Tutorial will describe manually unpacking yoda Protector newest version 1.03.3. Tutorial will focus on main yP problem, running protected file under debugger.

Introduction:

As I said, yoda's Protector 1.03.3 is last yoda protector version and author has decided to stop project. This tutorial will show how anti-debug tricks can be easy avoided and bypassed.

Target and some tools are needed :

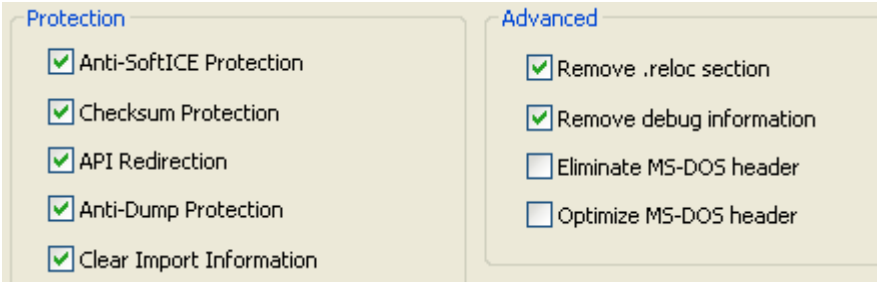
- OllyDbg 1.10
- LordPE
- ImpREC or PE Tools
- Windows XP
- Target : Yoda's Protector 1.03.3

yoda Protector is based on yoda's Cryptor frame, only that new tricks are added. Old tricks are :

- PE header erasing (which is pointless)
- CRC checking (code and file)
- IsDebuggerPresent check
- API redirecting and destroying import information.

New tricks are terminating Olly and possible freesing Windows XP.

Yoda's Options and new tricks :

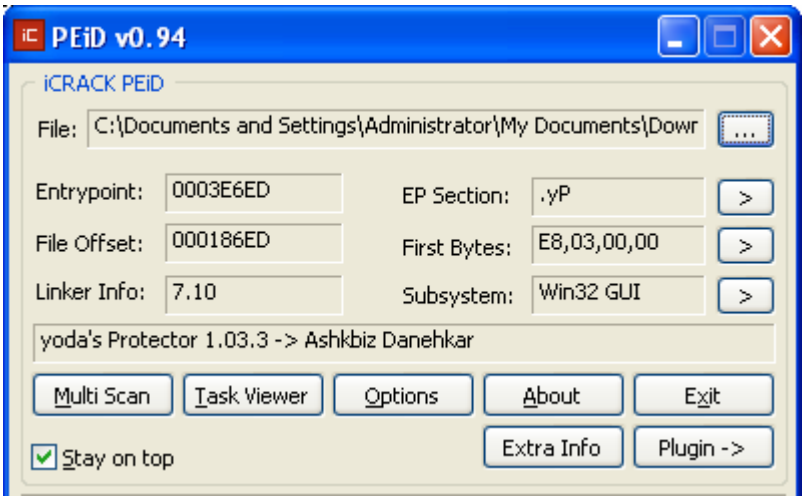


Let's see how Olly is killed. Protector is using combination of API's to get PID number of all running processes. Then it search for process that started (ollydbg in our case) and terminate it. It compares PID of that process with it's own PID. If those PID's are not same, it will terminate that process.

Second trick is more annoying. Protector will use BlockInput API before any other check. That API blocks input devices (mouse, keyboard, ..) so we are blocked from our system. Then protector will do other checks and decrypting. If in meantime protector stops on some exception or Olly is found , our system will wait for us to take action but we cannot do nothing except restart windows. If everything is passed fine, protector will again use BlockInput API to unblock input devices. Pretty smart trick.

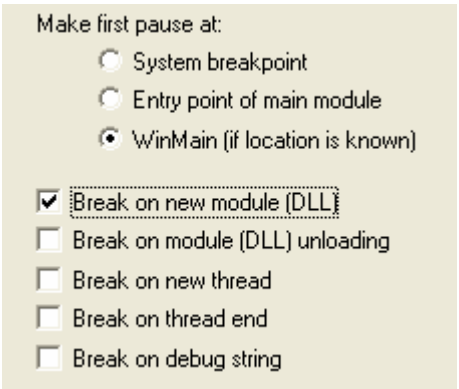
Target analysing :

– First, we use PeiD to analyse and get more target's information :

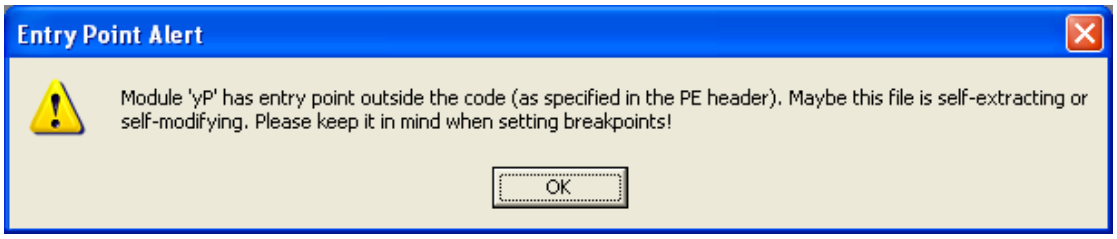


Reaching OEP :

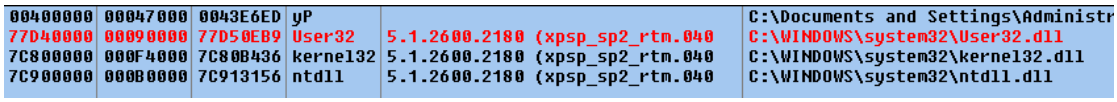
– Before manual unpacking, you need configure OllyDBG's options follow below image to OllyDBG break at module **user32.DLL**, prevent **BlockInput** API excutation:



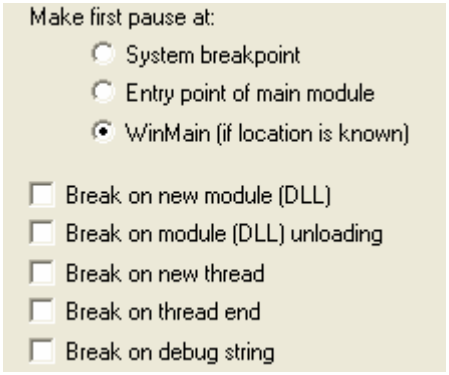
– Load target into OllyDBG, you see a meesagebox appear, press OK to continue :



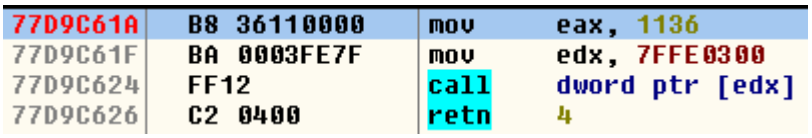
– Then press F9 untill you see that User32.dll is loaded:



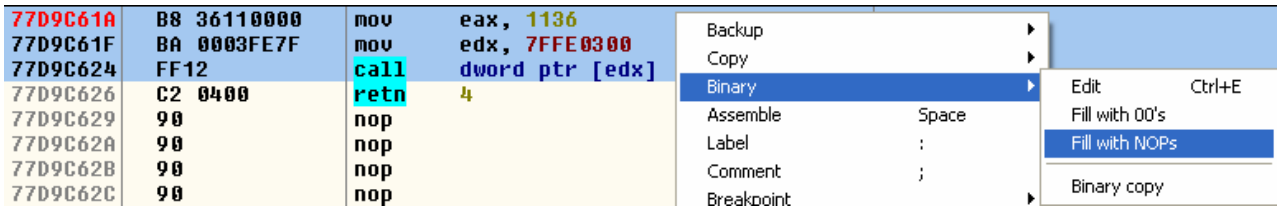
– After, you configure OllyDBG's options as orginal :



– After that we can uncheck option for breaking on new module. Now we need just to patch BlockInput API so it doesn't block devices. Simply select, "go to, expression" and enter BlockInput. Ok and we land in **user32.dll** on that API (this looks on my system):



– To kill API, just NOP all to RETN 4:



– Press F2 to set BP at RETN 4 :

```
77D9C61A   90              nop
77D9C61B   90              nop
77D9C61C   90              nop
77D9C61D   90              nop
77D9C61E   90              nop
77D9C61F   90              nop
77D9C620   90              nop
77D9C621   90              nop
77D9C622   90              nop
77D9C623   90              nop
77D9C624   90              nop
77D9C625   90              nop
77D9C626   C2 0400         retn    4
```

– We have killed this API and with that we avoid blocking devices, but we need to prevent Olly killing. There is similar simple solution for that. yoda uses CreateToolhelp32Snapshot to get all processes and couple others to walk trough all processes. But it uses GetCurrentProcessId to get PID of itself. Then yoda will check is process who started it has same PID as itself (ei. did protected file started trough some debugger or not) and if not, it will terminate that process. We can do next to prevent killing Olly.

– Open LordPE or PE Tools and get PID of OllyDbg.exe. In my case, PID is 888 (your computer is different).

```
c:\program files\internet download manager\i...   000008A4   00400000   00040000
c:\program files\microsoft office\office10\win...  000009A4   30000000   00A41000
e:\cracking\tools\ollydbg collection\ollydbg_...   00000888   00400000   00164000
e:\cracking\tools\lord pe 1.4\lordpe.exe          00000E64   00400000   00036000
```

– "Go to , expression", enter **GetCurrentProcessId** and click ok. You are in API:

```
7C80994E   64:A1 18000000   mov     eax, dword ptr fs:[18]
7C809954   8B40 20          mov     eax, dword ptr [eax+20]
7C809957   C3               retn
```

– That api will return PID of protected file, but I will patch it to return Olly PID. And our protected file will think that it is Olly itself, check:

```
7C80994E   64:A1 18000000   mov     eax, dword ptr fs:[18]
7C809954   8B40 20          mov     eax, dword ptr [eax+20]
7C809957   C3               retn
7C809958   90               nop
7C809959   90               nop
7C80995A   90               nop
7C80995B   90               nop
```

Backup ▸
Copy ▸
Binary ▸ — Edit Ctrl+E; Fill with 00's; Fill with NOPs
Assemble Space
Label :

– You edit instruction at address **7C80994E** follow :

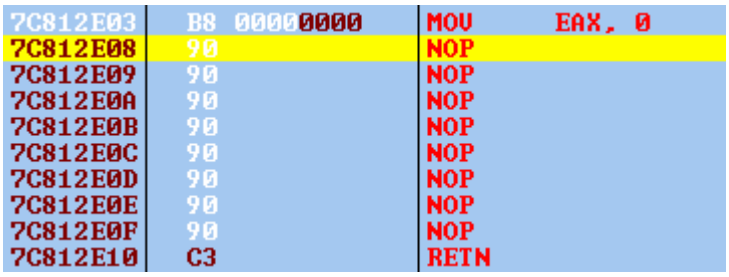
MOV EAX, OllyDBG's PID (mean is 888)

```
7C80994E   B8 88080000   MOV     EAX, 888
7C809953   90            NOP
7C809954   90            NOP
7C809955   90            NOP
7C809956   90            NOP
7C809957   C3            RETN
```

– Continue, you must kill **IsDebuggerPresent** by one of all :

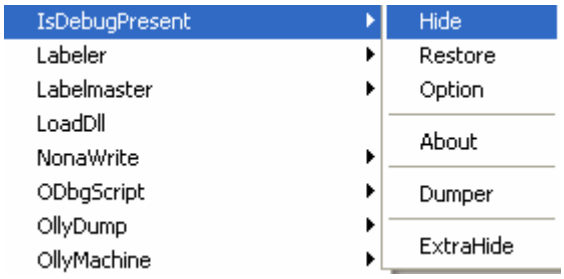
Method 1 : (Manual Fix)

– Press CTRL + G, and type **IsDebuggerPresent**, you NOP and edit to **MOV EAX, 0** :



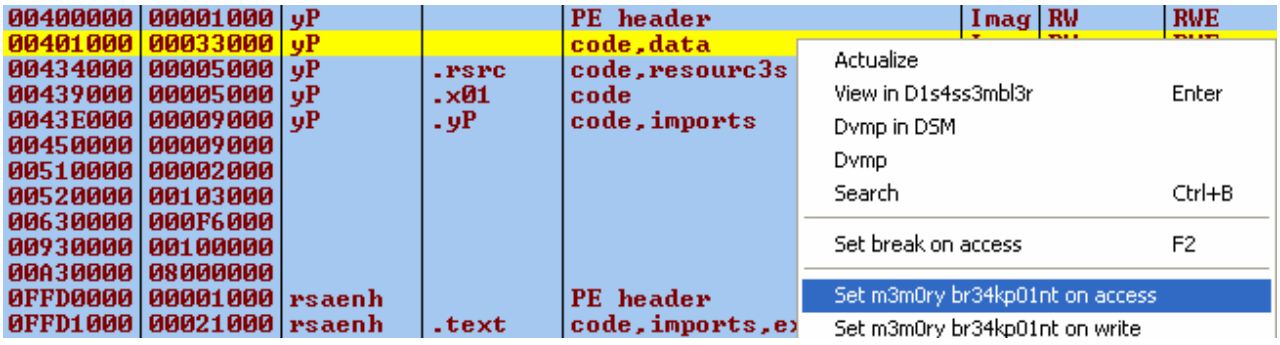
Method 2 : using OllyDBG's plugin

– We need use plugin to hide Olly from IsDebugerPresent check :



– Press F9 to run target (remember you break at BP on BlockInput – RETN 4). We will stop two times on bp on patched BlockInput API.

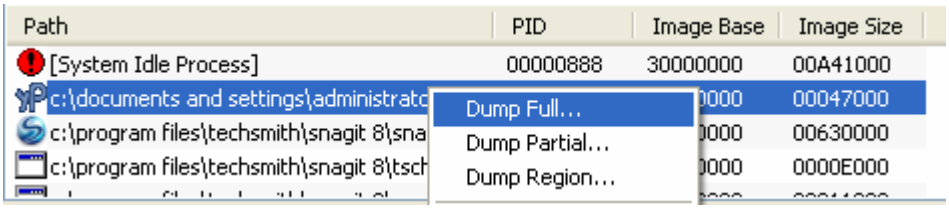
– Press ALT + M to open "Memory Map" window and place memory bp on access on first section and run. OEP is reached:



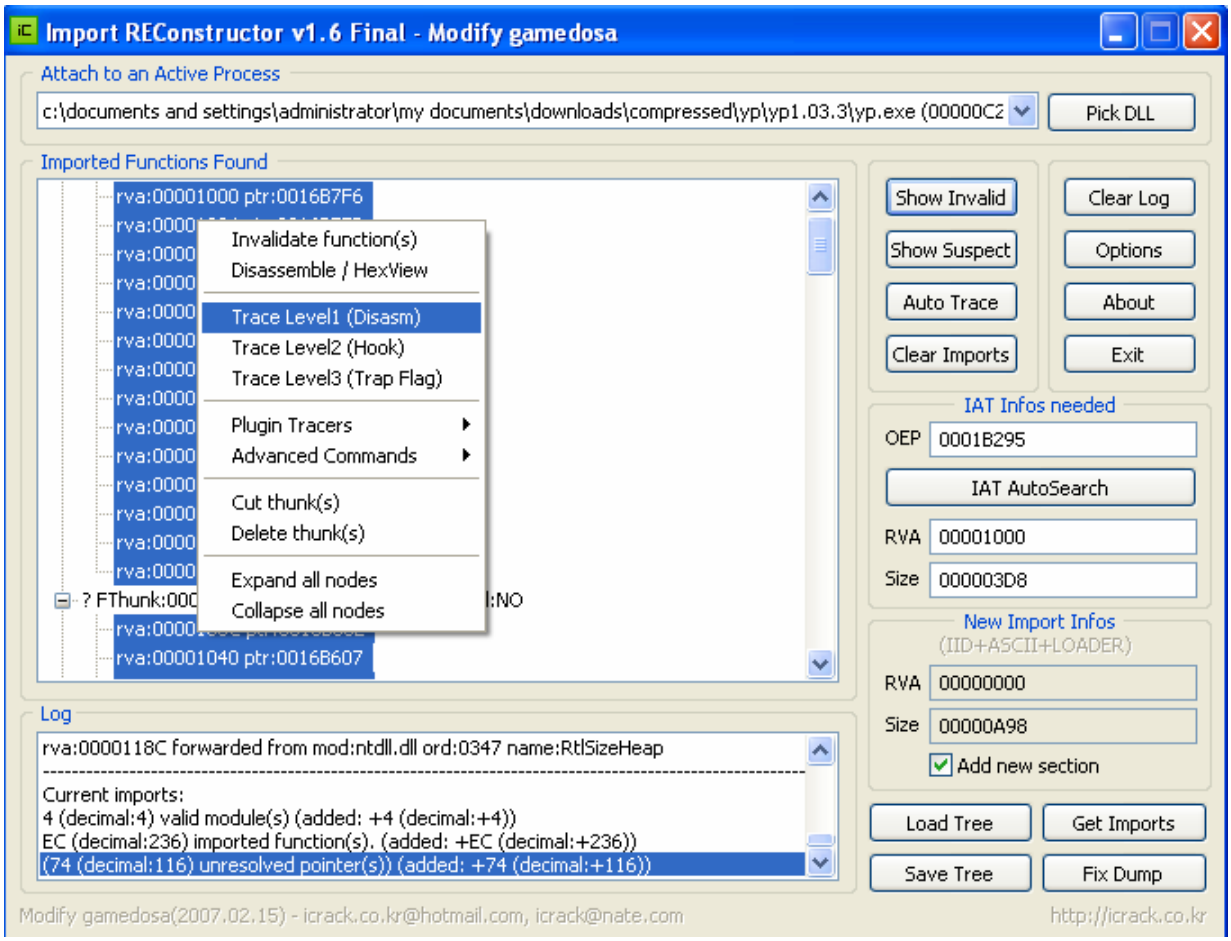
– Press F9 to RUN target, you break target's OEP :

```
0041B295   6A 60          PUSH   60
0041B297   68 D05B4000    PUSH   405BD0
0041B29C   E8 670D0000    CALL   0041C008                          yP.0041C008
0041B2A1   BF 94000000    MOV    EDI, 94
0041B2A6   8BC7           MOV    EAX, EDI
0041B2A8   E8 C3FCFFFF    CALL   0041AF70                          yP.0041AF70
0041B2AD   8965 E8        MOV    DWORD PTR SS:[EBP-18], ESP
0041B2B0   8BF4           MOV    ESI, ESP
0041B2B2   893E           MOV    DWORD PTR DS:[ESI], EDI
0041B2B4   56             PUSH   ESI
0041B2B5   FF15 48114000  CALL   NEAR DWORD PTR DS:[401148]       kernel32.GetVersionExA
```

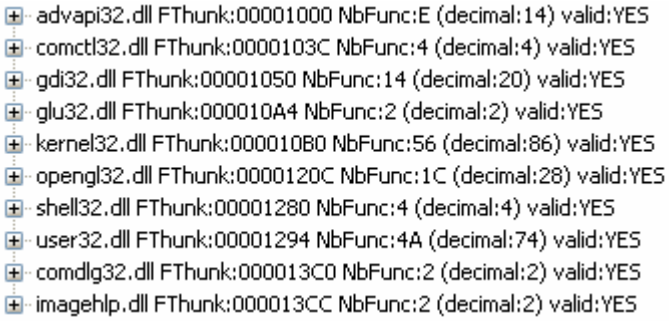
– You d**ump Full** by PE Tool or Lord PE, default it save with name **dumped.exe** :



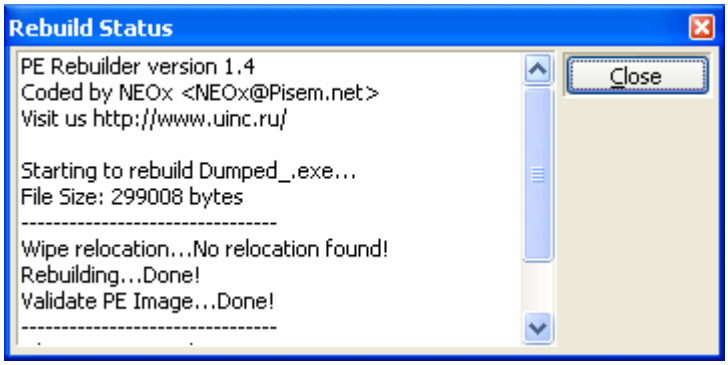
– Using ImpREC to Fix Import. There are very bad import, right click choose **Trace Level1 (Disasm)** follow :



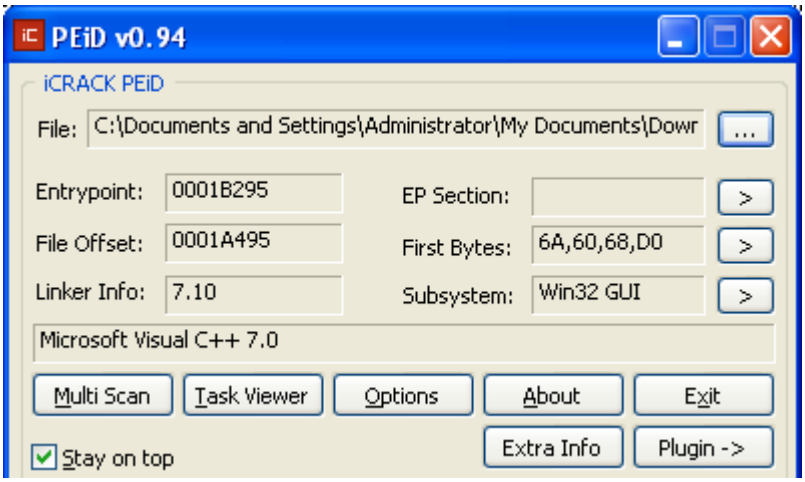
– After fixing :



– Press **Fix Dump**, and choose file **dumped.exe** to finish Fix IAT. You can use Lord PE or PE Tools to Rebuild PE :



– Using PeiD retest and you know Yoda's is coded by language **Microsoft Visual C++ 7.0**.



Thanks to haggar's tutorial about Yoda's Protector 1.03.3

– Unpacking done !