

바이러스 분석 과 전용 백신 제작 (Virus analysis and production to vaccine)

2006. 4. 18

By Maxoverpro[max](장상근)

maxoverpro@paran.com

<http://www.maxoverpro.org>

1. 서 론

2001년 Slammer 웜으로 인한 ‘1.25 인터넷 대란’이 일어날 만큼 현재 사이버 세상에서는 바이러스, 스파이웨어, 트로이, 웜, Bot 등. 많은 악성코드가 퍼져있어 인터넷에 연결되어 있는 것 자체가 위협에 노출되어 있는 것이나 다름없으며 악성코드는 진화하면서 보안 패치나 업데이트가 나오기 전에 취약점을 이용한 공격법이 나오는 Zero-Day가 현실이 되어 버렸다.

이 문서는 바이러스 패턴을 분석하는 방법에 대해 알아보겠으며 바이러스를 제작하거나 하는 방법은 기술하지 않는다. 이 문서에서 사용된 바이러스는 악성코드가 없이 주로 바이러스에서 나타나는 일반적인 현상들을 보기 위해 만든 실험용 바이러스임을 염두 해두길 바라며 실험용 바이러스를 치료하기 위해 바이러스를 분석하고 그에 따른 전용 백신을 제작하는 방법에 대해서 알아보도록 하겠다.

2. 본 론

Max는 자기의 메일로 “안녕! 오래간만이야~”라는 제목의 메일을 열었고 메일에 첨부된 loveme.exe 라는 파일을 실행시켰다. 할 일을 다하고 Max는 컴퓨터를 끄고 다음날 컴퓨터를 켜 보니 컴퓨터가 30초 뒤 컴퓨터가 재부팅 되는 현상이 발생되어 바이러스에 걸렸다고 의심을 갖게 되었고 Max의 컴퓨터에는 백신이 설치되어 있긴 하지만 최신 업데이트가 안된 백신이고 바이러스도 잡히지 않았다. 그래서 Max는 바이러스를 치료하기 위해 전용 백신을 만들기로 했다.

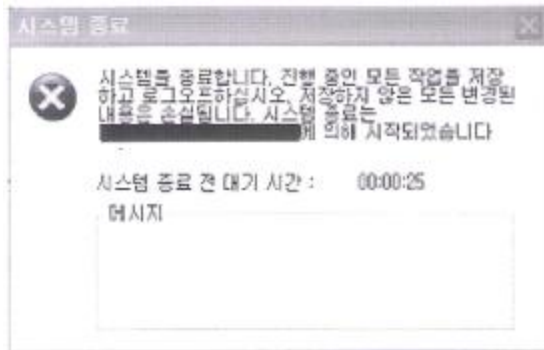
✓ 분석

위 바이러스 분석에 앞서 바이러스등의 악성코드들을 어떻게 탐지해내야 할지에 대해 알아보도록 하겠다. 현재 악성코드를 탐지하는 방법으로 알려진 악성코드에 대한 탐지와 알려지지 않은 악성코드의 탐지 방법으로 나누어져있다.

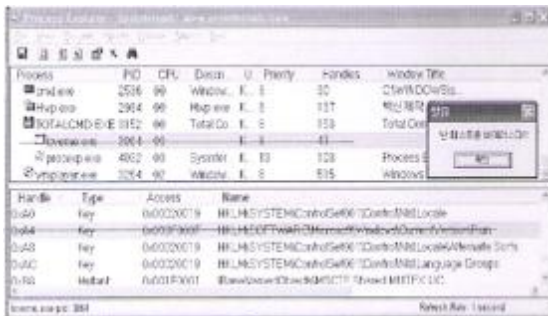
알려진 악성코드 탐지방법				알려지지 않은 악성코드 탐지방법		
패턴	시뮬레이션	CRC.	지능형	면역	악의적 행동	예상

이 문서에서 다루는 바이러스는 패턴 방식으로 접근하는 것으로 패턴에는 프로세스 상태, 레지스트의 접근, 파일, 네트워크 상태에서의 나오는 일정한 패턴등을 찾아내 악성코드를 탐지해 내는 방법이다.

- 기초적인 분석법



첫째, loveme.exe를 실행시킨 후부터 컴퓨터를 켜고 옆의 화면이 나오면서 계속 30초가 지나면 컴퓨터가 재부팅이 되는 현상을 확인을 하였다.



둘째, Process Explorer라는 Process 모니터링 프로그램으로 모니터링 해보면 loveme.exe가 실행되면서 어떠한 행동을 하는지 옆의 화면처럼 확인할 수 있다.



셋째, 옆의 그림은 TCPView라는 것을 통해 현재 어떤 프로세스가 어떤 포트를 열고 어떤 상태인지를 확인할 수가 있는데 현재 loveme.exe가 실행 중인 상태에서 loveme.exe의 프로세스가 사용하는 포트가 5678로 LISTENING되어 있는 상태로 보아 바이러스가 5678포트로 백도어 기능을 수행하거나 네트워크를 통해 바이러스를 전파할 수 있는 가능성을 확인하였다.

- 상세 분석

디코딩한 코드가 긴 관계로 바이러스의 중요 부분만 보기로 하겠다.

00401346	51	PUSH ECX	Disposition
00401347	8D55 FC	LEA EDX, DWORD PTR SS:[EBP-4]	Handle
00401348	52	PUSH EDX	oSecurity = NULL
00401349	6A 00	PUSH 0	Access = KEY_WRITE
0040134D	8B 06002000	PUSH 200006	Options = REG_OPTION_NON_VOLATILE
00401352	6A 00	PUSH 0	Class = NULL
00401356	6A 00	PUSH 0	Reserved = 0
00401358	8B 00E42000	PUSH DWORD 004200E4	Subkey = "Software\Microsoft\Windows\CurrentVersion\Run"
0040135D	6B 02000000	PUSH WORD 00000002	hKey = HKEY_LOCAL_MACHINE
00401362	FF15 88524200	CALL DWORD PTR DS:[-6AD9&P132.RegCreateKeyEx@]	hProcess = 0
00401368	8B45 CC	LEA EAX, DWORD PTR SS:[EBP-34]	BufSize
00401369	5B	PUSH EBX	Buffer
0040136C	EB 0F050000	CALL LOWDWORD 00401A30	ValueType = REG_SZ
00401371	83C4 04	ADD ESP, 4	Reserved = 0
00401374	83C0 01	ADD EAX, 1	ValueName = "max.virus"
00401377	50	PUSH EAX	hKey
00401378	8B40 CC	LEA ECX, DWORD PTR SS:[EBP-34]	hProcess
00401379	51	PUSH ECX	hKey
0040137C	6A 00	PUSH 0	hProcess
0040137E	6A 00	PUSH 0	hProcess
00401380	6B F8004200	PUSH LOWDWORD 004200F8	hProcess
00401385	8B55 FC	MOV EDI, DWORD PTR SS:[EBP-4]	hProcess
00401388	52	PUSH EDX	hProcess
00401389	FF15 00524200	CALL DWORD PTR DS:[-6AD9&P132.RegSetValueEx@]	hProcess
0040138F	8B45 FC	MOV EAX, DWORD PTR SS:[EBP-4]	hProcess
00401392	FF15 84524200	CALL DWORD PTR DS:[-6AD9&P132.RegCloseKey@]	hProcess
00401393	6A 00	PUSH 0	hProcess
00401398	6B 80000000	PUSH WORD 00000000	hProcess
004013A2	6A 02	PUSH 2	hProcess
004013A3	6A 00	PUSH 0	hProcess
004013A4	6A 00	PUSH 0	hProcess
004013A8	6B 00000000	PUSH WORD 00000000	hProcess
004013AB	6B F4004200	PUSH LOWDWORD 004200F4	hProcess
004013B0	FF15 0C524200	CALL DWORD PTR DS:[+6AEEH&L32.CreateFile@]	hProcess
004013B6	8B45 F0	MOV EDI, DWORD PTR SS:[EBP-10]	hProcess
004013B9	6A 00	PUSH 0	hProcess
004013BB	8B40 F4	LEA ECX, DWORD PTR SS:[EBP-C]	hProcess
004013BE	51	PUSH ECX	hProcess
004013BF	8D55 CC	LEA EDX, DWORD PTR SS:[EBP-34]	hProcess
004013C2	52	PUSH EDX	hProcess
004013C3	EB 68000000	CALL LOWDWORD 00401A30	hProcess
004013C8	83C4 04	ADD ESP, 4	hProcess
004013CB	50	PUSH EAX	hProcess
004013CC	8B45 CC	LEA EAX, DWORD PTR SS:[EBP-34]	hProcess
004013CF	5B	PUSH EBX	hProcess
004013D0	8B40 F0	MOV ECX, DWORD PTR SS:[EBP-10]	hProcess
004013D3	51	PUSH ECX	hProcess
004013D4	FF15 40524200	CALL DWORD PTR DS:[+6AEEH&L32.WriteFile@]	hProcess
004013D8	8B55 F0	MOV EDI, DWORD PTR SS:[EBP-10]	hProcess
004013DB	52	PUSH EDX	hProcess
004013DE	FF15 40524200	CALL DWORD PTR DS:[+6AEEH&L32.CloseHandle@]	hProcess
004013E0	6A 00	PUSH 0	hProcess
004013E3	6B 00004200	PUSH WORD 00420000	hProcess
004013E8	6B F4004200	PUSH LOWDWORD 004200F4	hProcess
004013F0	8B45 08	MOV EAX, DWORD PTR SS:[EBP+8]	hProcess
004013F3	50	PUSH EAX	hProcess
004013F4	FF15 60644200	CALL DWORD PTR DS:[+6A5F&H32.MessageBox@]	hProcess

00401398	51	PUSH ECX	hProcess
00401399	8D55 FC	LEA EDX, DWORD PTR SS:[EBP-4]	hProcess
0040139C	52	PUSH EDX	hProcess
0040139D	6A 00	PUSH 0	hProcess
0040139E	6A 00	PUSH 0	hProcess
004013A0	8B 06002000	PUSH 200006	hProcess
004013A3	6A 00	PUSH 0	hProcess
004013A4	6A 00	PUSH 0	hProcess
004013A8	6B 00000000	PUSH WORD 00000000	hProcess
004013AB	6B F4004200	PUSH LOWDWORD 004200F4	hProcess
004013B0	FF15 0C524200	CALL DWORD PTR DS:[+6AEEH&L32.CreateFile@]	hProcess
004013B6	8B45 F0	MOV EDI, DWORD PTR SS:[EBP-10]	hProcess
004013B9	6A 00	PUSH 0	hProcess
004013BB	8B40 F4	LEA ECX, DWORD PTR SS:[EBP-C]	hProcess
004013BE	51	PUSH ECX	hProcess
004013BF	8D55 CC	LEA EDX, DWORD PTR SS:[EBP-34]	hProcess
004013C2	52	PUSH EDX	hProcess
004013C3	EB 68000000	CALL LOWDWORD 00401A30	hProcess
004013C8	83C4 04	ADD ESP, 4	hProcess
004013CB	50	PUSH EAX	hProcess
004013CC	8B45 CC	LEA EAX, DWORD PTR SS:[EBP-34]	hProcess
004013CF	5B	PUSH EBX	hProcess
004013D0	8B40 F0	MOV ECX, DWORD PTR SS:[EBP-10]	hProcess
004013D3	51	PUSH ECX	hProcess
004013D4	FF15 40524200	CALL DWORD PTR DS:[+6AEEH&L32.WriteFile@]	hProcess
004013D8	8B55 F0	MOV EDI, DWORD PTR SS:[EBP-10]	hProcess
004013DB	52	PUSH EDX	hProcess
004013DE	FF15 40524200	CALL DWORD PTR DS:[+6AEEH&L32.CloseHandle@]	hProcess
004013E0	6A 00	PUSH 0	hProcess
004013E3	6B 00004200	PUSH WORD 00420000	hProcess
004013E8	6B F4004200	PUSH LOWDWORD 004200F4	hProcess
004013F0	8B45 08	MOV EAX, DWORD PTR SS:[EBP+8]	hProcess
004013F3	50	PUSH EAX	hProcess
004013F4	FF15 60644200	CALL DWORD PTR DS:[+6A5F&H32.MessageBox@]	hProcess

00401398	51	PUSH ECX	hProcess
00401399	8D55 FC	LEA EDX, DWORD PTR SS:[EBP-4]	hProcess
0040139C	52	PUSH EDX	hProcess
0040139D	6A 00	PUSH 0	hProcess
0040139E	6A 00	PUSH 0	hProcess
004013A0	8B 06002000	PUSH 200006	hProcess
004013A3	6A 00	PUSH 0	hProcess
004013A4	6A 00	PUSH 0	hProcess
004013A8	6B 00000000	PUSH WORD 00000000	hProcess
004013AB	6B F4004200	PUSH LOWDWORD 004200F4	hProcess
004013B0	FF15 0C524200	CALL DWORD PTR DS:[+6AEEH&L32.CreateFile@]	hProcess
004013B6	8B45 F0	MOV EDI, DWORD PTR SS:[EBP-10]	hProcess
004013B9	6A 00	PUSH 0	hProcess
004013BB	8B40 F4	LEA ECX, DWORD PTR SS:[EBP-C]	hProcess
004013BE	51	PUSH ECX	hProcess
004013BF	8D55 CC	LEA EDX, DWORD PTR SS:[EBP-34]	hProcess
004013C2	52	PUSH EDX	hProcess
004013C3	EB 68000000	CALL LOWDWORD 00401A30	hProcess
004013C8	83C4 04	ADD ESP, 4	hProcess
004013CB	50	PUSH EAX	hProcess
004013CC	8B45 CC	LEA EAX, DWORD PTR SS:[EBP-34]	hProcess
004013CF	5B	PUSH EBX	hProcess
004013D0	8B40 F0	MOV ECX, DWORD PTR SS:[EBP-10]	hProcess
004013D3	51	PUSH ECX	hProcess
004013D4	FF15 40524200	CALL DWORD PTR DS:[+6AEEH&L32.WriteFile@]	hProcess
004013D8	8B55 F0	MOV EDI, DWORD PTR SS:[EBP-10]	hProcess
004013DB	52	PUSH EDX	hProcess
004013DE	FF15 40524200	CALL DWORD PTR DS:[+6AEEH&L32.CloseHandle@]	hProcess
004013E0	6A 00	PUSH 0	hProcess
004013E3	6B 00004200	PUSH WORD 00420000	hProcess
004013E8	6B F4004200	PUSH LOWDWORD 004200F4	hProcess
004013F0	8B45 08	MOV EAX, DWORD PTR SS:[EBP+8]	hProcess
004013F3	50	PUSH EAX	hProcess
004013F4	FF15 60644200	CALL DWORD PTR DS:[+6A5F&H32.MessageBox@]	hProcess

디코딩한 부분에서 레지스트리를 조작, 파일 생성을 확인하였으며 아래 부분 디코딩 부분에서는 소켓을 열어두는 행위를 확인하였다.

기타 악성코드 등에서 사용할 수 있는 아래의 API를 알아둘 필요가 있다.

```

*****+++++***** IMPORT MODULE DETAILS *****
Import Module 001: KERNEL32.dll

Addr:0002592A hint:0156 Name: GetStringTypeW
Addr:00025918 hint:0153 Name: GetStringTypeA
Addr:00025902 hint:01E4 Name: MultiByteToWideChar
Addr:0002589C hint:027C Name: SetStdHandle
Addr:0002588C hint:00E9 Name: GetACP
Addr:00025880 hint:00EF Name: GetCPInfo
Addr:0002588C hint:00AA Name: FlushFileBuffers
Addr:000258B4 hint:0241 Name: SetConsoleCtrlHandler
Addr:0002594C hint:01EF Name: LCHMapStringA
Addr:00025502 hint:0034 Name: CreateFileA
Addr:000254F6 hint:02DF Name: WriteFile
Addr:000255F6 hint:0131 Name: GetEMCP
Addr:000254E8 hint:0019 Name: CloseHandle
Addr:000258A2 hint:026A Name: SetFilePointer
Addr:00025892 hint:02B8 Name: VirtualAlloc
Addr:00025902 hint:011A Name: GetLastError
Addr:00025874 hint:01A2 Name: HeapReAlloc
Addr:00025868 hint:0199 Name: HeapAlloc
Addr:0002586C hint:022F Name: RtlUnwind
Addr:0002594E hint:02DF Name: VirtualFree
Addr:00025842 hint:019F Name: HeapFree
Addr:00025834 hint:019B Name: HeapCreate
Addr:00025826 hint:019D Name: HeapDestroy
Addr:0002590C hint:0100 Name: GetEnvironmentStringsW
Addr:000257F4 hint:0106 Name: GetEnvironmentStrings
Addr:00025626 hint:007D Name: ExitProcess
Addr:00025634 hint:0298 Name: TerminateProcess
Addr:00025640 hint:00F7 Name: GetCurrentProcess
Addr:0002565C hint:0126 Name: GetModuleHandleA
Addr:00025670 hint:0150 Name: GetStartupInfoA
Addr:00025682 hint:00CA Name: GetCommandLineA
Addr:00025694 hint:0174 Name: GetVersion
Addr:000256A2 hint:01B8 Name: IsBadWritePtr
Addr:000256B2 hint:01B5 Name: IsBadReadPtr
Addr:000256C2 hint:01A7 Name: HeapValidate
Addr:000256D2 hint:0051 Name: DebugBreak
Addr:000256E0 hint:0152 Name: GetStdHandle
Addr:000256F0 hint:01AD Name: InterlockedDecrement
Addr:00025708 hint:01F5 Name: OutputDebugStringA
Addr:0002571E hint:013E Name: GetProcAddress
Addr:00025730 hint:01C2 Name: LoadLibraryA
Addr:00025740 hint:01B9 Name: InterlockedIncrement
Addr:000257F8 hint:0174 Name: GetModuleFileNameA
Addr:0002576E hint:026D Name: SetHandleCount
Addr:00025780 hint:0115 Name: GetFileType
Addr:0002578E hint:02AD Name: UnhandledExceptionFilter
Addr:000257AA hint:00B2 Name: FreeEnvironmentStringsA
Addr:000257C4 hint:00B3 Name: FreeEnvironmentStringsW
Addr:000257D8 hint:02D2 Name: WideCharToMultiByte
Addr:0002595C hint:01C0 Name: LCHMapStringW

```

```

Import Module 002: USER32.dll

Addr:00025532 hint:0282 Name: TranslateMessage
Addr:00025546 hint:012A Name: GetMessageA
Addr:00025514 hint:0059 Name: CreateWindowExA
Addr:00025566 hint:0172 Name: RegisterClassA
Addr:00025578 hint:015E Name: LoadIconA
Addr:00025584 hint:013A Name: LoadCursorA
Addr:00025592 hint:00E4 Name: DefWindowProcA
Addr:000255A4 hint:01E1 Name: MessageBoxA
Addr:0002551E hint:0095 Name: DispatchMessageA

Import Module 003: GDI32.dll

Addr:000253BE hint:011F Name: GetStockObject

Import Module 004: ADVAPI32.dll

Addr:00025528 hint:0184 Name: RegSetValueExA
Addr:0002553A hint:015B Name: RegCloseKey
Addr:000255FA hint:011F Name: RegCreateKeyExA

Import Module 005: WSOCK32.dll

Addr:00000003 hint:0003 Name: closesocket
Addr:00000013 hint:0010 Name: send
Addr:00000010 hint:0010 Name: recv
Addr:00000001 hint:0001 Name: accept
Addr:00000000 hint:0000 Name: listen
Addr:00000002 hint:0002 Name: bind
Addr:00000009 hint:0000 Name: htons
Addr:00000008 hint:0000 Name: htonl
Addr:00000017 hint:0017 Name: socket
Addr:00000019 hint:0010 Name: WSAPoll
Addr:00000014 hint:0014 Name: WSACleanup

```

- Kernel32.dll

CreateFile, CopyFile, DeleteFile, Movefile
 FindFirstFile, FindNextFile,
 WriteProcessMemory, ReadProcessMemory,
 CreateProcess, OpenProcess, GetStartupinfo,
 TerminateProcess, CreateRemoteThread

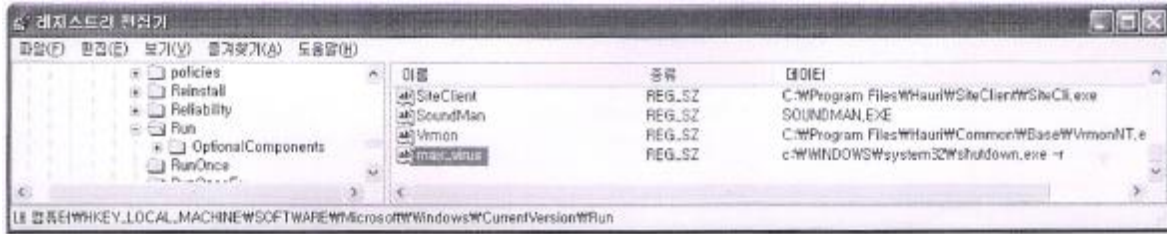
- user32.dll

SetWindowsHook, CallNextHook

- advapi32.dll

RegCreateKey, RegOpenKey, RegSetValue, RegDeleteValue, RegEnumKey,
 RegEnumValue, CreateService, DeleteService, inet_addr, htons, gethostbyname

loveme.exe를 워치럼 디어셈블 해보면 프로그램의 작동구조와 무슨 짓을 하는지 알 수 있었다. 정리하자면 loveme.exe를 실행하게 되면 C:\W 아래 max_virus.txt라는 파일이 생기며, 'Software\W\Microsoft\W\Windows\W\CurrentVersion\W\Run'이란 레지스트리에 무언가가 생긴 것을 확인 할 수 있다. 확인해 보면 계속 부팅이 되는 원인을 알 수 있었고, 그리고 소켓을 사용하여 특정 포트를 열어두는 행위를 확인할 수 있었다.



'Software\Microsoft\Windows\CurrentVersion\Run'의 위치에 max_virus라는 이름으로 'c:\WINDOWS\system32\shutdown -r'라고 값이 들어가 있는 것을 확인했다.

이제 실험용 바이러스의 실체를 알았으니 쉽게는 그냥 max_virus란 이름의 레지스트리를 지우고 max_virus.txt를 삭제하고 loveme.exe를 삭제하면 되지만, 다른 누군가는 혹시나 바이러스에 고생 하고 있을 사람을 위해 전용백신을 제작하는 법을 알아보도록 하겠다.

✓ 전용 백신 제작.

바이러스 분석을 통해 실험용 바이러스 loveme는 loveme.exe를 실행하면 C:\W에 loveme.exe 파일과 max_virus.txt를 생성하며 레지스트리의 ...

'Software\Microsoft\Windows\CurrentVersion\Run'의 위치에 max_virus라는 이름으로 'c:\WINDOWS\system32\shutdown -r'를 생성하여 컴퓨터가 재부팅 되도록 shutdown이 자동으로 실행되어 30초 뒤에 계속 부팅되는 현상이 발생하였다. 이 발생된 현상을 해결하기 위해서는 작동중인 loveme 바이러스 프로세스를 찾고 바이러스와 관련된 파일과 해당 레지스트리를 지워줌으로서 해결 할 수 있으며 전용 백신에 관한 것은 아래의 소스를 참고해 주길 바란다. 그리고 바이러스를 치료할 때에는 [안전모드]로 부팅해서 실행 시켜주기 바란다. 그 이유는 처음 실행시 레지스트리에 있는 shutdown이 윈도우에 들어가면서 작동되지 않는 이유도 있으며 일반 모드에서는 죽여도 계속 프로세스가 살아나는 경우도 있기 때문에 [안전모드]에서 작업해 주도록 한다.

```

전용 백신 소스

#include "stdafx.h"
#include <tlshlp32.h>

BOOL KillProcess(char *pname);
int GetProcessList(char *pname);

//바이러스 관련 파일 삭제 처리 함수.
int DelFile(char *path)
{
    int count = 0;
    char buf[255];

    FILE *fp = NULL;
    sprintf(buf, "%s", path);
    fp=fopen(buf, "r");

    if (fp == NULL )
        return 0;

    fclose(fp);
}
    
```

```
        unlink(buf);
        return 1;
    }

    int kill_count = 0;

    //프로세스 열이 오는 함수.
    int GetProcessList(char *pname)
    {
        int Index = 0;
        HANDLE hProcessSnap = NULL;
        DWORD Return = FALSE;
        PROCESSENTRY32 pe32 = {0};
        HANDLE hProcess;

        if ( (hProcessSnap=CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0)) == INVALID_HANDLE_VALUE)
            return 0;

        pe32.dwSize = sizeof(PROCESSENTRY32);

        if (Process32First(hProcessSnap, &pe32))
        {
            {
                DWORD Code = 0;
                DWORD dwPriorityClass;
            do
            {
                hProcess = OpenProcess (PROCESS_ALL_ACCESS, FALSE, pe32.th32ProcessID);

                dwPriorityClass = GetPriorityClass (hProcess);

                if ( strcmp(pe32.szExeFile, pname) == 0 )
                    return pe32.th32ProcessID;

                CloseHandle (hProcess);
            }
            while (Process32Next(hProcessSnap, &pe32));
        }
        CloseHandle (hProcessSnap);
        return -1;
    }

    //프로세스 죽이는 함수.
    BOOL KillProcess(char *pname)
    {
        int pid = GetProcessList(pname);

        if (pid == -1 )
            return false;

        HANDLE hProcess = OpenProcess( PROCESS_ALL_ACCESS, FALSE, pid );

        if (hProcess == NULL )
            return false;

        if(!TerminateProcess( hProcess, 1 ))
        {
            CloseHandle( hProcess );
            return false;
        }

        CloseHandle( hProcess );
        kill_count++;
        return true;
    }

    int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
        int tdel = 0, delent = 0;
    }
```

```
int kill_count_save = 0;
char pristr[1024];

//레지스트리 삭제 부분.
HKEY dkey;
::RegOpenKey(HKEY_LOCAL_MACHINE, "Software\\Microsoft\\Windows\\CurrentVersion\\Run", &dkey);
::RegDeleteValue(dkey, "max_virus");

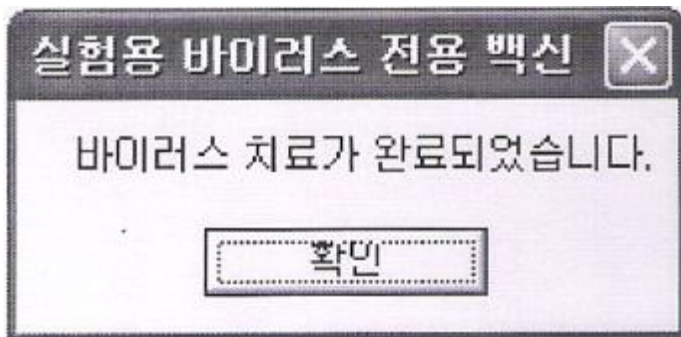
//프로세스 종료 부분.
KillProcess("LOVEME.EXE");
KillProcess("loveme.exe");

//파일 삭제 부분.
DelFile("c:\\WWmax_virus.txt");
DelFile("c:\\WWloveme.exe");

sprintf(pristr, "바이러스 치료가 완료되었습니다. Win");
MessageBox(NULL, pristr, "실험용 바이러스 전용 백신", MB_OK);

return 0;
}
```

✓ 전용 백신 실행



3. 결론

이 문서에서는 간단한 실험용 바이러스를 통해 일반적인 바이러스 현상을 분석·확인하고 해당 바이러스를 치료할 수 있는 전용 백신 제작을 제작하는 기본적인 방법을 다뤘지만 점점 진화하는 악성코드와 심지어 사용자의 승인 없이 몰래 설치된 안티스파이웨어가 사용자의 PC에 있지도 않은 악성코드 치료를 위해 과금을 물리는 양심적이지 못한 보안 소프트웨어등 인터넷에 떠도는 검증되지 않은 소프트웨어와 파일 하나하나 여는 것에 대해 주의하길 바라는 의미에서 쓰여졌다. 차후 이 문서에 더하여 바이러스에 감염된 파일을 바이러스로부터 복구 하는 기술과 간단한 몇 가지 바이러스 패턴과 스팸 메일을 네트워크에서 걸러내는 기술에 대해서 다루도록 하겠다. 우선 이 문서를 통해 통합 백신이나 안티 스파이웨어 등을 만드는데 약간이나마 도움이 되길 바란다.