Panchan's Mining Rig: New Golang Peer-to-Peer Botnet Says "Hi!"

akamai.com/blog/security/new-p2p-botnet-panchan

Akamai Security Research

June 15, 2022

Written by: Stiv Kupchik

Executive summary

- Akamai security researchers discovered **Panchan**, a new peer-to-peer botnet and SSH worm that emerged in March 2022 and has been actively breaching Linux servers since.
- Panchan is written in Golang, and utilizes its built-in concurrency features to maximize spreadability and execute malware modules.
- In addition to the "basic" SSH dictionary attack that is commonplace in most worms, this malware also harvests SSH keys to perform lateral movement.
- Akamai security researchers were able to gain access to the malware's communication protocol and its administration panel, and use them to analyze the infection scope of the malware.
- The most common victim vertical of Panchan (after telecom/VPS) is education. We assume collaborations among different academic institutes might cause SSH keys to be shared across networks, which may explain why this vertical tops the list. Akamai security researchers reached out to the abuse emails associated with each victim IP.
- To avoid detection and reduce traceability, the malware drops its cryptominers as memory-mapped files, without any disk presence. It also kills the cryptominer processes if it detects any process monitoring.
- Based on the malware's activity and victim geolocation, admin panel language, and the threat actor's Discord user's activity, we believe the threat actor is Japanese.
- Akamai MFA can mitigate the risk presented by SSH key harvesting. In addition, configuring strong SSH passwords should stop the malware in its tracks since it uses a very basic list of default passwords to spread. We have also published IOCs, queries, signatures, and scripts that can be used to test for infection.

Introduction

Panchan's mining rig is a feature-packed Golang botnet and cryptojacker. Its peer-to-peer protocol is straightforward — plaintext over TCP, yet it is effective enough to decentralize the botnet. It is also capable of persistence and perseverance, and of monitoring evasion.

Baked into the malware is a "godmode" — an admin panel that is capable of editing the mining configuration, which is then dispersed to the rest of its peers. To prevent unwanted tampering, a private key is required to access godmode, which is then used to sign the mining configuration. The malware contains a public key that is used to verify the supplied private key. **The admin panel is written in Japanese, which hints at the creator's geolocation**.

The botnet introduces a unique (and possibly novel) approach to lateral movement by harvesting of SSH keys. Instead of just using brute force or dictionary attacks on randomized IP addresses like most botnets do, the malware also reads the *id_rsa and known_hosts* files to harvest existing credentials and use them to move laterally across the network.

The malware is written in Golang and uses Golang's concurrency features for most of the main logic — by running them as concurrent Go routines. The threat actor is on top of new Go releases — the earliest detected malware version (from March 2022) was compiled using Go 1.17.7 (released February 2022), while the latest sample was compiled using Go 1.18 (released March 2022). In addition, Go 1.18 had some changes made to its internal data structures, so neither any of the online tools nor the IDA disassembler could parse the malware correctly and match function name to function pointer. In Appendix A: Short dive into Go reversing, we describe how we overcame this.

In this report, we will describe the malware's capabilities in detail, how we detected it, and the process we went through to attempt to attribute it.

For a list of IoCs you can see the Github repository here.

Malware activity

The Akamai Security Research Team proactively monitors botnet and malware activity on our global sensor network. We first noticed Panchan's activity on March 19, 2022. The malware's peer-to-peer communication and wormability drew our attention, and warranted further investigation.

While reverse engineering the malware, we developed scripts to "tune in" to the botnet network, which allowed the team to gather a full list of infected machines (botnet peers). We found 209 peers, 40 of which are currently active.

While the targets are dispersed worldwide, there seems to be a heavier concentration of targets in Asia.

Our attribution of the threat actor's Japanese origins (detailed later) may explain the larger number of targets in Asia. Since it doesn't seem that there is an organization behind this malware, it is plausible that it is easier for the threat actor to stick to the close and familiar.

Looking at victim verticals, most of the victim IPs are registered under their hosting/VPS platform, so there isn't much information there. The most common vertical among monitored victims was education. This might be due to poor password hygiene, or it could be related to the malware's unique lateral movement capability with stolen SSH keys. Researchers in different academic institutions might collaborate more frequently than employees in the business sector, and require credentials to authenticate to machines that are outside of their organization/network. Strengthening that hypothesis, we saw that some of the universities involved were from the same country (e.g.,Spain) and others were from the same region (e.g., Taiwan and Hong Kong).

Exploring the malware's features

Infection vector — SSH worm

The malware is capable of self-spreading through SSH. It has two methods of generating targets and authentication details:

Existing SSH keys

> The malware looks under the running user *HOME* directory for SSH configuration and keys. It reads the private key under *~HOME/.ssh/id_rsa* and uses it to attempt to authenticate to any IP address found under *~HOME/.ssh/known_hosts*. This is a novel credential harvesting method we haven't seen used in other malware.

Brute-forcing credentials

> The malware can randomize IP addresses and attempt a dictionary attack using a predetermined user and password list. The brute-forcing spreader is spawned in a separate process multiple times, limited only by the OS's set limit on open files. The usernames and passwords are fairly simple — combinations of default strings like "ubuntu," "root," "user," '‘debian," "pi," etc.

After a successful authentication to the target, the malware creates a hidden folder with a random name under the root directory /, and copies itself to the hidden folder with the name **xinetd** using sftp.

The malware then remotely executes the copied binary on the target machine (using *nohup*) and passes it a list of peers over the command line. **After a successful infection, the malware initiates an HTTPS POST operation to a Discord webhook**, which is probably used for victim monitoring.

Peer-to-peer communication

he botnet's peer-to-peer protocol is fairly simple. Everything is sent in plaintext over **TCP port 1919**. All peers listen on that port **and create a rule to allow it in iptables**. Each message begins with pan-chan's mining rig hi! and ends with finish. Between them, the malware sends configuration commands separated by newlines. There are only two configuration options that we've seen: *sharepeer* and *sharerigconfig*.

sharepeer is fairly straightforward — it is followed by an IP, which is then added to the malware's internal peer list.

sharerigconfig is followed by a base64-encoded string, which is actually a JSON structure that encodes the mining configuration, and a signature of that configuration:

The signature is validated using an internally saved public key to ensure authenticity. The communication logic is also simple — upon a connection to or from a peer, the malware parses its memory-saved configuration (that was obtained earlier, when it started executing), generates the message string, and sends it. It also receives a similar message from the other side and parses it. New peers are added to its peer list, while configuration is overwritten if it is of a newer version.

To check for updates, the malware periodically connects to its saved peers.

Godmode

This is probably the most unique feature in the malware: It has an administrative panel built directly into the malware's binary. To launch it, we need to pass the malware the string godmode as the first command-line argument (followed by a peer list).

Access checking

Even though the admin panel is baked into the malware, it can't be accessed by just anyone. To prevent unwanted access to the panel, the malware first requests a private key, and only after validation can we access its interface.

We didn't have the needed private key, so instead we patched the program to skip key validation and accept any supplied private key (it only required a single JZ to JMP modification).

Admin panel: stats

After supplying the private key and "logging in," we are greeted with a status screen about the current configuration.

The first section is peer stats, which are contacted before godmode is launched, based on the peer list that is passed on the malware's command line (we didn't supply a peer list while analyzing it, hence the zero).

The second section is the cryptomining configuration. It is in the same format as the mining configuration that is sent between peers, but with Japanese text instead of English. The language difference is likely because of ease of programming — printing Japanese text is simple, but parsing it is harder, so the malware creator used English in the configuration that is sent between peers.

Finally, we are given a menu with the following options:

1. Refresh the status screen
2. Print the active peer list
3. Update miner settings
4. Exit

Fileless miner

The malware deploys two miners — **xmrig** and **nbhash**. Both miner binaries come base64-encoded inside the malware binary itself and are extracted and executed during runtime. There is some novelty to the execution, however, as **the miners aren't extracted to the disk at all**. Instead, the malware uses the UNIX function *memfd_create* to create a memory-mapped file with the miner binary content, so it can be executed directly from memory without having a traceable filesystem path. From the configuration we extracted from various botnet peers, it seems that the malware uses NiceHash for its mining pools and wallets. NiceHash wallets are not blockchain wallets, so **we can't see transaction and mining details on them to gauge actual revenue.**

Anti-kill

The malware catches Linux termination signals (specifically SIGTERM — 0xF and SIGINT — 0x2) that are sent to it, and ignores them. This makes it harder to terminate the malware, but not impossible, since SIGKILL isn't handled (because it isn't possible, according to the POSIX standard, page 313).

Anti-monitor

This module is internally called *antitaskmanager*, but contrary to its name, it doesn't interfere with task manager operation. Instead, the malware continuously looks for the processes *top* and *htop*. Upon finding them, it terminates the miner processes that are currently running.

Persistence

The malware copies itself to */bin/systemd-worker* and creates a systemd service with the same name. This is probably done to mimic legitimate systemd services to reduce suspicion and avoid investigation.

Attribution

There is an additional screen that is presented in the godmode panel, while validating the private key.

The copyright claim is quite interesting — it both mentions Panchan and has an actual Discord server! Following the link, it seems we can actually join that server, and we also get the actual Discord username that Panchan uses. We assumed this server is the same one that the malware reports to after a successful SSH connection.

We joined the server, hoping to find information about the threat actor, and see the Discord notifications that the malware sends as part of the infection flow. We found none of it — the main chat was empty, except for a greeting from another member that occurred in March. It could be that other chats are only available to higher privileged members of the server, and that's why we didn't see them. The only useful information we found was that the server was created at the beginning of March 2022, very close to our first observation of the malware.

Looking up other activity from that user, we could also find him active in Privex's (a VPS provider) Discord server.

Besides regular VPS, Privex also offers VMs that are pre-installed with blockchain node software. This might mean that the threat actor is using them to host their own server, or perhaps is actively targeting their VMs for cryptojacking.

Detection and mitigation

To assist with detection, we have created a repository with IOCs and Yara and Snort signatures that can be used to test for infection. We have also developed a bash script that can be run on a VM. It looks for the following Panchan indicators:

- The process systemd-worker
- The process xinetd, if it runs from a different path than /bin or /sbin
- Processes listening on TCP port 1919

In addition, outgoing communication over TCP ports 3380 and 3387 might indicate traffic to the cryptomining pool.

For readers who want to proactively defend their networks, we have the following recommendations:

- Use secure and complex passwords. The malware uses a very limited number of default username and password combinations that shouldn't be configured on any production machine. Creating strong passwords can greatly reduce the impact of the malware.
- Configure MFA where possible. Using an MFA would prevent any unauthorized login attempt. Akamai MFA can help protect against SSH key harvesting, as well.
- Segment your network where possible. Although it's legitimate to have machines open to the internet over SSH, it is wise to control who is allowed to connect to them from the internet, as well as who they are allowed to connect with inside the network. Configuring those access controls reduces the impact a breached machine can have on the network, as well as reduces the overall attack surface.
- Monitor your VMs' resource activity. Botnets such as this, whose end goal is cryptojacking, can raise machine resource usage to abnormal levels. Constant monitoring can alert you to suspicious activity. In the case of Panchan, resource usage monitoring would have also terminated the cryptomining entirely.

Appendix A: Short dive into Go reversing

Go executables are compiled statically, which means that all the executable's dependencies are compiled directly into the binary. This creates massive binaries with a lot of functions (for reference, our malware was 30 MB with approximately 3,700 functions).

To assist with stack traces, Go has a pclntab structure that matches function names and pointers inside the binary. This exists even in stripped binaries, so we can use it to find function names.

In Go 1.18, this structure changed. Where before it held pointers to locations in the binary, now it holds instead offsets from specific locations — function pointers are now offsets from the first Go function (which is pointed in the pcln structure commented as text_start in the picture above), name pointers are offsets from the start of the function-name array, and both are referenced by offsets into a different array that holds function data.

Written by

Akamai Security Research