08.Debugging with W32Dasm, RVA, VA and offset, using LordPE as a hexeditor

2012년 1월 28일 토요일

오후 8:27

Hello everybody.

모두들 안녕.

Welcome to this Part 8 in my series about reversing for newbies/beginners.

나의 초보자 reversing series Part 8에 온 것을 환영해.

This "saga" is intended for complete starters in reversing, also for those without any programming experience at all.

이 "saga"는 완벽히 reversing 초보자를 맞춰서 만들어졌다. 또한 어떠한 programming 경험이 없어도 된다.

Lena151 (2006)

Set your screen resolution to 1152\*864 and press F11 to see the movie full screen !!!

Again, I have made this movie interactive.

You screen 해상도를 1152\*864로 설정해 그리고 full screen으로 movie를 보기 위해 F11를 눌러

So, if you are a fast reader and you want to continue to the next screen, just click here on this invisible hotspot. You don't see it, but it IS there on text screens.

그래서, 네가 이것을 빨리 읽고 다음 screen을 보고 싶다면, 보이는 hotspot 여기를 눌러. 보고 싶지 않을 때는 여기에 두지마.

Then the movie will skip the text and continue with the next screen.

Movie는 text와 다음 screen을 skip할 수 있다.

If something is not clear or goes too fast, you can always use the control buttons and the slider below on this screen.

무언가 명확하지 않거나 빨리 넘기고자 할 때, 항상 control button과 이 screen 밑에 있는 slider 바를 사용해.

He, try it out and click on the hotspot to skip this text and to go to the next screen now!!!

도전해봐. 그리고 이 text와 다음 screen을 보기 위해 hotspot을 click해.

During the whole movie you can click this spot to leave immediately

이 movie 어디에서나 즉시 떠나기 위해 이 spot을 click 할 수 있다.

1. **Abstract**

In this part 8, we will reverse a "real" application to learn something about working with the disassembler/debugger W32Dasm and LordPE as hexeditor, while studying the reversing of a registration scheme.

이번 part 8에서, Reversing의 등록 계획을 공부하는 동안 우리는 뭔가 배우기 위해서 disassembler/debugger W32Dasm과 LoadPE로 "real" application을 reverse 할 것이다.

That is because indeed, the best practice is found in real applications.

Real application에서 최고의 연습을 찾았다.

For better comprehension and if you are a newbie, I advise you to first see the previous parts in this series before seeing this movie.

더 좋은 이해력을 위하고 네가 만약에 초보라면, 나는 이 movie를 보기 전에 이 series의 첫번째 part부터 보라고 조언한다.

The goal of this tutorial is to teach you something about a program's behaviour.

이 tutorial의 목표는 program's behaviour를 가르치기 위함이다.

In my search not to harm authors, I found the program ArtGem v1.1 which was discontinued in 2003 and not downloadable any longer.

나의 연구는 제작자에게 해가 되지 않는다, 나는 ArtGem v1.1이 2003년 이후로 더 이상 download 되지 않는 것을 찾았다.

Taking a look in the specialized media, I also found this application to be "cracked" already.

특화된 media를 가져봐, 나는 또한 이 application의 "cracked" 된 버전을 이미 찾았다.

Here, this application is only chosen because it is ideal for this tutorial in reversing and it is targeted for educational purposes only.

여기, 이 application은 오직 선택됐다. 왜냐하면 이것은 이 tutorial은 reversing과 교육적인 목적의 target으로 이상적이다.

I hope you will exploit your newly acquired knowledge in a positive way.

너에게 네가 얻은 새로운 지식을 긍정적인 방향으로 사용하기를 바란다.

In this matter, I also want to refer to Part 1.

이 문제는, Part 1을 참고하기를 바란다.

Set your screen resolution to 1152\*864 and press F11 to see the movie full screen !!!

Again, I have made this movie interactive.

You screen 해상도를 1152\*864로 설정해 그리고 full screen으로 movie를 보기 위해 F11를 눌러

So, if you are a fast reader and you want to continue to the next screen, just click here on this invisible hotspot. You don't see it, but it IS there on text screens.

그래서, 네가 이것을 빨리 읽고 다음 screen을 보고 싶다면, 보이는 hotspot 여기를 눌러. 보고 싶지 않을 때는 여기에 두지마.

Then the movie will skip the text and continue with the next screen.

Movie는 text와 다음 screen을 skip할 수 있다.

If something is not clear or goes too fast, you can always use the control buttons and the slider below on this screen.

무언가 명확하지 않거나 빨리 넘기고자 할 때, 항상 control button과 이 screen 밑에 있는 slider 바를 사용해.

He, try it out and click on the hotspot to skip this text and to go to the next screen now!!!

도전해봐. 그리고 이 text와 다음 screen을 보기 위해 hotspot을 click해.

During the whole movie you can click this spot to leave immediately

이 movie 어디에서나 즉시 떠나기 위해 이 spot을 click 할 수 있다.

1. **Tools and Target**

The tools for today are : W32Dasm, LordPE and … your brain.

오늘의 tools은 : W32Dasm, LordPE and 너의 두뇌

W32Dasm was a program developed by Ursoft and sold as shareware.

Ursoft에 의해 W32Dasm은 발전됐다. 그리고 shareware가 판매된다.

It was once a beloved tool for many reverser but was passed by other tools years ago.

많은 reverser가 사랑하는 도구였지만 몇 년 전부터 다른 tools에 의해 사용되지 않았다.

It would be a shame not to show some work with this program as it can still come in handy occasionally.

그것은 아직도 가끔 유용하게 사용할 수 있는데 이 program과 함께 작품을 보여주는 게 부끄럽다.

The company no longer exists but the program can still be found at

Company는 더 이상 존재하지 않는다. 그러나 program은 여전히 찾을 수 있다.

<http://www.cracklab.ru/download.php?action=get&n=MzA=>

REMARK: this is the patched version, or else you won't be able to debug with the program!

주목: 이 patch된 version은, 네가 원하던 program과 함께 하는 debug가 아닐 수 있다.

LordPE is a free tool and can be down'ed at

LordPE는 무료 툴이고 여기에서 다운 가능하다.

<http://www.cracklab.ru/download/php?action=get&n=MjA=>

Again, the brain is your responsibility ;)

다시, 너의 두뇌는 너의 책임감이다.

Todays target is a program called ArtGem1.1 Because it can no longer be downloaded, I have included it in this package for research.

오늘 target program은 ArtGem1.1로 불린다. 왜냐하면 그것은 더 이상 download 되지 않는다, 나는 이 package에 연구를 위해 포함했다.

1. **Behaviour of the program**

What you see here this time is a lot different from what you're used to see. Indeed, it is W32Dasm that is opened.

이 시간에는 무엇이 보이나? 네가 지금까지 봐왔던 것과 꽤 다를 것이다.정말, 이것은 W32Dasm을 열어놓아서 그렇다.

W32Dasm is basically a disassembler (also called a deadlister because it makes a "deadlist") but it has some nice debugging features too. Let's load and disassemble ArtGem first

W32Dasm은 기본적으로 disassembler다.(deadlister라 불리기도 한다. 왜냐하면 그것은 "deadlist"가 만들었기 때문이다.) 그러나 이것은 괜찮은 debugging 특징을 가진다. ArtGem을 먼저 Load 하고 disassembler 하자.

시스템 생성 대체 텍스트: 騙uRSottw32Dasmyer'op魄ram脯sassemblerlDebu鱗r■■■■■■■■【ㅍㅍ그
Usas,em벼erEditPrO)ectDebugSearchGOtoExecLJteTextFtjn功onsH.'Da出Ref눙Heb
퍼멥聊뾔！繃닒찌釗＿鍋〕〔巳찌啣釉‘咽壓」L」。쪄이
DiaasseZ亡》ly。frile=Ar乙Gem-슬xe
L꽈
r'-
-- --
00001ㅇ00'CQdeSize
0006DOO아Data512·
000'9000
00007000
-- --
COdeㅇff의e乙
D‘헙‘侊fa슬t
N냅祉》er。fob〕ect..00（》4(dec)'I교ageb',.=()()40000Oh
恥끄느ct01:.t.xt
Ob)ec℃02=.rdaLa
曲）ec'03:.da℃a
物）ec乙04='rarc
효VA:00001000ㅇffset:氓》001000Size:0006900OFlaq의:C000002O
묘VA:000이노000ㅇff,e:r()00히뢸曲OSize쵸0000300OFlag홉:40000（》40
묘VA:0006D000ㅇff듸e乙:000'D000Size호0000700OFlaggrC000O()40
요VA:OOQA10O()楗fae乙쵸O00'74ㅇO()Size:0003500()Flagar40000040
+++++++++++++++++++XZNUINFㅇ요XATIㅇN++++++++++++++++++
ThereA=eNo뵤enu묘e，。u=c솥，inThi의샤mlicati。n
+++++++++++++++++DIAIDGINrㅇ묘XATIㅇN++++++++++++++++++
Th.r.A＝슬N。Dial。g요츨，。urc。＇
inThi.쟈》plica헙i。n
+++++++++++++++++++工MPㅇ요T크D
NUI祉’er。fl꺽Por七ed뵤。dule,-
rUNCTIㅇNS++++++++++++++++++
7 (de。功‘'L-
I쵸쑤。rtH。dule001:
I2聊》。r:H。dule002:
I1聊沁r乙쯔이iule003:
I죠鬱ar=Module00샐:
I2먀》。r乙Moclul즐005:
I찌por℃MQdul.006:
IZ.por七MOdul.O0'7:
XZ묘NZL32-dll
US꾜요32
GDI32.
c。mdlg
紊
Asyㅇucanse윅W32Dasmhasdoneagood
」obandshㅇwsustheres미tsstartingwith
dll
ADVAP工32_dll
SH크LL32.dll
C（羚鍈rrL32_dll
somegeneralinformatiorlfoundintheheader
oftheexeandal-theimports.Afterthat,we
canfindadisassemblyofthecodesectiㅇ『ㅏ
Let'5haVeabriefIOOk.
+++++++++++++++++++IMpㅇ묘T》粲）r)UL므DZTAILS+++++++++++++++
I끄따。r乙M。dule001rXE勵궁EL32.dll
Addr:0006BF42hinL《OO3F》Nam슨:Cr.'t츨Mut슬xA
Add=:0006크F32hin:《OllB:.Name:C。tL。。a1Ti쿄e
Addr=0006BFIAhin헙《OZSD)Na.m.:S솥t毓‘rr·ntDir·cL。ryA
Addr:000'BFOAhin헙《01'D)Na교e=C슨七TickC。unt
Addr:0OO6SEFChin乙10181》Na교e=Gl。ba끄노11。c
Addr:00O6BEZOhin乙별OOB4》Na脈e쵸FreeLlbrary
ㅡ삐삐∼
써－
■■■■■■■■■뻐

As you can see, W32Dasm has done a good job and shows us the results starting with some general information found in the header of the exe and all the imports.

네가 볼 때, W32Dasm은 좋은 능력을 가졌다. 그리고 exe의 header와 모든 imports에서 약간의 시작하는 일반적인 정보를 찾은 results 보여준다.

After that, we can find a disassembly of the code section.

그 후에, 우리는 disassembly의 code section을 찾았다.

Let's have a brief look.

간단히 보자.

시스템 생성 대체 텍스트: N』山》erofㅇb」ect,.0004《dec)'I쿄쵸geba,e.OO40000Oh
曲기느ctOI:.text
Ob)ec七02=.rda七a
山〕ec乙03=.da℃a
飽）ec乙04:*r'rc
효VA:000OI000Offset=0000100（》512·r00069000Flag의:戚00000釵》
묘WL000덞노000ㅇffseL=凹0石A000Size:0000300OFlag,:40000040
효VA:0006DOO()楗f,e℃=OO06DOOOSize=000070ㅇOFlag으=C0000()40
m礖쑈OㅇOAIOO()Offae乙흐00074000512슬:00035000Flag,:40000（】40
+++++++++++++++++++X크1ㅍy工IJFC7XAT工얘++++++++++++++++++
SOmeirlfoabOUt
theSeCtiㅇns
The:eAreNO브enU묘e,OU:Ce,inThi,ADDliCa乙IOn

Some info about the sections

시스템 생성 대체 텍스트: +++++++++++++++++++IMpㅇ묘T》杖）DUL흐DZTAILS+++++++++++++++
andtheimport$
I2聊》。r乙贓。dule001rX뜨勵｛EL32,dll
Addr:0006BF42hin'《OO3F》Nam츨:Cr슬‘乙습贓u헙．xA
Addr:000'BF32hin헙《OllB》NaㅠLe=Ge乙L。c'IT劾．e
Addr:0OO6BFIAhin七《OZSD)N∥e:S솥LCurr솥ntDir.cLOryJL
Addr=（】00'BF（】Ahint《01'D》N·飇rC솥七TickC。unt
Addr:0006B흐rChin乙뱁0181)Naㅍ階:Gl。ba끄노11。e
Addr섰》0O6BEZOhin乙(OOB4)Name:FreeLlbrary

And the imports

Let's take a look and scroll down

밑을 보기 위해서 Scroll 내려봐.

시스템 생성 대체 텍스트: TheStartㅇfthe
disaSSembledCOdeSeCtiㅇn
:l」U'U술UUt》 yU
::l'94ㅇiㅇOC90
:004ㅇ1ㅇO:9:
:004ㅇ1ㅇㅇEgC
즈。0401ㅇ隧…90
nOP
nop
nop
nop
nop
*묘．f솥r·nC츱d
-:）돗a1oㅇa
bya
cALLatAddre므드
〕）뢸醜01그
pugh

The start of the disassembled code section

Disassemble된 code section의 시작 부분이다.

All right, let's now study the application and some features of this debugger together

좋아, 이제 application과 debugger의 특징을 함께 공부하자

시스템 생성 대체 텍스트: W32Dasm
Deblj麗eroptions
Enab논COmmandUneonLQ盼d
Debugon沙ThisP『ocess
Di펴斗Pr어ramGenelatedExc印츨眈ㄶ
EnabbsleakonDeateProcess /
LnaDIebreaKOnLX.rrOCeSS .
LnaDleb!eaKOnLOaOUㄴㄴ 노
Ena바e8reak(mUnL。adDLL
EnabbBreah勵Dea毓eThlead
Enab始Break예Ex鷺Thlead
陽H隨隨楗
Debugger
야）tiㅇns
…그[-。
rrrrrL
Cancel

Debugger options

Debugger options

시스템 생성 대체 텍스트: －、
W32脯smLIStofstringDataltems
巨」回因
TOSearchDisasse社》lyf。rStringData,D이lble
(((
(%dx%d)'
一1"
…』■■■■■■■■■■탭■■■
」『
YOUCar1SearCh
thrㅇughal-thestringS
券bOOI
轍mk
ataα
lcl。치
copyAll1coPyyi히

You can search through all the strings

너는 모든 string을 통해 찾을 수 있다.

시스템 생성 대체 텍스트: 己컨’탬SearCh
taRef눔Heb
LOadPrOC한｀'
A勵toan勸VeProceSS俱
녀0
Seethatyou
canalsoattach
toanaCtiVe
proCess,justlike
in이ly(seelater)
긔nglebtepInto
,IOna그 _ 굳→tAddr슨드=
ZO rJ
F F
蝕傲》Singlest印Thru
AutoSinglest印Im。
IaO
:3D
"＾一1111-.궈수'.r^'－。（. 炸
決恥卯erOPbonS··

See that you can also attach to an active process, just like in Olly (see later)

너는 또한 첨부한 active process 를 볼 수 있다. Olly와 비슷하다.(나중에 보자)

:)

시스템 생성 대체 텍스트: 「에raln【》isassemblerIDe抛驪e딥r
Search Goto b(eCUteText Fun土Ion'
꺼：IEIP:Jㅒ6OE79isin汕dule:Artt놂m·exe
Hl고츱g' 됴“4'
Fm놀
"
細
繃l騙11711쁜l繃―낢
Li仇。죠간팅r7屎：=d功lanL》驪r"
re헙
gra쿄助、七ryPOin윕'*..*''*
push른bp
m。vebp,e,p
pugh-一一E万f[-
pughㅍ-一EB二[
pu,h-'04石긷:-른-l
m。veaX'd낚。rdpLrfs=【0
pu,heax
ea」L=00000000
.bx='ffd·000
eCX==OO12ffbO
EDX·7CgOEB94
e의1-004·’O'7
edl=OOEbb쨌'O
e야尸OO12fffO
eap==()O12ffc4
株杜흖枓태’'=:：麗；縣
「saurc.鞠rD."Di.PI 드쁘외
箋鬪［ :,'
·R．·騙·＇
술P．·5..
.t$Tkr·
U’츨r漱軟늅1
00400000 '
'-k.
vPJ.
U'OrAddrZ
【e의P+0000000']一O(）石bb7'O
【eap+00000005]一00섶a7077
[eaP+0000()OOCI一7ffd슬000
【·，P수凹OOOO1이-eOS펩a6。d
r..＂수nnnnnnl'1 - nn,찻fF尸층
」
S슴丁．·n七효츨g
匿
Bp',c。pylCI.arf
I Nonese'씨
隧7뜨틔助d.-＞陣矗궁즈뜨의」즈의으뾔
…
→l
ㅜheinfoscreenfrom
SOurcerorDa윕aDispZ
cop피
ACtiV.
씌
DLL,c。。vI
thedebuggerbringsus
notonlygeneralinfo
butalsospeCificillfo
abouttheregisters
군＇ 「■－
IDis[1
臘1ㅣ
ㅁAZ-
야准r-
漱idre의，:,C81'D4r
char【에1】:"P"
鱗7ㅇ요r):SfS4e85（》,
COD=rDU,heaX
iainMOdUI슬: kerne132_dll
낚醱D:2850'BYTE:50
닙룁ㅗㅗ
XOr
죠놈OV
표（》V
InCV
and
죠놀OV
,hl
add
끄노OV
ghr
ZnOV
dvordp乙r[
edx,edx
dl,ah
d낚。rdptr【-
u,b＾ㅗ28
Tr0001pr。ce,,OPr。qran'Entrvp。Int.·lp:004'Oe기
EvOOISLO·dinqDLLC（껴CTL32·DLLO·ddr:77390000土．
t。Curren℃Elp
Am杻JI32_dll^
C（갸仗江L32.dll
r-r＇너1＂ㅊ얏ㅐ11낟
PrOCCroa乙．Brk
pr。c크xltBrk
ThrdCrea乙eBrk
ThrdZXitBrk
Dl.LL。adBrk
力LLUnl.QadBrk
겯겨
ㅗ·■■■■■■■■■■■■■■
늬
드1'edx
eCX'
eCX'
d힁rord
eCX'
eCX'
d낚Ord
eaX'
d낚Ord
e룔1'
.의1
eaX
00000OFF
p헙r【에49F'1SC】,
솥CX
:004'OE7CPu,hrFrrFrrF
:00춈r》E'!pu,hOO4'B168
:()O춈‘OES3Pu,h00465크BO
:004'OE88죠느OveaX, dvordD七rf의=【000000001
.
edX
ptr[O49F':
p헙r
e51
1'ecx
1,eaX
En'bl슴DOCU끄힙OnL츨dA꼬ID·헙all.
Enabl츨UnD。cum.n윕슬dAPID슬tall,
ZnableLoc'1runc=i。nDe:all. 一
StopAutOOrlApl
5℃epIn℃。"rep"In의truetionBypa，의
GOtOAddr츱·'
pa乙ChCOde
견므뾔
FLrLL
XOr
pu,h
C.11
pㅇp
t·,t
끄노e
pugh
UV숑b요ㄴ‘U
.CX
·aX. eaX
J4b
O〕
笙희笙뇌蠟뙤
-'-r？딥r놈r꺼』r'r놀「■무r씸7p딩r놈't=-'r노r놈r놈r｀궂눔「쨍f널－ :- 『·：·- 슈·1「‘--- -

The info screen from the debugger brings us not only general info but also specific info about the registers

Info screen은 Debugger에서 오직 일반적인 정보뿐만 아니라 특별한 register에 대한 정보들도 가져온다.

시스템 생성 대체 텍스트: L냥Z노 r노
Let'5haVealoㅇkatthe
COmmanCISCreerl.
ThisscreenalsodisPlays
theCOdearOUndEIP
r『000000001
En큅bl츨
Enable
UnDOCUn뇨·nL슬dAPI
u의
LOCal
COtOAddreS,
StopAutOOrl
StepIn℃O"r
rUnC乙1OnD숩乙ail
Apl
pa헙ChCOde
·p"
Instruc七1。nBypa,,
FrrLL
筮희笙희
쬐뙤二프

Let's have a look at the command screen.

Command screen을 봐.

This screen also displays the code around EIP

이 screen은 또한 EIP code에 대하여 보여준다.

시스템 생성 대체 텍스트: -“닐뿌끄‘」汕느→으乙느
급孕닐돌갸뜨b '
―一戚뻐－천므쯔」二ㅋ긋네
COd.Addr슬되흥 :Oㅇ4EOZ791의inMOdUI슬Ar乙G슬죠〔＿exe
"나：ㅍ炤5
.
b
-－터l
」
"O맨SOE6FandedX,d낚。rdp乙r[ecx+O시
竅》O춈‘OE'2醜따ed='eax
:OO460E74〕e00씰‘OE,8
른=솝따
:00흩‘OE76x。r
→-二004602伊8ret
"O。4'O민9＿됴」맨
－一一＝00습‘OE7A．。v
:O()'6OZ7CPu,h
eaX' 슨aX
LOa도
if,Dal1.
push
pu,h
끄노OV
‘으므르...._
·꺄’'·’p
FFFrFFrF
004'Bl'8
004'5크BO
.aX. dVOrdDLrf의그『000000001
E3B
788
EEE
000
'''
괘멩 괘탭걺쨍
000
000
……、－·
치ㅡ
늬
7nabl츨DOCu=.nt즐dAPID슬乙all'
Enabl츨Ur1DOcuㅍ｀.n乙．dAp工D.tall,
ZnableL。calrunc乙IOrlDe乙츨11, we
S헙OPAu업OㅇnApl
SLepIn℃。·rep"Instruc仁ionByPa,,
COtOAddr츨．,
pa七ChCOd손
FrrLL
NotiCethat이Iyuses
thesameshortCuts:)

Notice that Olly uses the same shortcuts :)

Olly는 같은 shortcut 사용하는 것을 보여준다.

시스템 생성 대체 텍스트: r一
Addr슬aa 죠Oㅇ4石02791요in쯔OdUleA：乙G슬끄｀_eXe
:00흩SOE'F
2004'OE72
:00흩‘OE'4
:（》04'OE,'
deF
Ca희
섭
二 ee솝'
뾔
孤길心鞠嵋’e'
:004'OE79
乙00춈塔OE'A
:00쐽6OE7C
:004'OE텼E
:00춈'OEe3
:OO46OE88
<
andedx'dvordp헙r【ecx+041
醜聊》edx'이‘X
기’004石OE'sr: ＇·엥 '
XOreaZ' eax
J궁·＝··· ”∼'::-
．。v.bp,츨，p
pu,hrFFFFFrr
pu.hOO46B168
pu,hOO46S크BO
죠驢OVeaX. d냠OrdD乙rf의r【000OOUUUI
Znabl습
En즐b1o
Znable
DOCU죠L·nL즐dA꼬1D.ta쵸1,
UnDOCUn느·nt츨dApID슬tall의
○k.Letlsstudy
in
LOCal
stop
S업ep
AUtO
ㅇn
"r
rUnC乙1OnDe乙ail요
Apl
theapPlication
thedebugger
In乙O
ep"In,truc乙i。nByPa,,
Te
늬 FLFLL
윕e
笙희笙희쬐쬐E드

Ok. Let's study the application in the debugger

Ok. Debugger에서 Application을 공부하자.

시스템 생성 대체 텍스트: 一嫩隘T蘇麻欌痲7一→길→
!Firstscreen15anagscree『1.
bringsususefu∥irlfoalready
dw'
'ZX크mEL32·C＊乙V.r,iorl,ㅇrd:
-
'2BA14'00 Calldy
XOred뵤
m。vdl,
EEF〕긷〕EE 끄戚》Vd낚O
쵸노C'I 슨OX
딩
→잇乂빵．
UNREGISTEREDIR티키STERNOW!

Aha! First screen is a nag screen.

먼저 nag screen이다.

This brings us useful info already

이미 유용한 정보를 가져온다.

And after some seconds, the main screen appears

그리고 몇 초 후에, main screen이 보인다.

:)

Mmmm, so the nag screen is also the about box screen !

음, about box screen 또한 nag screen이다.

:)

시스템 생성 대체 텍스트: O
痲倆츱un楸le樞＞
.4미冒도
En略【『egi샀lalior∥nf이mliontour"acl..exp
「】‘」
「7ㅍF＿一」＇r맥
L쓰쁘뜨7논
．·■뺨
.
-딩-수
·
- 〔
누E乙컁의캔〕拭
-
一
亡
수·
-
l■빽
R쓰r한콱斗.
lrLJ즈．JJJ쐰으l
' ㄻ삐ㄶ’m묍습1
- WlaOI■·11
일 '
I ■
수-一- ㅏ
Theregistratiㅇnboxwhich
硼溢慈
뉩·slzs?oouo:끝빽
I I
■ l
■ I
므ㅇ=·"仔·
IhavealreadyexPloreda"ttle:)
腱藪SterArt（놂m
.一려∼논
\;
Name:
以》mpar汐：
圓纘巧기
ier■a
]J꺼r그Jj ·
K톄－Code:
-11．· ._
「石「〕cancel-
。그、수그]
r/
0「）a이t
■■
■■

The registration box which I have already explored a little :)

Registration box는 내가 이미 약간 탐험했다.

All right. This should give us enough information. Let's go!

좋아. 이것은 우리에게 충분한 정보를 주었다. 시작하자!

1. **Finding the patches**

Well, indeed, why not see what the registration looks like. Let's see it …

좋아, 우리는 왜 이 화면처럼 등록된 화면을 볼 수 없나? 보자.

시스템 생성 대체 텍스트: 軾口
>
ㅓ쁘·
ㅁ
戚Gem쐰
《｀「】2
그Untitle교뺨
볍치 돌『
En＇이feg미latloninform副ion山
뱀·
뱁．괘
·
ur"a하｀exp
/f
' '
수-nn傘
' 1
뱀」 …』
蕣2
倉珊〃짊
긱－[
·
·
·'一-f
yㅁ含』力'겹孑
amlal’枇’ng쑈」
-■■．닐ㄴr겯 L티禮L티큔ㄴr. 니＇· ㄴ티큔
- e Ln 괘U e e ·nV
·lz 이 In 시냇 嶼 륵ln
휘 표 U 기· 겠 ·:l
' O 기 "" 수。 C
- n r( a S a
糾 - e nP
"1·■．·I “톄 수수
■■■익r■·■ n논 ··
-긔
,---'
. ■
n ■
W32Dasmhasastring
SearChfUnCtiontOO!
。一→수
Name:
:(
C。mpar珊：
0
Inv혀id之yl
·1]仍
t수·t·∼
칭 ∼
K톄．C出e:
CanCel
'…中。。찌
- ㅁ L豚―
'ㄷ仇IH백刑니귀미
匹긔
r---
〕「
.

:(

W32Dasm has a string search function too !

W32Dasm은 string search function이 있다.

시스템 생성 대체 텍스트: 脾
뺑쟁
1n
一 -- ■■■■■■■■■■
= :b.r E즈r·：' l勿닌D
、 I^ 밸＾ I「꿸「F’샐『＝7一
工희－그새盼r치·…비■·溜l뾔
r : 나；소。nalOr
늉》
OF·
O없놉
OH.'
」，l됴큐
ㅏ랙,．■뉴
=：성ㅗ낌소Or-드1
仇珊p
reL
//..''*
praqraZn助1七ryPOin七'.'.*'*'
:00쨍
=004
:00'
:ㅇㅇ4
쌔더
F耐鹹aITI．톄
Sealchin
仔AIl
I^-mPortrefefence
Cancel
L O小elrelerence
n티lporlrefefence
J一Matchcase
Di『ecUon
Up盧Down

시스템 생성 대체 텍스트: 00''OE-
BadneWS?
/.*''*'*'*'.*'*''.**.prog:
''.'
004
004
004
00'
DO4
T텨αpnvalid心y!1NotFound
Fh小＂ha｀·--nV하idK砂
「searchin
-－『녹
回幟
'＇꽃
004
004
004
004
Oㅇ4
004
盧AIl 「며脯『reference
「ㅣmPortrefe『encenExporl'efefence
CanCel
l「M副chcase
Di『e이ion
-＿텔P「D。wn
NolW32DasmalsosearcheSinone
direCtiorl(like이ly)andbecauseWeare
herealmOStattheendOftheCOde
sectiㅇrl건etls『lowsearchupwards
깻［찬쨋．d驢한rdptr

Bad news ?

No ! W32Dasm also searches in one direction (like Olly) and because we are here almost at the end of the code section, let's now search upwards

아니야! W32Dasm 또한 한 방향으로(Olly와 같이) 찾는다. 그리고 왜냐하면 여기는 거의 code section의 끝이다. Search를 위쪽으로 해보자.

시스템 생성 대체 텍스트: 즈」닐。 』텬｀뱀뿡』■‘ 』‘』닙L』‘卜‘ 』｀7탭｀卜＇탭．
｀닐＾』」
T'e:Xt「bUndOnUn6:109150
―囹찌
Fh小시ha【：--n·I·．」―.1」，하·’
－、 a 4 괘펭
· 't 씩J힁」
R·
Sealchln
盧AlI
尸ㅣmPortrelef레ce
CanCel
104
10'
fO4
「미糖l『eference
nExporlrefefence
「MatchCa$e
「Dire햐ion
효·
回「D。硼

:)

시스템 생성 대체 텍스트: Z【jRSoftW32Dasmver
USaSSefn벼er ㅌdit
pro청쥰杜
아ect
1《》Program【》isassemblerl【》ebug伊r
DebuOSearchG（嫩》ExecuteTextFl.rCbort드
半암
驪
締눠뇌뉘一
R￡’
.
>'
마h
n없」戮’硼驪J
H티섭배∥，-
同눕
체溜
댜나놈
J매
(9ㅇ‘38石O' Er
pu듸h
죠O·10
'p。，，노bl.8헙r세gData묘effr蛔DataOb쵸－》＂ArtCem"
X》0435'09'SOCF94'00
l
pu의h-）쵸石rgOC
r。＇slbleStrin7Data
】：익曆L
只effra크r
：版덕j그스ㅁ易
:OO438E그E E드〕IF946그〕
:0043:1-
〕ata
－》"Invalid
戚믄y
묘솥fe:enceTO=US크요32．쯔츱，,age3이‘A,ㅇrd:OIB크h
I
:uu，。．bㅗ풀''ㅗ。＾나씽ㅗ쨍勿uu Ca』ㅗdvar。P乙r
:OO43861A〔A01 Pu의h
:OO43361C53 Pu,hebx
uu'""'^'-

:)

It is rather crowded here. I won't need the register info screen right away because I always take an overview first. So, I'll put in away for better visibility ….

이곳은 꽤 복잡해 보인다. 나는 옳은 방법에서 register info screen이 필요하지 않다. 왜냐하면 나는 항상 먼저 관점을 가지기 때문이다. 나는 좀 더 좋은 방향으로 놓을 것이다.

시스템 생성 대체 텍스트: '드굿E그-
pu듸h
p。，，ㅗbl.s七r以qDataP슨ffr蛔Da헙aQb그－》－표r乙C이n"
·닙틀닐'닙큐O（큐닐」
（걍09、＇'J7빽맣｛팝닐닐
Strin7그ata
E깃rCFg4600
pu,h
니닙쨍밍쫠'J닐짐＇
二죠i-r石〔L
: :ra크r
：蘆ㅍ‘
Data
工묘스n료
' raSSible
Cb〕一》＂Invalidkey'I'
맬j e
고
그00438‘旺
〕）료33'13
pu의h
push
기46F9C츠
eCX
'묘솥fe＝·nc솥T。：US크R32.M·，，翩구eB이‘A,ㅇrd섰UB크h
勵ght.Pressdownbuttㅇnㅇnthe
keypadtoscr이―downandtakea-001<.
' p즐f츤,'n．수T。· TT且쩟묘且그7ndni츨1＾。 Ord·On효댜h

Right. Press down button on the keypad to scroll down and take a look.

좋아. Scroll down을 위해 Keypad에 있는 Down button을 눌러. 그리고 봐.

Study the code.

Code를 공부하자.

시스템 생성 대체 텍스트: Calld낚。rdp첩r[
ebX
d낚。rdp℃r
]'
n h
= ㅌ V
그 U 門1
”上m
Thismea『15thatthisPieCeof
Cㅇde15reaChedfroma
conditiㅇnaljumpatVA438578
'h
。rdptr[O춈石A.」「】
」3CO
5B
Here15aretUrn.
r뜨 slcl』盼04ㅇC
e크꾸’
」
프큔FE CZ'』DO
COdeAddrea듸:Oㅇ4602791흥in눴
Studythecod.
」
:《】04'OE6F·nd
:O()4'OE72에뿌
一一요OO460E74〕e
:00460트，6x。r
:OO460E7Bre乙
·dㅈ－
edㅈ，
d힁＇Ord
솥aX
OO46OE'8
ea표＇ eax
:Oㅇ439塔叭…크B.:IDA8494'00
:曲438E:l'7 E므스:
. po,'ibl.
YButhowdo눅t
:00'3a609姃（wegethere구,p
...5ㅇwegetherefrㅇmthis
conditionaljump(a-ways
keepaneyeonthe
jumps,espeCially
theconditiㅇnal」umps!)
'p。의，lbles七riㄲgData꾜．ffr帥DataObaee》”InvalidXey'"
-
: 뜨프[EEp「 ==ㅍ『「 』 DU의h
j Z' DU의n eCX
'/，… --- --
무■■．샐요涵뀀〔■념■미볍댐■·볘■■■■■■■
표》04石OE7Am。v·bp,츨．p
:004'O트7Cpu,hrrFFFFFF
:004'OE'EPushOO46B168
:00흩‘OE83pu,ha（》465크BO
:00쨍6OEBB휩L。veaX. d낚。rd
<,.-」■■·ㄴ ．볍．ㄴ'
크nabl.DOCun'.nt츨dAPIDO乙al
En쵸bl츨UnD。cuz．츨nt츨dAPID.t
EnableLOCalrUnC七IOnDe乙al
S헙OPAutOㅇnApl
S七epIn乙。"rep"In,tructlㅇn
FrrrL
TheprogramexecutesthisPieceofcodeto
pushthelllnvalidKey!''stringonthestack
:OD·
】 l墨i笙히
St·
Int
F'
pu,hebx

The program executes this piece of code to push the "Invalid Key!" string on the stack

But how do we get here?

Program은 이 조각 code들은 "Invalid Key!" string을 stack에 넣는 것을 실행한다. 우리는 여기에 어떻게 올까?

Here is a return…

여기 return 이다.

… so we get here from this conditional jump (always keep an eye on the jumps, especially the conditional jumps !)

… 그래서 우리는 이 조건 jump에서 여기를 얻을 수 있다.(항상 눈을 jump에 유지해라, 특별한 조건 jump !)

This means that this piece of code is reached from a conditional jump at VA 438578

Hence, we need to go to VA 00438758 to see why we jumped here. So, scroll up

이이 조각 code들은 조건 jump VA 438578에서 여기로 이르렀다.

그리하여, 우리는 VA 00438758를 보기 위해 jump 하는 것이 필요하다. 그래서, scroll 올려봐.

Huh? So, we jumped over the goodboy !!!

음? 그래서, 우리는 goodboy로 jump 했다.

Still better news. Scroll up to the cond jump …

좋은 뉴스다. Scroll을 조건 jump까지 열려 봐.

There we arrived.

우리가 도착했다.

This is the cond jump to the messagebox

이 조건 jump는 messagebox로 jump 한다.

Proceeded by a

TEST EAX, EAX

Which decides on jumping or not !!!

진행하다.

TEST EAX, EAX

이것이 jump할지 말지 결정한다.

… and I suppose you already understand it's all decided in the proceeding call ???

이 진행중인 call에서 이것이 결정한다는 것을 너는 이미 이해했다고 생각한다.

Press F2 to set a BP exactly like in Olly(Shown in yellow)

BP를 set 하기 위해 Olly에서 처럼 F2를 눌러 (yellow로 보인다)

Because we want to go look IN the call what sets EAX and makes us jump to the badboy !!!

왜냐하면 우리는 이 call에서 EAX를 무엇이 set하고 badboy로 jump하는지 보는 것을 원한다.

시스템 생성 대체 텍스트: :OO43B55E
:卯439S石2
:⑩43SS'3
호凹438S64
:⑩4385'9
:OO43856D
:·〕。〕4385〔2
eax'd날。rdp乙r[esp＋그프]
edx
eaX
Ob4‘曲5O
·。x,d낚。rdptr【e,p+'배
eCX
:ㅇO
385'3
版5,6
8D44〔1[_
52
SO
E8E78202ㅇ0
SD4C242S
51
컨SR뜨》긴.-FFr'!
a3C41―그
B드Dl그
CE떼）쵸=8그-
노wc차｀4ㅓ
lea
pu,h
pu'h
C·11
lea
pu,h
call
add
乙e의七
그e
lea
e의p'
eaX' eaX
:mD43댕녹
:.꽈4395
ecil,
_그一3므532
그I）홑3皇’''
_〔 슨CX'
d낚crdㅑ「스 〔e5F+
ㅏ:
J〕
勵ght.ThisIinejS
breakpointecLWecanpress
runtoseethebadboyand
．·}.
므뾔卜－튿
thenregisteragaintobreakin
thisBP
COde緻죠dr슴，,r후CgO효B9415inMOdUlen乙dll*dll
」
byLePtr
d닿ㄸdp
」
:7CgOEBgori。p
:7CgOEB91noP
=,CgOEB92n。p
:7CgOZB93n。p
21raa乙Sy의℃이없rall묘e乙《)
:)
·ㅡ r X
- t '
- p·
FrrrL
:⑩춈385AI33.
:⑩439SA3 42
:OO4385A4FZ
:OO4385A5AE
:OO4385A〔r7D소
:⑩43B5A849
료）04385A93BDI
=（】O439SAB.2E3
maval,
leaedl'
addal,
OreCX'
In。vbyte
XOr·aX'
inCedX
repnZ
,Ca,b
【．,p＋슨dx+-l,·1
r>CgO료B94ze헙
:후CgOEB'51슨a츨，p,d낚。rdp귑r【츨，P]
r7CgOEBgCleae:p,d낚。＝dp七r【。，P】
r7CgOEBA()n。p
:7CgO므8AIㅍ。p
논CgOEBAZnoD
no乙
deC
a댜》
jb
eCX
< ∼
Enabl·
굿nabl·
Enable
恥C耐혓軫pl
Url＝덥Cㅛeㅍ℃拓윗
LOCalrtㅍ노C七ion
D·업ail.
ID.taila
eCX
edx,ecx
JO측385
LIS
COtOAddr.',
StopAutOㅇnApl
StepIntc'"rep"In,tructiorl
pa乙ChCOd슨
ByPa,,
TernLina乙e
. F죤ferenC.O
貝CO「〕노- .-a- Or
y=dltl。「a그:=nPa℃A〕
길3프프
츤에438SAI》8D542444
leaedㅈ’d날。rdp업r【eap+,」
笙희笙희蠟蠟
Pau,eI견쑤In
L耐109059P01226of2377Co出脯柚嫩0043856E叩ffseI00嚼56Ehin馳Ar.
탭■■
■탭■■

Right. This line is breakpointed. We can press run to see the badboy and then register again to break in this BP

좋아. 이 line에 breakpoint를 설치했다. 우리는 badboy를 보기 위해 run을 누른다. 그리고 다시 등록하기 위해 BP에서 멈춘다.

:)

We break in the BP !

우리는 BP에 멈췄다.

So far, so good. W32Dasm also offers a possibility to save BP's, stepped code, etc

Let me show you how first

지금까지 좋아. W32Dasm은 또한 BP를 저장할 가능성을 제공한다. Code를 진행하자.

먼저 어떻게 보는지 보여주겠다.

시스템 생성 대체 텍스트: ．躇
」「
巨回
여눠aan.n
―啄ㅉ碼n
▼-
筐】d口，
九
U》
=IExam伊es
야始미終
ge（珊end
吸
Bu『b札비ad
公
晦ndocLjrner맡en
＇〕
Dezeco叫딥er
딩
INFO;
W32Dasmnotonlysavestheasmfi-easa
*.alffilebUtWIllmeanwhileaISㅇSaVethe
comPleteprojectWithBP’뒤Steppedcod타
etCasa*.WPjfile-->easyifsㅇmething
happens(likethe*.uddfilefor이―y).
BTW,youcanonlysavethedebugging
projectfilewhiledebuggingofCOurse
M「
netw뻐dOC出es
BBestandsnaam
OpslaanaIStyPe:
陣軸ㅁ
IA'nUatR“仁alf)
[EI嫩Iaan-
21紬―一-

INFO :

W32Dasm not only saves the asm file as a \*.alf file but will meanwhile also save the complete project with BP's, stepped code, etc as a \*.wpj file --> easy if something happens (like the \*.udd file for Olly).

W32Dasm은 alf file 일 때 오직 asm file로만 저장하는 것이 아니다. 그러나 BP와 함께 완벽한 project를 저장할 수 있다. Code를 계속 진행하자. \*.wpj file ---> 무슨 일이 일어나기 쉽다.( Olly의 \*.udd file과 같이)

BTW, you can only save the debugging project file while debugging of course

By the way, 물론 debugging 하는 동안 오직 debugging project file을 저장할 수 있다.

시스템 생성 대체 텍스트: 「－---------------------- '.
opsl硼nals
여닒aarlln:
J
百而諦
＝》Exam나es
야닐鄒目s
geㅇpend
아
Bu『큰aLlblad
夕
胸ndoCurnenten
前
Dezeco昨납er
YeP.W32Dasmpr아沁sesanacceptable
nam이WhyShㅇ미dwechangeit구 O
님 L
M聊
netw래dOC出es
BBest纛、出naam
勵轍輛
oPslaanals嫩”「l綸脯R“陸비
一FE니物Iaan-
一［〕서nuleren-
驪미

Yep. W32Dasm proposes an acceptable name, why should we change it? :)

예, W32Dasm은 괜찮은 이름을 제안했다.우리는 왜 바꾸려고 하는가?

시스템 생성 대체 텍스트: -
_-..--.....-
.
Ok.Webrㅇkein
theBPOntheCal-.
StepF7inthecal-
toexPIㅇrewhere
EAX15Set.
deAddres,
:Oㅇ43E5〔El므In뵤OdUI슬Ar헙G슬끄t.eXe
:OO43aS'2
:00씰355'3
:OO43e564
:00룔3B5'9
:凹43B5石D
Hp'2004385石E
:00펩385,3
:00펩3B576
Z()O룔3a5'a
r()O43BS7E
:OO43BSBZ
《
pu,hedx
pu,heaX
CallAr七C손n'_OO46OBSO
leaecx-d낚。rdp℃r【e,p+28그
pu,hecx
eallArtCe·＿OO434COO
add·．p'00000010
乙e.乙O요X' 솥표X
乙닝-
rFrFFFFF
.ptr
' eaX
【．.p
그。00츤38601
lea·dl'd낚。rdp윕r
OreCX. FFFFrrFr
JI
En르bl슴
En흩bl츨
7nable
DOCU죠｀On乙츨dAPID·L'11,
UnDOCU죠L슴nL츨dAPID슬tall.
LOCal
[.,p+4펩】
》
C。Dvl
- 盼OLO ^QQ【슨S,
FrrLr
eCX
8590
S헙。P
S七．p
AUtO
In七O
On
-r
軌mC七10끄De七ai18
Apl
Pa헙ChCOd슨
ep"In，七ruc七ionByPa,,
쵸lt1o「a_
dvcrdFㄴr【egF+
Au祉히An騶添，l 探l杖키
르므＿뜨」竺느씌 z의-.-－쁘－」
竺」阿

Ok. We broke in the BP on the call. Step F7 in the call to explore where EAX is set.

Ok. 우리는 이 call의 BP에서 멈췄다. EAX가 set되는 곳을 F7을 눌러 call에서 찾자.

시스템 생성 대체 텍스트: :OO434BFE90
:OO434BFF90
nop
nop
"se=습rer1C슨굽
즘38SC드
_- 딤ㄷ .-.
gEgB
:에434COO
r（》O'34Cㅇ3
:Oㅇ434CO'
:0043수CD므
:i〕4뜨순C꽈9
83ZC24
8B44242B
드3
SUb
a驢）V
pu듸h
pu요h
pu프h
esp'
eaX'
ebX
ebp
츨,1
e:X'
edl
;1궁＝r；쌨
f-,O24
d낱。rdptr【e횰p＋궁S]
dwc【dF헌= 【e'x】
pLr
그길：:rd
Mmmmm,thislookslikea
VerifiCatior1OftheSerial.Ok.
NㅇwstepF8toexPIㅇreand
seewhatal-goesonher티
C으쑈슨요크쑈r슨드드 _:.D샐34CD그그5 소rlYOdUle요：'-뜨
긔 J
·1,
닙｀뀀·느」」·일‘
【e,p十:】'ecx
Ftr[e色x+ 1
【e크F+ 〕,e：나：
r흰= 즈'드x+ 1
re섭F수 〕,ㄷ＝-=
卜‘』‘'蠟．"+-」
e,p,이）0000휘）
:nC흽＇
Cmp
W놀OV
끄청e
mOV
te,t
끄노e
XOr
bytep乙r【e,p+.E]
'1' "
d낚Ord
p乙r【e,P十
】,edx
al,bytePtr【eap+
al,al
00000024
d낫。rdpㄷr【츨，p+28]
edx,edx
쾨
거）습룅솝뺑BE슬add
:OO434BrCret
:OO434BrDn。p
:OO434BrEnOP
:OO434BrFnㅇp
=OO434COOSUb
:00펩34CO3m。v
:0043뱁CO7pu,h
:0043'COSpu,h
:00춈34CO9pu,h
:OO434C이노m。v
<'
e，隅，
.a섶'
·bX
.bp
·,1
eCZ.
d낚OrdDtr 【ea표1
En츱bl츨DOCU죠힙．n귑츨dAPIDO乙all쐬
F
〃In'7「‘l htlt· "t' r츨첵”+'Hy+

Mmmmm, this looks like a verification of the serial. Ok.

Now step F8 to explore and see what all goes on here.

음, serial을 검증하는 것 같다. Ok.

이제 탐험하기 위해 F8을 누르자. 그리고 어디로 가는지 봐라.

시스템 생성 대체 텍스트: 乙
URSoftW32DasmVer1Opr（띳ram【》isassemblerm驪沈鉀
侖 JJn
e －커닙
nR Z-ㄷ
醱／飇
뻬흴姿
INF○:W32DasmkeepstrackofthestepS
take『ISofar.RedVA====steppedbefore·
:卯쨀34COO
:OO434CO3
그에‘34C0'7
'
그Oㅇ‘3샐COa
:00샐34CO9
=()0434CㅇA
d3EC24
eB44多螺
__OㅇO■〕24
d낱。rdptr【e,p+
S5
56
gBOC
aube의p,
n戚》V.aX'
pu,hebx
pu,hebp
n』，he,i
麗흽OVeCX'
d얗。rdp七r【른ax】

INFO : W32Dasm keeps track of the steps taken so far. Red VA == stepped before.

W32Dasm은 step의 track을 유지한다. Red VA == 전에 실행됐다.

시스템 생성 대체 텍스트: ],
.b=」：+
],
le:x+
넌쑈｝:
난〕X
SeemsIikepreparinga
certai『ISpotfrㅇmtheserial
inAㄴandthenComparing
:e=r+
Lㅌsp+ 】,
누｛痲覇FF〕

Seems like preparing a certain spot from the serial in AL and then comparing

With 2Dh (ascii "-")

AL의 serial부터 중요한 spot을 준비하는 것 같다. 그리고 비교한다.

2D는 (ascii "-")

And we jump if this spot is not "-"

Ok. Let's jump …

이 spot이 "-"이 아니라면 jump한다.

Ok. 계속 jump 하자.

시스템 생성 대체 텍스트: 』O434D3D,4소4
je〕04于윗긔팟
*묘ef슨re끄tedbyA（미ncQndㅗ乙ㅗ：八＾ㅗ。r l='.dㅗ℃ㅗ餓길al仇mra乙ACiclr슨드三슬=
H'434C2C<c》, :43슥C3S《c>, 뜨r43춈D25《C/, :〕=4）一DZC(C' 료》ㄴ43슥D3E별C
1.이」즐3욤Dsr《cl, .닐닐쨍돌숨p66《C>, .닐습숨솝슥氏쵱숨《C,, '닐솥숨솝一DBO《C,
=00쨍34DSr
=（》0434D905드
:OO434D91SD
그그04무도E크2 2드
〕기긺「-E〕4 =한
pOFedl
pope,1
p。pebp
XOreaX' eaX
pcpebx
adde,p' --
ret ''‘∼
...ㅇ아）5,atoncetㅇthe
endOftherㅇUtine
insteadofcomparing
therestOftheSerial
cㅍp솥dl
그슨
.引
34긔
rOO434】

...oops, at once to the end of the routine instead of comparing the rest of the serial

웁스, Serial의 나머지를 검사하는 대신에 routine의 끝으로 왔다.

See all the other jumps here if a bad serial is detected along the routine

Bad serial은 이 routine에 의해서 밝혀졌다면 여기의 모든 다른 jump를 봐라.

시스템 생성 대체 텍스트: ＇노l건샌l -
''l ㄴ·드I -
OH。21… J.
Unlesswehelpthe
prㅇgramalittleinanother
direCtiㅇnofCOUrSE'Let'S
dㅇthatandtryitㅇut.
픈aX
e51
아결F
룝。뚝
eb표
esp
FㅍF
add
ret
Ancltherlfinallyher이beforeretuming,
EAX15XORecL
ㅡ－
ㅡ－
RememberXOREAX，ㅌAX-->ㅌAX
0
쬐I탭·－따’른.'
．■탭딤딤〔 탭딤
rr「edl,
COdeAddr슬as :OO434D92i츨inMOdUleArtC습m.eXe
」
r'칙
I4D,
ItisthisX○REAX，ㅌAXthatlwantto
NOP.Thatway,EAXWi∥notbezero
whenreturning·(OrINCAL,oretc)
r그b
'그a
ar1G
醜邙＞
끄、e
g騙P
」n.
add
표〔》V
IZLUI
Sar
InOV
P。p
,hr
add
쵸江〕V
eaX'
eaX'
edx,
eaX'
.dx,
eeX
edX
dvcrdptr【e의F수-】
eaX
얍
:0043홑D87C2聊》·cx'00000^BF
竅）O씹3홑DSD〕e00흩34DA3
:OO434DSFpOP슨dl
:氓）434DgOPope,1
:班》434D91po낏숟bp
:O（】434D92ㅈ。reaㅈ。eax
:OO434D94p。p습bx
:00'3춈D95adde'p,0000002펠
:00'34DgBre乙
:00펩3흩D99ㅁ聊》edi'0000000r
rOO43홑DgCle0043흩n죠3
FrrrL
edl,
J청＜卜펭2.」才노
eCX'
eaX'
eCX
edx'
eaX'
·dl
·aX'
edx'
e의1'
」〕l:그F
4EC一E
<, '
Enabl츨DOCUㅍ｀슬nL습dAPID슬乙ail,
Enabl·UnDOCU죠L·n윕OdAPID습tail'
ZnableLOCalrU효C乙IOnDe乙alls
스」
07
.dX
StOPAutOOrlApl
5乙epIn℃cl·rep"In，乙r'dc℃lorlByPa,,
1F
eaX
edX
笙희笙뇌뾔麗
》
COpy-
。。：。Add【…―
Pa:chCOd.-
Ter祗na'e-
竺」些」
「－」－n.:·燈흴．nn』。』＾。ㄱ乙｀n"'-.。nn｀』h。｀ㄴ졉［「；I-.＾」F一一

And then finally here, before returning, EAX is XORed.

그리고 마지막으로 여기, EAX를 XOR 후에 returning 한다.

Remember XOR EAX, EAX --> EAX == 0

기억해 XOR EAX, EAX --> EAX == 0 이다.

Unless we help the program a little in another direction of course. Let's do that and try it out.

우리가 program을 다른 방향으로 가게 돕지 않는 한. 그것을 실행해 봐.

It is this XOR EAX, EAX that I want to NOP. That way, EAX will not be zero when returning. (Or INC AL, or etc)

이 XOR EAX, EAX를 NOP 하기를 원한다. 이 방법은, returning 할 때 EAX가 zero가 되지 않는다. (아니면 INC al, etc)

시스템 생성 대체 텍스트: …-一一一一一一一一一一一一一一一一 ------------------－一 \
W32DasmCOI】epatcher
DE．團
프Ip
CUrren乙Ina七rUCLIOna乙Elp
XOr슨쵸X' ．쵸X
ZnLerNe닺In희헙rt1Ct1OnB슨10힁〃
I一 ■-一一一 一
Co（요ePa乙ChLI．乙1ㅍg
l■
■■■．님■--－■■■노·－
C순쑈－eXe
Thisisthepatchscreen
fromW32Da5m.
Herewecanmakepatches
intheCodeinmemㅇry.
O
t
O
G
긔긔
cl。。e-
－덞n텬■
1ㅍ驪11
sar
죠놈OV
pop
'hr
add
겸脯》V
''^'
eCX
슨dx,
-빔J〔1- ■■J
eaX,edx
솥dl
솥aX,IF
edx'eaX
eal'edㅈ
StOPAut。
StepIn乙。
ㅇn
"r
Apl
ep"In，헙ruc헙i。nByPa,,
pa仁ChCOd슨
rL
笙희笙희蠟陋
·_『노_'_ 7'Lr날씸』히」「노r놀＇놀r크L「놀【《-_'r놀r놀r놈히』「느r놈「r-
-『．】_·'

:)

This is the patch screen from W32Dasm Here we can make patches in the code in memory.

W32Dasm의 patch screen 이다. 우리는 memory의 code에서 patch 할 수 있다.

시스템 생성 대체 텍스트: J'''
口聊》
公｀e
.dl,
YOUAreAboutTO除汕什Q업eLOCabon
:004꾀【）92TtlruButNotInck넙．9LOCa耿r
：氓》43《）94.................AreYOUSUre7
의Ub
add
mOV
스rld
eaX'
ea표＇
edx'
츨aX'
eCX
edX
d날Ord
」a
r
t ·
p
j(AndCOnfirmthepatcheS
youwanttㅇmake·
:OO434DgC
,＜』．'
Enabl.갸OCUㅍ밍．
꾜『노룔bl츨Ur뇨no「＇U
Fr

And confirm the patches you want to make

그리고 네가 만든 patch를 확인할 수 있다.

시스템 생성 대체 텍스트: 섭래금「꺼＇「F『了Tll■Rp卜【루【．（큐r.r『낌nF래．즐FF「＇r뀀＇rn슨「「．r
Indeed.
X○REAX，ㅌAX
15NOPed
:0043홑DS7
:0043춈DSD
:OO434DBF
:OO434DgO
=OO434D91
‘으연욕典）혁료
:OO434D승3
:00춈3'D94
:0043'D95
:00춈3흩D98
:00펭3홑D99
·
Cㅍ딴
〕·
pop
pop
pop
E]
nop
pop
'dd
r슨t
CmD
·α’O0000ABF
OO434DA3
edl
e의1
ebp
헙r【슨，p+-E】
F
ebX
e,p'0000002수
」
.dl, 0000000F
《
멨·蠟驪．驪텼혔

Indeed.

XOR EAX, EAX is NOPed

정말

XOR EAX, EAX는 NOP 됐다.

Also place a BP on this line.

또한 이 line에 BP를 설치해.

Press F2

시스템 생성 대체 텍스트: F擊’
COdeAddres듸 표》0434D9218in뵤OdUI슬
…＝흐，’므볍볍니
ArtG슬m-eXe
」
:00펭3'DS'
:OO434DSD
:OO434DSF
=00펠34D90
=OO434D91
초00'34D,2
섰》O펩34D93
:00쐽3'Dg춈
:00춈3'D95
=()O춈3一DgS
=OO434D99
。聊》ec'-00000ABr
그。OO434nA3
p。pedl
p。pesl
popebp
nop
pop
add
ret
솥bX
e,p
0000002'
<
크nabl슬
Zrlabl·
7nable
CmD손dl.
』ㅣ
00000O0r
D。cuㅍ느·nt습dA'I그e仁aㅗl도
Urll)OC
LOCal
Ancltrythepatch.
PreSSrUn.
》
copy-
COtOAddre프,
StopAutO0rl》｀
StePIn℃O"rep"
pa乙ChCOde
Instruc乙lorlByPa,,
Ter교ina乙e
긔 FrrLL
笙희笙희쬐麗
三넌」
뺐．■■■■■■■■■■■■■■ ■

And try the patch.

Press F2

F2를 눌러.

Patch를 해.

F2를 눌러.

시스템 생성 대체 텍스트: 습莎礪〔麻驪磅
「宇－-－맬탭’■．■·떻．'---＝∼－■
닐섭흩디l?
Er나ellegi샀lalioninf이m비ion10unf。ck●xP..「】．」
---l■．··냅■■탭탭뺙―■ -―닙■■뺨＇··r·‘rl--
I
즌
一.一中
r
…
「눙녠驪차＇의《
f令、
一』』』』
baliasin
W『a
P締
'I/ 브
―아흙
緘
C－깁 탭■뀀■따
순ㅏ
닝夕瑯
u＼一
-.·소-
r[
巨면
－미■■■■■r뱁．．낟뱁뺍「 뉜工그끄「찧
치Ze
smoo出
)
'OUn
angle
lp,e::u，성
-Spa이nQ
仙 。 의
∥I鷺
1
수·。'
r,L『a승e '
이O-ㅇ
딛
U7
·．·밈
C
a
p
Tor-
［「
■■■■■■■■■

;)

This seems okay.

Ok가 나타났다.

1. **Using LordPE as hexeditor and patching the exe**

Compared to Olly where it is easy to make permanent changes in a file, it is impossible to do it with W32Dasm.

어느 file에서 영원히 변화할 곳을 만드는 게 쉬운지 Olly와 비교해. 이것은 W32Dasm으로 가능하다.

With W32Dasm, you can only patch memory, not on disk.

W32Dasm와 함께, 너는 오직 memory patch를 할 수 있다. Disk는 아니다.

So, we will need another tool to do that.

그래서, 우리는 이 일을 하기 위해 다른 tool이 필요할 것이다.

But let's first look in our debugger what and where changes are needed.

그러나 먼저 우리의 debugger를 무엇이 그리고 어디에서 변화가 필요한지 봐라.

So, click in the debugger window

Debugger windows를 click 해.

시스템 생성 대체 텍스트: FE
:卯‘34CSF
:00'34C90
:OO434D91
:OO434D9쌩
:OJ핑34p5프
:Oㅇ434D95
OrSeeitheFe 겨
stVA00434D9찧
…勺r
.dl
e,1
ebp
룔aX'
ebX
e룔p'
슨a二t
pop
add
AndnotiCethatWe
needtoedit
.3by
opCodes33CO
一一탭■L
톤뗀쁘섶므一
=rredl,
COdeAddr슬요룔 흐⑩434D92i의in》IOdUI슴Ar七C슬죠L.eXe
긔 -
乙乙P
jne
츨di,
크Ub
add
m。V
arld
=P
yte
c珊P
그ne
add
끄〔》V
1ㅍWll
'ar
mOV
pㅇp
,hr
add
n노OV
eaX'
eaX'
edx'
슬aX'
edㅈ，
eCX
edX
d낚Ord
atVA00434D92
:0043쵸De7
:00'34D8D
:OO434DeF
:OO434DgO
:OO434D91
Bp':OO434D92
빤’戮》0434D93
낌:00'34D94
.7:00춈34D95
:00춈3춈DgB
:OO434D99
<---
C2略》
〕슨
pop
pop
pop
「二F
nop
pop
·dd
re귑
c”三《】
ecx,00000ABr
0043쐽nA3
edl
e.l
ebp
ebX
e,p
00000024
t …
p·-
손dl. 0000000F
의 FrrLL
eaX
edl,
eCX'
eaX'
eCX
츨dㅈ,
eaX'
.dl
eaX'
edㅈ’
e,1'
7n츨bl습
En츱bl츨
Znable
DOCU표씸슬n乙츨dA꼬ID·乙ail.
U효DOCU俎·n乙슬dAPID.tail,
LOCal玖metiOr1D슨tail,
》
쬐
C。py-
COtOAdd【츱·뾔
S업OP
S乙ep
쵸U乙O
In윕O
ㅇnA꾜I
튜rep"In，乙ruc七1。nByPa,,
pa乙ChCOde
edX
:'9434DC9
〕O434DCA:J청
〕）흴34DCZ).:_
:.꽈一3스Ocr8EFr
WhiCh15atㅇffset
OO034D92
笙희笙희뾔蠟꽤큐
e e
<
Li閔：101593Pgll42of2377以님eD찮a（莩00434D92(90ff柏〔I:II:ID4【）92hinFile:

And notice that we need to edit opcodes 33 C0

그리고 opcode 33 C0의 edit가 필요하다.

at VA 00434D92

VA 00434D92

Or see it here at VA 00434D92

또는 VA 00434D92를 봐라.

Which is at offset 0034D92

Offset이 0034D92 다.

And NOP them !!!

그리고 그들을 NOP 하자.

*번역 주) VA, offset은 꼭 필요하다.*

INFO :

In Part 03, I explained you that an RVA is an offset from the base address at which an executable was loaded in memory.

Part 03에서, 나는 너에게 설명했다. RVA는 base address부터 offset이다. Memory에서 실행 가능하게 load 됐다.

This is not necessarily the same as the offset within the file on disk because of the section alignment requirements.

이것은 File안의 Offset과 필연적으로 같지 않다. 왜냐하면 section alignment가 있기 때문이다.

The PE header specifies the section alignment requirements for an executable image.

PE header는 executable image를 위해 section alignment가 필요하다는 것을 명시한다.

A section has to be loaded at a memory address that is a multiple of the section alignment.

Section은 다양한 section alignment의 memory address에 load됐다.

The section alignment has to be a multiple of the page size.

Section alignment는 다양한 page size를 가졌다.

This is because different sections have different page attribute requirements (read, write, execute, …).

이것은 왜냐하면 다른 section들은 다른 page attribute가 필요하다.(읽고, 쓰고, 실행, …)

Hence, a page cannot span section boundaries.

그리하여, page는 section 경계가 걸쳐있지 않다.

INFO :

Because the PE format always talks in terms of RVAs, it's difficult to find the location of the required information within a file.

왜냐하면 PE format은 항상 RVA term을 이야기 한다. 이것은 file에서 요구된 위치 정보를 찾기 어렵다.

It's a bit complicated to calculate the address for the given RVA in a memory-mapped file.

이것은 memory-mapped files에서 주어진 RVA address를 계산하기 꽤 복잡하다.

You first need to find out the section in which the given RVA lies.

먼저 RVA가 자리 하고 있는 section을 찾는 게 필요하다.

You can accomplish this by iterating through the section table.

Section table을 반복하는 것으로 이것을 완성할 수 있다.

Each section header stores the starting RVA for the section and the size of the section.

각 section header는 section과 section의 size가 시작되는 RVA에 저장된다.

A section is guaranteed to be contiguously loaded in memory.  
section은 memory에 인접하게 load되게끔 보장됐다.

Hence, the offset from the start of the section for a particular piece of data is bound to be the same whether the file is memory mapped or loaded by the operating system loader for execution.

그리하여, data의 특정 조각 section의 시작부분부터 OS에 의하여 실행되기 위하여 load 됐을 때 File이 memory mapped 되거나 loaded 될 때 offset은 같아질 가능성이 있다.

So, to find out the address in a memory-mapped file, you simply need to add this offset to the base address of the section in the memory-mapped file.

그리하여, memory-mapped file에서 address를 찾을 수 있다, 너는 간단히 memory-mapped file에서 base address의 section에 offset을 추가하기 위해 필요하다.

Now, this base address can be calculated from within the file offset of the section, which is also stored in the respective section header.

이제, 이 base address는 file offset의 section으로부터 계산되어진다. 그곳은 또한 각각의 section header에 저장된다.

시스템 생성 대체 텍스트: WhiletheRVAhere
Wㅇ미dbe00034D92
rㅇ醜34DeF
:에솥3펠CgO
X)O쵸34C91
:·》■〕4그갤r)?
(RVA=VA-1ma9ebase)
r瑩r
슨Dr
드프꽈：
드크X
X
'L
－匕
一r
一
rJ'
Remember:00434D92is驪「「「’
aVA(VirtualAddress)
요eferencec
그〕434드근쬔 _
nCㅇn쑈쵸tlㅇr·、
NFO:
PerhapSthisseemsal-verydiffic미ttㅇyou.
Hㅇwever,thereexisttwoeasyform미ae
tㅇCalCulatebetweenC)ffsetandVA:
:OO434D9983rFOF
:OO434DgC7405
:OO434DgE93FF1'7
:ㅇO434DA二태드EC
Offset=VA-1magebase-VirtualOffset*+
RaW○ffset*
의玆｝MOdL
00000ABF
끄L3
VA=Offset+1magebase+Virtual○ffset*-
Raw○ffset*
BTW,thinkaboutita5VAfㅇranexecutable
fi-emappedinmemㅇrybytheOS-oaderand
thinkabㅇUtOffsetfㅇraneXeCUtableㅇndisk.
0000002'
0000000r
Nㅇte[fㅇr*,SeeSeCtiㅇntable
D．乙all,
ID슬ta노1
De七ail희
「s七epIn乙。"reP''Ihs헙ruc七ionBy
ㅜhistim타RVAandoffsetareequal,
butthatis『lotalway$theCasel
뺑뺨’「一一一一一一一一一一一一큐拓》닙5〔F石茁「→－-
笙희笙희쬐
Li閔：101593Pgll42of2377Co出D勸연004型畔＠。ffse’函巾4D9첸n마：'F"
■■■■■■■

INFO :

Perhaps this seems all very difficult to you.

아마 이것은 너에게 매우 어렵다.

However, there exist two easy formulae to calculate between offset and VA :

그러나, 이것은 2가지 쉬운 공식이 offset과 VA를 계산하여 존재한다.

Offset = VA - Imagebase - Virtual Offset\* + Raw Offset\*

VA = Offset + Imagebase + Virtual Offset\* + Raw Offset\*

BTW, think about it as VA for an executable file mapped in memory by the OS loader and think about Offset for an executable on disk.

By the way, 운영체제 loader에 의해 memory에 매핑된 실행 파일을 위한 VA로 그것에 대해 생각하고 disk에 있는 실행 파일에 대한 offset을 생각해 봐.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note : for \*, see section table

번역 주) 여기 내용은 뭔가 다르다. <http://reversecore.com> 참조하라.

Remember : 00434D92 is a VA(Virtual Address)

기억 : 00434D92는 VA일 때(가상 주소)

While the RVA here would be 00034D92

RVA가 0034D92다.

(RVA = VA - Imagebase)

This time, RVA and offset are equal, but that is not always the case!

이 시간, RVA와 offset은 같다. 그러나 항상 같지는 않다.

And then at last comes the good news :

마지막으로 좋은 뉴스가 있다.

W32Dasm gives us both data :)

W32Dasm은 2개의 data를 준다.

So, no worries : use the offset in a hexeditor … but I'll show you a much easier solution :)

그리하여, 걱정하지 마 : hex editor에서 offset을 사용할 수 있다. 그러나 매우 쉬운 해결책을 보여줄 것이다.

First however, let's save this project (just in case) and terminate the debugging.

그러나 먼저, project를 저장하고(이번 경우에는) debugging을 종료하자.

The easy solution for all this is not to use a hexeditor: hexeditors use offsets.

쉬운 해결책은 Hex editor를 사용하지 않는 것이다. : hex editor의 offset을 사용하자.

Instead, use LordPE. LordPE has come really interesting features for this. Let's start.

정말, LordPE를 사용하자. LordPE는 정말 재미있는 기능이 있다. 시작하자.

REMARK:

More complicated work may require a real hexeditor though.

So, I have NOT said that hexeeidtors are useless … we will use them for other stuff.

주목:

좀 복잡한 일은 진정한 hex edtor가 필요할 것이다.

그래서, 나는 hex editor가 필용벗다고 말하지 않았다. 우리는 다른 구성에서 그것들을 사용할 것이다.

시스템 생성 대체 텍스트: Wearegoingtㅇ
"hexedit''theexe
驪【LordPEDeluxe1byyoda
.―녜。
Path
【쟀沕ml
【寸샀eml
＼왱쨌emrcoh斗샀em32＼豺nss.exe
\??\c＼쌔ndOW執s螂em32\c앍略．exe
PID
000仄）000
0(）【）00004
0()t)D0268
000田2E8
ㅣm젱eBase
00000000
00000000
485800D0
4A68凶00
-magesize
00000000
00000000
0000F000
0仄刃戚勵0
Reb예d麗
媤o'o媤o'o·
-P楸h
ImageBase
m꿩e히ze
0翩ons
曰톄떼

We are going to "hexedit" the exe

시스템 생성 대체 텍스트: …를
－→씸■■■·■■■■■■ -
드크r =>-
』■■■■■■■■매■■■■■．
빽■■■떼＇＇■탭■맣”＇·■．"＇■뻐뻑뻐∼
!NFO:
It이dyㅇuinPartO3whenwetalkedabout
theheaderthatWeWㅇuldCOmebaCktㅇ
this.Wel-,indeed,LordPE(andanyㅇther
tool11kethis:Pㅌtools,...)makesitveryeasy
tㅇchangethePEheader(andmore).Nㅇw,if
youwㅇuldwanttㅇmakeconversionS
betweenVAandㅇffsets,yㅇucaneasily
calc미atethemusingtheformulaefrom
befㅇreWiththedatafrㅇmtheSeCtiontable
BaS祗PEH閃deflnf이ma略b衲
鎰一
En驪以Point
ㅣmageBase:
sizeofㅣmage:
Base아COde:
Base아D副a·
Sec力餉Alignmert
FIleAlignmer丈
MaQ〔
Sub斗삵em:
Numbel이Sections:
T轍eoatestamP:
S始eofHeadlef竅
Charac삿efi샀祗s:
Checksum:
S砲e0IO야ionaIHeade'
Nu『n미RvaA庶tSlzes:
- 0002□
- 0004
-3909F9A9
100001000스1」
l 이0F」
1000000co」
! 00Eo J
0K
...thatCanbe
foUndhere
Saye
Sect眈眈
00000이0
...buttoday,I'mmoreinterested
inthelIaUtOmateCl''
s이utiㅇnforthisinhereo
- PE태i柚『
Bleak牲Enter
Reb너IdPE
Uns며t
DumpelS餓el
수댕
n
O
'r떽
p
0
r
一一一一一一一一
＾卜＾∥튠

INFO :

I told you in Part 03 when we talked about the header that we would come back to this.

우리가 header에 대해서 이야기 할 때 다시 돌아온다고 Part 03에서 말했다.

Well, indeed, LordPE (and any other tool like this : Petools, …) makes it very easy to change the PE header (and more).

좋아. LordPE(그리고 다른 tool도 있다. :petools, …)는 PE header를 매우 쉽게 바꿀 수 있다.(그리고 더)

Now, if you would want to make conversions between VA and offsets, you can easily calculate them using the formulae from before with the data from the section table

이제, VA와 offset을 변환하기를 원한다면, 쉽게 수식을 사용하는 사용하여 section table의 data로부터 계산할 수 있습니다.

번역 주)FLC(File Location Calculator):File 위치 계산기

….that can be found here

여기에서 찾을 수 있다.

...but today, I'm more interested in the "automated" solution for this in here :)

그러나 오늘, 나는 좀 더 흥미로운 "automated" 해결책을 보겠다.

And with the push of a button, LordPE will calculate all three : VA, RVA and offset !!!

PUSH button을 누르고, LordPE는 그곳을 계산할 수 있다 : VA, RVA and offset !!!

시스템 생성 대체 텍스트: \or이ㄸamfiles낢쑈B1ㅜhat'5theVA
r-Ba沈PEHe匯F*h【a勸cakt*tor】
J宿
澁蠱씸꼬
E楸汐PcHnL
ㅣr'nageBase:
size()flmage:
Base아Cc心e:
Base미D副a:
Sec力onAlignr
FII뱀卜gnme楸
Maglc;
i饍 댜희
-offsel:1 1 一
A改數ionalInfofm戡ion
Se히【죠
嘴

That's the VA

VA다.

...that we always know from debugging of course

우리는 debugging 과정에서 항상 알고 있다.

시스템 생성 대체 텍스트: bEF'
Bas祗PEHe
Inthiscas티
RVA==0ffset
bUtremarkthat
thiscertainlyis
notalwayssc)l
Er낚汐Pc닙nl:
-mageBase:
sizeoflmage:
Baset)fCode:
BaseofData
Sec力onAbgnr
F"eA矗gnment
M이Jr
〔프그 X版阿爾福
E巫］ ＼「’一7
臼墜의 -
층늬크
Sec劤ns
A改libonalInfolmation
That'seasyhuh?
I FLC一
Seclbn:
B한e驪
; ．驪毓
SB83C424C383FFOF
쁘흐드－」
c。mPa『e」
뇌

That's easy huh ?

쉽지 않아?

In this case, RVA == Offset but remark that this certainly is not always so !

이번 경우, RVA == offset 입니다. 그러나 항상 같지는 않다.

Okay, I hope this was clear enough and is understood now.

Ok, 이것은 충분히 명확하다. 그리고 이제 이해했다.

But look why I came to LordPE too: no need for another hexeditor, with LordPE, you can do this easily too !!!

LordPE로 온 이유를 봐라 : 다른 hex editor가 필요하지 않다. LordPE만 있으면 너는 쉽게 바꿀 수 있어 !!!

시스템 생성 대체 텍스트: rㅇO쵸34DBF
:OO434DgO
:00펭3춈C91
:OO434D92
:D〕팔〕뜨::크4
p。pedl
POp·,1
p。p·bp
XOr즌aX' eaX
p。pebX
addesp'
ret
聊뿌’변鸞헨＇논’
Bas始PE'H。］["eh'a勸calcubtor】
Er뱃汐Point
ㅣmageBase:
Size매mage;
Base이COde:
Base이D副a:
Sec力叩Aligry
F祗eAlignmen'
Mag(:
Acldresses
-vA:-
-RVA:-
-마fs이：-
-l _■■■■陋
陋礪祈 瓦」
---
「一一一一一一一一一
0K
SaY·
맨E
S枕劤ns
Dire햐ories
AddiUOnalInformation
FLC
Sec따沁n:
B蠟驪 띠工
text
TDSC
SB83C424C383FFOF
·.끄탱Edil
Compare
:)
노」
PEE痲1
Bleak뜨nter-
.Be瞰ld띤．1
＿■．UnS며’-
Dumpers枏el-
-
Indeed건et'sNOP
theseopCodeSWith
LㅇrdPEandSaVe
thefiletodisk.
AbOL낚
Ex' …

:)

;)

Indeed, let's NOP these opcodes with LordPE and save the file to disk.

정말, LordPE와 함께 이 opcode들을 NOP 하자. 그리고 disk에 file로 저장하자.

시스템 생성 대체 텍스트: BTW,theopCode
氷＿forN○P
j-r祀m『戚죤n
==90
毓【16[dit「X
빼네。
ㅡ-
圈딤
송I
圓、.D－白戱r
땋X
FnUCBBFOB3門VnVD4DOOS기쨍O
F D EOOZ｀ㅗO乙p C C E00000O계쨍 EOO
3BFO221944BFFr계쨍7DO
834DEEO8288FO乙p258o
5F83138C기쨍FIFOD42CO
OIBODSBICㅇOFC7451O
＾쨍n닐OFron죠OOB3FAUJSㅓㅗBㅓ홑기쨌F
7 pㄴOI Dg CrDS FrO｀홑OO COOSC F
F3CSBIBgB5llOBCO38
000OEZC28uJSO923계쨍586
FCllOBSFOOAS괘1,0계쨍C83
FIBCOSOSOㅇ乙UEC52453
341FODO3OSOO3E44ES
00200蝦JOSECO＾쨍OuJS기쨍蝦걀247
쵸J계졉｀ㅗZOA,'계쨍OZA7 FBB계쨍BZ
CU걀CCnUCCZO Brg5 F000050OC
4BIBO3O41CFCFCOBSB
2888002COOFIF39853
계쨍23755338DF46거l,C24C
CC5OFO48B3F2120525
335A계쨍29BDB蝦걀계맬7계쨉DrO계쨍乙O
00ㅇ7 F｀ㅗEOOUJUJ000Q계쨍n죠계쨍005계펭계쨍
BIFID154E78D8B88BB
더더COCSCO255ASE85계쨍88
OBFnhCAOOB계쨍rDOO계쨍｀홑OrOβ쌩 F'l
92FEID볍驪기쨍ZUJEOSO거쨍2F8
凸구 C쵸J7aJ71ㅗn〃BOOn냥A괘쨍OBCE계쨍
q· p'S F7 F COO乙01쟁OrOZ000계쨍 .=1
D르긍DEnhEZn닐gCO괘l,기쨍八UnVBCD
蝦J7D기l.OOU걀C000EOOZOCO0n珊00
E75계멩ZroaJOO하JOAC F계쨍｀솥CO
蝦J｀홑7CFDOOZaon닢rD괘쨍00｀홑SEC
00034DgO:
0003쌜恥뇔이
00034DBO:
000341)co:
00034DDO:
00034DEO:
00034DFO:
00034EO():
00034EIO:
曲O34E2ㅇ:
00034E3O:
00034E4O:
00034ESO:
00034E6O:
00034E7O:
00034ESO:
00034EgO:
00034EAO:
‘】ㄸ…l'^s차오·t·什
.Ui+A．藪TS.f令．;D
u勿오·usㄷAㄷ소·..ci
奐N÷즌A습·'쵸A츤..D《
6鷺5．【.a....+C飴＋
C＾÷（A'.．히《介＊I'..
．多勳콥．鷺CA＾·섕．..一
.．쑈ㅈ＄【｝죠무쵸鷺k．느k
驕k$．】.....[J쟈쵸
fiH洲《=·'H.－夕．,.A
...손－一夕夕〕．〕·｝《8
a·〕·댜＝$．뜨P욘'.竹□
L$$C슨一．夕夕f丸..A·，'
E·.．。L寧＜。NP；시aD
.A《n（ㄷ.디T寧L《LSH
C'l$HVP《TSLFgP덖츤
iα跡《nsx《NXJ쵸.E·
Aㄷ．디티；轍】3畛．..
Edibr
k各En눈l
劾匕PE
lns헨
)ers히vel
pbons
鞭fSet:OX00034D94
닌一
빼口떼

BTW, the opcode for NOP == 90

Opcode 는 NOP == 90

Save changes!

바뀐 것을 저장해!

Verify the changes

바뀐 것을 검증하자.

시스템 생성 대체 텍스트: 흩
뺙'' 컫砲’
Ba輛cPE규峻브쁘璽calct.bto『」
디
En心Pant
ㅣmageBase:
Sizeofㅣmage;
BaseofCcde:
Base()fD副a:
Sec肋躇騙gnl
F紬埇gnmen毓
MaQ〔
Addresses
-vA:ㅣ
-RVA:-
-마“비：-
0K
［而耳겨蘭乏
ㅣ
-
Ssve
SeC蘭n숭
Dire햐ories
出枕ionallnf
FLC
Se이bn:
B鮮e驪
TDSC
Comr)a『e
뇌
Bleak驢En눈l
RebL미dPE
Uns헨
…
…Dumpels鄒el
…
… 。이iorls

:)

BTW, always first make a backup file before reversing (also with Olly).

By the way, 항상 먼저 backup file을 만들고 reversing 하자(Olly 또한)

LordPE always overwrites the file without making a backup

LordPE는 항상 file에 덮어쓰고 backup을 만들지 않는다.

What rests now is …

어떻게 달려있느냐?

1. **Testing the patches**

Notice that W32Dasm has indeed also saved the project!

W32Dasm 은 project로 저장됐다.

시스템 생성 대체 텍스트: 坐삡회啣釜노
L痺Pr-s
CtrIL
rile
A胞XhtoanAdNe나OC.SS
．■．·섭■■■■히겸■펩。쫴J＇탬
닐＇ el
Dlaassemblyof
COd.ㅇff,e첩.
Da七aㅇff..t=
0000
000'D(
낟OlntTㅇC
”일 n니 n닐 n일 n닙
·． ·． ·■■ ·． ··
N띠山》erofㅇb그솥ct,.
습《q
飾그ectOI=·tex첩
山）ec七02=.rda乙a
出〕ec:03:.da℃a
ㅃ）e。七04=수r'rc
+++++++++++++++++++
ㅓ
Th손reAreN'
冬
수＋十수＋++++++++++++茯 ^。 →、
Ua차뜨겯．rUp댜Or恪．.
극
喚助q

And we land again on the EP

우리는 EP에 다시 도착했다.

Let's run the program in the debugger to see if everything is fine …

모든 것이 괜찮은지 보기 위해 Debugger에서 Program을 실행하자.

시스템 생성 대체 텍스트: 'l텁’비《珝''.르‘“즈뾔·召틸싫－·ㅗ－一－욜－一」됴섬‘-－」。‘'－一·－－」一∼｀_5兩눋
트뾔
드E(C〕
크
蝕
：뺙
:凹‘3펠CaFSF
:OO434C9ㅇSE
:OO434C91 :〕
:이〕】34r9그
Butperhapsyㅇuwㅇnder:
∥BUtthe이dCOde15Sti∥heFe7
Whathappenedtothechanges?
X二r
:00434:55
:〕0434D크.
pF
dd
'br
른a=.:, 드프
ebX
esp'
Heh타nevermincLㅜhis15becausewe
loadedthe이dprojectanddidn't-et
W32DasmdisaSSembletheeditedfile.
NICe니!W이andinthebreakpointatnew
Startup·Itmea『15thatthecodethatverifies
theserialtok『low"Am1registered힛’15run
921,inM。dul솥Ar七Gem.exe
l築赫－' -
헉｀ rOO43펭D91
BP*:00'3춈D92
pO
끄r
e,1
·bD
:O
의Ub
add
죠놓。V
ar노d
q邨》
r쟁e
。UP
그ne
add
교．OV
12「U
의ar
죠놈OV
pop
,hr
add
꾜戚》V
eaX' eCX
eax'edx
edx,d낚。rdpㄴr
.aX' 추
edx,e'x
솥dl'
34D93
34D94
nop
p。pebx
[e크며rThisisindeedtherealcodewith
갓）024
1꽈숲DABS드
一 "h스L츱 (
theNOPedXORㅌA×，ㅌAX
OOr
Znabl슨
!nable
En·bl솥
DOCUW노슨n헙손dApID슨
七all,
D슨tail의
》
copyl
eOX
eaX
UnD
L。y
OCUm슨ntedApl
API
leCX
edx,
eaX'
.dl
eaX'
edx'
e,1'
S七Op
St。P
AU乙O
IntO
50,pressRunagaintoseethe
res미tfrㅇmtheeditedcodel
T.n죠in'te
FLLrL
:⑩434DC콩SBCZ
료⑩434DC9SF'
:OO434D댜죙CIE81F
:O0434DCDO3DO
:01〕434DCFSBFZ
■■L
걀.11
edX
1F
eaX
edX
笙희笙희뾔뙤
;au：규卜R배
■■■■탭■■■■■■■탭미탭탭

Nice!!! We land in the breakpoint at new startup.

좋아!!! 우리는 startup의 breakpoint에 도착했다.

It means that the code that verifies the serial to know "Am I registered ?" is run

이 뜻은 code가 "등록됐는지?" serial을 검증했는지 실행하자.

But perhaps you wonder :

아마 네가 원하는 것은?

"But the old code is still here ? What happened to the changes ?"

"Old code는 아직 그대로인데? 무슨 변화가 있다는 거야?"

Hehe, never mind.

헤헤, 걱정 마.

This is because we loaded the old project and didn't let W32Dasm disassemble the edited file.

이것은 왜냐하면 우리가 old project를 load 했기 때문이야. 그리고 W32Dasm 에게 수정된 file을 disassemble 하라고 시키지 않았다.

This is indeed the real code with the NOPed XOR EAX, EAX

이것이 XOR EAX, EAX가 NOP된 real code다.

So, press Run again to see the result from the edited code!

그래서, 수정된 code에서 결과를 보기 위해 Run을 다시 한 번 눌러.

시스템 생성 대체 텍스트: 7n「귀雁F
[맥lㄴUd丸《…
目H―一
·
I 츤．
一
一
一
·午
-
11·令
-
。·11
L、
P'essFlFOIHelp
■ 1
1 '
·.-·-수
놓1츤J티〃
캔｝拭
2소倉亐갇
「】含』IJ느i
·『
겹슭
'-/ 毖 汨c
＾戱놂mo뱀솥98-2000byRLVis沁n
櫶눔
＇俎noolroun珂쌘aCIn
-수---'
■ .
■ 二
샐
솔·「
一눋거꺼y
一놀얏
'··r
뺍·甥삿
. -rn
7。tn孕resm.esl.'
一-,
N鄒。e:
ComF)a어r
轍衲151
남넌
OCe88O'놈
■■■■■■■■■■■■■■■■
以丁足落孔
〕7
맥「】
- '■■■■■■■■■■■■■■■■
J·논。-16*
·
수--
·
ItSeemS
everything
15fine!니
d -U -
· ·
-수--·,---
· ·
· '
·
.--
·
·
'＇닉l
·
-·-,--·
opaci砂
L맡、tㄷt令。
수…']J
一 I
희
一
노·■수■··
■J - 11■
r 「－겸r
떼

It seems everything is fine !!!

모든 것이 괜찮은 것을 봐 !!!

The register button has gone !

등록 button이 사라졌다!

In this part 8, the primary goal was to study the use of another debugger and patching a program's registration scheme using LordPE as hexeditor.

이번 part 8에서는, 중요한 목표는 다른 debugger를 공부하는 것이었다. 그리고 program's registration scheme을 hex editor인 LordPE를 사용하여 patch하는 것이었다.

RVA, VA and offset were handled too.

RVA, VA and offset은 다뤄졌다.

I hope you understood everything fine and I also hope someone somewhere learned something from this. See ma back in part 09 ;)

네가 모든 것을 좋게 이해했기를 희망한다. 또한 누구든지 어디서든지 이것에서 무언가를 배웠으면 좋겠다. Part 09에서 보자.

The other parts are available at

다른 parts는 사용 가능하다.

<http://tinyurl.com/27dzdn> (tuts4you)

<http://tinyurl.com/r89zq> (SnD Filez)

<http://tinyurl.com/l6srv> (fixdown)

Regards to all and especially to you for taking the time to look at this tutorial.

Lena151 (2006, updated 2007)

모두에게 안부를 전하고 특별히 이 tutorial에 시간을 투자해준 너에게 감사한다.