IT asset discovery and inventory

Dashzeveg Baatartsogt
Senior information security analyst at Unitel LLC

Agenda

IT assets
Service discovery by scan
Service discovery use minions
OS discovery

Beginning

For all types of admins (network, security, system etc) IT asset registration, service & IP discovery are important concepts.

vm2 10.[redacted] vm2-pc	62b91d95afd4800e210b8788	system unit	vm2	10.[redacted]	vm2-pc	[redacted]@unitel.mn	untrust	low	system unit
vm3-1 10.[redacted] vm3-pc	62b91d95afd4800e210b8789	system unit	vm3-1	10.[redacted]	vm3-pc	[redacted].r@unitel.mn	untrust	low	system unit
vm4 10.[redacted] vm1-pc	62b91d95afd4800e210b878a	system unit	vm4	10.[redacted]	vm1-pc	[redacted].r@unitel.mn	untrust	low	system unit

Benefit of IT asset management

- Improved performance
- Increased security
- Reduced costs
- Asset visibility

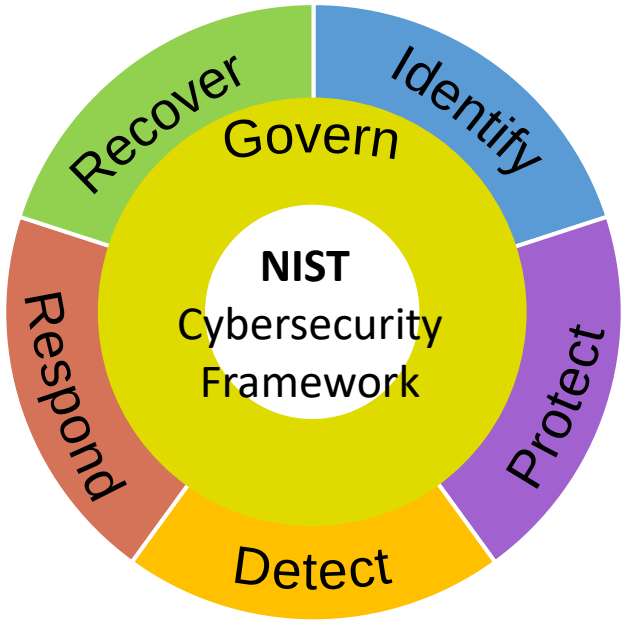
Another aspect of IT asset management is the need for accurate information. Therefore, when services change, IT assets are replaced, **it is necessary to update the Asset register**.

Best Practices: Our Approach in Focus

For identify and IT registration, we have several approaches that we can follow. And these approaches also require constant improvements and info updates from IT engineers.

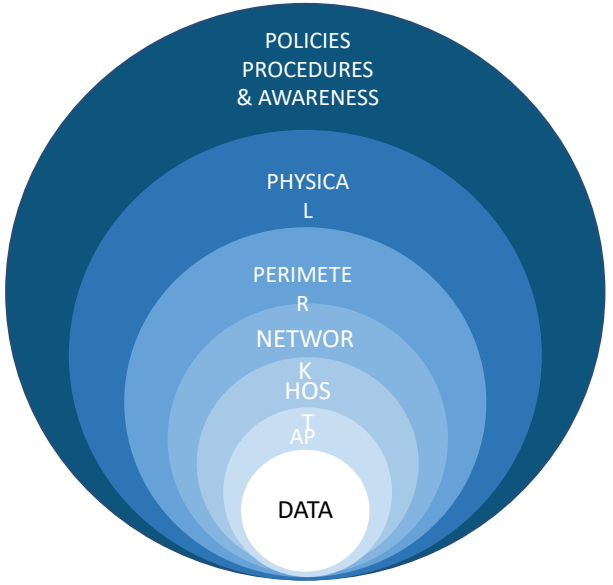
Management

The best-practice approach helps organizations manage their information security by addressing people, processes and technology



Strategy and Operation

The strategy roadmap, objectives, and intended outcomes have been framed using NIST* Cybersecurity Framework



Security Control Layers

All the infrastructure layers we work on are best represented in Defense-in-Depth approach

Controller

ID.AM - Assets (e.g., data, hardware, software, systems, facilities, services, people) that enable the organization to achieve business purposes are identified and managed consistent with their relative importance to organizational objectives and the organization's risk strategy

ID.AM-01 - Inventories of hardware managed by the organization are maintained

ID.AM-02 - Inventories of software, services, and systems managed by the organization are maintained

Requirement

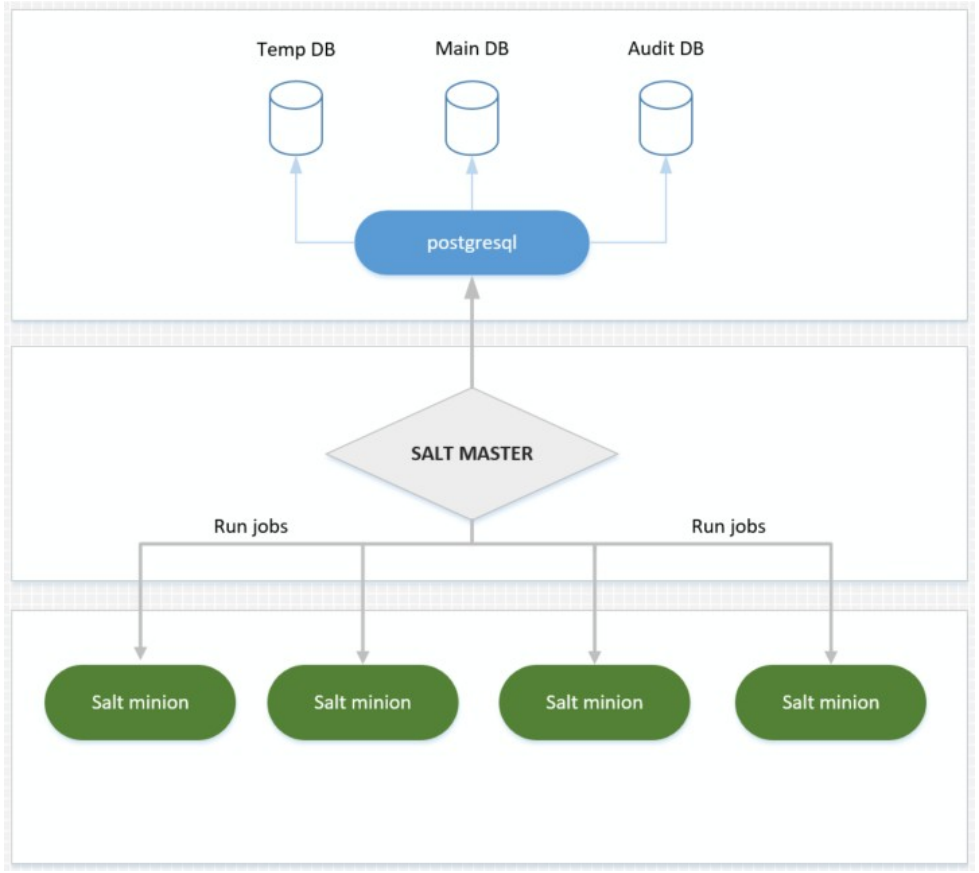
But how do you know changes if you have thousands of IT assets?
How to keep accurate IT asset registration info?

Depending on how many addresses and assets you need to manage, you may need to automate this. We tried some automation related IT asset registration use **SALT, PYTHON, POSTGRESQL.**

Our main principle is to detect changes and new asset arrivals based on port and service changes.

Service & IP discovery

Judging from the approaches, if our service discovery provides as much information as possible, it is good and it may be correct to analyze and register it in the asset.



We are doing host and service detection on about 2000 servers using NMAP. So, we used minions on each network pool in order to accelerate the job's time.

Service & IP discovery

For example, we have the following minions, and we created a user with the same name on the minions and saved the following file named pool.txt.

```
ubuntu@ubuntu:~$ sudo salt '*' test.ping
minion_10.21.60.163:
    True
minion_10.133.3.107:
    True
10.21.60.38_minion:
    True
```

pool.txt

```
[redacted]:/home/minion$ cat pool.txt
10.133.3.0/24
[redacted]:/home/minion$
```

Service & IP discovery

After that, will command the minions at the same time use below code, and for concurrent jobs use below package.

from concurrent.futures import ThreadPoolExecutor

All minions result will write in SALT master to one text named scanoutput.txt.

```python
import subprocess
from concurrent.futures import ThreadPoolExecutor

def run_command(command):
    process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    stdout, stderr = process.communicate()
    return stdout.decode(), stderr.decode()

commands = [
    "sudo salt '10.21.60.38_minion' cmd.run 'nmap -sV -sS -O -Pn -iL /home/minion/pool.txt'",
    "sudo salt 'minion_10.133.3.107' cmd.run 'nmap -sV -sS -O -Pn -iL /home/minion/pool.txt'",
]

with open("scanoutput.txt", "w") as output_file:
    with ThreadPoolExecutor(max_workers=len(commands)) as executor:
        results = executor.map(run_command, commands)

        for command, (stdout, stderr) in zip(commands, results):
            output_file.write(f"Command: {command}\n")
            output_file.write("Stdout:\n")
            output_file.write(stdout)
            output_file.write("\n")
            output_file.write("Stderr:\n")
            output_file.write(stderr)
            output_file.write("\n")
            output_file.write("="*80 + "\n")
```

Service & IP discovery

The result looking like that and included ip, service, some service servers etc.

```
Nmap scan report for 10.133.3.247
Host is up (0.00032s latency).
All 1000 scanned ports on 10.133.3.247 are filtered
MAC Address: 00:50:56:B5:01:11 (VMware)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for 10.133.3.249
Host is up (0.00092s latency).
All 1000 scanned ports on 10.133.3.249 are filtered
MAC Address: 00:50:56:B5:67:2C (VMware)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for hatuu (10.133.3.107)
Host is up (0.000025s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Network Distance: 0 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (66 hosts up) scanned in 472.60 seconds
```

-rw-rw-r-- 1 1000 442926 Jun 14 07:58 scanoutput.txt

Service & IP discovery

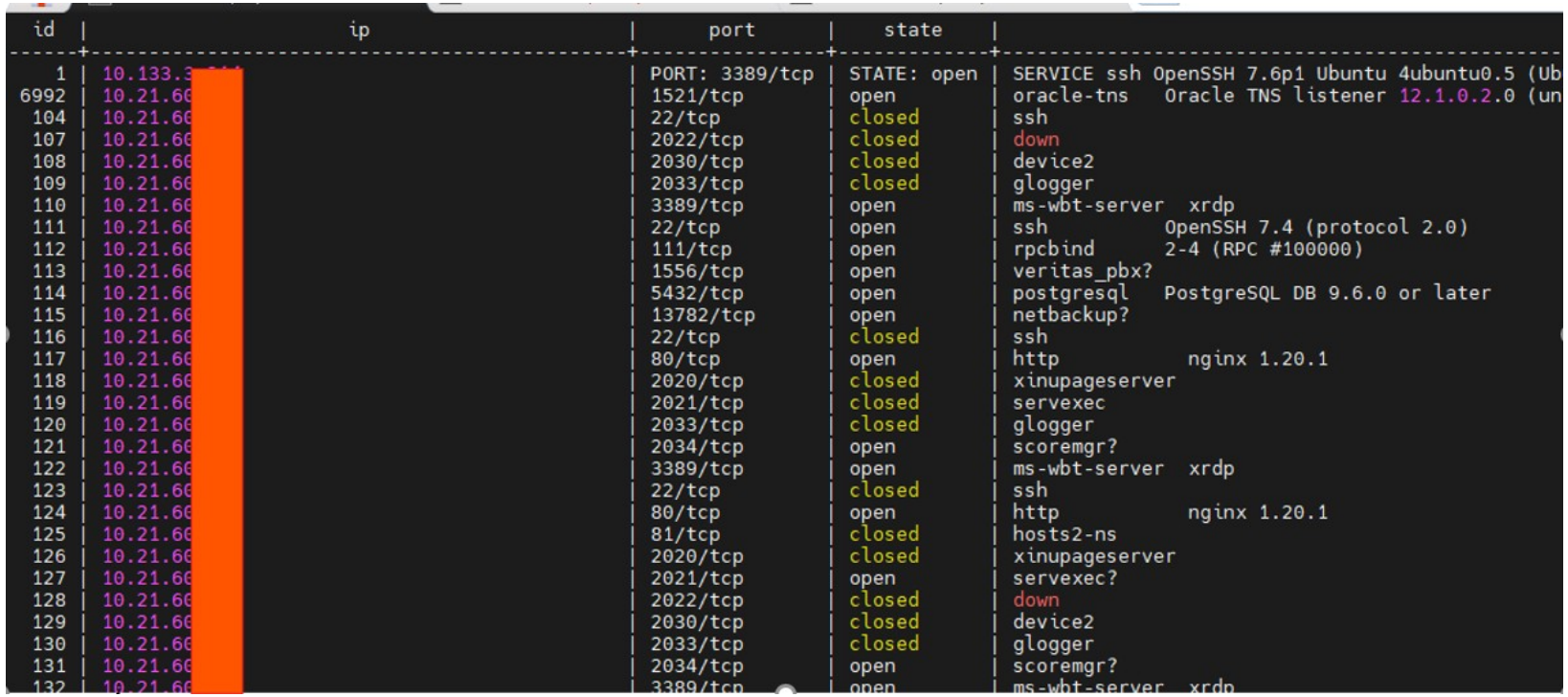
After getting the following results, we get the information we need with the code below. We get only port, states and services.

```python
import re
import psycopg2

def parse_scan_report(filename):
    ip_pattern = re.compile(r"Nmap scan report for (.+)")
    port_pattern = re.compile(r"(\d+/tcp|udp)\s+(\w+)\s+(.+)")

    data = []

    with open(filename, 'r') as file:
        current_ip = None
        current_ports = []

        for line in file:
            line = line.strip()
            ip_match = ip_pattern.match(line)
            port_match = port_pattern.match(line)

            if ip_match:
                if current_ip:
                    data.append((current_ip, current_ports))

                current_ip = ip_match.group(1)
                current_ports = []

            elif port_match:
                port = port_match.group(1)
                state = port_match.group(2)
                service = port_match.group(3)

                current_ports.append({'PORT': port, 'STATE': state, 'SERVICE': service})

        if current_ip:
            data.append((current_ip, current_ports))

    return data
```

Service & IP discovery

The results are first we written to the main DB using the code below and main db looks like this.

```
for ip, ports in data:
    if ports:
        for port_info in ports:
            port = port_info['PORT']
            state = port_info['STATE']
            service = port_info['SERVICE']
            cursor.execute(
                """
                INSERT INTO services (
ip, port, state, service)
                VALUES (%s, %s, %s, %s);
                """,
                (ip, port, state, service)
            )
```



id	ip	port	state	
1	10.133.[redacted]	PORT: 3389/tcp	STATE: open	SERVICE ssh OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (Ub
6992	10.21.60[redacted]	1521/tcp	open	oracle-tns Oracle TNS listener 12.1.0.2.0 (un
104	10.21.60[redacted]	22/tcp	closed	ssh
107	10.21.60[redacted]	2022/tcp	closed	down
108	10.21.60[redacted]	2030/tcp	closed	device2
109	10.21.60[redacted]	2033/tcp	closed	glogger
110	10.21.60[redacted]	3389/tcp	open	ms-wbt-server xrdp
111	10.21.60[redacted]	22/tcp	open	ssh OpenSSH 7.4 (protocol 2.0)
112	10.21.60[redacted]	111/tcp	open	rpcbind 2-4 (RPC #100000)
113	10.21.60[redacted]	1556/tcp	open	veritas_pbx?
114	10.21.60[redacted]	5432/tcp	open	postgresql PostgreSQL DB 9.6.0 or later
115	10.21.60[redacted]	13782/tcp	open	netbackup?
116	10.21.60[redacted]	22/tcp	closed	ssh
117	10.21.60[redacted]	80/tcp	open	http nginx 1.20.1
118	10.21.60[redacted]	2020/tcp	closed	xinupageserver
119	10.21.60[redacted]	2021/tcp	closed	servexec
120	10.21.60[redacted]	2033/tcp	closed	glogger
121	10.21.60[redacted]	2034/tcp	open	scoremgr?
122	10.21.60[redacted]	3389/tcp	open	ms-wbt-server xrdp
123	10.21.60[redacted]	22/tcp	closed	ssh
124	10.21.60[redacted]	80/tcp	open	http nginx 1.20.1
125	10.21.60[redacted]	81/tcp	closed	hosts2-ns
126	10.21.60[redacted]	2020/tcp	closed	xinupageserver
127	10.21.60[redacted]	2021/tcp	open	servexec?
128	10.21.60[redacted]	2022/tcp	closed	down
129	10.21.60[redacted]	2030/tcp	closed	device2
130	10.21.60[redacted]	2033/tcp	closed	glogger
131	10.21.60[redacted]	2034/tcp	open	scoremgr?
132	10.21.60[redacted]	3389/tcp	open	ms-wbt-server xrdp

Service & IP discovery

From now on, we will only record changes, and only changed data such as service, ip, etc. will be added to the main DB.

We are running a 1 week scheduled scan, and if there are no changes, we will keep the previous base DB. If there is a change, we will know about it and add the change to the main asset.

For this we are using temp DB and temp DB will store the new result and compare it with main DB.

```
def insert_into_temp_table(cursor, data):
    for ip, ports in data:
        if ports:
            for port_info in ports:
                port = port_info['PORT']
                state = port_info['STATE']
                service = port_info['SERVICE']
                cursor.execute(
                    """
                    INSERT INTO temp_services (ip, port, state, service)
                    VALUES (%s, %s, %s, %s);
                    """,
                    (ip, port, state, service)
                )
        else:
            cursor.execute(
                """
                INSERT INTO temp_services (ip, port, state, service)
                VALUES (%s, %s, %s, %s);
                """,
                (ip, 'N/A', 'N/A', 'Currently have not service')
            )
```

Service & IP discovery

If there is a change, we will update it to the main DB, and if there is a new IP, we will insert it.

```
# Insert/update main table if with new data
cursor.execute(
    """
    SELECT ip, port, state, service
    FROM temp_services;
    """
)
new_rows = cursor.fetchall()
for ip, port, state, service in new_rows:
    cursor.execute(
        """
        SELECT ip, port, state, service
        FROM services
        WHERE ip = %s AND port = %s;
        """,
        (ip, port)
    )
    existing_row = cursor.fetchone()
    if existing_row is None or existing_row != (ip, port, state, service):
        if existing_row is not None:
            operation = 'U'
        else:
            operation = 'I'
        if existing_row is None or existing_row[3] != 'Currently have not service':  # Check if the old data was not 'Currently have not service'
            cursor.execute(
                """
                INSERT INTO services (ip, port, state, service)
                VALUES (%s, %s, %s, %s)
                ON CONFLICT (ip, port) DO UPDATE
                SET state = EXCLUDED.state,
                    service = EXCLUDED.service;
                """,
                (ip, port, state, service)
            )
```

Service & IP discovery

And we have an audit DB, and only according to the changes, we can write I and U queries to the audit DB and see the changes as an alert.

```
    if operation == 'I':  # Only insert into audit table for actual insert operations
        cursor.execute(
            """
            INSERT INTO services_audit (ip, port, state, service, operation)
            VALUES (%s, %s, %s, %s, 'I');
            """,
            (ip, port, state, service)
        )
    elif operation == 'U':  # Always insert into audit table for update operations
        cursor.execute(
            """
            INSERT INTO services_audit (ip, port, state, service, operation)
            VALUES (%s, %s, %s, %s, 'U');
            """,
            (ip, port, state, service)
        )
```

GOAL

We can see only changes from auditDB, and it means that we can simplify our work by relying only on changes when renewal assets.

```
asset=#
asset=#
asset=# SELECT * FROM services_audit
WHERE audit_timestamp >= NOW() - INTERVAL '5 minute';
     4929 | 2024-06-14 07:58:51.124839 | 10.133.[redacted] | 5988/tcp | closed | wbem-http                                                | I
     4930 | 2024-06-14 07:58:51.124839 | 10.133.[redacted] | 902/tcp  | open   | ssl/vmware-auth VMware Authentication Daemon 1.10 (Uses VNC, SOAP) | I

asset=#
asset=#
```

Asset registration is a task that should be regularly reviewed.

If your organization has thousands of assets, you can use this automation to follow up based on changes.

Hopefully this will help you avoid having to recheck assets that haven't changed at all.

Service discovery from minions

If you install minion-service at all assets, you can track service from master. But we need to exclude the default linux service, for example, we excluded it as follows.

```
ubuntu@ubuntu:~/SALT$ sudo salt-cp '*' default_service.txt /home/minion/default_service.txt
10.21.60.38_minion:
    ----------
    /home/minion/default_service.txt:
        True
minion_10.133.3.107:
    ----------
    /home/minion/default_service.txt:
        True
minion_10.21.60.163:
    ----------
    /home/minion/default_service.txt:
        True
ubuntu@ubuntu:~/SALT$
```


And should run this command:

sudo salt '*' cmd.run 'systemctl list-units --type=service --state=running | grep -v -F -f /home/minion/default_service.txt'

Service discovery from minions

We took the services of the minions with that code and wrote it to the DB. After that also use AuditDB as above and you can track service changes.

```
cmd = "sudo salt '*' cmd.run 'systemctl list-units --type=service --state=running | grep -v -F -f /home/minion/default_service.txt'"

def parse_salt_output(output):
    minion_data = {}
    current_minion = None

    for line in output.splitlines():
        if re.match(r'^minion_', line) or re.match(r'^\d+\.\d+\.\d+\.\d+', line):
            current_minion = line.strip(':')
            minion_data[current_minion] = []
        elif "LOAD   = Reflects" in line or "ACTIVE = The high-level" in line or "SUB    = The low-level" in line or "loaded units listed" in line:
            continue
        elif current_minion and line.strip() and not line.startswith('LOAD') and not line.startswith('UNIT'):
            # Parse the service data
            parts = line.split(maxsplit=4)
            if len(parts) == 5:
                minion_data[current_minion].append(parts)

    return minion_data

# Execute the Salt command
result = subprocess.run(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.P

if result.returncode == 0:
    output = result.stdout
    minion_data = parse_salt_output(output)

    for minion_id, services in minion_data.items():
        for service in services:
            cur.execute('''
                INSERT INTO service_status (minion_id, unit, load, active, sub, desc
                VALUES (%s, %s, %s, %s, %s, %s);
            ''', (minion_id, service[0], service[1], service[2], service[3], service

    conn.commit()
    print("Data inserted successfully.")
else:
    print("Error running Salt command:", result.stderr)

# Close the database connection
cur.close()
conn.close()
```

id	minion_id	unit	load	active	sub	description
1	minion_10.133.3.107	UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
2	minion_10.133.3.107	accounts-daemon.service	loaded	active	running	Accounts Service
3	minion_10.133.3.107	acpid.service	loaded	active	running	ACPI event daemon
4	minion_10.133.3.107	avahi-daemon.service	loaded	active	running	Avahi mDNS/DNS-SD Stack
5	minion_10.133.3.107	cbram.service	loaded	active	running	Cybereason av protection
6	minion_10.133.3.107	colord.service	loaded	active	running	Manage, Install and Generate Color Profiles
7	minion_10.133.3.107	cups-browsed.service	loaded	active	running	Make remote CUPS printers available locally
8	minion_10.133.3.107	cups.service	loaded	active	running	CUPS Scheduler
9	minion_10.133.3.107	cybereason-sensor.service	loaded	active	running	Cybereason sensor end-point protection
10	minion_10.133.3.107	gdm.service	loaded	active	running	GNOME Display Manager
11	minion_10.133.3.107	kerneloops.service	loaded	active	running	Tool to automatically collect and submit kernel crash signatures
12	minion_10.133.3.107	ModemManager.service	loaded	active	running	Modem Manager
13	minion_10.133.3.107	nessusagent.service	loaded	active	running	The Nessus Client Agent
14	minion_10.133.3.107	NetworkManager.service	loaded	active	running	Network Manager
15	minion_10.133.3.107	rtkit-daemon.service	loaded	active	running	RealtimeKit Scheduling Policy Service
16	minion_10.133.3.107	salt-minion.service	loaded	active	running	The Salt Minion
17	minion_10.133.3.107	snapd.service	loaded	active	running	Snap Daemon
18	minion_10.133.3.107	SplunkForwarder.service	loaded	active	running	Systemd service file for Splunk, generated by 'splunk enable boot-start'
19	minion_10.133.3.107	ssh.service	loaded	active	running	OpenBSD Secure Shell server
20	minion_10.133.3.107	switcheroo-control.service	loaded	active	running	Switcheroo Control Proxy service
21	minion_10.133.3.107	sysmon.service	loaded	active	running	Sysmon event logger
22	minion_10.133.3.107	systemd-timesyncd.service	loaded	active	running	Network Time Synchronization
23	minion_10.133.3.107	user@1000.service	loaded	active	running	User Manager for UID 1000
24	minion_10.133.3.107	whoopsie.service	loaded	active	running	crash report submission daemon
25	minion_10.133.3.107	wpa_supplicant.service	loaded	active	running	WPA supplicant
26	minion_10.21.60.163	UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
27	minion_10.21.60.163	apache2.service	loaded	active	running	The Apache HTTP Server
28	minion_10.21.60.163	containerd.service	loaded	active	running	containerd container runtime
29	minion_10.21.60.163	ModemManager.service	loaded	active	running	Modem Manager
30	minion_10.21.60.163	multipathd.service	loaded	active	running	Device-Mapper Multipath Device Controller
31	minion_10.21.60.163	mysql.service	loaded	active	running	MySQL Community Server
32	minion_10.21.60.163	ntp.service	loaded	active	running	Network Time Service
33	minion_10.21.60.163	open-vm-tools.service	loaded	active	running	Service for virtual machines hosted on VMware
34	minion_10.21.60.163	packagekit.service	loaded	active	running	PackageKit Daemon
35	minion_10.21.60.163	postfix@-.service	loaded	active	running	Postfix Mail Transport Agent (instance -)

OS detection

You can detect minions OS and model in several ways.
Example:

- sudo salt '*' grains.item os osrelease
- sudo salt ‘*’ cmd.run ‘cat etc/os-release etc

```
minion_10.21.60.163:
    PRETTY_NAME="Ubuntu 22.04.4 LTS"
    NAME="Ubuntu"
    VERSION_ID="22.04"
    VERSION="22.04.4 LTS (Jammy Jellyfish)"
    VERSION_CODENAME=jammy
    ID=ubuntu
    ID_LIKE=debian
    HOME_URL="https://www.ubuntu.com/"
    SUPPORT_URL="https://help.ubuntu.com/"
    BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
    PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
    UBUNTU_CODENAME=jammy
10.21.60.38_minion:
    PRETTY_NAME="Ubuntu 22.04.4 LTS"
    NAME="Ubuntu"
    VERSION_ID="22.04"
    VERSION="22.04.4 LTS (Jammy Jellyfish)"
    VERSION_CODENAME=jammy
    ID=ubuntu
    ID_LIKE=debian
    HOME_URL="https://www.ubuntu.com/"
    SUPPORT_URL="https://help.ubuntu.com/"
    BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
    PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
    UBUNTU_CODENAME=jammy
ubuntu@ubuntu:~/SALT$ sudo salt '*' grains.item os osrelease
```

OS detection

You can also use it in a number of ways, such as adding your OS version to your asset registry or checking for vulnerabilities in that OS version.

In our case we want to check our OS vulnerabilities and for this we used exploitdb's api.

```
git clone https://github.com/offensive-security/exploitdb.git /opt/exploitdb
ln -sf /opt/exploitdb/searchsploit /usr/local/bin/searchsploit
```

OS detection

We can check available exploits by OS use this code.

```python
import subprocess
import re

salt_command = "sudo salt '*' cmd.run 'cat /etc/os-release'"

result = subprocess.run(salt_command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)

# Extract stdout from the result
output = result.stdout

minion_pattern = r'(minion_[\d.]+|\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}_minion):'
version_id_pattern = r'VERSION_ID="([^"]+)"'
id_pattern = r'ID=([^\s]+)'

minion_matches = re.findall(minion_pattern, output)
version_id_matches = re.findall(version_id_pattern, output)
id_matches = re.findall(id_pattern, output)

for i, minion in enumerate(minion_matches):
    version_id = version_id_matches[i] if i < len(version_id_matches) else "Unknown Version"
    os_id = id_matches[i] if i < len(id_matches) else "Unknown OS"

    def search_exploits(os_name, os_version):
        search_term = f"{os_name} {os_version}"
        result = subprocess.run(['searchsploit', search_term], stdout=subprocess.PIPE, text=True)
        if result.returncode == 0:
            print(minion, result.stdout)
        else:
            print("Error searching exploits")

    search_exploits(os_id, version_id)
```

OS detection

Result is looking like below

```
minion_10.133.3.107 Exploits: No Results
Shellcodes: No Results

minion_10.21.60.163 Exploits: No Results
Shellcodes: No Results

10.21.60.38_minion Exploits: No Results
Shellcodes: No Results
```

If your version is exploitable, it will look like this.

```
ubuntu@ubuntu:~/SALT$ searchsploit centos 7
------------------------------------------------------------------------------------------ ----------------------------
 Exploit Title                                                                            |  Path
------------------------------------------------------------------------------------------ ----------------------------
abrt (Centos 7.1 / Fedora 22) - Local Privilege Escalation                                | multiple/local/38835.py
CentOS 7.6 - 'ptrace_scope' Privilege Escalation                                          | linux/local/46989.sh
CentOS Control Web Panel 0.9.8.836 - Authentication Bypass                                | linux/webapps/47123.txt
CentOS Control Web Panel 0.9.8.836 - Privilege Escalation                                 | linux/webapps/47124.txt
CentOS Control Web Panel 0.9.8.838 - User Enumeration                                     | linux/webapps/47125.txt
CentOS Web Panel 0.9.8.740 - Cross-Site Request Forgery / Cross-Site Scripting            | php/webapps/45822.txt
CentOS Web Panel 0.9.8.740 - Cross-Site Request Forgery / Cross-Site Scripting            | php/webapps/45822.txt
CentOS Web Panel 0.9.8.763 - Persistent Cross-Site Scripting                              | linux/webapps/46349.txt
CentOS Web Panel 0.9.8.789 - NameServer Field Persistent Cross-Site Scripting             | linux/webapps/46629.txt
CentOS Web Panel 0.9.8.793 (Free) / 0.9.8.753 (Pro) - Cross-Site Scripting                | linux/webapps/46669.txt
CentOS Web Panel 0.9.8.793 (Free) / v0.9.8.753 (Pro) / 0.9.8.807 (Pro) - Domain Field (Add | linux/webapps/46784.txt
Centos Web Panel 7 v0.9.8.1147 - Unauthenticated Remote Code Execution (RCE)              | linux/webapps/51194.txt
Centos WebPanel 7 - 'term' SQL Injection                                                  | linux/webapps/48212.txt
Linux Kernel (Debian 7.7/8.5/9.0 / Ubuntu 14.04.2/16.04.2/17.04 / Fedora 22/25 / CentOS 7.3 | linux_x86-64/local/42275.c
Linux Kernel (Debian 7/8/9/10 / Fedora 23/24/25 / CentOS 5.3/5.11/6.0/6.8/7.2.1511) - 'ldso | linux_x86/local/42274.c
Linux Kernel 2.4/2.6 (RedHat Linux 9 / Fedora Core 4 < 11 / Whitebox 4 / CentOS 4) - 'sock_ | linux/local/9479.c
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'aiptek' Nullpointer Dereference               | linux/dos/39544.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'cdc_acm' Nullpointer Dereference              | linux/dos/39543.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'cypress_m8' Nullpointer Dereference           | linux/dos/39542.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'digi_acceleport' Nullpointer Dereference      | linux/dos/39537.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'mct_u232' Nullpointer Dereference            | linux/dos/39541.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - 'Wacom' Multiple Nullpointer Dereferences     | linux/dos/39538.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - visor 'treo_attach' Nullpointer Dereference   | linux/dos/39539.txt
Linux Kernel 3.10.0 (CentOS / RHEL 7.1) - visor clie_5_attach Nullpointer Dereference   | linux/dos/39540.txt
Linux Kernel 3.10.0 (CentOS 7) - Denial of Service                                        | linux/dos/41350.c
Linux Kernel 3.10.0-229.x (CentOS / RHEL 7.1) - 'iowarrior' Driver Crash (PoC)           | linux/dos/39556.txt
Linux Kernel 3.10.0-229.x (CentOS / RHEL 7.1) - 'snd-usb-audio' Crash (PoC)              | linux/dos/39555.txt
Linux Kernel 3.10.0-514.21.2.el7.x86_64 / 3.10.0-514.26.1.el7.x86_64 (CentOS 7) - SUID Posi | linux/local/42887.c
Linux Kernel 3.14.5 (CentOS 7 / RHEL) - 'libfutex' Local Privilege Escalation             | linux/local/35370.c
Linux Kernel 4.14.7 (Ubuntu 16.04 / CentOS 7) - (KASLR & SMEP Bypass) Arbitrary File Read | linux/local/45175.c
Pure-FTPd 1.0.21 (CentOS 6.2 / Ubuntu 8.04) - Null Pointer Dereference Crash (PoC)        | linux/dos/20479.pl
------------------------------------------------------------------------------------------ ----------------------------
Shellcodes: No Results
```

Code

- https://github.com/Da5hka/inventory