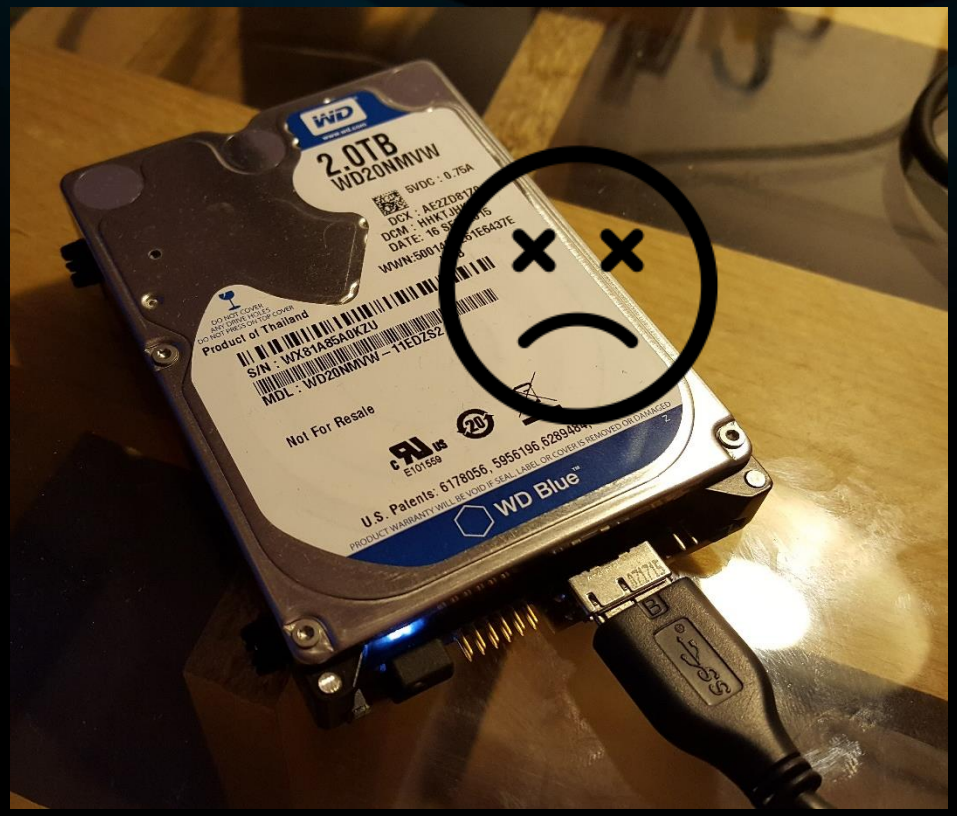
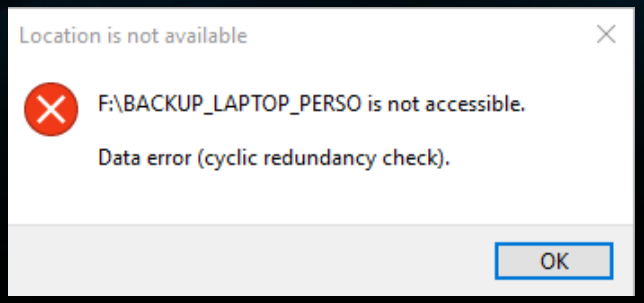
CRACKING SOFTWARE

For Fun & Profit

[[redacted] – 04.2018]

The beginning…

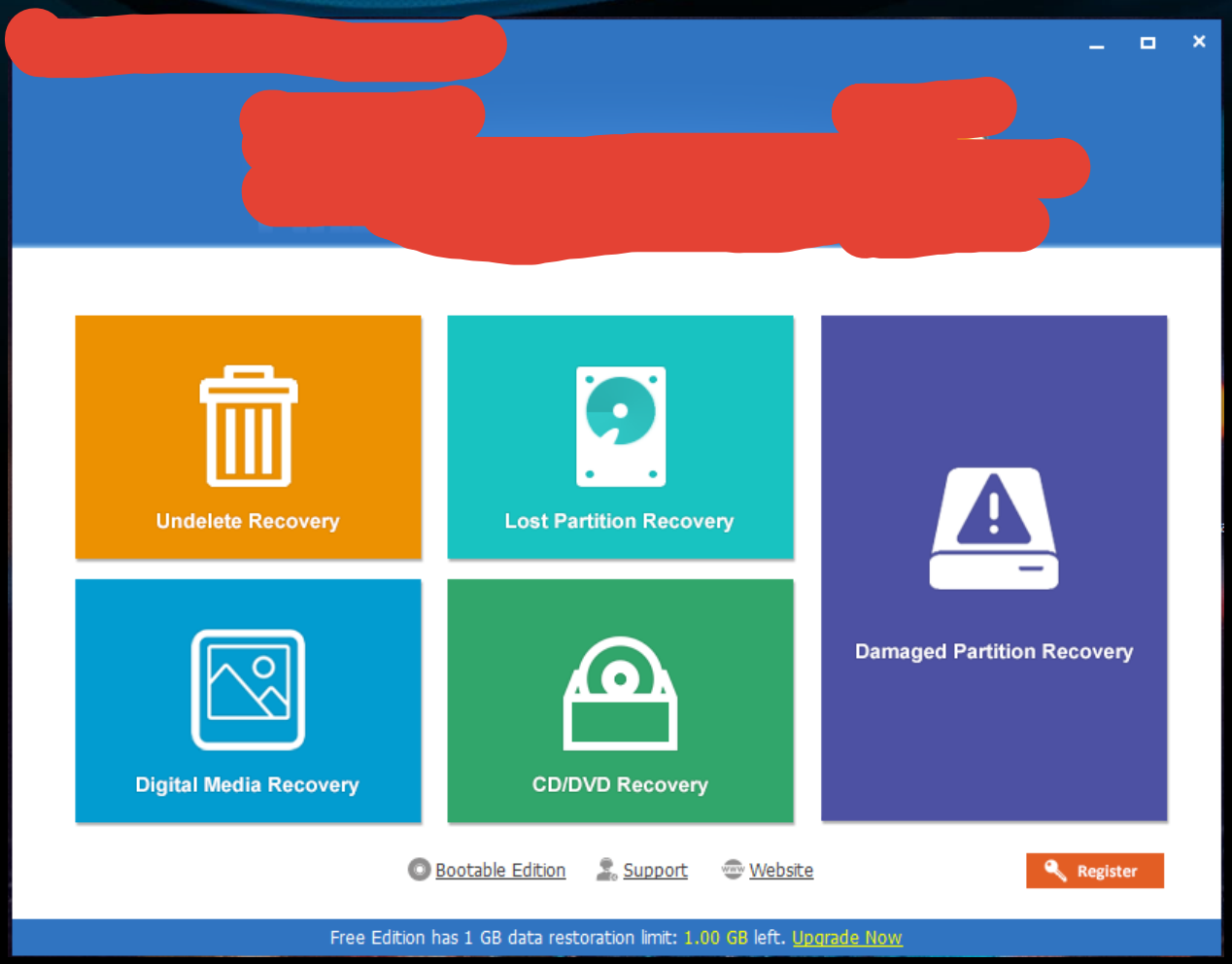




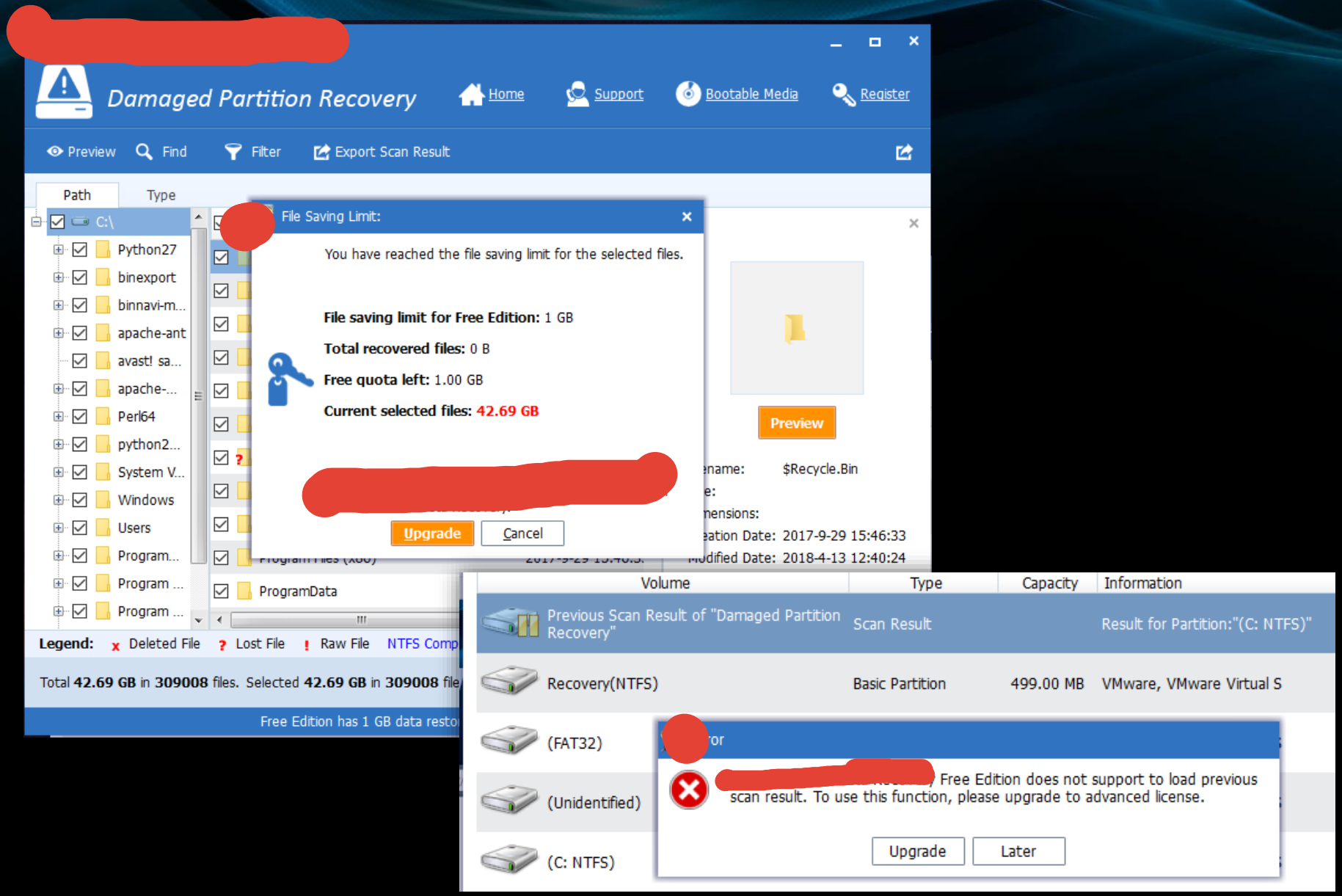
(almost) DEAD

1> INTRO

Working Software



But…


Damaged Partition Recovery
Home
Support
Bootable Media
Register
Preview
Find
Filter
Export Scan Result
Path
Type
C:\
Python27
binexport
binnavi-m...
apache-ant
avast! sa...
apache-...
Perl64
python2...
System V...
Windows
Users
Program...
Program ...
Program ...
ProgramData
Legend: Deleted File Lost File Raw File NTFS Comp
Total 42.69 GB in 309008 files. Selected 42.69 GB in 309008 file
Free Edition has 1 GB data resto
File Saving Limit:
You have reached the file saving limit for the selected files.
File saving limit for Free Edition: 1 GB
Total recovered files: 0 B
Free quota left: 1.00 GB
Current selected files: 42.69 GB
Upgrade
Cancel
Preview
name: $Recycle.Bin
ation Date: 2017-9-29 15:46:33
odified Date: 2018-4-13 12:40:24
Volume
Type
Capacity
Information
Previous Scan Result of "Damaged Partition Recovery"
Scan Result
Result for Partition:"(C: NTFS)"
Recovery(NTFS)
Basic Partition
499.00 MB
VMware, VMware Virtual S
(FAT32)
(Unidentified)
(C: NTFS)
Free Edition does not support to load previous scan result. To use this function, please upgrade to advanced license.
Upgrade
Later

I NEED TO CRACK THIS SOFT

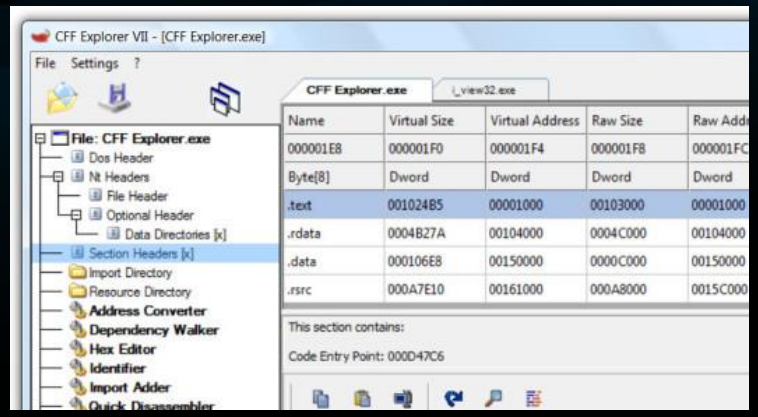
The Tools

- Static Analysis: ***IDA***

Last leak: IDA Pro 7.0.170914

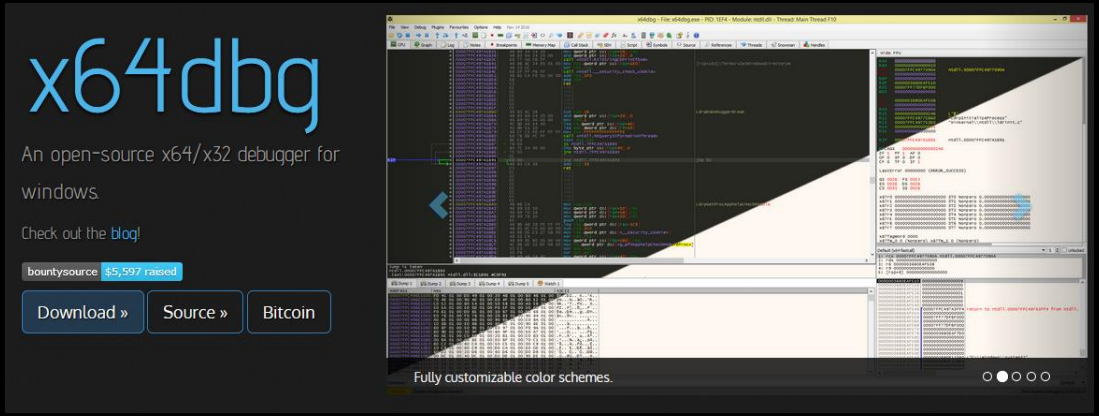
(x86, x64, include Hex-Rays Decompiler)

- PE Viewer / Editor: ***CFF Explorer***



http://www.ntcore.com/exsuite.php

- Debugger / Dynamic Analysis: ***x64dbg***

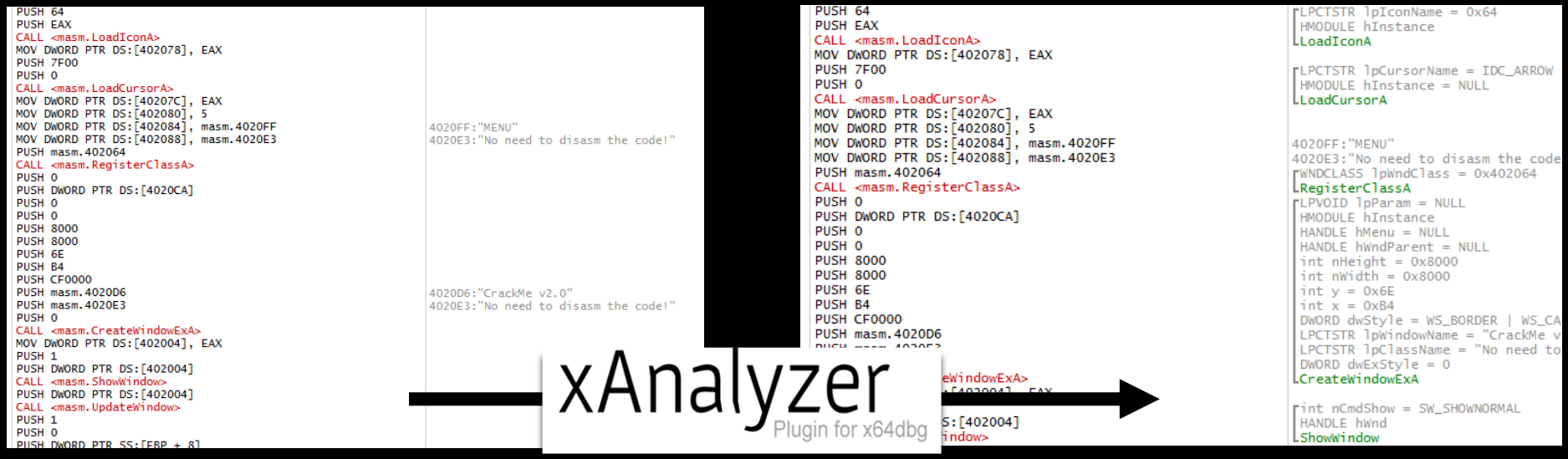


https://x64dbg.com/

x64dbg is awesome !

- Forget about ***Ollydbg*** and ***Immunity Debugger***
  - No support for x64 / No recent upgrades

- ***x64dbg*** is fully living project with regular updates (see github)
  - Nice interface, customizable
  - Includes graph view
  - Many plugins available: https://github.com/x64dbg/x64dbg/wiki/Plugins



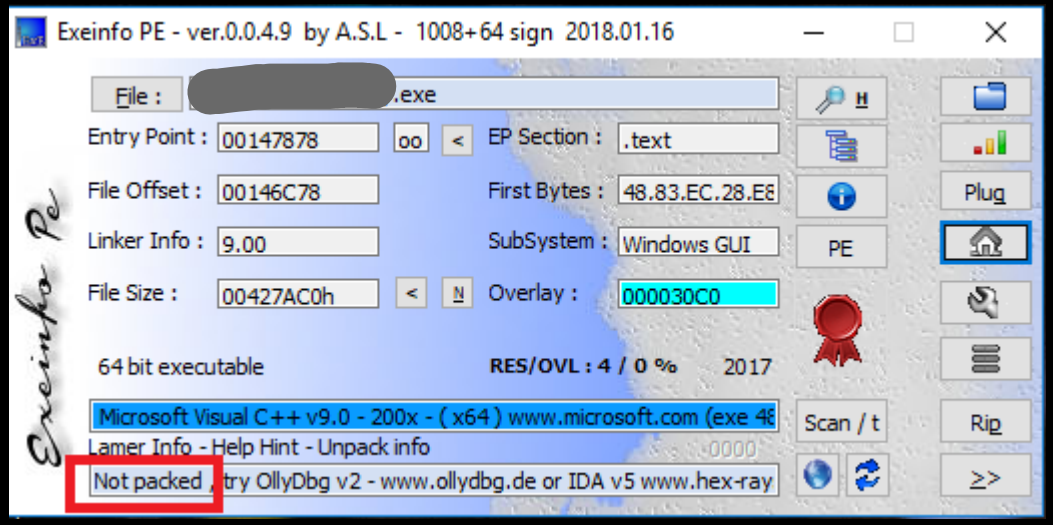
Before all, check for binary protection

- Check for the use of known packers/protectors
- ***PEiD*** is usually recommended, but not updated anymore !

=> ***ExeInfo PE*** has a large database of signatures and is still actively developed and maintained.

- For most common packers => tutorials available on the net.

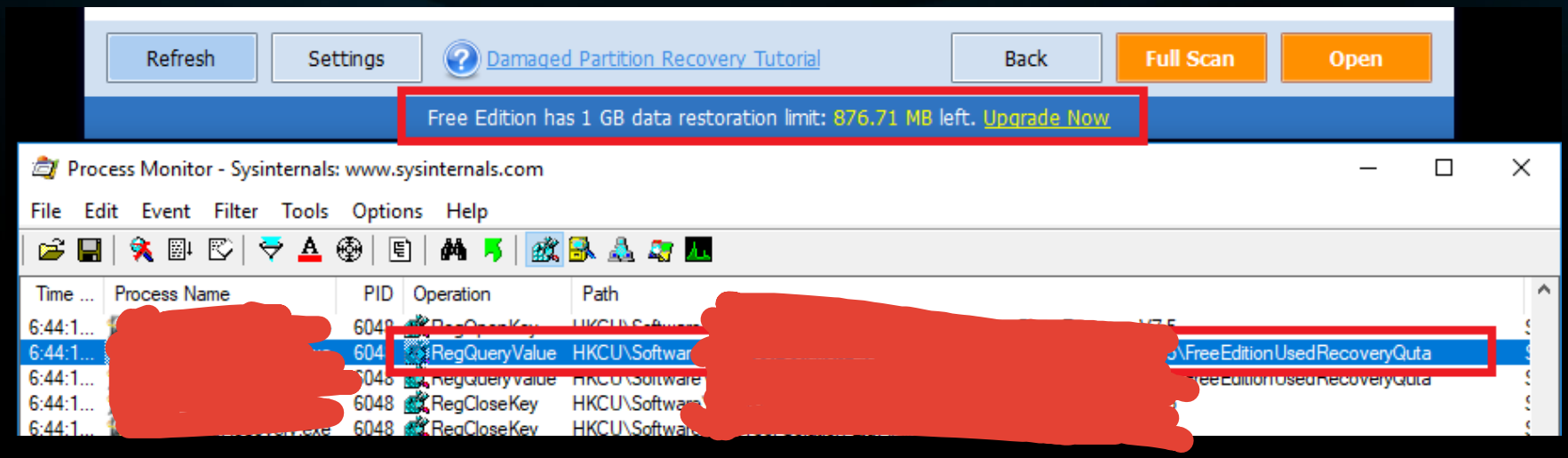
- Here, we are lucky, no packer is used …



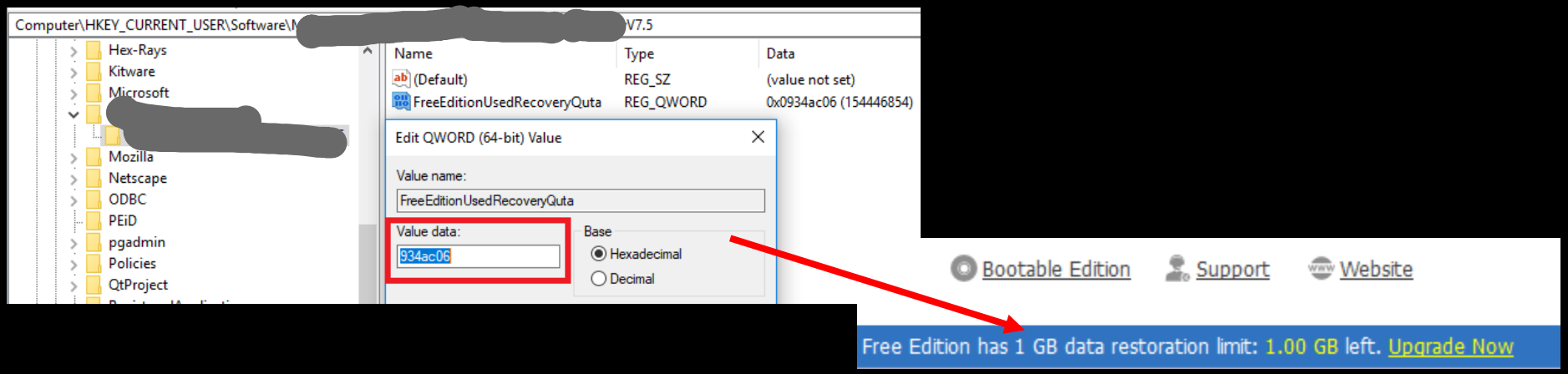
2> BASIC STUFF

The n00bz method

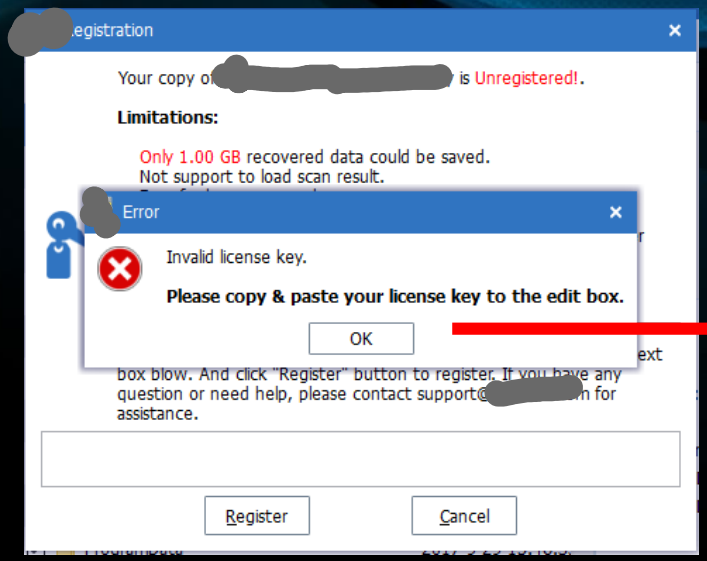
- Run the Free version & Monitor Registry activity with ***ProcMon***:



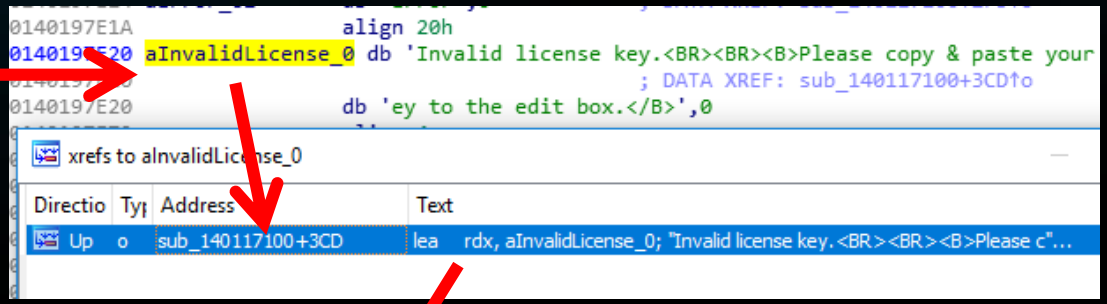
- Just replace the registry value by 0 to re-initialize the counter:

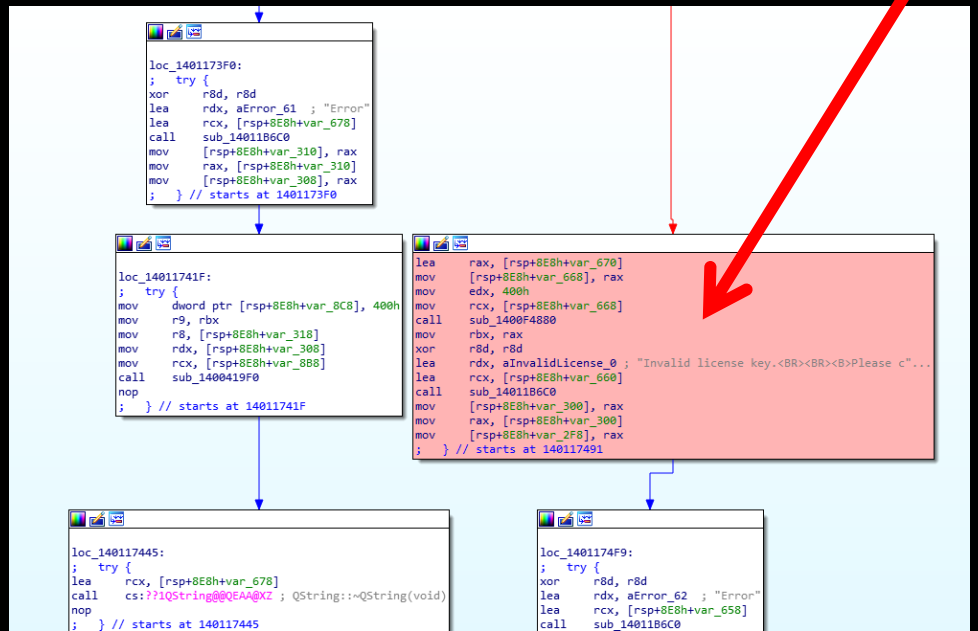


Locate the license key check routine



Search for string in IDA, go to XREF





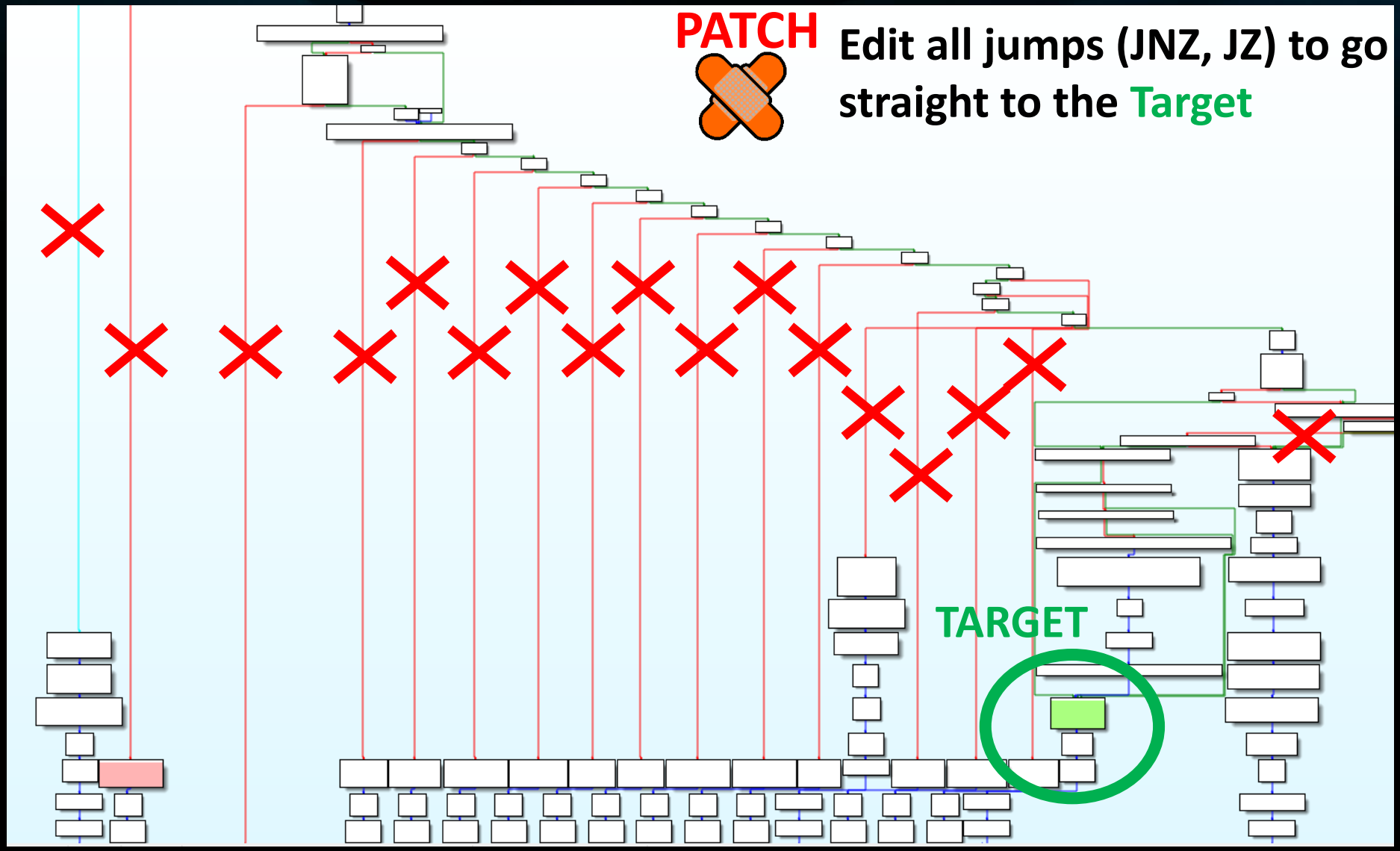
DeZoom

```
loc_140118930:
lea     rax, [rsp+8E8h+var_3A8]
mov     [rsp+8E8h+var_3A0], rax
mov     edx, 400h
mov     rcx, [rsp+8E8h+var_3A0]
call    sub_1400F4880
mov     rbx, rax
xor     r8d, r8d
lea     rdx, aThankYouForYou ; "Thank you for your registration!"
lea     rcx, [rsp+8E8h+var_308]
call    sub_14011B6C0
mov     [rsp+8E8h+var_30], rax
mov     rax, [rsp+8E8h+var_30]
mov     [rsp+8E8h+var_28], rax
;   } // starts at 140118922
```

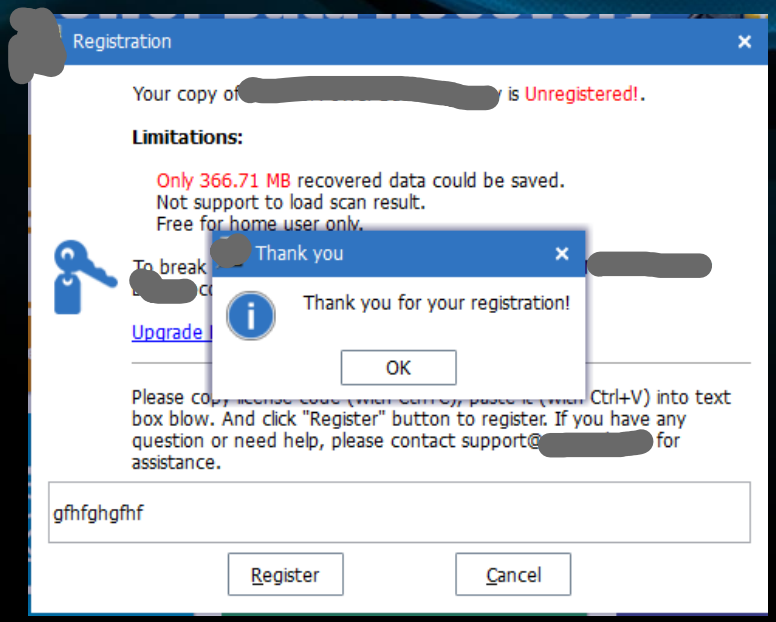
```
loc_1401177BF:
;   try {
cmp     [rsp+8E8h+var_8A8], 66h
jnz     loc_1401178A9
```

```
loc_1401178A9:
;   try {
cmp     [rsp+8E8h+var_8A8], 67h
jnz     loc_140117993
```

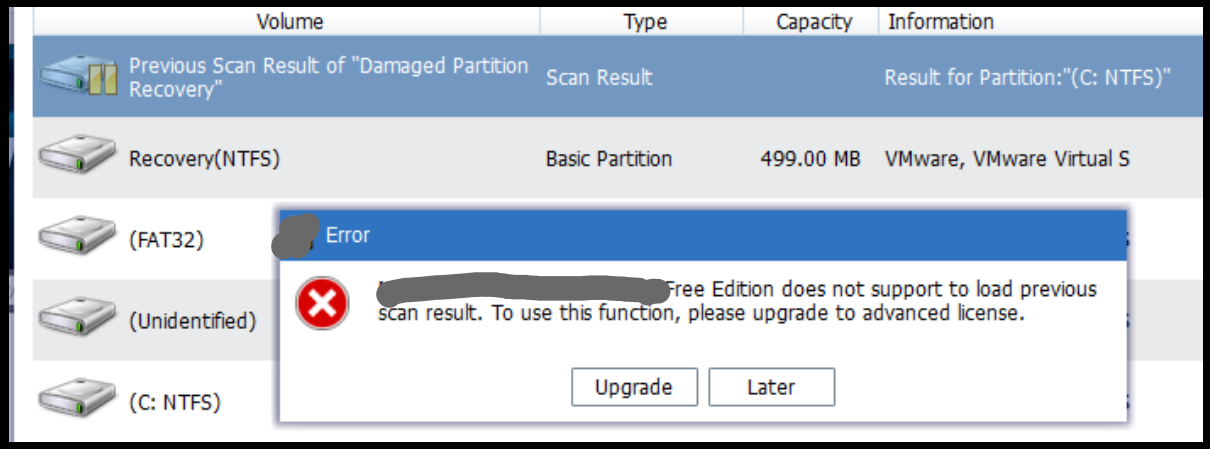
Patch the routine



Fail #1


Registration
Your copy of is Unregistered!.
Limitations:
Only 366.71 MB recovered data could be saved.
Not support to load scan result.
Free for home user only.
Thank you
Thank you for your registration!
OK
Upgrade
Ctrl+V) into text box blow. And click "Register" button to register. If you have any question or need help, please contact support@ for assistance.
gfhfghgfhf
Register
Cancel

But…


Volume
Type
Capacity
Information
Previous Scan Result of "Damaged Partition Recovery"
Scan Result
Result for Partition:"(C: NTFS)"
Recovery(NTFS)
Basic Partition
499.00 MB
VMware, VMware Virtual S
(FAT32)
(Unidentified)
(C: NTFS)
Error
Free Edition does not support to load previous scan result. To use this function, please upgrade to advanced license.
Upgrade
Later

3> LET’S GO DEEPER

What about licenses ?

https://w[redacted]

For Personal Users | For Business Users

Standard	Deluxe	Enterprise	Technician
Customer Reviews	Customer Reviews	Customer Reviews	Customer Reviews
Single license for 1 PC	Single license for 1 PC	Single license for 99 PCs	Single license for 299 PCs
1 Year Free Upgrade Service	Snap-in WinPE Bootable Builder	Snap-in WinPE Bootable Builder	Snap-in WinPE Bootable Builder
Commercial Use	Free life time Upgrade Service	1 Year Free Upgrade Service	1 Year Free Upgrade Service
Support Windows Server OS	Commercial Use	Commercial Use	Commercial Use
	Support Windows Server OS	Support Windows Server OS	Support Windows Server OS
U.S.$119.00	U.S.$199.00	U.S.$399.00	U.S.$499.00
Buy Now	Buy Now	Buy Now	Buy Now

Different license types

⇒ Check strings references in IDA again:

“Technician”, “Deluxe”…

Locate the license type retrieval

```
.rdata:000000014018B768 aBusinessStanda db 'Business Standard',0
.rdata:000000014018B768
.rdata:000000014018B77A                 align 20h
.rdata:000000014018B780 aBusinessDeluxe db 'Business Deluxe',0
.rdata:000000014018B790 aBusinessEnterp db 'Business Enterprise'
.rdata:000000014018B790
.rdata:000000014018B7A4                 align 8
.rdata:000000014018B7A8 aBusinessTechni db 'Business Technician'
.rdata:000000014018B7A8
```

xrefs to aBusinessTechni

Directio	Typ	Address	Text
Up	o	sub_1400C2820+507	lea rdx, aBusinessTechni; "Business Technician

We locate the function which maps license information to the corresponding name.

```
loc_1400C2C7B:          ; jumptable 00000001400C2ACB case 5
xor     r8d, r8d
lea     rdx, aBusinessDeluxe ; "Business Deluxe"
lea     rcx, [rsp+2B8h+var_1E8]
call    sub_1400C5E10
mov     [rsp+2B8h+var_A8], rax
mov     rax, [rsp+2B8h+var_A8]
mov     [rsp+2B8h+var_A0], rax
;   } // starts at 1400C2C68
```

```
loc_1400C2CD1:          ; jumptable 00000001400C2ACB case 6
xor     r8d, r8d
lea     rdx, aBusinessEnterp ; "Business Enterprise"
lea     rcx, [rsp+2B8h+var_1E0]
call    sub_1400C5E10
mov     [rsp+2B8h+var_98], rax
mov     rax, [rsp+2B8h+var_98]
mov     [rsp+2B8h+var_90], rax
;   } // starts at 1400C2CBE
```

```
loc_1400C2D24:          ; jumptable 00000001400C2ACB case 7
xor     r8d, r8d
lea     rdx, aBusinessTechni ; "Business Technician"
lea     rcx, [rsp+2B8h+var_1D8]
call    sub_1400C5E10
mov     [rsp+2B8h+var_88], rax
mov     rax, [rsp+2B8h+var_88]
mov     [rsp+2B8h+var_80], rax
;   } // starts at 1400C2D14
```

Graph pattern = Switch case

Patching the function used to check for license

Switch is made upon the value returned by function ***sub_14002F0E:***

```
mov     rcx, [rsp+2B8h+arg_0]
call    cs:?parent@QObject@@QEBAPEAV1@XZ ; QObject::parent(void)
mov     rcx, rax
call    sub_140017440
mov     [rsp+2B8h+var_270], rax
call    sub_1400307E0
call    ?_GetNumberOfVirtualProcessors@_CurrentScheduler@details@Concurrency@@SAIXZ ;
mov     [rsp+2B8h+var_260], eax
call    sub_1400307E0
call    sub_14002F0E0
movzx   eax, al
test    eax, eax
jnz     loc_1400C2A8D
```

License “Business Technician” corresponds to value = 7

```
sub_14002F0E0 proc near

var_18= qword ptr -18h

sub     rsp, 38h
call    sub_140030830
mov     [rsp+38h+var_18], rax
mov     rax, [rsp+38h+var_18]
mov     rax, [rax]
mov     rcx, [rsp+38h+var_18]
call    qword ptr [rax+8]
add     rsp, 38h
retn
sub_14002F0E0 endp
```

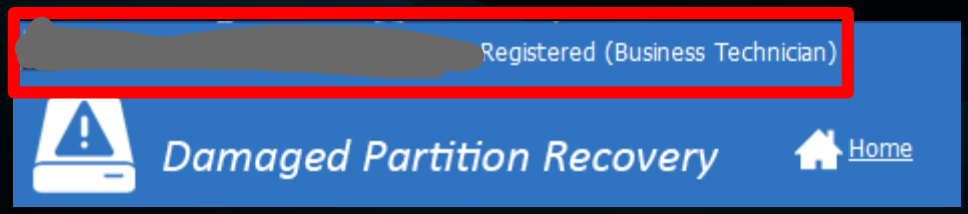
PATCH

```
licence_checker proc near

var_18= qword ptr -18h

sub     rsp, 38h
nop
nop
nop
nop
nop
nop
nop
mov     eax, 7
nop
add     rsp, 38h
retn
licence_checker endp
```

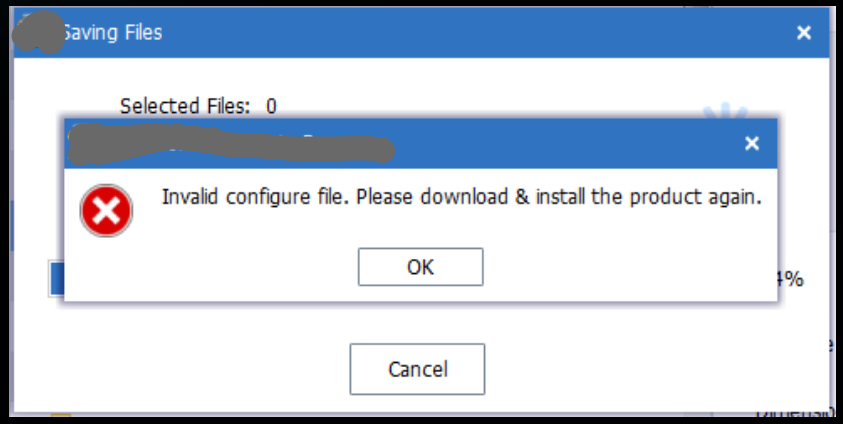
We make this function always returning

value = 7

Fail #2 ?



But, when saving files…



Features like accessing previously scanned device (not available in Free edition) are however working.

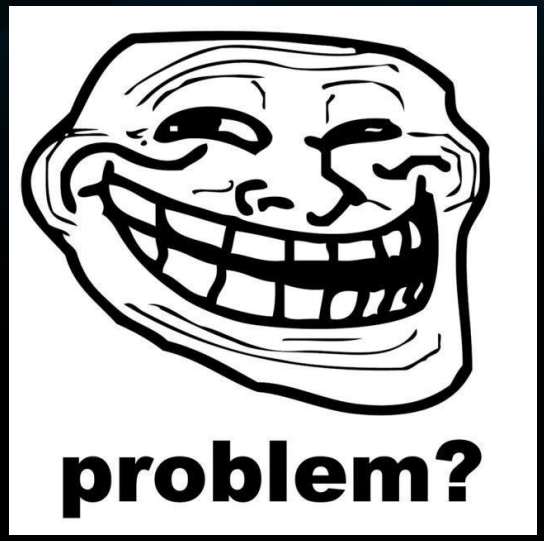
… BTW

[redacted] With Universal Crack

Share 29 Tweet 0 0 0

Download Now

[redacted] ll Version Cracked



Rohit says
December 30, 2017 at 5:09 pm

I installed the software and I took the crack (1) for my version and everything was alright until I found out that it says this error when I try to recover any file … Error- "Invalid Configure file. Please download and install the product again"

HELP ME OUTTTTT PLEASEEEE

Reply

4> TRACING

Instructions tracing

Idea: Run instructions tracing on suspected area (yellow block) with Free edition and our current Crack and make a **diff.**

Note: patching the jump (JZ) does not solve the problem => The copy takes place somewhere into the functions called from yellow block

```
loc_7FF680FB4353:
;   try {
call    display_file_tree
mov     rcx, rax
call    offset_1304
mov     [rsp+728h+var_C0], rax
mov     rax, [rsp+728h+var_C0]
mov     rax, [rax]
mov     rcx, [rsp+728h+var_C0]
call    qword ptr [rax+32]
mov     [rsp+728h+var_6C0], rax
call    sub_7FF680F007E0
lea     r8, [rsp+728h+var_6A0]
mov     edx, 8
lea     rcx, [rsp+728h+var_6C0]
call    call_check_mfh_file
lea     rax, [rsp+728h+var_368]
mov     [rsp+728h+var_360], rax
lea     rdx, [rsp+728h+str1] ; void *
mov     rcx, [rsp+728h+var_360] ; void *
call    cs:??0QString@@QEAA@AEBV0@@Z ; QString::QString(QString const &)
mov     [rsp+728h+var_B8], rax
mov     r8, [rsp+728h+var_B8]
lea     rdx, [rsp+728h+var_6A0]
mov     rcx, [rsp+728h+var__0]
call    change_val_offset144
mov     rcx, [rsp+728h+var__0]
call    return_val_arg_plus_144
movzx   eax, al
test    eax, eax
jz      loc_7FF680FB4552 ; problem: eax!=0
```

Branch taken in our current cracked version

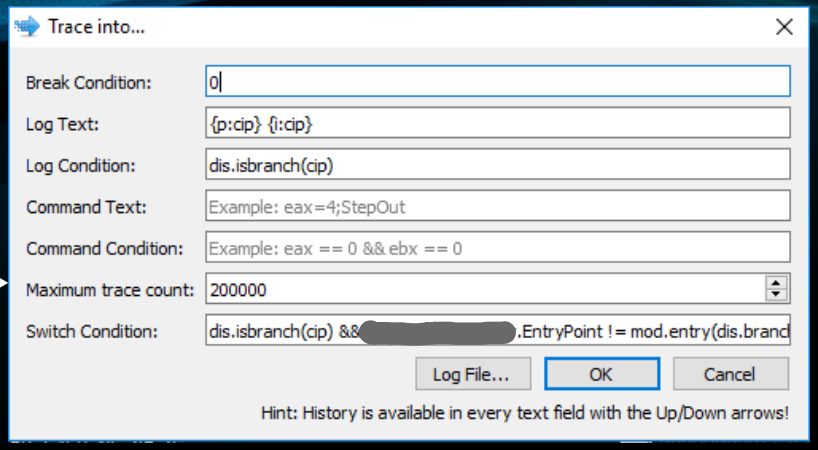
```
lea     rax, [rsp+728h+var_358]
mov     [rsp+728h+var_350], rax
mov     edx, 400h
mov     rcx, [rsp+728h+var_350]
call    copy_pointers
mov     rbx, rax
xor     r8d, r8d
lea     rdx, aInvalidConfigu ; "Invalid configure file. Please download"...
lea     rcx, [rsp+728h+var_348]
call    string_format
mov     [rsp+728h+var_B0], rax
mov     rax, [rsp+728h+var_B0]
mov     [rsp+728h+var_A8], rax
;   } // starts at 7FF680FB4353
```

```
loc_7FF680FB4552:
;   try {
mov     rcx, [rsp+728h+var__0]
call    cs:?result@QDialog@@QEBAHXZ ; QDialog::result(void)
test    eax, eax
jz      loc_7FF680FB4809
```

```
lea     rax, [rsp+728h+var_328]
mov     [rsp+728h+var_320], rax
mov     dl, 20h
lea     rcx, [rsp+728h+var_2FE]
call    char_to_byte
movzx   edx, byte ptr [rax]
lea     rcx, [rsp+728h+var_300]
call    cs:??0QChar@@QEAA@UQLatin1Char@@@Z ; QChar::QChar(QLatin1Char)
```

Trace into

Config x64dbg to “Trace into” (follow function calls) BUT **only inside the main binary** (do NOT trace into DLLs)



dis.isbranch(cip) &&
[redacted].EntryPoint != mod.entry(dis.branchdest(cip))

```
00007FF67FDC4388 | 4C 8D 84 24 88 00 00 00 | LEA R8, QWORD PTR SS:[RSP + 88]
00007FF67FDC4390 | BA 08 00 00 00          | MOV EDX, 8
00007FF67FDC4395 | 48 8D 4C 24 68          | LEA RCX, QWORD PTR SS:[RSP + 68]
00007FF67FDC439A | E8 A1 AD F4 FF          | CALL [redacted]7FF67FD0F140
00007FF67FD0F140 | 4C 89 44 24 18          | MOV QWORD PTR SS:[RSP + 18], R8
00007FF67FD0F145 | 89 54 24 10             | MOV DWORD PTR SS:[RSP + 10], EDX
00007FF67FD0F149 | 48 89 4C 24 08          | MOV QWORD PTR SS:[RSP + 8], RCX
00007FF67FD0F14E | 53                      | PUSH RBX
00007FF67FD0F14F | 48 83 EC 70             | SUB RSP, 70
00007FF67FD0F153 | 48 C7 44 24 58 FE FF FF | MOV QWORD PTR SS:[RSP + 58], FFFFFFFFFF
00007FF67FD0F15C | 48 8B 05 C5 8D 3A 00    | MOV RAX, QWORD PTR DS:[7FF6800B7F28]
00007FF67FD0F163 | 48 33 C4                | XOR RAX, RSP
00007FF67FD0F166 | 48 89 44 24 68          | MOV QWORD PTR SS:[RSP + 68], RAX
00007FF67FD0F16B | 48 8D 4C 24 28          | LEA RCX, QWORD PTR SS:[RSP + 28]
00007FF67FD0F170 | FF 15 F2 70 14 00       | CALL QWORD PTR DS:[<&std::basic_string<
00007FF67FD0F176 | 90                      | NOP
00007FF67FD0F177 | 4C 8D 44 24 28          | LEA R8, QWORD PTR SS:[RSP + 28]
00007FF67FD0F17C | 8B 94 24 88 00 00 00    | MOV EDX, DWORD PTR SS:[RSP + 88]
00007FF67FD0F183 | 48 8B 8C 24 80 00 00 00 | MOV RCX, QWORD PTR SS:[RSP + 80]
00007FF67FD0F18B | E8 D0 C8 FF FF          | CALL [redacted].7FF67FD0BA60
00007FF67FD0BA60 | 4C 89 44 24 18          | MOV QWORD PTR SS:[RSP + 18], R8
00007FF67FD0BA65 | 89 54 24 10             | MOV DWORD PTR SS:[RSP + 10], EDX
00007FF67FD0BA69 | 48 89 4C 24 08          | MOV QWORD PTR SS:[RSP + 8], RCX
00007FF67FD0BA6E | 53                      | PUSH RBX
00007FF67FD0BA6F | 57                      | PUSH RDI
00007FF67FD0BA70 | 48 81 EC C8 01 00 00    | SUB RSP, 1C8
00007FF67FD0BA77 | 48 C7 84 24 18 01 00 00 | MOV QWORD PTR SS:[RSP + 118], FFFFFFFFF
00007FF67FD0BA83 | 48 83 BC 24 E0 01 00 00 | CMP QWORD PTR SS:[RSP + 1E0], 0
00007FF67FD0BA8C | 74 0A                   | J[redacted]7FF67FD0BA98
00007FF67FD0BA8E | 83 BC 24 E8 01 00 00 00 | CMP DWORD PTR SS:[RSP + 1E8], 0
00007FF67FD0BA96 | 7F 07                   | [redacted]0BA9F
00007FF67FD0BA9F | 48 8D 4C 24 68          | [redacted] 68]
00007FF67FD0BAA4 | E8 77 43 00 00          | [redacted]FD0FE20
```

[...]
[...]
zZzZzzzZZ

More than 100 000 instructions !!

(including many loops and function calls)

=> Will take too much time to diff & analyze

Basic block tracing

- **Basic block:** straight-line code sequence with no branches (no jumps) in except to the entry and no branches (no jumps) out except at the exit.

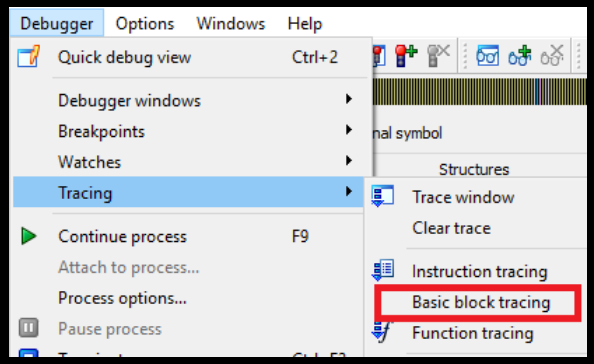
- Tools:
  - ***BinNavi*** (https://github.com/google/binnavi): Old and IDA exporter not working on IDA 7.x
  - ***IDA*** Plugin ***MyNav*** (https://github.com/joxeankoret/mynav): Really great, might be useful to find code path between 2 locations (see http://joxeankoret.com/blog/2010/05/02/mynav-a-python-plugin-for-ida-pro)
  - ***IDA*** feature:

Debugger > Tracing > Basic block tracing

(on newer versions of ***IDA***)

Use "Local Windows Debugger" to make it working.

https://www.hex-rays.com/products/ida/support/tutorials/replayer/trace_replayer.pdf



Basic block tracing Diff with IDA

Current Cracked Version

```
var_48= qword ptr -48h
var_38= dword ptr -38h
var_30= qword ptr -30h
var_28= byte ptr -28h
var_20= qword ptr -20h
var_18= qword ptr -18h
var_10= qword ptr -10h
arg_0= qword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h

; FUNCTION CHUNK AT 00007FF68103F940 SIZE 00000019 BYTES
; FUNCTION CHUNK AT 00007FF68103F960 SIZE 0000001C BYTES

mov     [rsp+arg_10], r8
mov     [rsp+arg_8], rdx
mov     [rsp+arg_0], rcx
sub     rsp, 58h
mov     [rsp+58h+var_18], 0FFFFFFFFFFFFFFFEh
```

```
sub     rsp, 20h
mov     rbp, rdx
mov     rcx, [rbp+
call    cs:??1QStr
add     rsp, 20h
pop     rbp
retn
; END OF FUNCTION
```

```
loc_7FF680FEE46C:
mov     [rsp+58h+var_38], 8
mov     rax, [rsp+58h+arg_0]
mov     byte ptr [rax+90h], 0
call    sub_7FF680F007E0
lea     r8, [rsp+58h+var_38]
lea     rdx, [rsp+58h+var_30]
mov     rcx, [rsp+58h+arg_8]
call    sub_7FF680EFF220
movzx   eax, al
test    eax, eax
jnz     short loc_7FF680FEE489
```

```
mov     rax, [rsp+58h+arg_0]
mov     byte ptr [rax+90h], 1
```

```
loc_7FF680FEE489:
lea     rax, [rsp+58h+var_28]
mov     [rsp+58h+var_20], rax
mov     rdx, [rsp+58h+arg_10] ; void *
mov     rcx, [rsp+58h+var_20] ; void *
call    cs:??0QString@@QEAA@AEBV0@@Z ; QString::QString(Q
mov     [rsp+58h+var_10], rax
mov     r8, [rsp+58h+var_10]
mov     rdx, [rsp+58h+var_30]
mov     rcx, [rsp+58h+arg_0]
call    sub_7FF680FEE510
nop
```

```
loc_7FF680FEE4AC:       ; void *
mov     rcx, [rsp+58h+arg_10]
call    cs:??1QString@@QEAA@XZ ; QString::~QString(void)
jmp     short loc_7FF680FEE4F8
```

```
loc_7FF680FEE4ED:       ; void *
mov     rcx, [rsp+58h+arg_10]
call    cs:??1QString@@QEAA@XZ ; QString::~Q
```

Free Edition

```
var_48= qword ptr -48h
var_38= dword ptr -38h
var_30= qword ptr -30h
var_28= byte ptr -28h
var_20= qword ptr -20h
var_18= qword ptr -18h
var_10= qword ptr -10h
arg_0= qword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h

; FUNCTION CHUNK AT 00007FF65F5DF940 SIZE 00000019 BYTES
; FUNCTION CHUNK AT 00007FF65F5DF960 SIZE 0000001C BYTES

mov     [rsp+arg_10], r8
mov     [rsp+arg_8], rdx
mov     [rsp+arg_0], rcx
sub     rsp, 58h
mov     [rsp+58h+var_18], 0FFFFFFFFFFFFFFFEh
```

```
mov     rbp, rdx
mov     rcx, [rbp+
call    cs:??1QStri
add     rsp, 20h
pop     rbp
retn
; END OF FUNCTION
```

```
loc_7FF65F58E46C:
mov     [rsp+58h+var_38], 8
mov     rax, [rsp+58h+arg_0]
mov     byte ptr [rax+90h], 0
call    sub_7FF65F4A07E0
lea     r8, [rsp+58h+var_38]
lea     rdx, [rsp+58h+var_30]
mov     rcx, [rsp+58h+arg_8]
call    sub_7FF65F49F220
movzx   eax, al
test    eax, eax
jnz     short loc_7FF65F58E489
```

```
mov     rax, [rsp+58h+arg_0]
mov     byte ptr [rax+90h], 1
```

```
loc_7FF65F58E489:
lea     rax, [rsp+58h+var_28]
mov     [rsp+58h+var_20], rax
mov     rdx, [rsp+58h+arg_10] ; void *
mov     rcx, [rsp+58h+var_20] ; void *
call    cs:??0QString@@QEAA@AEBV0@@Z ; QString::QString(QS
mov     [rsp+58h+var_10], rax
mov     r8, [rsp+58h+var_10]
mov     rdx, [rsp+58h+var_30]
mov     rcx, [rsp+58h+arg_0]
call    sub_7FF65F58E510
nop
```

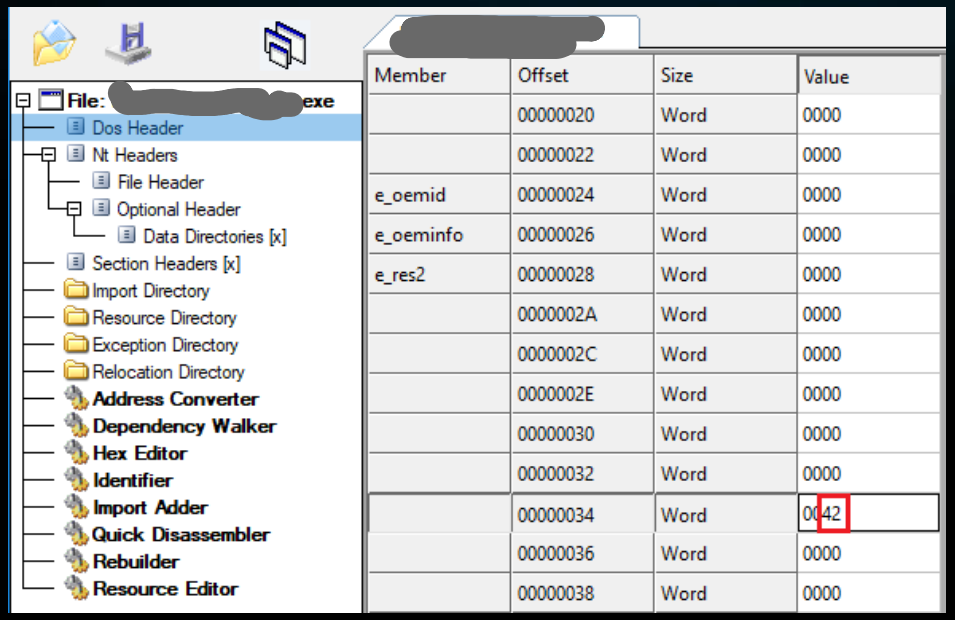
```
loc_7FF65F58E4AC:       ; void *
mov     rcx, [rsp+58h+arg_10]
call    cs:??1QString@@QEAA@XZ ; QString::~QString(void)
jmp     short loc_7FF65F58E4F8
```

```
loc_7FF65F58E4ED:       ; void *
mov     rcx, [rsp+58h+arg_10]
call    cs:??1QString@@QEAA@XZ ; QString::~QS
```

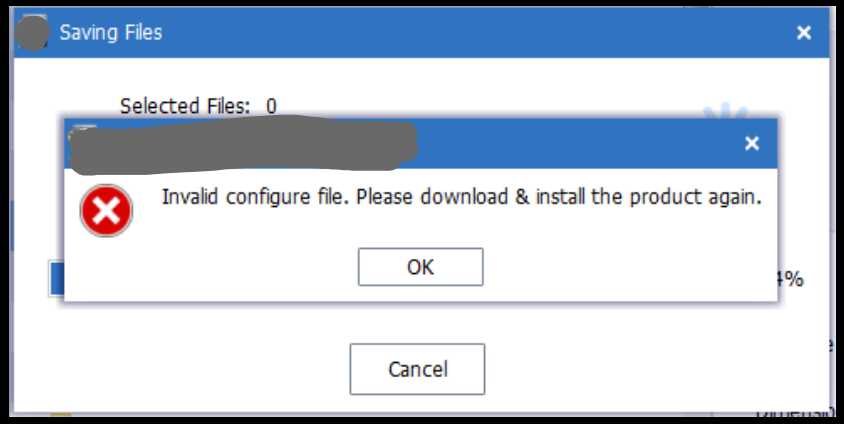
5> PROTECTION IDENTIFICATION

A little test

We take original version and just change 1 (useless) byte inside PE header:



And then we run again the soft, and check the “Saving Files” feature => Same error message as in our current crack



=> There must be a Binary Integrity Check implemented inside the soft !

Finding where the integrity check occurs

We run ProcMon on file system activity, just before the error message is displayed:

3188	ReadFile	C:\Program Files	.exe.mfh
3188	ReadFile	C:\Program Files	exe.mfh
3188	ReadFile	C:\Program Files	.exe
3188	ReadFile	C:\Program Files	
3188	ReadFile	C:\Program Files	\RawObjects.dll
3188	ReadFile	C:\Program Files	\efs.dll
3188	ReadFile	C:\Program Files	\efs.dll
3188	ReadFile	C:\Program Files	\Pas2.dll
3188	ReadFile	C:\Program Files	\Pas2.dll
3188	ReadFile	C:\Program Files	\PASServer.dll
3188	ReadFile	C:\Program Files	\PASServer.dll
3188	ReadFile	\Device\Harddisk0\DR0	
3188	ReadFile	C:\$Directory	

Call Stack:

Event Properties

Event | Process | Stack

Frame	Module	Location
K 0	FLTMGR.SYS	FltDecodeParameters + 0x19d2
K 1	FLTMGR.SYS	FltDecodeParameters + 0x1578
K 2	FLTMGR.SYS	FltDecodeParameters + 0x117b
K 3	FLTMGR.SYS	FltDecodeParameters + 0xf1e
K 4	ntoskrnl.exe	IofCallDriver + 0x59
K 5	ntoskrnl.exe	NtDeviceIoControlFile + 0xe9e
K 6	ntoskrnl.exe	NtReadFile + 0x68a
K 7	ntoskrnl.exe	setjmpex + 0x7e03
U 8	ntdll.dll	ZwReadFile + 0x14
U 9	KERNELBASE.dll	ReadFile + 0x76
U 10	QtCore4.dll	QString::contains + 0x610
U 11	QtCore4.dll	QFile::readData + 0xb3
U 12	QtCore4.dll	QIODevice::read + 0x2b6
U 13	QtCore4.dll	QIODevice::readAll + 0x11b
U 14		+ 0x3019d
U 15		+ 0x2ff13
U 16		+ 0x2c121
U 17		+ 0x2f3d4
U 18		+ 0x11e499

It reads the content of the main binary (.exe) and of 4 DLLs.

We check this last occurrence into the main .exe, in x64dbg

[redacted] exe file reading

Call responsible for binary reading

```
00007FF7515F0182  >  48 8D 4C 24 60     LEA RCX, QWORD PTR SS:[RSP + 60]
00007FF7515F0187     FF 15 9B 70 14 00  CALL QWORD PTR DS:[<&QString::~QString>]
00007FF7515F018D  >  48 8D 54 24 30     LEA RDX, QWORD PTR SS:[RSP + 30]
00007FF7515F0192     48 8D 4C 24 38     LEA RCX, QWORD PTR SS:[RSP + 38]
00007FF7515F0197     FF 15 AB 6B 14 00  CALL QWORD PTR DS:[<&QIODevice::readAll>]
00007FF7515F019B     90                 NOP
00007FF7515F019E     48 8D 4C 24 30     LEA RCX, QWORD PTR SS:[RSP + 30]
00007FF7515F01A3     FF 15 8F 6B 14 00  CALL QWORD PTR DS:[<&QString::size>]
00007FF7515F01A9     89 44 24 48        MOV DWORD PTR SS:[RSP + 48], EAX
00007FF7515F01AD     48 8D 4C 24 38     LEA RCX, QWORD PTR SS:[RSP + 38]
                                        ORD PTR DS:[<&QFile::close>]
                                        , QWORD PTR SS:[RSP + 1A0]
                                        , DWORD PTR DS:[RAX]
                                        RD PTR SS:[RSP + 48], EAX
```

QByteArray QIODevice::readAll()

Reads all remaining data from the device, and returns it as a byte array.

This function has no way of reporting errors; returning an empty QByteArray can mean either that no data was currently available for reading, or that an error occurred.

```
Address          Hex                                              ASCII
0000000010523040 01 00 00 00 C0 7A 42 00 C0 7A 42 00 00 00 00 00  ....ÀzB.ÀzB....
0000000010523050 58 30 52 10 00 00 00 00 4D 5A 90 00 03 00 00 00  X0R.....MZ......
0000000010523060 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00  ....ÿÿ..¸.......
0000000010523070 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  @...............
0000000010523080 00 00 00 00 00 00 00 00 00 00 00 00 42 00 00 00  ............B...
0000000010523090 00 00 00 00 00 01 00 00 0E 1F BA 0E 00 B4 09 CD  ......º.´.Í
00000000105230A0 21 B8 01 4C CD 21 54 68 69 73 20 70 72 6F 67 72  ¸.LÍ!This progr
00000000105230B0 61 6D 20 63 61 6E 6E 6F 74 20 62 65 20 72 75 6E  am cannot be run
00000000105230C0 20 69 6E 20 44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A  in DOS mode....
00000000105230D0 24 00 00 00 00 00 00 00 D6 6F AB 74 92 0E C5 27  $.....Öo«t..Å'
00000000105230E0 92 0E C5 27 92 0E C5 27 B5 C8 45 27 93 0E C5 27  ..Å'..Å'µÈE'..Å'
00000000105230F0 8C 5C 41 27 97 0E C5 27 2F 41 53 27 96 0E C5 27  \A'..Å'/AS'..Å'
0000000010523100 9B 76 50 27 90 0E C5 27 9B 76 46 27 86 0E C5 27  vP'..Å'.vF'..Å'
0000000010523110 B5 C8 A8 27 93 0E C5 27 B5 C8 BE 27 82 0E C5 27  µÈ¨'..Å'µÈ¾'..Å'
0000000010523120 9B 76 56 27 81 0E C5 27 92 0E C4 27 43 08 C5 27  vV'..Å'..Ä'C.Å'
0000000010523130 9B 76 41 27 22 0E C5 27 8C 5C 51 27 93 0E C5 27  vA'".Å'.\Q'..Å'
0000000010523140 9B 76 54 27 93 0E C5 27 52 69 63 68 92 0E C5 27  vT'..Å'Rich..Å'
0000000010523150 00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00  ........PE..d...
0000000010523160 1B 80 CD 59 00 00 00 00 00 00 00 00 F0 00 22 00  .ÍY........ð."..
0000000010523170 0B 02 09 00 00 50 17 00 00 06 2B 00 00 00 00 00  .....P....+.....
0000000010523180 78 78 14 00 00 10 00 00 00 00 00 00 40 01 00 00  xx..........@...
0000000010523190 [redacted] exe file 00 00 00 00 00 00 00 00  ................
00000000105231A0 00 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00  ................
```

QByteArray

Integrity Checks implementation

We put a breakpoint on previous function call, and then run ***x64dbg*** step by step (F8) to begin analysis of integrity checks:

```
CALL QWORD PTR DS:[<&QByteArray::begin>]
MOV RDX, RAX
MOV EAX, DWORD PTR SS:[RSP + 68]
IMUL EAX, DWORD PTR SS:[RSP + 20]
CDQE
ADD RDX, RAX
MOV R8D, DWORD PTR SS:[RSP + 20]
LEA RCX, QWORD PTR SS:[RSP + 78]
CALL QWORD PTR DS:[<&QByteArray::fromRawData>]
NOP
```

Read 0x6A5E0 (=435680) bytes from the binary (will loop over it)

```
QByteArray QByteArray::fromRawData(Buffer, 0x6A5E0)
```

```
MOV R8D, 1
LEA RDX, QWORD PTR SS:[RSP + 78]
LEA RCX, QWORD PTR SS:[RSP + D0]
CALL QWORD PTR DS:[<&QCryptographicHash::hash>]
```

Hash #1

```
MD5(partial_binary)
```

```
MOV RAX, QWORD PTR SS:[RSP + 130]
MOV QWORD PTR SS:[RSP + 138], RAX
MOV R8D, 2
MOV RDX, QWORD PTR SS:[RSP + 138]
LEA RCX, QWORD PTR SS:[RSP + 70]
CALL QWORD PTR DS:[<&QCryptographicHash::hash>]
```

Hash #2

```
SHA1(MD5(partial_binary))
```

```
LEA RCX, QWORD PTR SS:[RSP + D0]
CALL QWORD PTR DS:[<&QByteArray::~QByteArray>]
LEA RDX, QWORD PTR SS:[RSP + 70]
LEA RCX, QWORD PTR SS:[RSP + 50]
CALL [redacted]7FF7CA1F0AD0
NOP
LEA RCX, QWORD PTR SS:[RSP + 70]
CALL QWORD PTR DS:[<&QByteArray::~QByteArray>]
NOP
LEA RCX, QWORD PTR SS:[RSP + 78]
CALL QWORD PTR DS:[<&QByteArray::~QByteArray>]
```

Additional processing on hashes … (continue in the code with many functions and sub-functions called after)

Constant	Value	Description
QCryptographicHash::Md4	0	Generate an MD4 hash sum
QCryptographicHash::Md5	1	Generate an MD5 hash sum
QCryptographicHash::Sha1	2	Generate an SHA-1 hash sum

Let’s make a step back

One of the calling function (higher in call stack) returns what looks like the concatenation of several pseudo-hashes:

```
00007FF7CA1EC11C  E8 7F 3D 00 00   CALL [redacted]7FF7CA1EFEA0
00007FF7CA1EC121  48 89 84 24 78 01  MOV QWORD PTR SS:[RSP + 178], RAX
00007FF7CA1EC129  48 8B 84 24 78 01  MOV RAX, QWORD PTR SS:[RSP + 178]
00007FF7CA1EC131  48 89 84 24 80 01  MOV QWORD PTR SS:[RSP + 180], RAX
00007FF7CA1EC139  4C 8B 84 24 70 01  MOV R8, QWORD PTR SS:[RSP + 170]
00007FF7CA1EC141  48 8B 94 24 80 01  MOV RDX, QWORD PTR SS:[RSP + 180]

RAX 0000000000DFACD0
RBX 0000000000000000
RCX 0000000000DFABC0
RDX 0000000006D57360
RBP 00000000067BDE10
RSP 0000000000DFAC1

QWORD PTR [RSP+178]=[0000000000DFAD88]=1A
rax=0000000000DFACD0

.text:00007FF7CA1EC121 [redacted].exe:$2C121 #2B521

Default (x64 fastcall)
1: rcx 0000000000DFABC0
2: rdx 0000000006D57360
3: r8 0000000000000001
4: r9 0000000006FBA500
```

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct

Address	Hex	ASCII
0000000006D57360	01 00 00 00 F0 00 00 00 A0 00 00 00 0D F0 AD BA	. . . . ð . . . . . .ð.º
0000000006D57370	7A 73 D5 06 00 00 00 00 00 F0 61 00 39 00 66 00	zsÕ. . . . . ða 9 f
0000000006D57380	31 00 30 00 63 00 65 00 61 00 36 00 39 00 37 00	1 0 c e a 6 9 7
0000000006D57390	31 00 30 00 66 00 65 00 38 00 38 00 65 00 38 00	1 0 f e 8 8 e 8
0000000006D573A0	39 00 64 00 33 00 31 00 33 00 39 00 33 00 34 00	9 d 3 1 3 9 3 4
0000000006D573B0	36 00 63 00 36 00 34 00 66 00 34 00 30 00 65 00	6 c 6 4 f 4 0 e
0000000006D573C0	61 00 62 00 39 00 61 00 38 00 32 00 39 00 63 00	a b 9 a 8 2 9 c
0000000006D573D0	31 00 39 00 39 00 66 00 37 00 65 00 34 00 30 00	1 9 9 f 7 e 4 0
0000000006D573E0	35 00 35 00 37 00 31 00 61 00 66 00 39 00 62 00	5 5 7 1 a f 9 b
0000000006D573F0	37 00 61 00 65 00 64 00 32 00 63 00 61 00 63 00	7 a e d 2 c a c
0000000006D57400	32 00 38 00 64 00 36 00 36 00 34 00 33 00 32 00	2 8 d 6 6 4 3 2
0000000006D57410	36 00 33 00 34 00 35 00 65 00 61 00 32 00 66 00	6 3 4 5 e a 2 f
0000000006D57420	65 00 65 00 63 00 33 00 38 00 62 00 65 00 32 00	e e c 3 8 b e 2
0000000006D57430	38 00 36 00 66 00 31 00 63 00 33 00 39 00 65 00	8 6 f 1 c 3 9 e
0000000006D57440	37 00 62 00 31 00 65 00 37 00 65 00 30 00 37 00	7 b 1 e 7 e 0 7
0000000006D57450	65 00 65 00 30 00 34 00 38 00 63 00 61 00 34 00	e e 0 4 8 c a 4
0000000006D57460	38 00 61 00 65 00 35 00 33 00 61 00 35 00 34 00	8 a e 5 3 a 5 4
0000000006D57470	65 00 31 00 35 00 35 00 35 00 39 00 34 00 34 00	e 1 5 5 5 9 4 4
0000000006D57480	61 00 35 00 34 00 37 00 39 00 61 00 31 00 37 00	a 5 4 7 9 a 1 7
0000000006D57490	32 00 39 00 63 00 38 00 34 00 34 00 35 00 66 00	2 9 c 8 4 4 5 f
0000000006D574A0	31 00 30 00 61 00 63 00 37 00 32 00 61 00 61 00	1 0 a c 7 2 a a
0000000006D574B0	32 00 32 00 30 00 36 00 66 00 00 00 0D F0 AD BA	2 2 0 6 f . . . .ð.º
0000000006D574C0	0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA	.ð.º.ð.º.ð.º.ð.º

Pointer

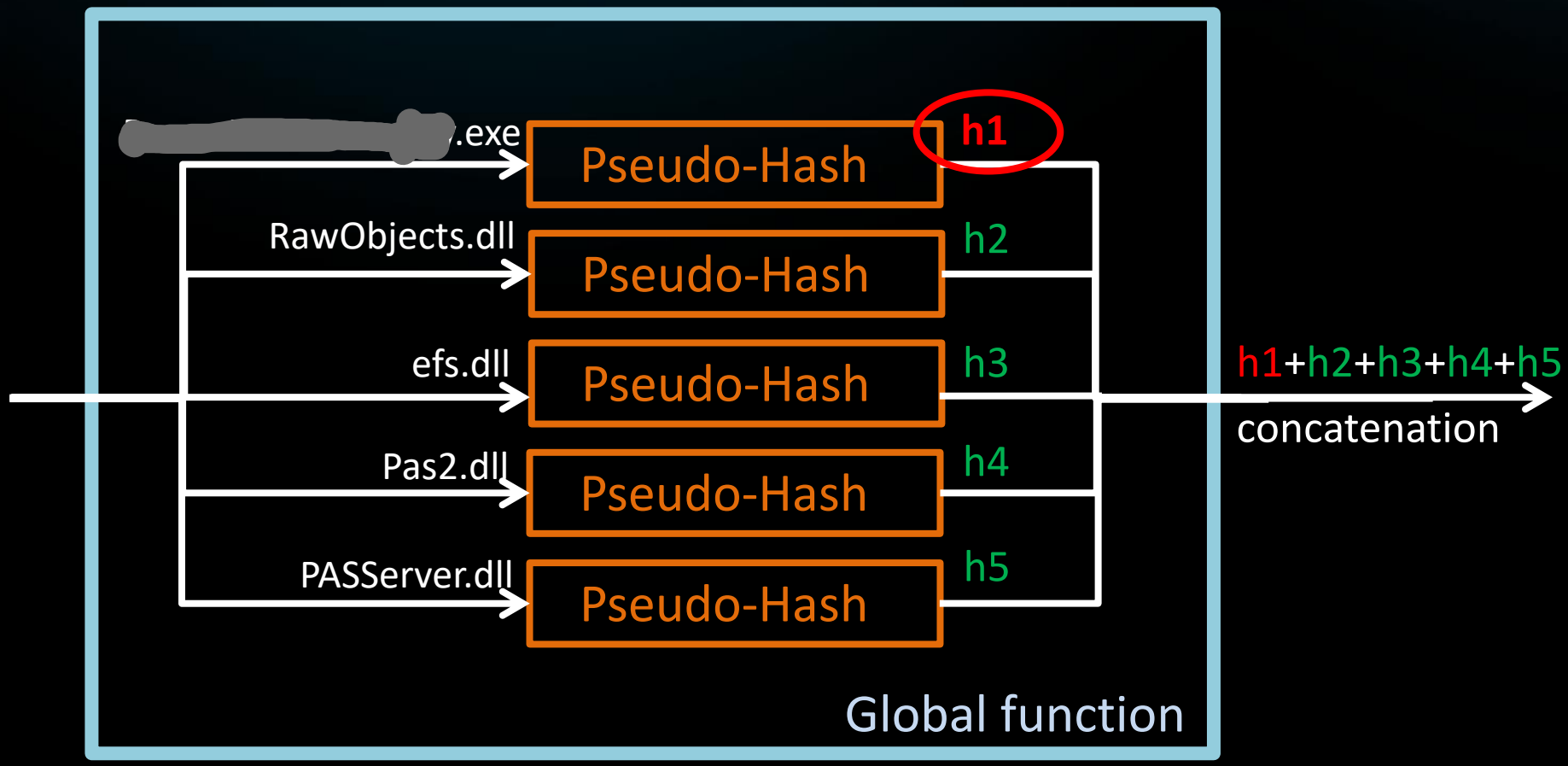
Return in current crack

Return in original version

ASCII
BA . . . ð . . . . . .ð.º
00 *ëE. ð0 4 5
00 6 6 f f 7 d 8 e
00 5 1 e 1 6 e c d
00 1 3 9 b 6 d f 7
00 2 8 8 a 7 4 0 e
00 a b 9 a 8 2 9 c
00 1 9 9 f 7 e 4 0
00 5 5 7 1 a f 9 b
00 7 a e d 2 c a c
00 2 8 d 6 6 4 3 2
00 6 3 4 5 e a 2 f
00 e e c 3 8 b e 2
00 8 6 f 1 c 3 9 e
00 7 b 1 e 7 e 0 7
00 e e 0 4 8 c a 4
00 8 a e 5 3 a 5 4
00 e 1 5 5 5 9 4 4
00 a 5 4 7 9 a 1 7
00 2 9 c 8 4 4 5 f
00 1 0 a c 7 2 a a
BA 2 2 0 6 f . . . .ð.º
BA .ð.º.ð.º.ð.º.ð.º

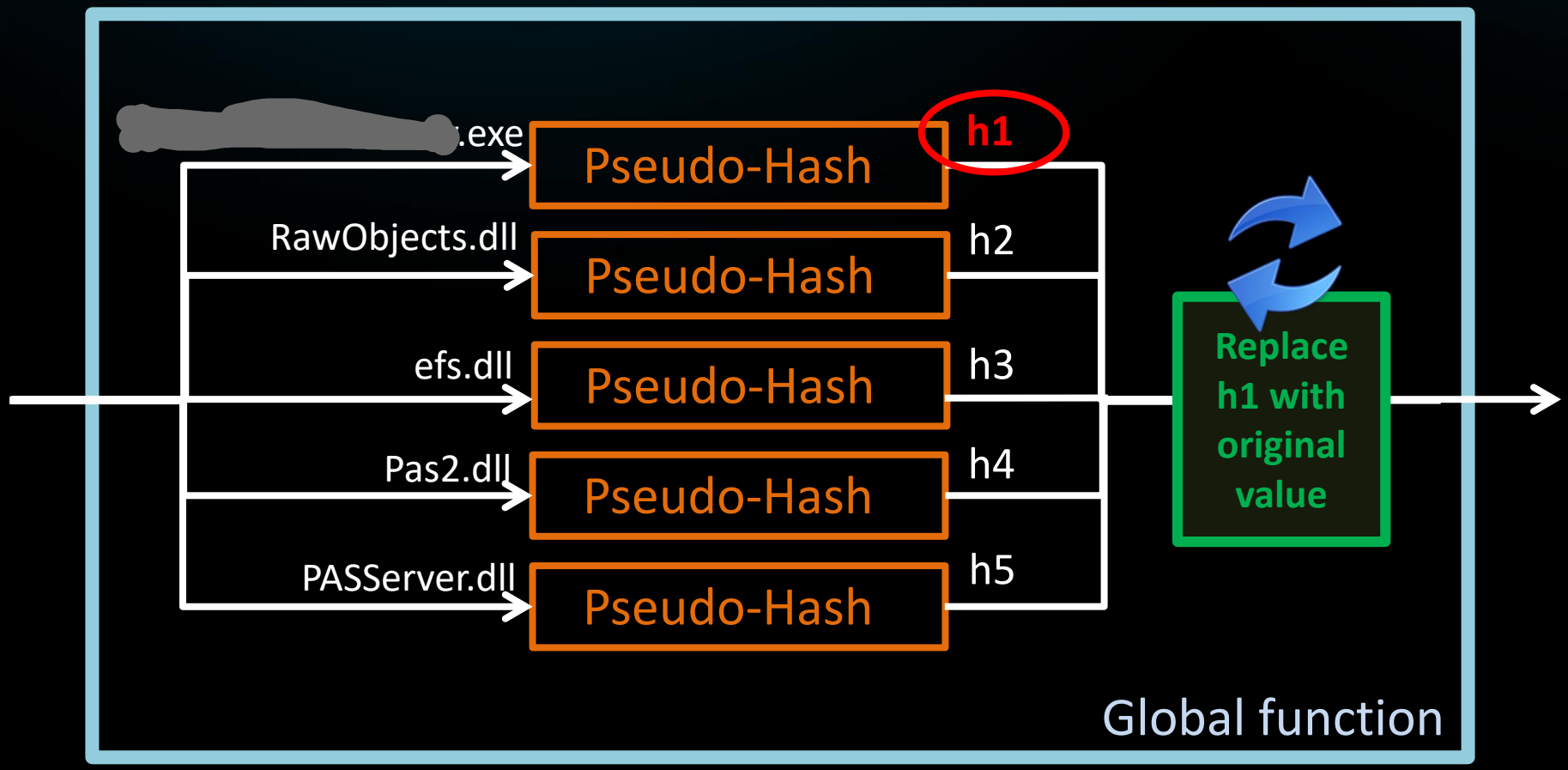
Summary

Because we edited [redacted].exe, the corresponding pseudo-hash differs => If we edit it in memory while debugging, no error message is displayed anymore !



6> PROTECTION BYPASS

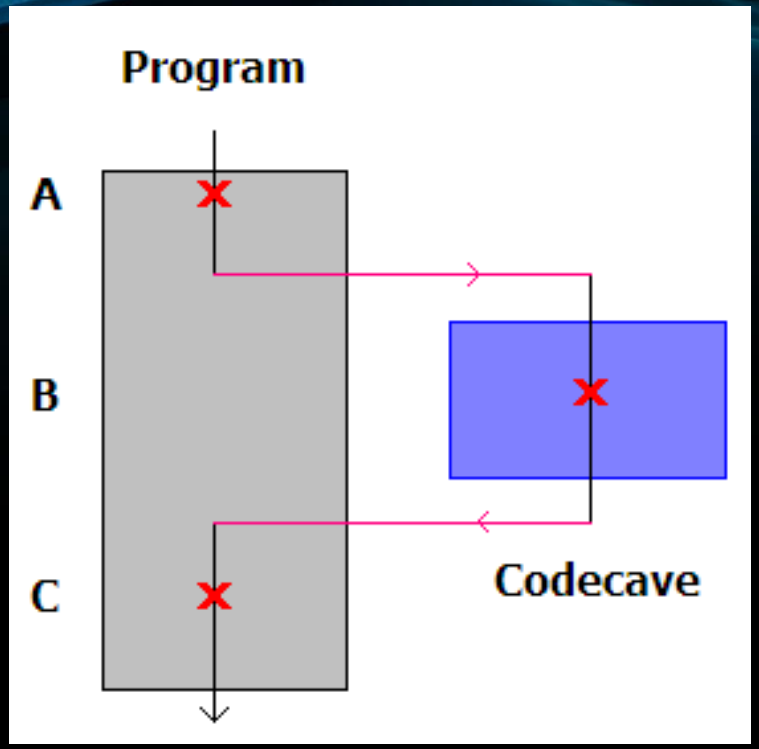
Making the change permanent



Code Cave (1/2)

- We will insert ASM code ("Code Cave") that will change the pseudo-hash h1 with its original value.

- A Jump to this "Code Cave" is added at the end of the "Global function"



- Empty code at the end of binary where we can host the "Code Cave":

```
00007FF7D1175FAF    CC                  INT3
00007FF7D1175FB0    48 8D 0D D1 1F 26   LEA RCX, QWORD PTR DS:[7FF7D13D7F88]
00007FF7D1175FB7    48 FF 25 72 05 00   JMP QWORD PTR DS:[<&VariantClear>]
00007FF7D1175FBE    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FC0    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FC2    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FC4    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FC6    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FC8    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FCA    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FCC    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FCE    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FD0    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FD2    00 00               ADD BYTE PTR DS:[RAX], AL
00007FF7D1175FD4    00 00               ADD BYTE PTR DS:[RAX], AL
```

Code Cave (2/2)

```
00007FF61FF6FF60   FF 15 C2 72 14 00   CALL QWORD PTR DS:[<&QString::~QString>]
00007FF61FF6FF66   48 8B 44 24 78      MOV RAX, QWORD PTR SS:[RSP + 78]
00007FF61FF6FF6B   E9 50 60 14 00      JMP [redacted].7FF6200B5FC0
00007FF61FF6FF70   48 83 C4 68         ADD RSP, 68
00007FF61FF6FF74   C3                  RET
00007FF61FF6FF75   CC                  INT3
```

JUMP to Code Cave

```
00007FF6200B5FAF   CC                  INT3
00007FF6200B5FB0   48 8D 0D D1 1F 26   LEA RCX, QWORD PTR DS:[7FF620317F88]
00007FF6200B5FB7   48 FF 25 72 05 00   JMP QWORD PTR DS:[<&VariantClear>]
00007FF6200B5FBE   00 00               ADD BYTE PTR DS:[RAX], AL
00007FF6200B5FC0   52                  PUSH RDX
00007FF6200B5FC1   56                  PUSH RSI
00007FF6200B5FC2   57                  PUSH RDI
00007FF6200B5FC3   53                  PUSH RBX
00007FF6200B5FC4   48 8B 38            MOV RDI, QWORD PTR DS:[RAX]
00007FF6200B5FC7   48 83 C7 1A         ADD RDI, 1A
00007FF6200B5FCB   48 8D 35 8C EE 25   LEA RSI, QWORD PTR DS:[7FF620314E5E]
00007FF6200B5FD2   48 C7 C2 00 00 00   MOV RDX, 0
00007FF6200B5FD9   8B 1C 16            MOV EBX, DWORD PTR DS:[RSI + RDX]
00007FF6200B5FDC   89 1C 17            MOV DWORD PTR DS:[RDI + RDX], EBX
00007FF6200B5FDF   48 83 C2 04         ADD RDX, 4
00007FF6200B5FE3   48 83 FA 40         CMP RDX, 40
00007FF6200B5FE7   75 F0               JNE [redacted]7FF6200B5FD9
00007FF6200B5FE9   5B                  POP RBX
00007FF6200B5FEA   5F                  POP RDI
00007FF6200B5FEB   5E                  POP RSI
00007FF6200B5FEC   5A                  POP RDX
00007FF6200B5FED   E9 7E 9F EB FF      JMP [redacted].7FF61FF6FF70
00007FF6200B5FF2   90                  NOP
00007FF6200B5FF3   90                  NOP
```

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct

ss	Hex	ASCII
FF620314E5E	30 00 34 00 35 00 36 00 36 00 66 00 66 00 37 00	0.4.5.6.6.f.f.7.
FF620314E6E	64 00 38 00 65 00 35 00 31 00 65 00 31 00 36 00	d.8.e.5.1.e.1.6.
FF620314E7E	65 00 63 00 64 00 31 00 33 00 39 00 62 00 36 00	e.c.d.1.3.9.b.6.
FF620314E8E	64 00 66 00 37 00 32 00 38 00 38 00 61 00 37 00	d.f.7.2.8.8.a.7.

Original value of pseudo-hash for [redacted].exe

WIN