

Forum:[Hack Forums](#)

Thread: [Which programming language is best for malware development?](#)

dtb1031231 - March 26, 2024, 3:20 pm

Hello everyone, I want to become a malware developer and all I know is Python, JavaScript, a tiny amount of C and I can build basic applications in PHP.

I want to create a RAT but I don't know what language is best for creating a RAT.

If anyone could reply and give me an answer that would be appreciated.

Kuso - March 26, 2024, 3:59 pm

I definitely know plenty of malware have been written with Javascript. I seen people reverse engineer JS malware plenty.

I would probably think JS would be one of the popular languages due to the browser ecosystem and how authentication is done across multiple browsers including cookies, JWTs and session cookies and the browser ecosystem is surrounded by JS.

I know other languages can for sure work with these data objects as well, but I'm mainly a SE so I'm pretty capped at this point.

However a Rat you would probably see other languages for that.

Dystortion - March 26, 2024, 4:42 pm

(Mar 26, 2024 03:20 PM)dtb1031231 Wrote: [\[CITE\]](#) Hello everyone, I want to become a malware developer and all I know is Python, JavaScript, a tiny amount of C and I can build basic applications in PHP.

I want to create a RAT but I don't know what language is best for creating a RAT.

If anyone could reply and give me an answer that would be appreciated.

TBH one of the C languages is probably the best option. Currently, writing one in C# myself right now, as long as you stay on top of debugging you should be all good.

Fludems - March 26, 2024, 6:05 pm

(Mar 26, 2024 03:59 PM)Kuso Wrote: [\[CITE\]](#) I definitely know plenty of malware have been written with Javascript. I seen people reverse engineer JS malware plenty.

I would probably think JS would be one of the popular languages due to the browser ecosystem and how authentication is done across multiple browsers including cookies, JWTs and session cookies and the browser ecosystem is surrounded by JS.

I know other languages can for sure work with these data objects as well, but I'm mainly a SE so I'm pretty capped at this point.

However a Rat you would probably see other languages for that.

Hey, can you give me more info I'm intrigued and I'm not sure what you mean.

Typically you can't exploit auth too much with javascript because there are things that prevent that which should be in place.

Here's a list, but I'm missing some:

CSRF Tokens: Sites should implement CSRF tokens to prevent CSRF attacks

CORS Policy: Cors Policy should be set correctly to prevent CSRF. This means web browsers won't send the request unless it matches the CORS policy the server has set. It'll do a pre-flight request before hand also to check if it's allowed to.

Cookies: SameSite Samesite attribute should be correctly set on cookies to prevent CSRF.

Cookies: HttpOnly If it's an auth cookie, it should be HttpOnly so javascript has no access.

JWT Cookie JWT's Should be hardened, if they aren't something should be stored that is hardened that allows server to check it's not stolen.

JWT Data Don't store data in the JWT that's sensitive at the same level as a password. For the love of god don't store permissions and roles here either, you should have this server side don't ask the user for what permissions they have.

JWT Secret Make the secret for the JWTs secure. The secret is the servers way of knowing it's a valid JWT. If the secret is exposed or bruteforced, attackers can just forge JWTs and gain access.

JWT Hash JWT's should be hashed, and you should check it was hashed with an algorithm when you receive the request.

Content-Security-Policy Should be used to prevent XSS. It specifies where JS can be retrieved from.

X-Content-Type-Options: nosniff header should prevent Mime Confusion for XSS attacks.

Referrer-policy headers not really XSS just a privacy thing, but they should be set.

Content-Type: text/html; charset=UTF-8 Content type headers should be set. The Charset is extremely important to prevent XSS.

Note on Content Type

Also worth nothing content-type application/json will make any code in a response not executable, but don't rely on that obviously this is a last resort protection, if you end up rendering the response somewhere, or running eval or unsafe functions against the response you'll make your users vulnerable.

Stateless auth

If you're using stateless authentication for the love of god add checks against the user context. There's loads of ways to check this, but you should be checking multiple things that might hint that the tokens are being used by someone who isn't the original user.

Google Chrome Auth Flow Exploited Recently

Recently Google Chrome had some sort of oAuth flow for getting your profile and history or something like this from Google.

The people who make & sell stealers were quick to implement a feature that stole these tokens from a victims filesystem.

This gave the attackers the ability to persist access to Google Accounts no matter how many times the victim changed the password. Luckily you could invalidate the tokens. If you logged the device out in your Google Account remotely it would invalidate the tokens. Using 'log out of all devices' is the better way.

RexBlasen - March 27, 2024, 3:18 am

The few 'RAT' related projects I've built have been with C# - mostly because my projects are less malicious and more of a joke so they tend to do more visual things like throwing dumb images up on the screen and blocking the mouse with a topmost popup etc. .NET makes all of that very easy.

You can also manage communication with your server in 1 line of code using HttpClient or WebClient so it's definitely the language to start with if you want to get something working quickly. (I seem to recall C++ and especially C being a bit of a task to get the likes of cURL working in comparison).

You mention you know a bit of PHP, that's perfect because I also used PHP on the server side of these projects - also extremely easy to get started. All you need really is a PHP page that when accessed by a client, will return a command string (if one has been provided by your control panel). You will need to setup an SQL database for this ideally.

I'd say C# (especially winforms) is the least secure of the 3 C languages but it's by far the easiest and you'd want to have your stub obfuscated anyway...

That's my input anyway, hope you find it helpful.

RedNode - March 27, 2024, 6:55 am

building a rat in c/c++ it also requires from you knowledge in GUI and networking, but there are also other options for you because of your knowledge in PHP you can build a botnet over HTTP and this requires from you c networking and PHP only of course never forget without knowing the windows API making a windows rat will be useless so learn winapi and ntapi optional

gregokis02 - March 27, 2024, 7:12 am

> I want to create a RAT but I don't know what language is best for creating a RAT.

The objective answer is C. There are plenty of reasons why this is the case, but mainly that it lets you manipulate memory at a very low level. There are no restrictions, no garbage collector, few dependencies. Granted, it will require you to have a deep understanding of programming languages and how they work at an operating system level.

More importantly to understanding C in my opinion, would be understanding operating systems because ultimately that is what you will be interacting with. If you're writing malware for Windows, you'll want to be familiar with system libraries provided by Microsoft and even better the Windows NT kernel. The actual functionality of a remote-access-tool is pretty simple... it's just some system calls and networking. The complicated bit comes when you need your code to run undetected by the user and anti-virus software. Running your code at the kernel-level, injecting it into other "deemed safe" processes, hooking onto libraries, are all valid strategies.

... so tl;dr, C is the most suitable for this. It has the least safeguards of any language, which is what you want when you're trying to write code that goes unnoticed. The downside of this, is that it's extremely easy to introduce bugs, system crashes, memory leaks/corruption.

p.s. JavaScript would be a valid language to use for web-based exploits, although I don't know if I'd consider that "malware".

happygonutty - March 27, 2024, 1:24 pm

IMHO the best language would be anything that you want as long as it works and there are no extended dependencies (statically compiled much better but larger size) aside from the already existing dependencies within your target machines. Performance doesn't matter anymore since its already 2024 (for effing-sakes my friends, please stop the C/C++ drama) and almost everybody owns more than 4GB RAM machines unless you want to target IoTs or lower tier machines.

The only three things that makes any malware successful and evade the latest AVs are the following:

1. Art of social engineering and some good persuasion skills.
2. Making or buying an advance or custom crypter.
3. Making or buying a legitimate exploit.

It's not that I am promoting the sellers in this community but you can make tools yourself if you are up for the challenge but if you are lazy, either you copy and enhance somebody's work OR just spend money (if you have the money) and buy tools and get the job done.

gregokis02 - March 27, 2024, 7:36 pm

(Mar 27, 2024 01:24 PM)happygonutty Wrote: [\[CITE\]](#) IMHO the best language would be anything that you want as long as it works and there are no extended dependencies (statically compiled much better but larger size) aside from the already existing dependencies within your target machines. Performance doesn't matter anymore since its already 2024 (for effing-sakes my friends, please stop the C/C++ drama) and almost everybody owns more than 4GB RAM machines unless you want to target IoTs or lower tier machines.

The only three things that makes any malware successful and evade the latest AVs are the following:

1. Art of social engineering and some good persuasion skills.
2. Making or buying an advance or custom crypter.
3. Making or buying a legitimate exploit.

It's not that I am promoting the sellers in this community but you can make tools yourself if you are up for the challenge but if you are lazy, either you copy and enhance somebody's work OR just spend money (if you have the money) and buy tools and get the job done.

I'd disagree with your statement about performance not mattering for malware. Maybe it doesn't matter on the host machine, but if your program eats up memory & CPU that will be (1) easier to detect by looking at stats and overall machine health (2) more challenging to hide in memory, especially if you use techniques like injecting it into a trusted process (3) a lot of exploits are memory based, so having fine-grain memory control is an asset.

In security, the lower level you go, the better. It offers you more control and is harder to guard against because there are more attack vectors.

happygonutty - March 27, 2024, 11:13 pm

(Mar 27, 2024 07:36 PM)gregokis02 Wrote: [\[CITE\]](#) I'd disagree with your statement about performance not mattering for malware. Maybe it doesn't matter on the host machine, but if your program eats up memory & CPU that will be (1) easier to detect by looking at stats and overall machine health (2) more challenging to hide in memory, especially if you use techniques like injecting it into a trusted process (3) a lot of exploits are memory based, so having fine-grain memory control is an asset.

In security, the lower level you go, the better. It offers you more control and is harder to guard against because there are more attack vectors.

With all respect to your point of view and I meant no harm but I totally disagree with you as well. A fun fact that I think you haven't realized is that any language you use, no matter if its C/C++, Rust, MASM32 or whatever magical programming language you think outperforms any language we all know, you would still be relying on the built-in APIs provided by the host machine.

I personally believe that you should do more research and study more in-depth before citing out your personal perspective and coming into an argument with anyone in the community. I encourage you to do more research how Win32 APIs work and how you use them like calling them via DLLs or depends what language you use.

I hope that these kinds of drama should totally stop and just be mature. Anyone can use any language they prefer, its just a matter how creative you are and how you would make it perform faster on your target machine.

And lastly (for the last time and I am not repeating again myself), we are not in the early 2000s that having a 2GB RAM is very expensive and outrageous. Lol. Nowadays even the poorest of the poor can even afford a 6GB or 8GB smartphone which can be infected by RATs written in Java 8 or C# which is less superior to your wicked C programming language. ?

dtb1031231 - March 28, 2024, 2:01 am

(Mar 26, 2024 03:20 PM)dtb1031231 Wrote: [\[CITE\]](#) Hello everyone, I want to become a malware developer and all I know is Python, JavaScript, node JS, obviously html, css and js, a tiny amount of C and I can build basic applications in PHP.

I want to create a RAT but I don't know what language is best for creating a RAT.

If anyone could reply and give me an answer that would be appreciated.

gregokis02 - March 28, 2024, 8:41 pm

(Mar 27, 2024 11:13 PM)happygonutty Wrote: [\[CITE\]](#) With all respect to your point of view and I meant no harm but I totally disagree with you as well. A fun fact that I think you haven't realized is that any language you use, no matter if its C/C++, Rust, MASM32 or whatever magical programming language you think outperforms any language we all know, you would still be relying on the built-in APIs provided by the host machine....I mean yes, but that is true with the majority of software. Of course you'll have dependencies that can bottleneck performance, but that doesn't negate all use cases for writing performant code. In the context of writing malware, it's advantageous to write it in a low level language for detectability. Can you get away with writing malware in a higher level language? Sure. But the question was what is the best programming language for malware. Try to write malware targeted at the windows kernel in Java 8 or C#.

(Mar 27, 2024 11:13 PM)happygonutty Wrote: [\[CITE\]](#) I personally believe that you should do more research and study more in-depth before citing out your personal perspective and coming into an argument with anyone in the community. I encourage you to do more research how Win32 APIs work and how you use them like calling them via DLLs or depends what language you use. My apologies! I hadn't realized the prestige of "hack forums", and that I needed to provide citations. In an effort to educate myself, can you elaborate on what you mean by "how Win32 APIs work and how you use them like calling them via DLLs"...? What exactly am I missing about Win32 APIs?

(Mar 27, 2024 11:13 PM)happygonutty Wrote: [\[CITE\]](#) I hope that these kinds of drama should totally stop and just be mature. Anyone can use any language they prefer, its just a matter how creative

you are and how you would make it perform faster on your target machine.

And lastly (for the last time and I am not repeating again myself), we are not in the early 2000s that having a 2GB RAM is very expensive and outrageous. Lol. Nowadays even the poorest of the poor can even afford a 6GB or 8GB smartphone which can be infected by RATs written in Java 8 or C# which is less superior to your wicked C programming language. ?There's no drama here. Your argument is that "no language is best for malware development, use what you want" and mine is "C is best for malware development". Programming languages, like tools, each serve a purpose and some are better than others for specific use cases... but OP (and everyone for that matter) can use whatever language he wants. He just asked for my opinion and I gave it.

Hackers - March 28, 2024, 8:55 pm

(Mar 27, 2024 11:13 PM)happygonutty Wrote: [\[CITE\]](#) relying on the built-in APIs provided by the host machine.

This isn't actually always true. There are ways to skip over standard APIs that are hooked into by services and use more lower level, internal funcs. You can also use assembly, opcodes (ex: <https://sparksandflames.com/files/x86InstructionChart.html>) , internal system calls, etc. This is still something that can be monitored, so sometimes an API func is just as good I'm sure, but depending on the system, version, and code, you are not always just relying on traditional APIs/libs/dependencies.

happygonutty - March 29, 2024, 1:35 pm

(Mar 28, 2024 08:41 PM)gregokis02 Wrote: [\[CITE\]](#) ...I mean yes, but that is true with the majority of software. Of course you'll have dependencies that can bottleneck performance, but that doesn't negate all use cases for writing performant code. In the context of writing malware, it's advantageous to write it in a low level language for detectability. Can you get away with writing malware in a higher level language? Sure. But the question was what is the best programming language for malware. Try to write malware targeted at the windows kernel in Java 8 or C#.If you ought to do something that requires to perform faster that doesn't rely on the built-in Win32 APIs by Windows, using C would totally be an advantage when it comes to performance but the problem is you would most likely still be relying on them 100% of the time unless you will invent the APIs yourself. If you would do that, that would be ridiculous because your malware would be heavy which is a disadvantage. And also, I have cited and recommended to use a crypter to evade AVs so using a high level language wouldn't be a problem that is why there are still some hackers out there still using outdated versions of Comet RAT or so and couldn't still be detected all thanks to a decent crypter.

(Mar 28, 2024 08:41 PM)gregokis02 Wrote: [\[CITE\]](#) My apologies! I hadn't realized the prestige of "hack forums", and that I needed to provide citations. In an effort to educate myself, can you elaborate on what you mean by "how Win32 APIs work and how you use them like calling them via DLLs"...? What exactly am I missing about Win32 APIs?

Well, it's not about being prestige but rather something to backup your claims like in real life and not just here in the forum. It's like you are attending a court hearing and you couldn't provide evidence. Anyway, if you are to make an executable per say that connects to the internet or depends on the programming language you are using by default it would automatically link to DLLs that it needs to use. I believe this is a topic in C about linkers and compilers, and I expect that you know this.

Since you claim that you are familiar with C and you have no idea how to call these APIs from DLLs, I guess you know the language itself but not totally in-depth but its alright since what we are rambling about is an educational process and I very much respect that.

(Mar 28, 2024 08:41 PM)gregokis02 Wrote: [\[CITE\]](#) There's no drama here. Your argument is that "no language is best for malware development, use what you want" and mine is "C is best for

malware development". Programming languages, like tools, each serve a purpose and some are better than others for specific use cases... but OP (and everyone for that matter) can use whatever language he wants. He just asked for my opinion and I gave it.

I am sorry my friend but I totally perceive this as big drama because you are defending something that you love (which is C) and that you love it so much, you would disagree with other people's opinion and negate them just like what you did to me and claim your opinion is so superior. Lol.

It doesn't matter to me to be honest, I just like putting myself into arguments or making one myself. And I meant no disrespect on your love of C and performance in come of malware development.

(Mar 28, 2024 08:55 PM)nodes Wrote: [\[CITE\]](#) This isn't actually always true. There are ways to skip over standard APIs that are hooked into by services and use more lower level, internal funcs. You can also use assembly, opcodes (ex: <https://sparksandflames.com/files/x86InstructionChart.html>) , internal system calls, etc. This is still something that can be monitored, so sometimes an API func is just as good I'm sure, but depending on the system, version, and code, you are not always just relying on traditional APIs/libs/dependencies.

With all respect to your creations since you are a seller with high reputation (very much kudos to that), its kinda 50/50 in my perspective because you as someone who offers services, sometimes there is a portion of code or functionality that you can make magic in order for it not to rely on APIs right then just opt to using op codes or instruction sets via assembly language? But sometimes, the rest of the portion of your code would still rely on the DLLs. Am I right? Please correct me on this if I am wrong.

gregokis02 - March 29, 2024, 4:38 pm

(Mar 29, 2024 01:35 PM)happygonutty Wrote: [\[CITE\]](#) If you ought to do something that requires to perform faster that doesn't rely on the built-in Win32 APIs by Windows, using C would totally be an advantage when it comes to performance but the problem is you would most likely still be relying on them 100% of the time unless you will invent the APIs yourself. If you would do that, that would be ridiculous because your malware would be heavy which is a disadvantage. I'm not sure how else I can explain it... You can write inefficient code, rely on the windows system libraries, and your code can be the bottleneck for performance. Whether it's because you're causing memory bloat, leaks, or the garbage collector just needs to take priority to clean things up, it's possible for your code to be the culprit - and managing memory yourself is one solution to this.

(Mar 29, 2024 01:35 PM)happygonutty Wrote: [\[CITE\]](#) And also, I have cited and recommended to use a crypter to evade AVs so using a high level language wouldn't be a problem that is why there are still some hackers out there still using outdated versions of Comet RAT or so and couldn't still be detected all thanks to a decent crypter. To cite means to provide a reference to a source of information... where on earth did you do that? Regardless, yes you could use a 3rd party tool to prevent detection. However that means that you're relying on another developer, and now plenty of other malware is using the exact same techniques to avoid detection.

(Mar 29, 2024 01:35 PM)happygonutty Wrote: [\[CITE\]](#) Well, it's not about being prestige but rather something to backup your claims like in real life and not just here in the forum. It's like you are attending a court hearing and you couldn't provide evidence. Anyway, if you are to make an executable per say that connects to the internet or depends on the programming language you are using by default it would automatically link to DLLs that it needs to use. I believe this is a topic in C about linkers and compilers, and I expect that you know this.

Since you claim that you are familiar with C and you have no idea how to call these APIs from DLLs, I guess you know the language itself but not totally in-depth but its alright since what we are rambling about is an educational process and I very much respect that.

-snip-The irony is that I believe you're the one who's out of their league, trying to explain concepts

they don't fully understand because of some superiority complex. Yes, the standard library in C is dynamically linked automatically. Yes, you need to add dynamic links for the windows API dlls in order to use them. How do you think other languages consume Win32 APIs...? Those APIs are written in C/C++. Other languages use wrappers or bindings in order to call them. As an aside I actually don't love C... I think there are plenty of problems with it. I primarily use go, rust, and typescript.

happygonutty - March 30, 2024, 9:45 pm

(Mar 29, 2024 04:38 PM)gregokis02 Wrote: [\[CITE\]](#) I'm not sure how else I can explain it... You can write inefficient code, rely on the windows system libraries, and your code can be the bottleneck for performance. Whether it's because you're causing memory bloat, leaks, or the garbage collector just needs to take priority to clean things up, it's possible for your code to be the culprit - and managing memory yourself is one solution to this.

And just like that you cannot defend something you are trying to defend because of lack of knowledge in malware development and programming in-depth. I believe you should educate yourself more on how things work then you will use the common flaws of a language as an excuse to prove your superiority. Whatever you say, whatever you try to argue with me, programs that you write and run on an operating system heavily rely on built-in APIs provided by the operating system. If what you are trying to say is true, then there shouldn't be different architectures and all binaries should run on any operating system without the need of compiling and linking from one operating system to another.

(Mar 29, 2024 04:38 PM)gregokis02 Wrote: [\[CITE\]](#) To cite means to provide a reference to a source of information... where on earth did you do that? Regardless, yes you could use a 3rd party tool to prevent detection. However that means that you're relying on another developer, and now plenty of other malware is using the exact same techniques to avoid detection.

True if you don't know how to use the 3rd party tool provided and you just rely on what is given and you do not know how to make it evasive towards AVs. I believe you should study more on reverse engineering and educate yourself more on how to use the 3rd party tools. It is very embarrassing for you to say this especially a reputable vendor is with us commenting on this thread. It's like you are saying that he is providing inefficient tools and people should not rely on his tools as you said, it is saturated and can easily be detected. Lol.

(Mar 29, 2024 04:38 PM)gregokis02 Wrote: [\[CITE\]](#) The irony is that I believe you're the one who's out of their league, trying to explain concepts they don't fully understand because of some superiority complex. Yes, the standard library in C is dynamically linked automatically. Yes, you need to add dynamic links for the windows API dlls in order to use them. How do you think other languages consume Win32 APIs...? Those APIs are written in C/C++. Other languages use wrappers or bindings in order to call them. As an aside I actually don't love C... I think there are plenty of problems with it. I primarily use go, rust, and typescript.

Nah, you are the one who shows superiority especially trying to defending a point that you cannot even defend. The answer to your questions lies on the first reply in this multi-quoted reply so I do not need to answer each question further.

And what's funny is that you are changing the topic and you are trying to say that you primarily use Go, Rust and Typescript. Lol.

LaysMeLimon - April 1, 2024, 7:54 am

(Mar 26, 2024 03:20 PM)dtb1031231 Wrote: [\[CITE\]](#) Hello everyone, I want to become a malware developer and all I know is Python, JavaScript, a tiny amount of C and I can build basic applications in PHP.

I want to create a RAT but I don't know what language is best for creating a RAT.

If anyone could reply and give me an answer that would be appreciated.

I would suggest C/C++, since you know a tiny amount of C, i think you should start being more comfortable coding in it and then you can get malware samples and study them. Thats what i'm doing.

Fludems - April 3, 2024, 11:43 am

(Apr 1, 2024 07:54 AM)LaysMeLimon Wrote: [CITE](#) I would suggest C/C++, since you know a tiny amount of C, i think you should start being more comfortable coding in it and then you can get malware samples and study them. Thats what i'm doing.

It's solid advice except, it depends on who you are targeting.

IE If we want to RAT phones C++ / C isn't a good choice.

If we want to RAT Windows, I would argue C++ / C is NOT a good choice.

It's the same reason why I would advise against writing the web server and a whole web app in C++ in 99.9% of situations.

What benefits are we getting by increasing the costs (either in time or general complexity) 10x fold?

I'd argue not many at all in the context of making a RAT in C++ for Windows Targets.

Use something with a good eco-system and a lot of packages you can leverage where it's done for you, and you'll ship whatever you're making a lot faster. C# using an older .NET Core / .NET Framework version is probably a better answer here.

Finding an older version that's most present on the majority of machines nowadays will also make it so your 'target audience' is bigger, if we use the very latest .NET Core a lot of people won't have the dependencies installed, and if we go back to .NET Framework 2.0 they also won't have the dependencies installed.

Here's some Pros & Cons IMO:

C++ Pros: Bigger target audience probably, doesn't require as much dependencies

Speed at runtime / optimisation

C++ Cons: 100X (made up figure) more difficult

100X (made up figure again) slower to ship

100x (and again :P) more fragile to different systems

Do everything from scratch

More likely to just fall over in general and not work as intended

I think the size of the target audience is fine without using C++ and speed isn't necessarily a problem as it's one executable doing some things on a PC, it's not like the executable that is placed

on 'Slave Machines' has to handle 500 other machines at the same time or something.

LaysMeLimon - April 3, 2024, 1:22 pm

(Apr 3, 2024 11:43 AM)Onyx Digital Wrote: [\[CITE\]](#) It's solid advice except, it depends on who you are targeting.

IE If we want to RAT phones C++ / C isn't a good choice.

If we want to RAT Windows, I would argue C++ / C is NOT a good choice.

It's the same reason why I would advise against writing the web server and a whole web app in C++ in 99.9% of situations.

What benefits are we getting by increasing the costs (either in time or general complexity) 10x fold?

I'd argue not many at all in the context of making a RAT in C++ for Windows Targets.

Use something with a good eco-system and a lot of packages you can leverage where it's done for you, and you'll ship whatever you're making a lot faster. C# using an older .NET Core / .NET Framework version is probably a better answer here.

Finding an older version that's most present on the majority of machines nowadays will also make it so your 'target audience' is bigger, if we use the very latest .NET Core a lot of people won't have the dependencies installed, and if we go back to .NET Framework 2.0 they also won't have the dependencies installed.

Here's some Pros & Cons IMO:

C++ Pros: Bigger target audience probably, doesn't require as much dependencies

Speed at runtime / optimisation

C++ Cons: 100X (made up figure) more difficult

100X (made up figure again) slower to ship

100x (and again :P) more fragile to different systems

Do everything from scratch

More likely to just fall over in general and not work as intended

I think the size of the target audience is fine without using C++ and speed isn't necessarily a problem as it's one executable doing some things on a PC, it's not like the executable that is placed on 'Slave Machines' has to handle 500 other machines at the same time or something.

I respectfully disagree with the points about using C++/C for creating malware like RATs. While it's true that the choice of language depends on who you're targeting, I think there are some things that haven't been considered.

First off, saying C++/C isn't good for RATs targeting phones or Windows seems a bit off. Actually,

plenty of sneaky malware has been written in C or C++, taking advantage of their low-level control and power.

The argument about C++/C being too complex and time-consuming compared to something like C# is valid, but it's kind of a trade-off. Sure, C# might be easier to work with, but you sacrifice some control and performance. With C++/C, you get more say over how your malware behaves and performs, which can be a big deal when you're trying to avoid getting caught by antivirus programs.

And about using older versions of .NET Framework/Core to make sure more people can run your malware, that could backfire. Older versions might have security holes that hackers and antivirus software both know about. So you might end up making it easier for someone to detect or even exploit your RAT.

As for the pros and cons of C++, yeah, it can be tougher to work with and might take longer to finish your project. But the benefits, like having more control and being sneakier on different systems, can make it worth it, especially when you're trying to make a RAT that flies under the radar.

Fludems - April 3, 2024, 1:56 pm

(Apr 3, 2024 01:22 PM)LaysMeLimon Wrote: [\[CITE\]](#) I respectfully disagree with the points about using C++/C for creating malware like RATs. While it's true that the choice of language depends on who you're targeting, I think there are some things that haven't been considered.

First off, saying C++/C isn't good for RATs targeting phones or Windows seems a bit off. Actually, plenty of sneaky malware has been written in C or C++, taking advantage of their low-level control and power.

The argument about C++/C being too complex and time-consuming compared to something like C# is valid, but it's kind of a trade-off. Sure, C# might be easier to work with, but you sacrifice some control and performance. With C++/C, you get more say over how your malware behaves and performs, which can be a big deal when you're trying to avoid getting caught by antivirus programs.

And about using older versions of .NET Framework/Core to make sure more people can run your malware, that could backfire. Older versions might have security holes that hackers and antivirus software both know about. So you might end up making it easier for someone to detect or even exploit your RAT.

As for the pros and cons of C++, yeah, it can be tougher to work with and might take longer to finish your project. But the benefits, like having more control and being sneakier on different systems, can make it worth it, especially when you're trying to make a RAT that flies under the radar.

Sorry, but I don't think you know what you're talking about. I don't mean it offensively there's plenty of topics you'd be more educated on than I am, but please be honest with yourself.

Mobile

Android is Kotlin / Java for Apps, iOS is Swift / Objective C. You can't just execute C++ Binaries on these devices.

.NET Version

Chromium on windows is built using C++ (I believe). This doesn't mean that websites need to be built in C++.

In the same way your malware executable can be built using Brainfuck and your control centre can

be built using Bash, they are completely agnostic of one another. Building malware in an older version of .NET Core / Framework does not make the control centre vulnerable, unless you also build your control centre with the vulnerable version for some stupid reason.

It's also way more likely to be vulnerable if you write the control centre in C++ because it's not memory safe, if you leverage a high level language all them attack vectors are taken care of.

C++ Counter Arguments AGAIN

Here's a list of Open Source RAT Projects:

https://github.com/alphaSeclab/awesome-rat/blob/master/Readme_en.md#36c965d7808be2799f70e4d910a17abd

All of the RAT's do not use C++ apart from the ones based on Gh0st which was written 10 years ago. The most popular rat Quasar uses C# and the rest use high level languages also like Go or Python. There's probably a reason for that obviously.

C++ is really not needed to be sneaky and secure you can do most if not all with higher level languages.

The point at which you need access of lower level stuff, is beyond my expertise, yours and OPs.

For performance, you're really not going to have performance issues that are best solved by using C++ in this scenario honestly.

DynamicW - May 2, 2024, 1:58 pm

I think that C/C# are generally the best languages for malware development i think.

BlackWidow1 - May 2, 2024, 5:32 pm

I think the language you already know well.