VMConnect: Malicious PyPI packages imitate popular open source modules | ReversingLabs

By Karlo Zanki, Reverse Engineer at ReversingLabsKarlo Zanki

Published: 2023-08-03 · Archived: 2026-04-05 20:11:31 UTC

ReversingLabs has identified several malicious Python packages on the Python Package Index (PyPI) open source repository. In all, ReversingLabs researchers uncovered 24 malicious packages imitating three, popular open source Python tools: *vConnector*, a wrapper module for pyVmomi VMware vSphere bindings; as well as *eth-tester*, a collection of tools for testing ethereum based applications; and *databases*, a tool that gives asyncro support for a range of databases.

Based on the research team's observations, the campaign began on or around July 28, 2023, when the first of the malicious packages were published. It continues to the current day, with new, malicious PyPI packages posted on a daily basis, as prior packages are detected and removed.

In contrast to other, recent supply chain campaigns, such as Operation Brainleeches, the malicious packages that make up this campaign display evidence of a concerted effort to deceive developers. They achieve this by implementing the entire functionality of the modules they are imitating and standing up corresponding and linked GitHub projects that omit the malicious functionality found in the PyPI release package.

This is not the first time that we have observed such behavior. In June, 2022, for example, we discovered an npm malicious package, *maintenancewebsite*, which used a similar approach to hide cryptomining features. The VMConnect campaign is the latest example of open source modules being used to propagate malicious code, and more evidence that security assessments of open source code repositories may miss these nuanced attacks.

The ReversingLabs research team is continuously monitoring open-source package repositories for instances of malicious code tampering, the planting of malicious packages or dependencies and other forms of software supply chain attacks. This work involves both automated and human-led scanning and analysis of packages published in the most popular public package repositories like npm, PyPI, Ruby and NuGet.

Historically, the vast majority of the malicious supply chain campaigns we have identified were found on the npm open source repository, which is home to the lion’s share of open source projects and developers. However, in recent months other platforms, in particular, the Python Package Index (PyPI) have seen increased malicious activity. In February, for example, ReversingLabs researchers discovered 41 malicious PyPI packages posing as HTTP libraries, with some mimicking popular and widely used libraries. In March, we encountered a malicious PyPI package named termcolour, a three-stage downloader published in multiple versions that co-opted the name of a defunct PyPI package. Then, in May, PyPI briefly stopped accepting new submissions after it was overwhelmed with malicious submissions.

ReversingLabs threat researchers have identified a new malicious PyPI campaign that includes a suspicious VMConnect package published to the PyPI repository.

Suspicious behavior detected

This package was declared suspicious during routine scanning by ReversingLabs Titanium Platform, a powerful static analysis engine capable of extracting various types of metadata from a wide range of file formats.

Besides the various types of metadata, the ReversingLabs Titanium Platform is also capable of extracting behavior indicators, making it easier to understand functional capabilities of a file. And that capability is what drew our attention to the *_init_.py* file within VMConnect. Code inside VMConnect's *__init__.py* file is capable of creating a process, decoding data using the Base64 algorithm and converting binary data to its string representation — a behavior commonly used in obfuscation. This combination of behaviors is what triggered the initial detection and prompted further investigation.

Figure 1: Behavior indicators extracted from VMConnect package.

A detailed look at the *__init__.py* file confirmed the presence of malicious functionality inside the package. That started with that Base64 encoded string, which gets decoded and executed in another process.

Figure 2: Execution of Base64 encoded string inside the *__init__.py* file.

When we decode the string, we discovered that it contains a download URL which is modified based on the information collected from the host machine. The substring *paperpin3902* in the command and control URL is replaced with a string containing the first letter of the host's platform name, username and a random, 6 character-long string.

Figure 3: C2 URL template, extracted from the Base64 encoded string, gets modified based on host machine information.

The decoded and executed Base64 string contains an endless execution loop which contacts the command and control (C2) server and attempts to download another Base64 encoded string with additional commands. If it succeeds, that code is executed and the loop repeats, with the C2 server polled by the infected host for new commands after a preconfigured sleep period.

Figure 4: Command fetching and execution loop extracted from the Base64 encoded string present in the *__init__.py*file.

Although the C2 server was live at the time of this research, the research team did not observe it serving any commands. Since the command fetching is performed in an endless loop, it is possible that the operator of the C2 server uploads commands only after the infected machine is determined to be interesting to the threat actor. Alternatively, the C2 server could be performing some type of request filtering. For example, attackers may filter requests based on the IP address of the infected machine to avoid infecting targets from specific countries.

Github misdirection

The VMConnect package was published on July 28th, by a developer named *Hushki Manager* — a throw-away PyPI account created the same day when its only package was published.

Figure 5: PyPI profile of hushki502 user.

Despite that, the actor did put a lot of effort into making the package look trustworthy. For example, it has a legitimate-looking description, and that description corresponds to the functionality present inside the package. Threat actors often don't make the effort to tidy up the project's description on the PyPI website.

Instead, they often rely on simple typosquatting of package names in order to trick enough developers into installing their creation. For example, the threat actors behind the recently discovered Operation Brainleeches campaign made minimal efforts to disguise their malicious packages, using default file names like index.html and DEMO.txt likely copied from phishing kits and minimally altered before being published.

Figure 6: VMConnect package description.

In this campaign, however, the attackers took a more studied approach. In addition to reusing the description from the actual packages, the attackers properly set links to a Github source code repository which was created by the same author on the same day. The Github project site looks trustworthy, with a description matching the one on the PyPI site and several commits and nothing obviously suspicious in the published files.

Figure 7: Github profile of hushki502 user.

Finally, the malicious actors took care to hide the malicious nature of this tool by omitting malicious functionality from the *__init__.py* file that was published to the Github repository.

Figure 8: Benign version of *__init__.py* file present in Github repository.

The use of corresponding GitHub repositories to create the impression of a legitimate open source package is something we have seen before - and there's a good explanation for why threat actors are taking the trouble to do this.

Historically, many supply chain security solutions have relied on source code reviews of third party libraries. Attackers, therefore, have an incentive to throw code scanning tools and manual code reviewers off their scent. For reviewers, PyPI projects tend to look more trustworthy if they have a corresponding link to a Github repository that doesn't seem arbitrary.

In the case of the malicious PyPI packages, the attacker created a phony Github repository with an identical name as the PyPI package and copied the entire functionality from the legitimate Github project into the corresponding PyPI project. However, the malicious functionality is not present within the source code. It is only by scanning the artifacts used in the build process that this threat would have been detected.

ReversingLabs observed a similar tactic used in the case of the npm coinminer discovered a year ago. In that campaign, the content of the release package - a temporary "maintenance mode" website" — was also different from the content hosted in the corresponding source code repository. As with the current PyPI packages, malicious content was added to the legitimate public source code resulting in a PyPI release package that looks much the same as the open source code it is built on, but with some subtle (and malicious) changes that are easily overlooked.

The lesson for development and application security teams is that release packages can (and do) contain malicious functionality which isn't present in open source repositories and therefore can't be detected with source code scanning or manual source code reviews alone. These teams need a method for detecting suspicious content in the final release packages themselves to avoid falling victim to supply chain attacks such as this VMConnect PyPI campaign.

Imitation: The sincerest form of thievery

It is also educational to take a closer look at the methods the attackers in this case used to disguise their malicious intent, including dressing their malicious wares up to imitate legitimate and widely used open source packages.

As we noted above, the packages we detected mimic a variety of well known PyPI packages with wide distributions, but serving very different ends. The common thread connecting the PyPI packages that were imitated appears to be nothing more than their popularity, measured in monthly downloads.

In each case, the attackers disguised their PyPI packages to look like these widely used and legitimate tools. That included copying the package description and pasting it into their imitation packages, simply replacing the legitimate package name with the name of their impostor package.

From the standpoint of a threat researcher or incident responder, their lack of effort makes the job of detecting the ruse easier. In our research, a Google search for the first sentence of the description immediately revealed the corresponding, legitimate package that was being mimickedAs visible in Figures 9 and 10, the descriptions of the PyPI packages and Github projects are identical, aside from the project name.

Figure 9: Github description of the malicious VMConnect project.

Figure 10: Github description of the legitimate vConnector project.

In this case, vConnector is a fairly popular package, first published nine years ago, with the last modification being committed to the Github repository almost four years ago. Download stats for this package show that it has almost 40 thousand downloads per month. A large number of monthly downloads combined with the lack of recent maintenance make this package a very good target for impersonation.

On July 19, seven different versions of a package titled *osinfopkg* were published to the PyPI repository and used to develop and test the malicious functionality that eventually got included into the VMConnect package.Since this was a testing package, the malicious actor didn't give too much care into making it look trustworthy, even though it did create a dedicated Github repository containing two more projects.

Then, on July 31, another package named *ethter*, was released. In this case, the threat actor impersonated the popular package eth-tester — a collection of tools for testing ethereum based applications which has more than 60 thousand monthly downloads. As the ReversingLabs research team observed with the VMConnect package, the ethter description was copied verbatim from the eth-tester package, with only the package name replaced.

Here again, a phony Github repository was created and malicious functionality was hidden in the *utils/based.py* file inside the corresponding PyPI package. The malicious file isn't present in the Github source code repository, only in the PyPI package. The threat actor also released the latest version of their malicious package with the

1.10.1b1 version number. That is in line with the version numbering convention of the impersonated package, and is higher than the latest released version of the impersonated package. The same approach is applied to the malicious *quantiumbase* package impersonating legitimate databases package.

Attackers play Whac-a-Module

Fortunately for Python developers, none of the packages we detected were available for download for very long. Most, including the VMConnect package, were removed from PyPI within one to three days of being posted.

Because of limitations in the PyPI platform, we do not know the reason that these packages were removed: whether they were detected by internal systems operated by PyPI itself, whether reports from ReversingLabs or other firms led to the takedowns, or whether the malicious actors themselves removed the packages. Efforts to get a definitive answer from PyPI on the circumstances that led to the various take downs were not successful.

However, we have observed that new malicious packages get published to PyPI on a daily basis, as soon as the previous package is reported and removed from the PyPI repository. In other words, despite the detection and removal of the packages described above, this malicious PyPI campaign is alive and ongoing. The quick response from threat actors to replace malicious PyPI packages suggests that this is a well-planned and organized campaign and that the malicious actors probably have a prepared list of packages suitable for imitation.

Questions linger on who, what, and why

There are lingering questions about key elements of this malicious, supply chain campaign. Among them: who (or what) posted the malicious packages, what the ultimate objective of the attacks is and why the campaign was launched.

On the question of who, we do not have sufficient telemetry to pinpoint a threat actor or actors as being responsible for the latest campaign. By publishing the IOCs including command and control infrastructure we have collected, however, we hope that others may be able to connect those with evidence of known attacks and threat actors to help fill in the full picture behind this campaign.

Previous supply chain attacks utilizing PyPI, npm and other public repositories range from sophisticated (IconBurst) to unsophisticated (Operation Brainleeches). While the VMConnect campaign doesn't have many of the telltale signs of a sophisticated nation-state campaign, it is not a copy and paste "script kiddie" operation either. More data is required to reach conclusions about who or what is responsible for this PyPI malicious campaign.

Similarly, the research team was unable to obtain the second stage deliverable used in these attacks. As noted, the compromised endpoints ReversingLabs observed merely polled C2 servers waiting for further commands (and presumably downloads). But we were unable to observe any active exchanges, which could indicate that the malicious actors were not actively using the infrastructure, or that the compromised endpoints we controlled were not of interest to them.

Lacking any visibility into the later stages of this campaign, it is impossible to know what its ultimate purpose was: theft of sensitive data or intellectual property? Surveillance? Ransomware? All of the above? More data that

reveals the full breadth of this campaign is needed before we can speculate on its intent.

Indicators of Compromise (IOCs)

Indicators of Compromise (IoCs) refer to forensic artifacts or evidence related to a security breach or unauthorized activity on a computer network or system. IOCs play a crucial role in cybersecurity investigations and cyber incident response efforts, helping analysts and cybersecurity professionals identify and detect potential security incidents.

The following IOCs were collected as part of ReversingLabs investigation of the VMConnect software supply chain campaign.

Command and Control (C2) domains and IP address:

45.61.139.219

ethertestnet.pro

deliworkshopexpress.xyz

PyPI packages:

package_name	version	SHA1
VMConnect	1.1.7	b0095f149951241c6e11e0d1be1f74e8cdfbdbb2
VMConnect	1.1.7	2ff1b3aa2dbff6d87447b250a8d19241e7853ab0
osinfopkg	0.0.2	67226da423ab4a2c97b2d008dec45280aaa5fdf5
osinfopkg	0.0.2	146942c5dbaba55be174b1bfb127410e332caa03
osinfopkg	0.0.3	0eb79e80c51c0e14be3620dfb237f7b53160a292
osinfopkg	0.0.3	bc2d48d6d9eeaf0b29625683942e90dfd2b75723
osinfopkg	0.0.4	9a276ca3678898f5596166416f7e709a2064e95c

osinfopkg	0.0.4	658605988c7afd9adf437fb64ff682cb4190f144
osinfopkg	1.0.1	5f03b73d56528ecbc3f24b8e7daec6b3d3370834
osinfopkg	1.0.1	19684554e4905bb3cf354a5d5a0f00d696f38926
osinfopkg	1.0.2	e531121b137182453f0d120be860ad882d2dc0a7
osinfopkg	1.0.2	b1f2d50be0aca0672475488d77c6f71a1b0633f8
osinfopkg	1.0.3	de4e9efeace6ff76dc00a166dca152dc3021d799
osinfopkg	1.0.3	664f0913a5952eeb77373f83e090fab7e94aa45e
osinfopkg	1.0.4	bd7ba47f730c2bc33afa67a39d9cbe3768f62426
osinfopkg	1.0.4	0dc723e77a5b97183a90eaecb62c9b7341e483ed
ethter	0.9.1b1	6bf76b01bd17f370cd3f9947135bf250597d1ac1
ethter	0.9.1b1	497df2fd2dba324be04cc57f50a3170b532aa70c
ethter	1.10.1b1	d404a55f1f7fbcd8b3156a84ebcf97c57ba24b95
ethter	1.10.1b1	9588affaf9d85e2141b9d76b914d9f89a8292574
quantiumbase	0.7.0	dbc14c3ac0528a8aeb6edba8a0b2792dab131102
quantiumbase	0.7.0	0b7b4444f820e9990dfeb5e2080321b5f25a9785

quantiumbase	0.8.1	e6494b9a91862191556d77022e5577ddbe749ef4
quantiumbase	0.8.1	a1b039f88c385f5c5eec2ef1701251c7341b1fcd

Conclusion

These latest examples of malicious packages on the PyPI platform are typical in many ways. They imitate popular and legitimate open source packages with tens of thousands monthly downloads and hide malicious functionality inside Base64 encoded strings which downloads additional commands from a C2 server. The ReversingLabs research team has seen many variations on this attack pattern in recent months.

What does this mean for security teams and development organizations? The VMConnect campaign is a reminder that threats lurk on open source repositories, and that standard practices like automated source code reviews are not enough to smoke them out.

Source code for the malicious modules published to GitHub was non malicious, but significantly different from the content of the release packages that eventually get compiled into the release artifacts as third party dependencies. Clues that something was amiss — like Base64 encoded strings — were easy to spot, if you know where to look for them. Still, such indicators often escape notice by conventional application security testing and manual code reviews.

The ReversingLabs Software Supply Chain Security platform can assist you in security assessments, helping sniff out signs of malicious functionality, assess third party package dependencies, and provide a wide range of behavior indicators. These indicators can be extracted from various file formats and used to detect this type of threat before it results in malicious code being deployed in your environment.
