

Snakes & Ladders: the offensive use of Python on Windows

theta.co.nz/news-blogs/cyber-security-blog/snakes-ladders-the-offensive-use-of-python-on-windows/

- [Home](#)
- [News & Blogs](#)
- [Cyber Security Blog](#)
- Snakes & Ladders: the offensive use of Python on Windows



04/12/2020

As Microsoft further integrates Python into its ecosystem, there are concerns around the offensive use of it. Can these offensive attacks be mitigated and are the current control mechanisms enough to stop this happening?

Recent news that Python's founder (and previous BDFL) Guido Van Rossum is working with Microsoft - along with the prospect of integration between Python and Windows - is an interesting and ultimately good development for fans of the versatile and incredibly popular language. However, the offensive use of Python on Windows remains an edge case to be considered (as some have) by defensive and offensive security practitioners alike.

The 'state of the art'

Theta's Cyber Security Practice has several forward-leaning observations around the offensive use of Python in Windows environments, which may only accelerate in relevance as Microsoft further integrates the language into its ecosystem. Much of the offensive tradecraft and detection engineering efforts of the world are currently (and rightly) focused on things like **.NET** and **C#**, along with coverage of "classic" windows **LOLBins** (so-called "living off the land binaries" – such as bitsadmin, certutil, scrobj and mavinject etc).

This comes after the instrumentation and subsequent transition away from PowerShell and cmd.exe before that. However, Python currently remains a good way to deploy and execute tooling and scripts onto Windows hosts: either as a scripting interface or as a fully-fledged interpreter.

Traditional use of offensive Python has been focused around its usage on dedicated Linux hosts for pentesting, particularly for network or packet manipulation, or in the delivery of fully functioning and otherwise "fat" remote access trojans (or RATS) to systems. While MITRE ATT&CK has the sub technique T1059.006 - Command and Scripting Interpreter: Python, few are familiar with its capabilities - specifically in relation to Windows. Cobalt Strike's flexible Agressor Scripts (its own native scripting language for modification and extension of the popular offensive software) comes with a hook for Python as an example.

Year of the snake

Unless it's been disabled, Python can be installed from the Windows store (programmatically via ``winget Python.Python``, if winget is deployed – which itself is likely to see an uptick in adoption and worthy of practitioners' understanding). Whilst installed Python packages **do** touch disk and **can** be discovered by investigators, there aren't many reasons to hunt for them.

We've observed first hand the ability to directly download offensive modules via pip onto otherwise orchestrated or monitored systems. EDR and AV are likely to turn a blind eye to packages like Skelsec's Pypykatz – an implementation of the infamous Mimikatz (admittedly with some reduced functionality), the ever-useful Impaket (GetUserSPNs.py) or FOXIT's Bloodhound.py ingestor module. Rounding out offensive capabilities, there are also Python implementations of psexec and wmiexec (part of the lsassy project).

None of any of these items would be expected to be on a normal corporate workstation and whose presence would cause alarm to a human analyst.











In systems where the Windows store isn't available, Python can be downloaded from the web. Doing so is extremely unlikely to trigger any detection by either host or network instrumentation: proxies, firewalls or other security controls simply do not classify IDEs as malicious - and nor should they.

Unsettlingly, we've been able to conduct a complete kerberoast of a domain from a single host using Python from the Windows Store without any need for lateral movement, privilege escalation or defensive evasion; all while AV was running on the machine.

Defensive Advice

Unlike PowerShell, Python's logging isn't well understood or documented. Execution or Transcript logging doesn't exist as it does with modern PowerShell, nor is there any equivalency of AMSI. Python requires the inclusion of additional modules to write out to the Windows Event Log, making it a dark spot for introspection and analysis – which is ironic given the extremely heavy use of Python in the forensics world for *performing* analysis.

From an analysis standpoint, finding the following on-disk is cause for some alarm and would be considered high fidelity sign of malicious activity if there wasn't legitimate testing being carried out.

 pypykatz.exe	11/06/2020 2:00 pm	Application	91 KB
 minidump.exe	11/06/2020 2:00 pm	Application	91 KB
 msldap.exe	11/06/2020 2:00 pm	Application	91 KB
 ccache2kirbi.exe	11/06/2020 2:00 pm	Application	91 KB
 ccacheedit.exe	11/06/2020 2:00 pm	Application	91 KB
 ccacheroast.exe	11/06/2020 2:00 pm	Application	91 KB
 getS4U2proxy.exe	11/06/2020 2:00 pm	Application	91 KB
 getTGS.exe	11/06/2020 2:00 pm	Application	91 KB
 getTGT.exe	11/06/2020 2:00 pm	Application	91 KB
 kirbi2ccache.exe	11/06/2020 2:00 pm	Application	91 KB

Source: %LOCALAPPDATA%\Programs\Python\Python<Version>\Scripts\

We debated releasing some Yara to look for these indicators. But rather than focusing on brittle detections for attributes like file names (remember the pyramid of pain), it's likely to be more useful if we understand what types of systems should be installing fully-fledged interpreters - particularly on servers where workloads should be fairly static. Albeit less straightforward advice: the reasons for non-technical users to suddenly deploy Python on a Windows endpoint remain limited, although the effort or cost to get that level of insight and visibility across an estate isn't trivial.

For those with hunt programs, developing Python-specific hunts or including Python artefacts in them may prove fruitful, along with potentially building out specific detections for Python execution. Hunting across

%LOCALAPPDATA%\Programs\Python\Python<Version>\Scripts\ for evidence of pip module names could prove a simple start. Detections based on pypsexec or similar lateral movement tools are also likely to be higher fidelity signs of something amiss than their non-Python equivalents – although this moves towards network instrumentation or in-memory detections.

At a more abstract level, looking for other languages (particularly ones with Windows interpreters or script interfaces) with these sorts of techniques is a valid exercise as more and more offensive research about [different](#) frameworks and [languages](#) available to practitioners continues to surface. This type of over-the-horizon scanning may not be relevant for what's happening today, but new offensive tradecraft has been shown to be adopted rapidly by sophisticated adversaries and red teams once it's been shown to work; understanding what's going on in the Twittersphere and GitHub provides some degree of early warning.

System administrators and security practitioners should also be cognizant of the Windows Store and develop a strategy for its use in their organisations. Microsoft's traditional model of application deployment being a free-for-all is slowly getting a viable alternative, but as with all things, there is an edge case around malicious usage to be considered. This goes beyond just Python. The opportunity to use WSL (or WSL2) to do ill is large and an [interesting topic](#) all of its own (going all the way to having [Kali Linux](#) built-in). Although, in our experience, this is more likely a trap for curious internal users than serious adversaries.

Summary

Further research by the community is required into the additional forensic opportunities available with Python Scripting on Windows; very little is currently available in the public domain. We can only hope that Microsoft places extra attention on working towards logging parity between PowerShell and Python in Windows. Especially if it's going to get further integrated into the Windows ecosystem.

Until then, you should strap yourselves in. If offensive use of Python and supporting features like NuGet goes the same way as PowerShell, then we're all in for a rough ride.

[Speak to us](#) at Theta if you want to know more about advanced cyber threats, defensive strategies and solutions to protect your organisation.

Lead author: Hamish Krebs, Lead Consultant



Hamish has spent time across Australia and New Zealand responding to advanced threat actors; running large DFIR engagements in complex environments. He's also designed and deployed a variety of security solutions such as SIEMs and EDR suites across APAC.