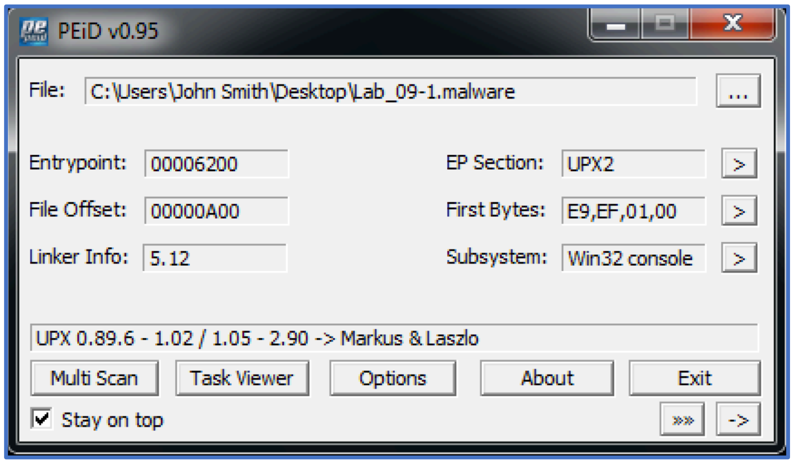
Lab 09-1.malware

1. Is there a name for the packer used to protect this sample?

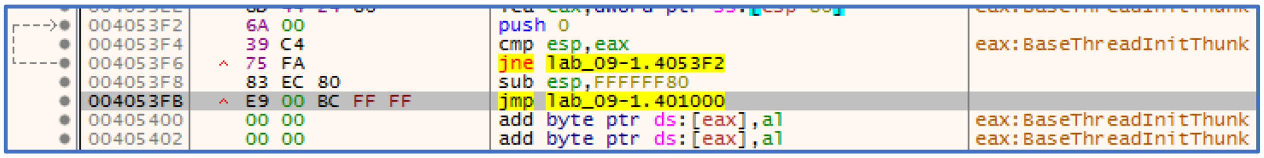
This sample is packed with UPX.



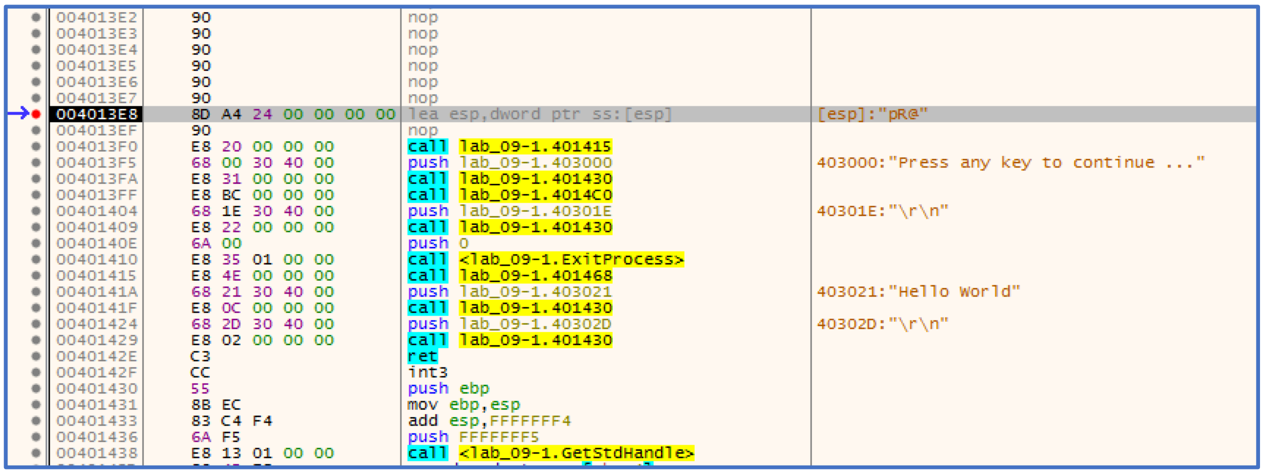
```
File: C:\Users\John Smith\Desktop\Lab_09-1.malware

Entrypoint:  00006200        EP Section:  UPX2
File Offset: 00000A00        First Bytes: E9,EF,01,00
Linker Info: 5.12            Subsystem:   Win32 console

UPX 0.89.6 - 1.02 / 1.05 - 2.90 -> Markus & Laszlo
```

2. What is OEP?

OEP is at 0x401000. Realistically, OEP can be anything between 0x401000 and 0x4013E8, since this area is a NOP sled.



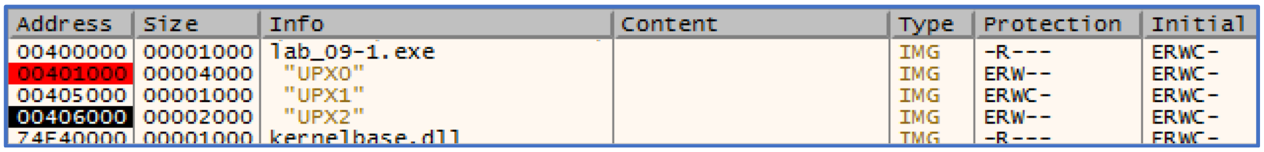
```
004053F2    6A 00             push 0
004053F4    39 C4             cmp esp,eax                     eax:BaseThreadInitThunk
004053F6  ^ 75 FA             jne lab_09-1.4053F2
004053F8    83 EC 80          sub esp,FFFFFF80
004053FB  ^ E9 00 BC FF FF    jmp lab_09-1.401000
00405400    00 00             add byte ptr ds:[eax],al        eax:BaseThreadInitThunk
00405402    00 00             add byte ptr ds:[eax],al        eax:BaseThreadInitThunk
```



```
004013E2   90                     nop
004013E3   90                     nop
004013E4   90                     nop
004013E5   90                     nop
004013E6   90                     nop
004013E7   90                     nop
004013E8   8D A4 24 00 00 00 00   lea esp,dword ptr ss:[esp]     [esp]:"pR@"
004013EF   90                     nop
004013F0   E8 20 00 00 00         call lab_09-1.401415
004013F5   68 00 30 40 00         push lab_09-1.403000           403000:"Press any key to continue ..."
004013FA   E8 31 00 00 00         call lab_09-1.401430
004013FF   E8 BC 00 00 00         call lab_09-1.4014C0
00401404   68 1E 30 40 00         push lab_09-1.40301E           40301E:"\r\n"
00401409   E8 22 00 00 00         call lab_09-1.401430
0040140E   6A 00                  push 0
00401410   E8 35 01 00 00         call <lab_09-1.ExitProcess>
00401415   E8 4E 00 00 00         call lab_09-1.401468
0040141A   68 21 30 40 00         push lab_09-1.403021           403021:"Hello World"
0040141F   E8 0C 00 00 00         call lab_09-1.401430
00401424   68 2D 30 40 00         push lab_09-1.40302D           40302D:"\r\n"
00401429   E8 02 00 00 00         call lab_09-1.401430
0040142E   C3                     ret
0040142F   CC                     int3
00401430   55                     push ebp
00401431   8B EC                  mov ebp,esp
00401433   83 C4 F4               add esp,FFFFFFF4
00401436   6A F5                  push FFFFFFF5
00401438   E8 13 01 00 00         call <lab_09-1.GetStdHandle>
```

3. **What method did you use to find OEP?**

I ran the program to completion and looked at which sections in memory were modified.



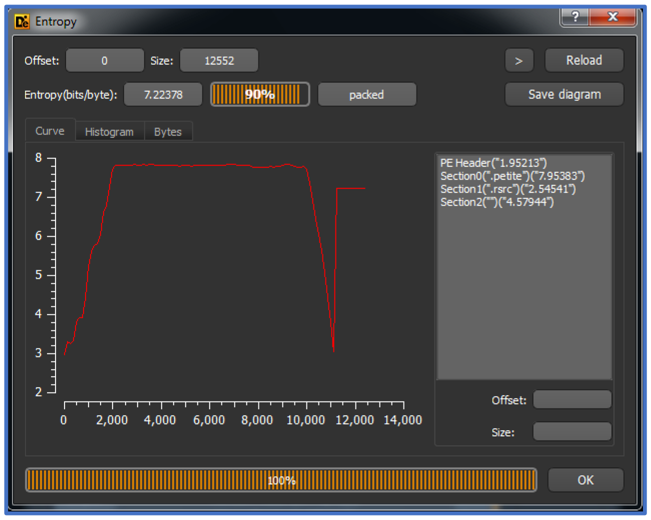
Address	Size	Info	Content	Type	Protection	Initial
00400000	00001000	lab_09-1.exe		IMG	-R---	ERWC-
00401000	00004000	"UPX0"		IMG	ERW--	ERWC-
00405000	00001000	"UPX1"		IMG	ERWC-	ERWC-
00406000	00002000	"UPX2"		IMG	ERW--	ERWC-
74E40000	00001000	kernelbase.dll		IMG	-R---	ERWC-

I then observed the contents of this memory section, and realized the top was a NOP sled. Setting a hardware execution breakpoint at the end of the NOP sled reveals the program eventually reaches this point.

Lab 09-2.malware

1. **What are two indicators of this sample being packed?**

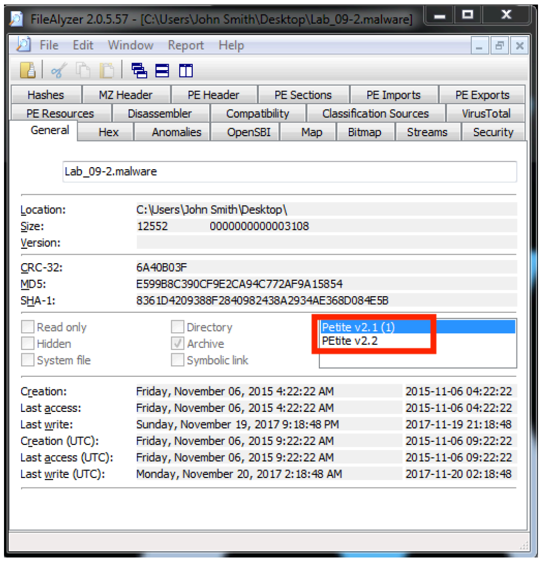
One indicator this executable is packed is the extremely high entropy value of the PE's biggest section. Another indicator is the non-typical section names (.petite and lack of .text and .data).



```
PE Header("1.95213")
Section0(".petite")("7.95383")
Section1(".rsrc")("2.54541")
Section2("")("4.57944")
```

2. **What is this program packed with?**

This program is packed with a packer called PEtite 2.2.



Lab_09-2.malware

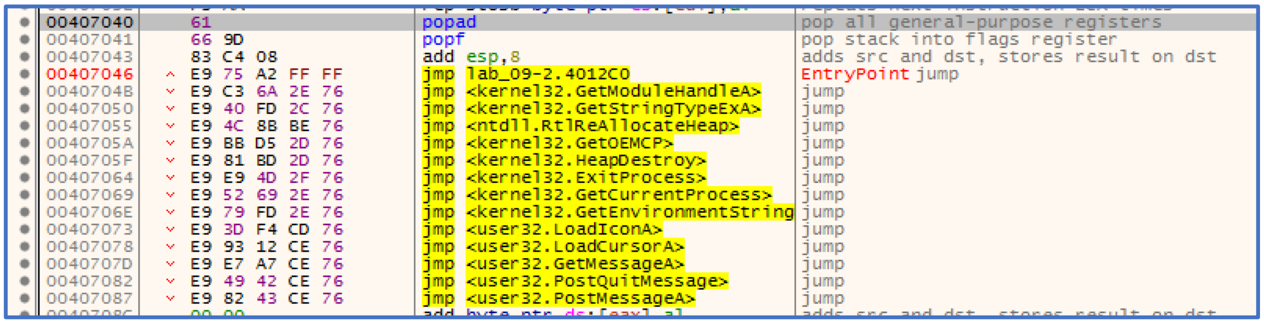
Location:	C:\Users\John Smith\Desktop\	
Size:	12552	0000000000003108
Version:		
CRC-32:	6A40B03F	
MD5:	E599B8C390CF9E2CA94C772AF9A15854	
SHA-1:	8361D4209388F2840982438A2934AE368D084E5B	

☐ Read only ☐ Directory
☐ Hidden ✓ Archive
☐ System file ☐ Symbolic link

Petite v2.1 (1)
PEtite v2.2

Creation:	Friday, November 06, 2015 4:22:22 AM	2015-11-06 04:22:22
Last access:	Friday, November 06, 2015 4:22:22 AM	2015-11-06 04:22:22
Last write:	Sunday, November 19, 2017 9:18:48 PM	2017-11-19 21:18:48
Creation (UTC):	Friday, November 06, 2015 9:22:22 AM	2015-11-06 09:22:22
Last access (UTC):	Friday, November 06, 2015 9:22:22 AM	2015-11-06 09:22:22
Last write (UTC):	Monday, November 20, 2017 2:18:48 AM	2017-11-20 02:18:48

3. **What is OEP?**



```
00407040    61               popad                               pop all general-purpose registers
00407041    66 9D            popf                                pop stack into flags register
00407043    83 C4 08         add esp,8                           adds src and dst, stores result on dst
00407046  ^ E9 75 A2 FF FF   jmp lab_09-2.4012C0                 EntryPoint jump
0040704B  v E9 C3 6A 2E 76   jmp <kernel32.GetModuleHandleA>     jump
00407050  v E9 40 FD 2C 76   jmp <kernel32.GetStringTypeExA>     jump
00407055  v E9 4C 8B BE 76   jmp <ntdll.RtlReAllocateHeap>       jump
0040705A  v E9 BB D5 2D 76   jmp <kernel32.GetOEMCP>             jump
0040705F  v E9 81 BD 2D 76   jmp <kernel32.HeapDestroy>          jump
00407064  v E9 E9 4D 2F 76   jmp <kernel32.ExitProcess>          jump
00407069  v E9 52 69 2E 76   jmp <kernel32.GetCurrentProcess>    jump
0040706E  v E9 79 FD 2E 76   jmp <kernel32.GetEnvironmentString  jump
00407073  v E9 3D F4 CD 76   jmp <user32.LoadIconA>              jump
00407078  v E9 93 12 CE 76   jmp <user32.LoadCursorA>            jump
0040707D  v E9 E7 A7 CE 76   jmp <user32.GetMessageA>            jump
00407082  v E9 49 42 CE 76   jmp <user32.PostQuitMessage>        jump
00407087  v E9 82 43 CE 76   jmp <user32.PostMessageA>           jump
```

OEP is 0x4012C0.

4. **What method did you use to find OEP?**

I found OEP by letting the program run and looking at the end of the last section. There was an import-table like structure, with a bunch of jumps into Windows modules. One of the jumps was into user-code, and x32dbg had labeled it EntryPoint. This is a section-hop, and it was right below a *popad* instruction, which further confirm this is likely the OEP.

5. **Write a script (any language) to unpack this program.**

See 'Code' directory.

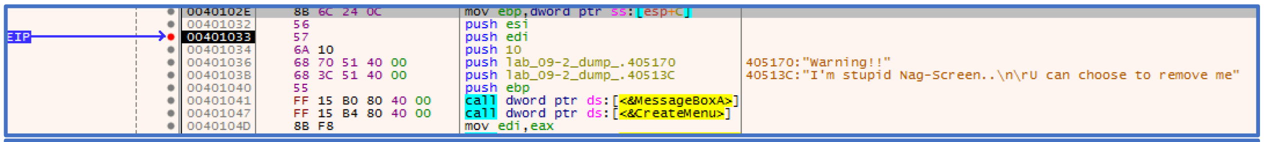
bsdiff and *bspatch* were used to create patch files http://www.daemonology.net/bsdiff/.

6. **Remove the nag screen and enable the secret menu item, briefly explain how you did it.**

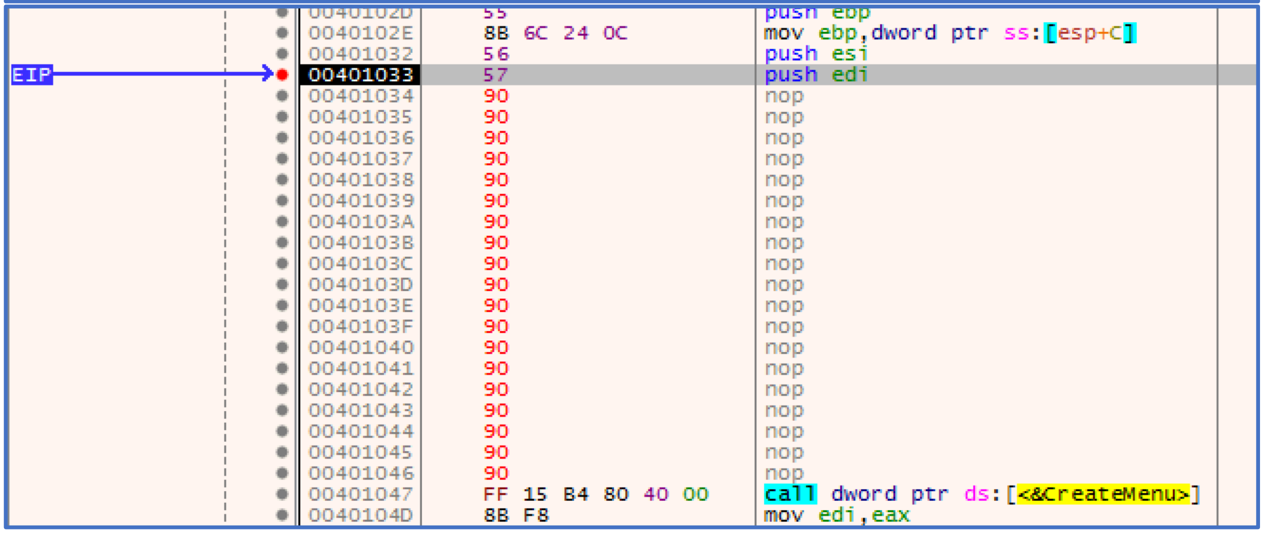
Removing the nag screen simply involved NOP'ing the *MessageBox* function call, and enabling the secret menu item involved changing the 'uFlags' argument of the *AppendMenu* function from 0x1 to 0x0 (MF_GRAYED to MF_ENABLED).

AppendMenu MSDN page: https://msdn.microsoft.com/en-us/library/windows/desktop/ms647616(v=vs.85).aspx

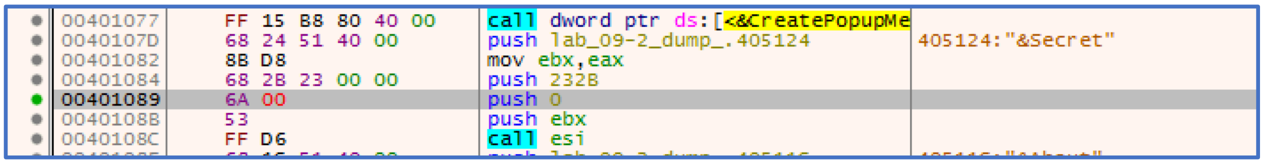
The following pictures show before and after patches for the MessageBox and secret menu.



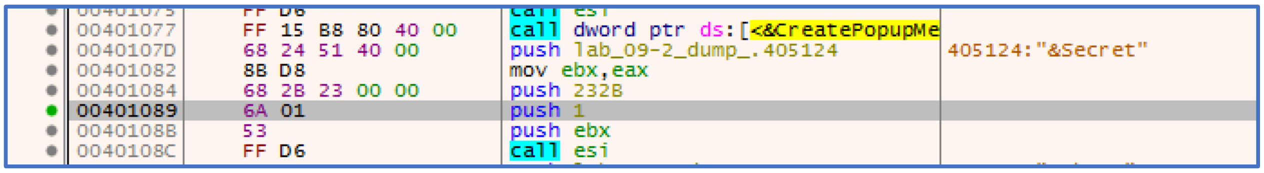
```
         0040102E   8B 6C 24 0C          mov ebp,dword ptr ss:[esp+C]
         00401032   56                   push esi
EIP ---> 00401033   57                   push edi
         00401034   6A 10                push 10
         00401036   68 70 51 40 00       push lab_09-2_dump_.405170         405170:"Warning!!"
         0040103B   68 3C 51 40 00       push lab_09-2_dump_.40513C         40513C:"I'm stupid Nag-Screen..\n\rU can choose to remove me"
         00401040   55                   push ebp
         00401041   FF 15 B0 80 40 00    call dword ptr ds:[<&MessageBoxA>]
         00401047   FF 15 B4 80 40 00    call dword ptr ds:[<&CreateMenu>]
         0040104D   8B F8                mov edi,eax
```



```
         0040102D   55                   push ebp
         0040102E   8B 6C 24 0C          mov ebp,dword ptr ss:[esp+C]
         00401032   56                   push esi
EIP ---> 00401033   57                   push edi
         00401034   90                   nop
         00401035   90                   nop
         00401036   90                   nop
         00401037   90                   nop
         00401038   90                   nop
         00401039   90                   nop
         0040103A   90                   nop
         0040103B   90                   nop
         0040103C   90                   nop
         0040103D   90                   nop
         0040103E   90                   nop
         0040103F   90                   nop
         00401040   90                   nop
         00401041   90                   nop
         00401042   90                   nop
         00401043   90                   nop
         00401044   90                   nop
         00401045   90                   nop
         00401046   90                   nop
         00401047   FF 15 B4 80 40 00    call dword ptr ds:[<&CreateMenu>]
         0040104D   8B F8                mov edi,eax
```



```
00401077   FF 15 B8 80 40 00    call dword ptr ds:[<&CreatePopupMe
0040107D   68 24 51 40 00       push lab_09-2_dump_.405124          405124:"&Secret"
00401082   8B D8                mov ebx,eax
00401084   68 2B 23 00 00       push 232B
00401089   6A 00                push 0
0040108B   53                   push ebx
0040108C   FF D6                call esi
```



```
00401075   FF D6                call esi
00401077   FF 15 B8 80 40 00    call dword ptr ds:[<&CreatePopupMe
0040107D   68 24 51 40 00       push lab_09-2_dump_.405124          405124:"&Secret"
00401082   8B D8                mov ebx,eax
00401084   68 2B 23 00 00       push 232B
00401089   6A 01                push 1
0040108B   53                   push ebx
0040108C   FF D6                call esi
```