

Forum:[Hack Forums](#)

Thread: [Analysis of a VAC module - Part 2](#)

JPieky - December 7, 2015, 6:06 pm

Analysis of a vac module

FileSize: 58,880 Bytes

.text hash: 0x59DC680

Number of scans: 4

TimeStamp: 0x564B49A8h Tue Nov 17 15:37:12 2015

Scan Number 2.

For those of you curious about part one.

It was a analysis of the first scan of this module in which it stored the modules of a target process.

Just another heads up:

For anyone interested, I run lionsneck.com and use my knowledge of vac to keep my builds undetected.

The second scan calls EnumProcesses to get up to 500 process ids.

It then passes the ids into a subroutine that decrypts the xored string:Quote:

"\x10*\x0F+;,\r'-*;3\x17081,3?*710"

into

Quote:NtQuerySystemInformation

Once decrypted it passes the handle to ntdll.dll and the function name to GetProcAddress to get the function pointer.

If the pointer is valid, it calls NtQuerySystemInformation in a loop passing SystemHandleInformation and a buffer from VirtualAlloc.

While the NTSTATUS of the call is STATUS_INFO_LENGTH_MISMATCH it calls VirtualFree on the buffer allocated and Reallocates it with a increased size of 0x100000 bytes. If there are no open handles from the received buffer, it calls GetLastError and returns.

After all the calls are successful, it iterates through the handles comparing them to a last process id. It starts with the process id of 1 so there will always be at least 1 hidden process id counted.

As it iterates the handles, it checks the process id associated with the handle to that of the ids return from EnumProcesses. If it finds a process id that has a open handle but isn't returned in EnumProcesses. It appends the process id to the process id buffer up to 500 ids.

This method essentially finds PIDs of hidden processes.

Lastly the subroutine frees the handle buffer with VirtualFree.

Next the scan iterates all the process ids including the hidden process ids and opens a handle to them with OpenProcess with the flags PROCESS_QUERY_LIMITED_INFORMATION and PROCESS_DUP_HANDLE. The bare minimum needed for querying information about the processes.

Once open, it calls GetProcessImageFileName to obtain the full path of the main executable module.

Immediately after it calls GetProcessTimes and stores the creation time.

Since it has everything it needs, it calls CloseHandle to free the handle it just opened.

Then the scan checks the length of the image path and separates the image name from its path.

It does a quick hash on the image name and its path separately using this method:

```
Code int __cdecl HashString(char *szInput, int ilInputLength)
{
    int iCount; // edx@1
    int result; // eax@1
    char *szTemp; // esi@2

    iCount = ilInputLength;
    result = 0x45D71892;
    if ( ilInputLength )
    {
        szTemp = szInput;
        do
        {
            result = ((unsigned __int8)*szTemp++ | 0x20) + 0x21 * result;
            --iCount;
        }
        while ( iCount );
    }
    return result;
}
```

Once hashed, it stores the image name, path, and both hashes into buffer up to 0x4000 bytes.

tl;dr This scan gets the file name and path of all processes, including hidden ones.

I'm glad to provide any info on vac and I hope it was educational for you guys

iamvexatious - December 7, 2015, 8:02 pm

Since you're still advertising lionsneck could you reply to ;

(Nov 27, 2015 06:23 AM)iamvexatious Wrote: [\[CITE\]](#) Fairly sure this is a VGnosis, Fadedcoder or that tier0 shit, No one has had shitty GUI style and all his websites lookalike and push features in the same way, Regardless you payed for it to be advertised on reddit, That's just silly.

(Nov 27, 2015 06:39 AM)iamvexatious Wrote: [\[CITE\]](#) They both seem to have a hardon for glow esp and metamorphic Code as features, Tier0 and lionsneck both have a pretty similar feature list, tier0 and lionsneck have an nearly identical advertising for crosshair, RCS and both have features under a "Physics" tab (which is silly regardless) So either segnosis is a personal hero of James's or they're pretty tight ;)

aethi - December 7, 2015, 8:07 pm

posting public info about vac on a forum where nobody will understand to boost ego and push p2c

whoa!!

Vuno - December 7, 2015, 8:48 pm

Boosting your pay2use cheat with VAC information. So what happens when the next provider banwave come, and you still get banned? Never underestimate VAC. Many have done it, and many failed. The best way to succeed is to not feel superior.

01010wow - December 8, 2015, 1:13 am

(Dec 7, 2015 09:07 PM)aethi Wrote: [\[CITE\]](#) posting public info about vac on a forum where nobody will understand to boost ego and push p2c

whoa!!

Stop shaming Vuno & the others :(
I'm sure vuno knows what it says

i n v a l i d - December 8, 2015, 1:36 am

Yeah, nobody here cares to learn this stuff, but thanks for the info again anyway.

I'm also curious why so many of you nerds think 'pure asm' shit is going to help you in any way? You also realize doing asm like that defeats the entire purpose of compiler optimizations? I'd take the compiler over the shitty p2c pea-brains around here any day.

Freak1234 - December 8, 2015, 3:28 am

(Dec 8, 2015 02:36 AM)i n v a l i d Wrote: [\[CITE\]](#) Yeah, nobody here cares to learn this stuff, but thanks for the info again anyway.

I'm also curious why so many of you nerds think 'pure asm' shit is going to help you in any way? You also realize doing asm like that defeats the entire purpose of compiler optimizations? I'd take the compiler over the shitty p2c pea-brains around here any day.

word. However the best things are pay2pasters that use STL objects all over the place creating major overhead and using asm in other parts "to save performance" lmao

Narvikz - December 8, 2015, 4:48 am

(Dec 8, 2015 02:36 AM)i n v a l i d Wrote: [\[CITE\]](#) Yeah, nobody here cares to learn this stuff, but thanks for the info again anyway.

I'm also curious why so many of you nerds think 'pure asm' shit is going to help you in any way? You also realize doing asm like that defeats the entire purpose of compiler optimizations? I'd take the compiler over the shitty p2c pea-brains around here any day.

Polymorphic code is the reason people talk about ASM, not ASM itself... Of course no one's done it though.

Narvikz - December 8, 2015, 5:29 am

(Dec 8, 2015 06:27 AM)Kerb Wrote: [\[CITE\]](#) CTRL + V _asm nop everywhere in project, bye VAC! ? My CTRL button broke of always pressing it. And I guess C and V key switches are at half their durability so. I have to use my mouse now and that shit just takes too much time. Fuck pasta I'm gonna be a gangster instead.

supex0 - December 8, 2015, 5:30 am

(Dec 8, 2015 06:27 AM)Kerb Wrote: [\[CITE\]](#) CTRL + V _asm nop everywhere in project, bye VAC! ? HOW DID YOU....

/me checks for intruders in his apt

Narvikz - December 8, 2015, 5:31 am

Oh and btw. Everyone knows ASM in C++ is detected...

Freak1234 - December 8, 2015, 8:48 am

(Dec 8, 2015 06:31 AM)Narvikz Wrote: [\[CITE\]](#) Oh and btw. Everyone knows ASM in C++ is detected...

nono only __asm (inlined style is detected). Embending it as shellcode or .asm files are fine

invalid - December 8, 2015, 5:06 pm

(Dec 8, 2015 04:28 AM)Freak1234 Wrote: [\[CITE\]](#) word. However the best things are pay2pasters that use STL objects all over the place creating major overhead and using asm in other parts "to save performance" lmao

stl is not slow...

cdr1za - December 22, 2015, 3:06 pm

(Dec 7, 2015 09:48 PM)Vuno Wrote: [\[CITE\]](#) Boosting your pay2use cheat with VAC information. So what happens when the next provider banwave come, and you still get banned? Never underestimate VAC. Many have done it, and many failed. The best way to succeed is to not feel superior.

"never underestimate vac"

i've never been vac banned once unintentionally and i've been cheating for over 3 years