

Forum:[Hack Forums](#)

Thread: [Hidden Security Risks in Python Code Snippets :O](#)

SIM-5432905 - November 15, 2023, 1:30 am

There is a critical issue regarding recent "open source" Python code snippets that could pose serious security risks.

An example:

<https://pastebin.com/uYwezGvb>

At first glance, this code may seem harmless, but the devil is in the details. The real threat lies within a malicious library called "pypiele."

<https://pastebin.com/BUKtxb2g>

Here's the catch: when you innocently run a script that includes this code, the malicious library is secretly downloaded onto your system, residing in a hidden location. Then, when you execute the seemingly innocent script, the harmful code operates in tandem with the legitimate one, potentially compromising your privacy and security.

What makes this particularly sinister is that the malicious library can wear different names each time, making it hard to detect.

To stay safe, follow this simple step:

1. Visit [pypi.org](<https://pypi.org/>).
2. Look up the names of libraries used in any open source Python program you come across to ensure their legitimacy.

Stay informed, stay safe.

TL;DR: Always verify the legitimacy of libraries in open source Python programs on [pypi.org](<https://pypi.org/>) to avoid hidden security threats.

jackrabbit - November 15, 2023, 3:46 am

Man this isn't only a Python problem. This goes for pretty much all modern programming languages.

This could happen with npm, and when you npm install a package, that package can run scripts during the install. you don't even have to execute it.

I guess same for python with its setup.py, they tend to run GCC to compile native code, but could easily run malware just from pip install.

Also rust has a metaprogramming feature, kind of like the c preprocessor but it's just more rust that compiles and runs before the real program is compiled. So just compiling rust could execute a malicious cargo package.

Plus there's a whole attack about making a typo during your npm/pip install which can easily download a malicious package you didn't mean to. An attacker just needs to make a package with some malware that runs on package install and push to npm/pypi with a title similar to popular packages.

SIM-5432905 - November 15, 2023, 4:11 am

(Nov 15, 2023 04:46 AM)jackrabbit Wrote: [\[CITE\]](#) Man this isn't only a Python problem. This goes for pretty much all modern programming languages.

This could happen with npm, and when you npm install a package, that package can run scripts during the install. you don't even have to execute it.

I guess same for python with its setup.py, they tend to run GCC to compile native code, but could easily run malware just from pip install.

Also rust has a metaprogramming feature, kind of like the c preprocessor but it's just more rust that compiles and runs before the real program is compiled. So just compiling rust could execute a malicious cargo package.

Plus there's a whole attack about making a typo during your npm/pip install which can easily download a malicious package you didn't mean to. An attacker just needs to make a package with some malware that runs on package install and push to npm/pypi with a title similar to popular packages.

The fact that a simple typo during package installation can lead to unintended downloads of malicious packages is crazy

Tentco - November 15, 2023, 4:58 am

(Nov 15, 2023 02:30 AM)m3gapi Wrote: [\[CITE\]](#) There is a critical issue regarding recent "open source" Python code snippets that could pose serious security risks.

An example:

<https://pastebin.com/uYwezGvb>

At first glance, this code may seem harmless, but the devil is in the details. The real threat lies within a malicious library called "pypiele."

<https://pastebin.com/BUKtxb2g>

Here's the catch: when you innocently run a script that includes this code, the malicious library is secretly downloaded onto your system, residing in a hidden location. Then, when you execute the seemingly innocent script, the harmful code operates in tandem with the legitimate one, potentially compromising your privacy and security.

What makes this particularly sinister is that the malicious library can wear different names each time, making it hard to detect.

To stay safe, follow this simple step:

1. Visit [pypi.org](<https://pypi.org/>).
2. Look up the names of libraries used in any open source Python program you come across to ensure their legitimacy.

Stay informed, stay safe.

TL;DR: Always verify the legitimacy of libraries in open source Python programs on [pypi.org](<https://pypi.org/>) to avoid hidden security threats.

This has been going around for a while now same issue with Nuget packages, at this point the package management team should just verify each package or restrict them certain privileges.

GoodNightWorld - November 15, 2023, 6:07 am

Wow...

I didn't even know that things like this could happen. Hard to stay safe these days.

Thanks!

Trisulfide - November 15, 2023, 8:26 pm

I get that it shouldn't be that easy to trick someone into downloading a malicious library, and that there should be a way around it but this is just another case of look at what you're running before you run it? It's the same as someone giving you a random exe saying it give you free money and you click it without doing any research. In the script provided, the library is imported and never used which should raises flags anyway.

Hackers - November 15, 2023, 9:00 pm

This makes it look obvious, the real danger is a package that has real functional use cases, but then can change behavior later. For example, a package that fetches crypto prices could work for months, and then one day an obscure piece of code is triggered - perhaps something such as a function that fetches a value from an API & then executes logic based on a value in that data. Something like this can be altered to trigger different code logic you're unaware of.

Well, perhaps a rule could be made if a crypto market price is returned as a negative number. This is even more tricky if it's disguised as something like "if the price is negative, something is wrong, fetch a new software update from the api". That seems logical because a user may think surely a package update in case of a negative crypto price is safe & a fix.

That's how a future security vulnerability could arise, by downloading a resource, executing it, etc - or for NON-immutable PHP or similar at-runtime software a file can be changed to update production in real time.

Imagine if the -1 crypto patch is actually a new script that breaks out of the project folder to find certain files in root. Or worst, what if your script is able to execute a root / sudo command to add a new SSH key, etc.

It's so important to verify trusted packages, and also to lock down production properly with user permissions, cages, whitelisting, etc.

jackrabbit - November 16, 2023, 9:45 am

More evidence about this issue, I just came across this, but it's not a new story:

<https://david-gilbertson.medium.com/im-harvesting-credit-card-numbers-and-passwords-from-your-site-here-s-how-9a8cb347c5b5>

tl;dr: guy made a useful npm package, added as a dependency to other projects. It scans forms for

credit card numbers and forwards them. To prevent being caught it only forwarded requests at night and at random and only if dev tools are closed so it never appears in network tab. GitHub project for code is clean, and package with bad code is pushed to npm.

(Nov 15, 2023 09:26 PM)Trisulfide Wrote: [\[CITE\]](#) I get that it shouldn't be that easy to trick someone into downloading a malicious library, and that there should be a way around it but this is just another case of look at what you're running before you run it? It's the same as someone giving you a random exe saying it give you free money and you click it without doing any research. In the script provided, the library is imported and never used which should raises flags anyway.

The problem is there's too many dependencies. A project I'm working on now has 30 direct dependencies which results in 655 npm packages. There's no way I could vet all 655 packages in a reasonable time. Thankfully my site doesn't collect any personal info to be any kind of threat.

Hackers - November 16, 2023, 3:06 pm

(Nov 16, 2023 10:45 AM)jackrabbit Wrote: [\[CITE\]](#) More evidence about this issue, I just came across this, but it's not a new story:

<https://david-gilbertson.medium.com/im-harvesting-credit-card-numbers-and-passwords-from-your-site-here-s-how-9a8cb347c5b5>

tldr: guy made a useful npm package, added as a dependency to other projects. It scans forms for credit card numbers and forwards them. To prevent being caught it only forwarded requests at night and at random and only if dev tools are closed so it never appears in network tab. GitHub project for code is clean, and package with bad code is pushed to npm.

The problem is there's too many dependencies. A project I'm working on now has 30 direct dependencies which results in 655 npm packages. There's no way I could vet all 655 packages in a reasonable time. Thankfully my site doesn't collect any personal info to be any kind of threat.

Yep, exactly, so many projects are probably overly vulnerable. I've noticed that a lot of Jr's go for newer packages because they're trendy, or recently updated, but a lot of times older packages are more than equipped to take on the task, and way more vetted, secure, etc.