기능을 제공할 수 있는 효

과를 주었다.

References

[1] IBM, *How SSL works*(2018), Retrieved May 23, 2018, from https://www.ibm.com/support/knowledgecenter.

[2] J. Y. Yang and H. K. Choi, “A study on the certificate verification testing in SSL/TLS implementations,” in *Proc. Symp. KICS*, pp. 138-139, Jeju Island, Korea, Jun. 2017.

[3] Y. R. Hong and D. S. Kim, “Content-based control of https mail for implementation of IT-Convergence security environment,” *J. Intell. Manufacturing (JIM)*, vol. 25, pp. 231-239, Apr. 2014.

[4] H. C. Jang and M. H. Park, “Decrypted traffic method for encrypted traffic based on ssl/tls of enterprise network clients,” in *Proc. Symp. KICS*, pp. 613-614, Gangwon-do, Korea, Jan. 2018.

[5] OpenSSL, *Manpages for 1.0.2*(2018), Retrieved Aug. 27, 2018, from https://www.openssl.org/docs/man1.0.2.

박 성 욱 (Sung-wook Park)

2018년 8월 : 숭실대학교 정보과학대학원 정보보안학과 석사 졸업

<관심분야> 정보보안, 컴퓨터공학, 통신공학

[ORCID:0000-0003-4177-5411]

박 민 호 (Min-ho Park)

2000년 2월 : 고려대학교 전자공학화 학사 졸업

2002년 2월 : 고려대학교 전자공학화 석사 졸업

2010년 2월 : 서울대학교 전기컴퓨터공학과 박사 졸업

2013년 3월~현재 : 숭실대학교 전자정보공학부 부교수

<관심분야> 전자공학, 컴퓨터공학, 통신공학

[ORCID:0000-0003-3033-192X]