소프트웨어 업데이트 유형별 위협요소 및 대응방안

이대성*

*부산가톨릭대학교

Threats and response measures according to the type of software updates

Daesung Lee*

*Catholic University of Pusan

E-mail : dslee@cup.ac.kr

요 약

지능형 타깃 지속 공격에 업데이트 서버가 유포수단으로 사용되고, 업데이트 프로그램은 악성코드의 실행 또는 안티바이러스 시그니처와 같은 애플리케이션 데이터의 조작에도 취약하기 때문에SW 업데이트 위협요소의 식별 및 방지대책이 시급하다. 본 논문에서는 국내외 SW의 업데이트 구조와 업데이트 과정의 취약성 공격 및 대응방안을 살펴보고, 국내 유명 SW의 업데이트 로그를 추출/분석하여 화이트리스트를 식별하는데 필요한 정상적인 프로그램의 업데이트 구성요소를 도출한다.

ABSTRACT

In case of APT attacks, the update server is being used as a means of dissemination, the update program is running malicious code or data in applications such as anti-virus signature is vulnerable to manipulation, SW Update threat identification and prevention measures are urgently required. This paper presents a natiional and international SW update structure, update process exploits and response measures to examine, Through the extraction/analysis of a domestic famous SW update log, we are willing to select the necessary component of the normal program update to identify a white list.

키워드

소프트웨어 업데이트, 위협요소, 화이트리스트, 로그 분석

Ⅰ. 서 론

인터넷 보급률의 급상승과 함께 프로그램의 자동 업데이트 기능은 국산 소프트웨어에서도 필수적인 요소가 되었으며, V3나 알약 같은 백신은 물론이고, 한글, 곰플레이어 등 모든 사용자 응용프로그램에 포함되어 있다. 업데이트란 문제의 예방 또는 해결에 도움을 주고, 컴퓨터의 작동 방식을 향상시키거나, 컴퓨팅 환경을 개선하기 위해 추가되는 소프트웨어이다. 소프트웨어 업데이트 메커니즘은 최종 사용자에게 업데이트 코드를 배포하는 데 필수적인 부분으로, 버그를 수정하고, 기능을 새로 추가하며, 취약성 공격의 대응책을 제공한다. 새로운 취약성이 드러나고 공격자가 그것을 임의대로 악용할 가능성이 있는 경우에는 신속한 업데이트 코드의 배포가 특히 중요하다. 많은 일반 응용프로그램에서 배포된 업데이트 메커니즘은 주어진 컴퓨터 보안 관행을 지키지 않기 때문에 공격에 취약하다. 한편 중요한 소프트웨어 업데이트도 그의 메커니즘이 복잡하거나 자동화되어 있지 않으면 많은 사용자들이 정규 업데이트를 하지 않을 수 있다.

본 논문에서는 다양한 업데이트 프로그램의 성능과 메커니즘을 조사하고 업데이트 유형의 특징을 파악한 후. 윈도우 환경에서 한글, 알집/알약, 곰플레이어 등과 같이 잘 알려진 국산 제품에 대한 업데이트 로그를 추출 및 분석한다. 궁극적인 목표는 응용프로그램의 업데이트 과정에서 안전한 것으로 증명된 입력 값만 허용하여 이들 프로그램의 정상적인 업데이트를 식별하는 화이트리스트를 도출하는 것이다.

본 논문의 구성은 1장의 서론에 이어 2장에서 국내외 소프트웨어의 업데이트 구조를 살펴보고, 3장에서는 업데이트 과정에서 발생하는 대표적 MITM(Man In The MIddle) 공격으로 프록시 하이재킹(Proxy hijacking)과 인젝션 공격(Injection attack)을 알아보며, 대응 방안으로 코드 서명(Code signing)과 SSL 통신에 대해 기술한다. 4장에서는 업데이트 과정에서 정상적인 입력 값만을 허용하는 응용프로그램의 화이트리스팅 디자인을 목표로 국내 유명 소프트웨어 7개의 업데이트 로그를 추출 분석하였으며, 5장에서 결론을 맺는다.

II. 응용프로그램 업데이트 구조

인터넷을 통한 자동 업데이트 소프트웨어의 기본 구조는, 일반적으로 서버에서 버전, 날짜, 파일 이름을 포함한 XML 형태의 파일을 생성하면, 클라이언트는 접속 시 또는 실행도중 주기적 시간 간격으로 서버에 요청하여 서버의 XML 파일을 수신한다. 클라이언트에서 XML 파일을 분석하여 현재 버전과 차이가 있는지 확인한 후, 새로운 버전이라 판단되면 업데이트 프로그램을 실행하고, 클라이언트는 종료된다. 업데이트 프로그램은 서버로부터 실행파일 및 기타 추가된 파일을 수신하여 새로운 파일로 복사한 다음, 클라이언트 프로그램이 실행되고 업데이트 프로그램은 종료된다. 이와 같은 업데이트 프로세스가 실제 국내외 상용 소프트웨어에서 다수 구현되고 있으며 안전을 위해 공개키 암호를 사용하여 서명하고 있다[2].

III. 업데이트 취약성 공격과 대응방안

사용자의 소프트웨어 패키지를 간편하게 최신 상태로 유지하기위해 도입된 자동 업데이트는 대부분의 패치를 적용하거나 응용 프로그램에 새로운 기능을 추가한다. 자동 업데이트 프로그램은 일정 주기로 또는 사용자의 요청에 의해 새로운 릴리즈나 패치를 점검하기위해 네트워크에 연결된다. 일반적으로 HTTP(Hyper Text Transfer Protocol)를 사용하여 업데이트가 다운로드 되며, 경우에 따라 보안접속을 위해 SSL(secure sockers layer)이 추가된다. 업데이트 절차가 안전하지 않은 방식으로 진행될 경우, 업데이터는 악성코드의 실행 또는 안티 바이러스 시그니처와 같은 애플리케이션 데이터의 조작에 취약성을 드러낸다. 이를 이용해 통신 채널에서 전송되는 데이터를 불법 수정하거나 거짓 데이터 스트림을 생성하는 등으로 쌍방의 의사소통을 가로채는 MITM(Man In The Middle) 공격 기법이 주로 사용되는 것으로 확인되었으며[2][4], 대표적인 공격으로 프록시 하이재킹(Proxy Hijacking)과 인젝션 공격(Injection Attack)이 사용되었다.

HTTP 다운로드를 훼손하여 특정 업데이트 과정을 악용하도록 만들어진 MITM 공격에 대한 대응기술로는 업데이트 트래픽의 무결성 및 인증과 관련된 코드 서명과 SSL 통신이 사용된다.

1. 코드 서명(Code Signing)

HTTP 코드 서명은 무결성과 인증을 보장하기 위해 배포전의 실행파일 또는 소스 코드에 디지털 서명하는 프로세스이다. 사용자에게 배포된 파일의 유효성은 파일과 함께 첨부된 전자 서명(digital signature) 및 제작자의 디지털 인증서(digital certificate 또는 public key certificate)의 검증을 통해 이루어진다. 디지털 인증서에는 제작자의 이름과 전자서명의 확인에 필요한 공개키, 유효기간, 그리고 해당 내용이 사실임을 증명하는 인증기관(CA, Certification Authority)의 이름과 그의 전자서명 등이 포함되어 있다. 파일의 출처는 인증서에 의해 보장된다. 원본 파일과 함께 파일 제작자의 개인키로 만든 서명 값이 첨부되고, 사용자는 제작자의 인증서에서 얻은 공개키, 원본 파일, 그리고 서명 값을 이용해 파일을 인증함으로써 파일이 제작자로부터 보내왔다는 것을 확신할 수 있다. 이 경우 일반적으로 해시 함수(hash function)로 메시지를 압축한 데이터를 송신자의 비밀키로 암호화하여 전자 서명을 생성하기 때문에 수신자는 송신자의 공개키로 전자 서명을 복호화한 값과 원래 메시지에 해시함수를 적용한 값을 비교하여 진위를 판단한다.

2. SSL 통신

SSL(Secure Socket Layer) 또는 TLS(Transport Layer Security)는 응용계층과 전송계층 사이에서 동작하며, 주로 다른 프로토콜과 연동하여 보안 서비스를 제공한다. HTTP상에 SSL이 구현된 HTTPS가 가장 일반적으로 사용된다. SSL의 궁극적인 목표는 기밀성, 무결성 인증 및 부인방지 기능을 제공하여 안전하지 않은 네트워크상에서 보안 채널을 생성하는 것이다. 클라이언트와 서버는 SSL 암호화 통신을 하기 위해 [그림 1]에서처럼 서로 메시지를 교환한다.

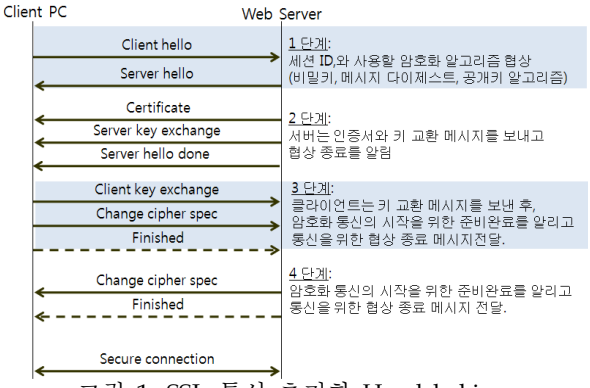

그림 1. SSL 통신 초기화 Handshaking

Ⅳ. 업데이트 취약성 공격 탐지 시스템

안티 바이러스 같은 보안 시스템은 화이트리스트나 블랙리스트를 이용하여 위험요소를 식별하고 필터링 한다. 지금까지는 블랙리스트 기반 보안 솔루션들이 주류를 이루었고, 여전히 주류를 이루고 있지만, 최근 보안 시스템 개발 업체들은 별도의 화이트리스트 기반의 솔루션을 출시하거나, 기존의 제품에 화이트리스트 방식을 통합하는 식으로, 화이트리스트가 새로이 주목받고 있다.

화이트리스트는 안전한 것으로 증명된 입력 값만 허용하기 때문에 위험성이 증명되지 않은 신종 악성코드를 차단할 수 있는 강력한 보안성을 유지한다. 하지만, 특정 프로그램의 사용만 허용하기 때문에 제한적인 환경에 적합하다. 즉, 수많은 애플리케이션을 이용하고 잦은 업데이트가 이루어지는 현실에서는 화이트리스트를 사용하여 관리의 한계가 있지만, 애플리케이션 수가 적고 변동이 크지 않는 특정 환경에서는 보안성 및 효율성을 확보할 수 있다[1].

응용프로그램 화이트리스팅은 승인되지 않은 악성 프로그램이 컴퓨터에서 실행되는 것을 방지하기 위해 디자인 된다. 화이트리스트에 특별히 선택된 실행파일과 소프트웨어 라이브러리(DLLS)만 실행될 수 있으며, 그 밖의 것들은 실행이 차단된다. 주로 악성 소프트웨어의 실행과 확산을 방지하기 위해 구현되지만, 승인되지 않은 악성 소프트웨어의 설치나 사용을 방지할 수 있다. 다양한 화이트리스팅 방법론은 일반적으로 응용프로그램 구성요소를 식별하여 화이트리스팅 정책과 결합시키는 몇 가지 공통된 방법에 기초를 두고 있으며, 인증서(certificate), 경로 값(path values), 해시 값(hash values), 서비스(services) 그리고 시스템 및 사용자 요소(system and user behavior) 등이 포함된다[3].

본 논문에서는 인터넷을 통해 업데이트를 수행하는 응용프로그램의 정상적인 상태를 식별하기 위한 화이트리스팅의 식별 기준이 되는 구성요소를 정의하기 위해 로그 분석 툴을 사용하여 국내 유명 소프트웨어(곰플레이어, 다음팟플레이어, 알집, 네이트온, 한글(hwp 2007), 알약, V3Lite)의 업데이트 실행 프로세스, 네트워크, 파일 및 레지스트리 정보 로그를 추출한 후 분석하였다. 본 분석에서 도출한 시스템상의 업데이트 파일 위치를 통해 업데이트 파일의 진위를 식별할 수 있는. 경로 값(path values)을 지정하고, 경로를 벗어나는 것은 유해한 것으로 간주하였다. 소프트웨어 업데이트 로그 추출 및 분석 방법은 다음과 같다.

- Windows XP OS상에서 Wireshark v1.6.5[5]와 Process Monitor v3.03[6]를 SW 업데이트 로그 추출 도구로 사용함
- 로그 분석에 사용된 국내 소프트웨어는 업데이트 형식에 따라 2가지 유형으로 분류되며, 새 버전이 기존의 전체 응용프로그램을 덮어쓰는 방식으로 업데이트가 실행되는 GOMPlayer, DaumPotPlayer, ALZip, NateOn,과 기존의 프로그램에 추가로 설치되는 증분 업데이트가 이루어지는 HWP 2007, ALYac, V 3 Lite임.
- 캡처 방식은 SW의 업데이트와 함께 Wireshark와 Process Monitor를 동시에 실행하여 .pcap 파일과 .PML 파일을 각 각 생성한 후, 분석을 위해 해당 업데이트 수행 프로세스와 관련된 로그만 추출함
- 추출 파일로부터 분석한 내용은 업데이트 유형, 실행 프로세스의 트리 구조, 업데이트 네트워크 기본 구조(client/main_server), 업데이트 진행과정의 네트워크 HTTP 로그(client/server IP, Port, URL), 그리고 네트워크로부터 전송된 패킷의 정확한 파일 경로를 추적한 파일 정보 및 레지스트리 정보임

Ⅴ. 결 론

화이트 리스트 기반 보안 시스템의 필터링 방식은 화이트리스트에 특별히 선택된 실행파일과 소프트웨어 라이브러리(DLLS)만 실행되고 그 밖의 것들은 실행이 차단된다. 화이트리스팅 방법론은 일반적으로 응용프로그램 구성요소를 식별하여 화이트리스팅 정책과 결합시키는 방법에 기초한다. 본 논문에서는 화이트리스트를 식별하는데 필요한 프로그램의 업데이트 구성요소를 정의하기위해 국내 소프트웨어 업데이트 로그를 추출하여 분석하였다. 본 분석 결과로부터 시스템상의 업데이트 파일 위치를 도출하여 업데이트 파일의 진위를 식별할 수 있는. 경로 값(path values)을 지정하고, 경로를 벗어나는 것은 유해한 것으로 간주할 수 있었다.

참고문헌

[1] Whitelist Security, Network Times, p153-162, 2010
[2] A. Bellissimo, J. Burgess, and K. Fu. Secure Software Updates: Disappointments and new challenges. In Proceedings of HotSec '06, p37–43. USENIX, July 2006.
[3] Dave. S, Application Whitelisting: Enhancing Host Security, A SANS Whitepaper, p1-14, 2009
[4] Bjoern M, Luettmann and Adam C. Bender, Man-in-the-Middle Attacks on Auto-Updating Software, Bell Labs Technical Journal 12(3), p131-138, 2007
[5] Wireshark: http://www.wireshark.org
[6] Process Monitor: http://technet.microsoft.com/en-us/sysinternals/bb896645