

Forum:[Hack Forums](#)

Thread: [\[Development thread\] Gasolyne - C/ASM crypter](#)

Jean-Jacques Rousseau - August 4, 2018, 9:50 am

Hey.

Yesterday I decided to code a crypter (for the lols of it), and I just finished the base. In this thread I will post my progress when developing this cryper.

Small details:

- Written in Visual C/C++ and x86_32 ASM.

Current features:

- Encryption/Decryption routine in x86_32.
- Malware is stored using the resource method.
- Malware is dropped to disk after decryption.

I was honestly expecting it to be fully detected when I first scanned it, however, it showed up as 11/32 scantime.

[i.gyazo.com/f227be7035813a8c5b5ea36733939830.png](#)

This is Gaudox crypted. The stub is compiled to debug - it has not been stripped.

What i am aiming for in the near future:

- Unique RunPE
- Changing the decryption routine to something like Luminous crypter, to mess upp the emulators
- Call the functions dynamically, bypass some hooks.

My end-goals:

- Calling the functions via syscalls for both x86_64 and x86_32 (still researching about the 0x33 segment!)
- Some Anti-debug/Anti-VM features.

The crypter is called Gasolyne. I dont know what I will do with it - might sell or release.

I am open to learning!

EDIT: A word.

To clarify for those who dont know how the Luminous crypter works, I will essentially brute-force key.

3DES - August 4, 2018, 11:04 am

Calling APIs dynamically wont do jack shit dude if you are trying to bypass with that. Preferably read syscall numbers from ntdll and use them to bypass hooks.

Jean-Jacques Rousseau - August 4, 2018, 11:35 am

(Aug 4, 2018 11:04 AM)3DES Wrote: [\[CITE\]](#) Calling APIs dynamically wont do jack shit dude if you are trying to bypass with that. Preferably read syscall numbers from ntdll and use them to bypass

hooks.

Yeah, that's why I wanted to use direct syscall, but I haven't really "mastered" the topic yet.

3DES - August 4, 2018, 11:45 am

You also might want to use LoadPE instead of runpe(ITS ANCIENT!)

int - August 4, 2018, 11:53 am

dropping the malware executable removes the entire point of running in memory

Jean-Jacques Rousseau - August 4, 2018, 12:00 pm

(Aug 4, 2018 11:53 AM)Jukeboxhook Wrote: [\[CITE\]](#) dropping the malware executable removes the entire point of running in memory

Lol, it was just the base, to make sure everything went fine. I am not a retard.

(Aug 4, 2018 11:45 AM)3DES Wrote: [\[CITE\]](#) You also might want to use LoadPE instead of runpe(ITS ANCIENT!)

It sure is, will add that in the end.

Uiqueblhats - August 4, 2018, 12:47 pm

Mate can you pm me the link to luminous crypter heard a lot of good stories about it

Jean-Jacques Rousseau - August 4, 2018, 2:12 pm

(Aug 4, 2018 12:47 PM)Uiqueblhats Wrote: [\[CITE\]](#) Mate can you pm me the link to luminous crypter heard a lot of good stories about it

I can't pm. The crypter has been posted on a load of forums (including this one) multiple times.

Uiqueblhats - August 4, 2018, 3:25 pm

(Aug 4, 2018 02:12 PM)Supercut Wrote: [\[CITE\]](#) I can't pm. The crypter has been posted on a load of forums (including this one) multiple times.

Post the link here only please

Jean-Jacques Rousseau - August 5, 2018, 5:16 am

(Aug 4, 2018 03:25 PM)Uiqueblhats Wrote: [\[CITE\]](#) Post the link here only please
showthread.php?tid=5722311

Uiqueblhats - August 5, 2018, 6:54 am

(Aug 5, 2018 05:16 AM)Supercut Wrote: [\[CITE\]](#) showthread.php?tid=5722311
thanks mate it will really help me

Jean-Jacques Rousseau - August 5, 2018, 6:55 am

(Aug 5, 2018 06:54 AM)Uiqueblhats Wrote: [\[CITE\]](#) thanks mate it will really help me
Happy coding!

Update: trying to implement a runpe, having some issues. The code looks just fine, however, my test file does not execute. Will look at other sources to see how they do it.

Twistd - August 5, 2018, 1:46 pm

What exact part of your crypter is assembly?

Jean-Jacques Rousseau - August 5, 2018, 11:49 pm

(Aug 5, 2018 01:46 PM)Twistd Wrote: [\[CITE\]](#) What exact part of your crypter is assembly?
The encryption/decryption routine, and soon, the syscalls (that will replace functions).

UPDATE: Implemented a RunPE, code gets executed. Everything looks fine!

New scan!

4/32

i.gyazo .com/ 8b1aaf859b39ea5eebc28c55dc26067b.png

I will add an anti-emulation method that Luminous invented. Hopefully it can bypass AVs runtime.

Twistd - August 6, 2018, 6:29 am

(Aug 5, 2018 11:49 PM)Supercut Wrote: [\[CITE\]](#) The encryption/decryption routine, and soon, the syscalls (that will replace functions).

UPDATE: Implemented a RunPE, code gets executed. Everything looks fine!

New scan!

4/32

i.gyazo .com/ 8b1aaf859b39ea5eebc28c55dc26067b.png

I will add an anti-emulation method that Luminous invented. Hopefully it can bypass AVs runtime. That will replace api calls you mean? I don't see any specific reason why you would want to use assembly for the encryption/decryption routine. Please elaborate

Anti emulation won't bypass anything runtime.

Jean-Jacques Rousseau - August 6, 2018, 8:53 am

(Aug 6, 2018 06:29 AM)Twistd Wrote: [\[CITE\]](#) That will replace api calls you mean? I don't see any specific reason why you would want to use assembly for the encryption/decryption routine. Please elaborate

Anti emulation won't bypass anything runtime.

Uh no, i will use syscalls and the api calls... zzz ROFL.

Because the smaller the size the better? I can make it as small as possible using ASM, and the fastest. Thats like asking why i dont write a decryptor in python and use py2exe LOL.

You probably dont know what i mean by "anti-emulation", because bypassing runtime is exactly what it does.

narc0tic - August 6, 2018, 11:24 am

(Aug 6, 2018 06:29 AM)Twistd Wrote: [\[CITE\]](#) That will replace api calls you mean? I don't see any specific reason why you would want to use assembly for the encryption/decryption routine. Please elaborate

Anti emulation won't bypass anything runtime.

Anti emulation does actually help with cloud runtime detections, Even the simplest anti emulation will kill avria cloud.

Are you generating the key using the anti emulation?

GLWS, I am very sure this will be really good.

Jean-Jacques Rousseau - August 6, 2018, 12:52 pm

(Aug 6, 2018 11:24 AM)narc0tic Wrote: [\[CITE\]](#) Anti emulation does actually help with cloud runtime detections, Even the simplest anti emulation will kill avria cloud.

Are you generating the key using the anti emulation?

GLWS, I am very sure this will be really good.

I was thinking about that, however, I am still debating which method to use. Bruting the key is a method that has been used with great success by Luminos.

The next update will prolly take some time, I am trying to reverse ntdll, lol.

Not sure if i will sell this, lol. Might release parts of it, but only in such a way that it is educational.

notluther - August 6, 2018, 1:09 pm

(Aug 6, 2018 12:52 PM)Supercut Wrote: [\[CITE\]](#) I was thinking about that, however, I am still debating which method to use. Bruting the key is a method that has been used with great success by Luminos.

The next update will prolly take some time, I am trying to reverse ntdll, lol.

Not sure if i will sell this, lol. Might release parts of it, but only in such a way that it is educational. Sell it. You won't regret it.

Twistd - August 6, 2018, 1:31 pm

Bruting the key is silly, eating unnecessary resources. There are much better ways.

You don't have the slightest clue about what you're writing. Smallest possible? Your arguments doesn't make sense. Everything you write doesn't make sense at all. Syscalls and api calls? Function syscalls? Do you even code at all? I've written runtime bypasses for over 30 antiviruses along with writing both asm and c/c++, so yea I know exactly what anti emulation is. I am probably the best resource for antivirus bypasses, crypters and native reverse engineering that you'll find on this forum.

Jean-Jacques Rousseau - August 6, 2018, 2:53 pm

(Aug 6, 2018 01:31 PM)Twistd Wrote: [\[CITE\]](#) Bruting the key is silly, eating unnecessary resources. There are much better ways.

You don't have the slightest clue about what you're writing. Smallest possible? Your arguments doesn't make sense. Everything you write doesn't make sense at all. Syscalls and api calls? Function syscalls? Do you even code at all? I've written runtime bypasses for over 30 antiviruses along with writing both asm and c/c++, so yea I know exactly what anti emulation is. I am probably the best resource for antivirus bypasses, crypters and native reverse engineering that you'll find on this forum.

Youre so full of yourself. I am not saying bruting the key is the best way, however it resulted in 0/13 for Luminos. I was being sarcastic about the syscall/api stuff. Do i really have to add /s? The syscalls will replace the API calls, lmao.

So by your logic, I should write the parts which already are in assembly, in C? What kind of logic is that? If i already know how to write it in assembly, why not implement it in assembly? The lower the better.

Twistd - August 6, 2018, 4:44 pm

(Aug 6, 2018 02:53 PM)Supercut Wrote: [\[CITE\]](#) Youre so full of yourself. I am not saying bruting the

key is the best way, however it resulted in 0/13 for Luminos. I was being sarcastic about the syscall/api stuff. Do i really have to add /s? The syscalls will replace the API calls, lmao.

So by your logic, I should write the parts which already are in assembly, in C? What kind of logic is that? If i already know how to write it in assembly, why not implement it in assembly? The lower the better.

Sarcastic about the syscall stuff? I seriously doubt that because that was too dumb to be sarcastic.

Theres absolutely no reason in this lifetime to encrypt your data with inline assembly.

The lower the better? You clearly dont know that the compiler optimizes your C\C++ to ASM much better than you can write your assembly anyway.

You claim that you know how to write in assembly, but you dont. Why dont you write in a language you actually understand?

I know you think it sounds super fancy to say that your crypter is written in ASM but its not.

I assume youre using XOR encryption? Why would you prefer using inline assembly for that instead of C? Do you really think its gonna be larger in C ? It wont.

It's literally 2 lines of code in C which then again is optimized by the compiler.

If you're looking for decreased size youre clearly looking at the wrong things.

Get rid of CRT and a couple other minor changes and youre cruisin with a stub at 6kb.

Yes anti-emulation can be used against some silly cloud antiviruses, just dont forget that they dont just use emulation, they've got their own crazy databases of scantime detections, along with hooks and heuristics based on api chains.

Anti-emulation also works against kaspersky SCANTIME. But against real RUNTIME on your local computer it wont matter at all.

A better alternative than brutin the key and exhausting its resources is to allocate a certain amount of memory, then checking that memory if its actually allocated or not. More efficient and will work just as good without wrecking the cpu.

If you think brutin a key made your crypter 0/13 runtime, then something is terribly wrong with your test because thats just not possible.

These scanners scan offline very often and don't forget to encrypt files that actually are highly detected, startup enabled with files like DarkComet and NetWire. Thats what i use when i test my shit.

I normally set up 30-35 snapshots with each snapshot having a different type of antivirus to do proper testing locally and online.

FalsifiableMeme - August 6, 2018, 5:55 pm

(Aug 6, 2018 04:44 PM)Twistd Wrote: [\[CITE\]](#) If you're looking for decreased size youre clearly looking at the wrong things.

Get rid of CRT and a couple other minor changes and youre cruisin with a stub at 6kb.

What is your opinion on the CRT? Removing it decreases stub size, but also makes it more difficult to FUD. By the time you add enough junk code to FUD it, the stub could be as big as with CRT. I'm not sure whether to use it in my crypter or just add massive amounts of junk

Twistd - August 6, 2018, 6:56 pm

(Aug 6, 2018 05:55 PM)FalsifiableMeme Wrote: [\[CITE\]](#) What is your opinion on the CRT? Removing it decreases stub size, but also makes it more difficult to FUD. By the time you add enough junk code to FUD it, the stub could be as big as with CRT. I'm not sure whether to use it in my crypter or just add massive amounts of junk

Why would it be more difficult to fud? thats wrong, you just need to learn to fud properly.
You don't junk your stub with C\C++ code, you junk it with false winapi calls and tiny inline assembly that breaks up your code.

The structure of your stub is whats going to matter the most how fast and easy you'll find those signatures.

There are also many types of detections and it takes time to understand how to approach them differently.

It takes some time, C\C++ is definitely more challenging to fud than C#. You'll get used to it after a while.

FalsifiableMeme - August 6, 2018, 7:49 pm

(Aug 6, 2018 06:56 PM)Twistd Wrote: [\[CITE\]](#) Why would it be more difficult to fud? thats wrong, you just need to learn to fud properly.

You don't junk your stub with C\C++ code, you junk it with false winapi calls and tiny inline assembly that breaks up your code.

The structure of your stub is whats going to matter the most how fast and easy you'll find those signatures.

There are also many types of detections and it takes time to understand how to approach them differently.

It takes some time, C\C++ is definitely more challenging to fud than C#. You'll get used to it after a while.

Thanks for replying. I said it's more difficult to FUD because when I disable CRT I have detections, but when I enable it, those detections are no longer there. I will follow the information you provided and try and FUD without CRT then. Also funny how you correctly predicted I came from C#, that is true I quit it many months ago.

Jean-Jacques Rousseau - August 6, 2018, 11:56 pm

(Aug 6, 2018 04:44 PM)Twistd Wrote: [\[CITE\]](#) Sarcastic about the syscall stuff? I seriously doubt that because that was too dumb to be sarcastic.

Theres absolutely no reason in this lifetime to encrypt your data with inline assembly.

The lower the better? You clearly dont know that the compiler optimizes your C\C++ to ASM much better than you can write your assembly anyway.

You claim that you know how to write in assembly, but you dont. Why dont you write in a language you actually understand?

I know you think it sounds super fancy to say that your crypter is written in ASM but its not.

I assume youre using XOR encryption? Why would you prefer using inline assembly for that instead of C? Do you really think its gonna be larger in C ? It wont.

It's literally 2 lines of code in C which then again is optimized by the compiler.

If you're looking for decreased size youre clearly looking at the wrong things.

Get rid of CRT and a couple other minor changes and youre cruisin with a stub at 6kb.

Yes anti-emulation can be used against some silly cloud antiviruses, just dont forget that they dont just use emulation, they've got their own crazy databases of scantime detections, along with hooks and heuristics based on api chains.

Anti-emulation also works against kaspersky SCANTIME. But against real RUNTIME on your local

computer it wont matter at all.

A better alternative than brutng the key and exhausting its resources is to allocate a certain amount of memory, then checking that memory if its actually allocated or not. More efficient and will work just as good without wrecking the cpu.

If you think brutng a key made your crypter 0/13 runtime, then something is terribly wrong with your test because thats just not possible.

These scanners scan offline very often and don't forget to encrypt files that actually are highly detected, startup enabled with files like DarkComet and NetWire. Thats what i use when i test my shit.

I normally set up 30-35 snapshots with each snapshot having a different type of antivirus to do proper testing locally and online.

> Sarcastic about the syscall stuff? I seriously doubt that because that was too dumb to be sarcastic.

I literally added "zzz ROFL." for you to understand that your question was redundant.

> You claim that you know how to write in assembly, but you dont. Why dont you write in a language you actually understand?

I know you think it sounds super fancy to say that your crypter is written in ASM but its not.

I am actually forced to write in assembly. Invoking a syscall is not possible in C, instead you're forced to write a piece in assembly. In worst case scenerio I will have to write it without mnemonics, because I will compile it to x86_32. "Why don't you write in a language actually understand" Now you're just being rude instead of being helpful.

>It's literally 2 lines of code in C which then again is optimized by the compiler.

Not really. And not in the future.

>A better alternative than brutng the key and exhausting its resources is to allocate a certain amount of memory, then checking that memory if its actually allocated or not. More efficient and will work just as good without wrecking the cpu.

Thank you

>If you think brutng a key made your crypter 0/13 runtime, then something is terribly wrong with your test because thats just not possible.

If i said that brutng the key alone was making it 0/13, then I was wrong.

Jean-Jacques Rousseau - August 12, 2018, 12:58 pm

Currently working on the syscall mechanism. I got some irl stuff coming up so dont expect anything soon.

Had some problems, couldn't debug them easily either, due to heavens gate. I was, however, successful in calling syscalls that take no arguments, e.g

```
DWORD WINAPI NtGetCurrentProcessorNumber(void) { // SCN: 0x0EC
    DWORD processnum;
    __asm {
        EnterHeavensGate() // we are now executing 64 bit code.
        mov eax, 0ECh // mov eax, 0x0EC
        EMIT(0x0F) EMIT(0x05) //syscall
        mov processnum, eax // mov processnum, eax
        LeaveHeavensGate()
    }
```

```
printf("processnum: %d\n", processnum);
return processnum;
}
```

I think i am passing my arguments wrong. I am also not aligning the stack properly.

Happy coding everyone!

05317832892 - August 13, 2018, 1:10 am

(Aug 12, 2018 12:58 PM)Supercut Wrote: [\[CITE\]](#) Currently working on the syscall mechanism. I got some irl stuff coming up so dont expect anything soon.

Had some problems, couldn't debug them easily either, due to heavens gate. I was, however, successful in calling syscalls that take no arguments, e.g

```
DWORD WINAPI NtGetCurrentProcessorNumber(void) { // SCN: 0x0EC
DWORD processnum;
__asm {
EnterHeavensGate() // we are now executing 64 bit code.
mov eax, 0ECh // mov eax, 0x0EC
EMIT(0x0F) EMIT(0x05) //syscall
mov processnum, eax // mov processnum, eax
LeaveHeavensGate()
}
printf("processnum: %d\n", processnum);
return processnum;
}
```

I think i am passing my arguments wrong. I am also not aligning the stack properly.

Happy coding everyone!

Use windbg to step through heavens gate and onto syscall for a WOW64 process. You can inspect the stack and the registers to see how to format the arguments.

What you are doing here is wrong because the syscall numbers are not hard coded. If syscall numbers didn't change then they would be used like they are on Linux. You need to read ntdll from disk, map it and then write your own getprocaddress implementation to find the VA of the function you want. Then if you are on 32 bit ntdll will start with

```
Mov eax,num
```

Which encodes to

```
?? XX XX XX XX
```

So just add 1 to your pointer and dereference to a DWORD to get the syscall num.

I know a lot of people recommend this method but for me I think a more reliable and easier method is just mapping ntdll and calling the function from your mapped copy.

Also debugging past heavens gate is not possible with x96dbg, immunity, olly etc. the only one I have found to work is windbg. If you don't like the command line and cba to use it then you can attach it with IDA and use it with a friendly GUI.

Jean-Jacques Rousseau - August 13, 2018, 1:16 am

(Aug 13, 2018 01:10 AM)ROOTED Wrote: [\[CITE\]](#) Use windbg to step through heavens gate and onto syscall for a WOW64 process. You can inspect the stack and the registers to see how to format the arguments.

What you are doing here is wrong because the syscall numbers are not hard coded. If syscall numbers didn't change then they would be used like they are on Linux. You need to read ntdll from disk, map it and then write your own getprocaddress implementation to find the VA of the function you want. Then if you are on 32 bit ntdll will start with

Mov eax,num

Which encodes to

?? XX XX XX XX

So just add 1 to your pointer and deterrence to a DWORD to get the syscall num.

I know a lot of people recommend this method but for me I think a more reliable and easier method is just mapping ntdll and calling the function from your mapped copy.

Also debugging past heavens gate is not possible with x96dbg, immunity, olly etc. the only one I have found to work is windbg. If you don't like the command line and cba to use it then you can attach it with IDA and use it with a friendly GUI.

Thank you for your help!

Yes i know that syscall numbers are not hardcoded, i was testing this on Windows 1803 x64.

Thank you for the technique to find syscall numbers! I will start developing it after i have solved what is wrong with my calls that pass arguments.

Happy coding!

05317832892 - August 13, 2018, 2:06 am

(Aug 13, 2018 01:16 AM)Supercut Wrote: [\[CITE\]](#) Thank you for your help!

Yes i know that syscall numbers are not hardcoded, i was testing this on Windows 1803 x64.

Thank you for the technique to find syscall numbers! I will start developing it after i have solved what is wrong with my calls that pass arguments.

Happy coding!

Don't bother. Just map and call ntdll.

Jean-Jacques Rousseau - August 13, 2018, 2:48 am

(Aug 13, 2018 02:06 AM)ROOTED Wrote: [\[CITE\]](#) Don't bother. Just map and call ntdll. I have heard they might be hooked?

Electron - August 13, 2018, 4:28 am

(Aug 13, 2018 02:48 AM)Supercut Wrote: [\[CITE\]](#) I have heard they might be hooked? they still bypass the usermode hooks until they implement syscall hooks.

Jean-Jacques Rousseau - August 13, 2018, 4:37 am

(Aug 13, 2018 04:28 AM)Noxious™ Wrote: [\[CITE\]](#) they still bypass the usermode hooks until they implement syscall hooks.

Syscall hooks are not possible on x64. SSDT is under patch guard.

Calling back register values is possible though, however, i have found a method to "bypass" it.

I will still implement syscalls. It is fun. Lol.

Electron - August 13, 2018, 5:17 am

(Aug 13, 2018 04:37 AM)Supercut Wrote: [\[CITE\]](#) Syscall hooks are not possible on x64. SSDT is under patch guard.

Calling back register values is possible though, however, i have found a method to "bypass" it.

I will still implement syscalls. It is fun. Lol.

Half of the AVs didn't implement same engines as x86 for x64 including scantime.

3DES - August 13, 2018, 6:24 am

(Aug 13, 2018 04:37 AM)Supercut Wrote: [\[CITE\]](#) Syscall hooks are not possible on x64. SSDT is under patch guard.

Calling back register values is possible though, however, i have found a method to "bypass" it.

I will still implement syscalls. It is fun. Lol.

What is this method of bypassing callbacks? Im interested on this one.

Jean-Jacques Rousseau - August 13, 2018, 8:02 am

(Aug 13, 2018 06:24 AM)3DES Wrote: [\[CITE\]](#) What is this method of bypassing callbacks? Im interested on this one.

Sorry. As much as I love learning and teaching, this is one of the things I will keep to myself. For now.

05317832892 - August 13, 2018, 10:34 am

(Aug 13, 2018 02:48 AM)Supercut Wrote: [\[CITE\]](#) I have heard they might be hooked?
If you read a fresh copy of ntdll from disk it won't be hooked.

Jean-Jacques Rousseau - August 13, 2018, 3:19 pm

(Aug 13, 2018 10:34 AM)ROOTED Wrote: [\[CITE\]](#) If you read a fresh copy of ntdll from disk it won't be hooked.

It is too high risk. Syscalls will do.

I dont know how to "read a fresh copy" since AVs replace them, and i cant include it in my crypter due to size

Electron - August 13, 2018, 4:19 pm

(Aug 13, 2018 03:19 PM)Supercut Wrote: [\[CITE\]](#) It is too high risk. Syscalls will do.

I dont know how to "read a fresh copy" since AVs replace them, and i cant include it in my crypter due to size

"I dont know how to "read a fresh copy" - manually mapping a dll?

"i cant include it in my crypter due to size" - it doesn't affect the stub size. (everyone using Windows OS has a copy of ntdll.dll lol)

"It is too high risk." - what? Where is the risk? They won't hook it.

SteamedHams - August 13, 2018, 5:19 pm

(Aug 6, 2018 06:56 PM)Twistd Wrote: [\[CITE\]](#) Why would it be more difficult to fud? thats wrong, you just need to learn to fud properly.

You don't junk your stub with C\C++ code, you junk it with false winapi calls and tiny inline assembly that breaks up your code.

The structure of your stub is whats going to matter the most how fast and easy you'll find those signatures.

There are also many types of detections and it takes time to understand how to approach them differently.

It takes some time, C\C++ is definitely more challenging to fud than C#. You'll get used to it after a while.

How can one guarantee that one stub they made is completely different from another one they made? For example, you see a some crypters that sell individual stubs. How can they guarantee there won't be a cross-detection? Are there tools to compare both files that are used?

Twistd - August 13, 2018, 11:30 pm

(Aug 13, 2018 05:19 PM)SteamedHams Wrote: [\[CITE\]](#) How can one guarantee that one stub they made is completely different from another one they made? For example, you see a some crypters that sell individual stubs. How can they guarantee there won't be a cross-detection? Are there tools to compare both files that are used?

In my previous c\c++\asm crypter I used LLVM ir bitcode for my USG. It managed to make 96% unique stubs. My usg was quite extensive and over 6 months work, I still managed to get cross detections so yes if you're buying a private stub even from the best and most advanced crypters you risk getting cross detections so I wouldn't really recommend buying private stubs from someone you don't really know, once coder gives you a fud stub you have no control over how unique they actually are from other he has sold earlier, plus detections might pop up later and already be on their way to get flagged.... coder fuds it and two days later detections already started appearing even if you didn't use it. My best advice is to make your own or pay someone to make it just for you. But I guess it all depends how serious you are about it.

Yes there are tools to measure the difference in percentage.

Jean-Jacques Rousseau - August 14, 2018, 1:14 am

(Aug 13, 2018 04:19 PM)Noxious™ Wrote: [\[CITE\]](#) "I dont know how to "read a fresh copy" - manually mapping a dll?

"i cant include it in my crypter due to size" - it doesn't affect the stub size. (everyone using Windows OS has a copy of ntdll.dll lol)

"It is too high risk." - what? Where is the risk? They won't hook it.

The ntdll.dll is obviously modified by AVs, a member has already said that dynamically calling is not the way to go.

Manually mapping ntdll is not a good idea unless you are thinking of extracting SCN.

I meant that, it probably is safer to include copies of ntdll in my crypter, rather than relying on system files that are probably modified.

(Aug 13, 2018 05:19 PM)SteamedHams Wrote: [\[CITE\]](#) How can one guarantee that one stub they made is completely different from another one they made? For example, you see a some crypters that sell individual stubs. How can they guarantee there won't be a cross-detection? Are there tools to compare both files that are used?

VX Heaven has great resources.

83.133.184.251/ virensimulation.org/vx88a0.html?id=ea14

Good luck finding the files though.

Electron - August 14, 2018, 2:12 am

(Aug 14, 2018 01:14 AM)Supercut Wrote: [\[CITE\]](#) The ntdll.dll is obviously modified by AVs, a member has already said that dynamically calling is not the way to go.

Manually mapping ntdll is not a good idea unless you are thinking of extracting SCN.

I meant that, it probably is safer to include copies of ntdll in my crypter, rather than relying on system files that are probably modified.

VX Heaven has great resources.

83.133.184.251/ virensimulation.org/vx88a0.html?id=ea14

Good luck finding the files though.

ntdll.dll that's the only one being modified by AVs is the original one that gets mapped in every process by default.

Why it's not a good idea? Tell me 1 AV name that hooks the manually injected ntdll, even game anti cheats don't, well there could be some I don't know of but the most of them don't.

And no, it's not safer to include the ntdll in your crypter because it's pointless, they can't just put hooks in a file on the disk. It's a better idea for you to get GMER or something else, install some AV like Avast/AVG, Emsisoft which performs usermode hooking and learn what how they really work. Don't rely only what the people say but check it yourself and then you will know for sure how it is.

josh28 - August 14, 2018, 11:28 am

(Aug 13, 2018 11:30 PM)Twistd Wrote: [\[CITE\]](#) In my previous cc++asm crypter I used LLVM ir bitcode for my USG. It managed to make 96% unique stubs. My usg was quite extensive and over 6 months work, I still managed to get cross detections so yes if you're buying a private stub even from the best and most advanced crypters you risk getting cross detections so I wouldn't really recommend buying private stubs from someone you don't really know, once coder gives you a fud stub you have no control over how unique they actually are from other he has sold earlier, plus detections might pop up later and already be on their way to get flagged.... coder fuds it and two days later detections already started appearing even if you didn't use it. My best advice is to make your own or pay someone to make it just for you. But I guess it all depends how serious you are about it.

Yes there are tools to measure the difference in percentage.
what tool, link?

Jean-Jacques Rousseau - August 14, 2018, 12:12 pm

(Aug 14, 2018 11:28 AM)josh28 Wrote: [\[CITE\]](#) what tool, link?
uhhh, he just said he used LLVM
llvm.org /docs/BitCodeFormat.html

Twistd - August 14, 2018, 12:28 pm

(Aug 14, 2018 12:12 PM)Supercut Wrote: [\[CITE\]](#) uhhh, he just said he used LLVM

llvm.org /docs/BitCodeFormat.html

I think he was talking about the tool used to compare, not llvm.

I uploaded the file comparer for you here, think i downloaded it from hackforums way way back:

<https://www.sendspace.com/file/ot7bek>

Hey F.B.I - August 14, 2018, 12:43 pm

I wanna give you a offer who will interest you in me opinion, pm me for details.

josha28 - August 18, 2018, 5:54 am

(Aug 14, 2018 12:28 PM)Twistd Wrote: [\[CITE\]](#) I think he was talking about the tool used to compare, not llvm.

I uploaded the file comparer for you here, think i downloaded it from hackforums way way back:

<https://www.sendspace.com/file/ot7bek>

thnx, yes I was talking about the file compare so I can check my usg. hope I don't get infected
dont know how Twistd is getting 96% my files are all 99% with that tool

Twistd - August 18, 2018, 10:11 am

(Aug 18, 2018 05:54 AM)josha28 Wrote: [\[CITE\]](#) thnx, yes I was talking about the file compare so I can check my usg. hope I don't get infected

dont know how Twistd is getting 96% my files are all 99% with that tool

I don't think you know what you're doing to be honest. What are you even comparing? Don't forget that you should always compare raw stub without payload attached, if you're comparing with encrypted payload and different key on each you're obviously gonna get false/wrong results. Pretty basic math.

Jean-Jacques Rousseau - August 18, 2018, 11:20 am

(Aug 18, 2018 05:54 AM)josha28 Wrote: [\[CITE\]](#) thnx, yes I was talking about the file compare so I can check my usg. hope I don't get infected

dont know how Twistd is getting 96% my files are all 99% with that tool

Please dont share your skiddy nonsense here.

Jean-Jacques Rousseau - December 5, 2018, 11:03 am

Started working on this again,

Wrote a very simple process hollowing technique, it does not support rebasing currently.

I call ntdll dynamically right now.

Got the syscall number extractor done. Started working on the x32 -> x64 syscaller just now.

No anti-emulation currently.

<https://spyralscanner.net/result.php?id=FO1EISc7A2> Gaudox

<https://spyralscanner.net/result.php?id=HYzV6niWUm> MessageBoxA called

Current stub size: 16KB

Jean-Jacques Rousseau - December 7, 2018, 6:41 am

Heavens Gate syscaller done.

Working on my own implementation of GetProcAddress, ESET is messing with my development (i think ESET returns 0x0 when retrieving CreateProcessW, but not CreateProcessA, lol?), by working

with the data directories.

Jean-Jacques Rousseau - August 18, 2019, 7:57 am

Crypter basically finished.

Info:

- payload is stored in a section, and its entropy is correct
<https://gyazo.com/e14043a004748e7cca6e41889fde2182>
- there are no scantime detections except defender that just showed up (will clean later)
- ring3 hook bypass (32 bit and 64 bit)
- payload is mapped with a twist

however, no .NET support

[Attention Required! | Cloudflare](#)

i have tested malwarebytes runtime with internet connection (can provide video if someone really is interested). Full bypass with # lite.

Bypass eset runtime as well, but gets detected later (# fault).

Stub size is: ~23kb

My anti emulation is currently weak. I will improve on that, and also my polymorphic engine (that will add junk code in assembly as well). The very base of the polymorphic engine is done (it recognizes instructions, however, i have not written substitutions yet).

Fulya Donmez - August 19, 2019, 7:38 am

How far have you gone with this project?

du0ngpv - August 19, 2019, 9:58 am

hope you release it (or sell with source code included)

Jean-Jacques Rousseau - August 27, 2019, 10:30 am

polymorphism implemented into project. 2-5% of the code is shared within builds. Comes with a cost: stub size increased from 16kb to 100-140kb.

this means the stubs are ~95-98% unique.