Unknown.exe

Jump into code in static disassembly then rename and comment on interesting assembly routines

String Anti Analysis

String stacking: hide strings on the stack (`push` instructions)

```
.text:00401151                push    67616C66h
.text:00401156                push    735F697Bh
.text:0040115B                push    5F796174h
.text:00401160                push    5F74756Fh
.text:00401165                push    5F6F6F74h
.text:0040116A                push    6574616Ch
.text:0040116F                push    746F675Fh
.text:00401174                push    746F6E5Fh
.text:00401179                push    676E6968h
.text:0040117E                push    5F6E695Fh
.text:00401183                push    625F796Dh
.text:00401188                push    6E696172h
.text:0040118D                push    7Dh ; '}'
```

- Values are little endian, LSB. Hex values appear backwards

Converting values of all sequential pushes from offset `00401151` to `00401188`

```
.text:00401151                push    'galf'
.text:00401156                push    's_i{'
.text:0040115B                push    '_yat'
.text:00401160                push    '_tuo'
.text:00401165                push    '_oot'
.text:0040116A                push    'etal'
.text:0040116F                push    'tog_'
.text:00401174                push    'ton_'
.text:00401179                push    'gnih'
.text:0040117E                push    '_ni_'
.text:00401183                push    'b_ym'
.text:00401188                push    'niar'
.text:0040118D                push    '}'
```

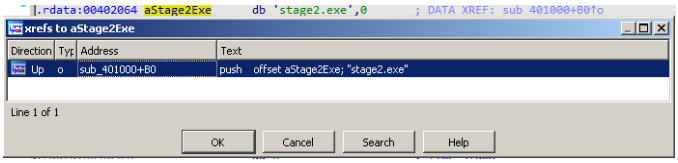
Looking for interesting strings

Interesting string: `.rdata:00402064 0000000B  C stage2.exe`

- Which code is referencing this string

Reference label: `aStage2Exe`

xrefs menu



Jumping to selected cross reference. Address `sub_401000` will take you to where the string is referenced at `004010B0`.

```
.text:004010B0                push    offset aStage2Exe ; "stage2.exe"
```

Trace back to the Start function

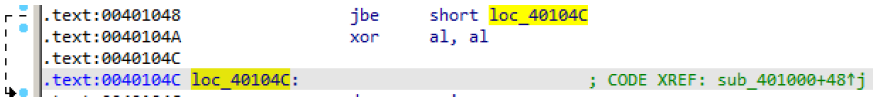
- Following the xrefs of the functions backwards until start function

First location label

```
.text:0040104C loc_40104C:                             ; CODE XREF: sub_401000+48↑j
```

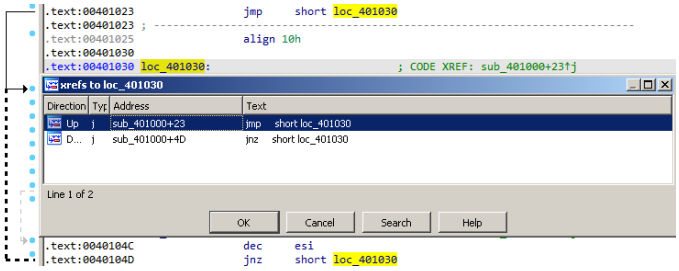
- xref function name to follow the branch or jump instruction (branch or jump instruction is jbe or jmp)

`loc_40104C` branches to this location

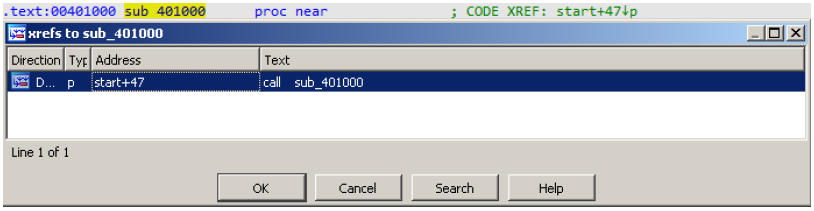


*Note: keep track of where these branch statements occur as apart of the flow.

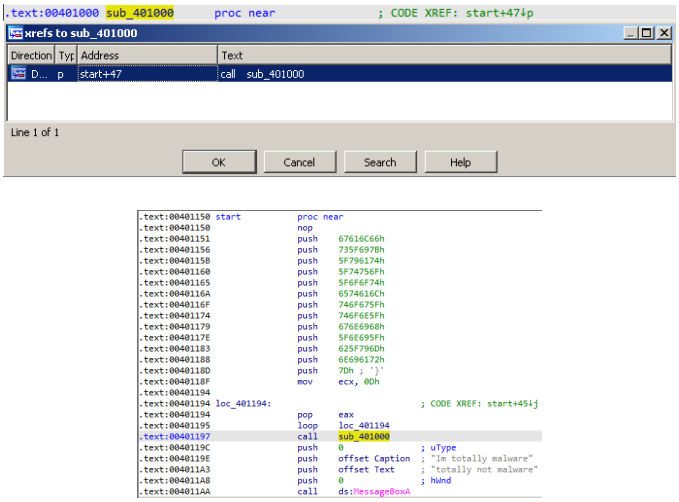
xref next location label



xref subroutine/function



Start function (sub_401000)



Analyze function sub_401000

- Need to capture when stage2.exe file is created and written

*Note: Remember that functions have their arguments pushed onto the stack before the function is called.

stage2.exe is concatenated together with the string referenced by the register esi

- esi is used as the first argument for `CreateFileA`

```
.text:004010B0                push    offset aStage2Exe ; "stage2.exe"
.text:004010B5                push    esi             ; lpString1
.text:004010B6                call    edi ; lstrcatA
.text:004010B8                push    0               ; hTemplateFile
.text:004010BA                push    80h ; '€'       ; dwFlagsAndAttributes
.text:004010BF                push    2               ; dwCreationDisposition
.text:004010C1                push    0               ; lpSecurityAttributes
.text:004010C3                push    0               ; dwShareMode
.text:004010C5                push    40000000h       ; dwDesiredAccess
.text:004010CA                push    esi             ; lpFileName
.text:004010CB                call    ds:CreateFileA
```

Push instructions = arguments

Function called: `CreateFileA`

esi register containing the new string is used as an argument for a call to `CreateProcessA`

```
.text:00401113                push    ecx             ; lpProcessInformation
.text:00401114                lea     edx, [ebp+StartupInfo]
.text:00401117                push    edx             ; lpStartupInfo
.text:00401118                push    0               ; lpCurrentDirectory
.text:0040111A                push    0               ; lpEnvironment
.text:0040111C                push    0               ; dwCreationFlags
.text:0040111E                push    0               ; bInheritHandles
.text:00401120                push    0               ; lpThreadAttributes
.text:00401122                push    0               ; lpProcessAttributes
.text:00401124                push    esi             ; lpCommandLine
.text:00401125                mov     eax, 5
.text:0040112A                push    esi             ; lpApplicationName
.text:0040112B                mov     [ebp+StartupInfo.cb], 44h ; 'D'
.text:00401132                mov     [ebp+StartupInfo.wShowWindow], ax
.text:00401136                call    ds:CreateProcessA
```

Interesting routines using control flow instruction (branch, jumps, calls)

Mapping

- Start
- Call `401000`
- Jmp `@401023`
- Jbe `@401048`
- Createfile `@4010CB`
- Create Process `@401136`

Other interesting API function patterns:

- Opening a file from the resource section
- FindResource
- SizeofResource
- LoadResource
- LockResource
- Creating a file
- GetEnvironmentVariable
- CreateFile
- WriteFile
- CloseHandle
- Sleep
- Starting a new process
- CreateProcess