Forensic Analysis of Cryptojacking in Host-based Docker Containers Using Honeypots

Javier Franco, Abbas Acar, Ahmet Aris, and Selcuk Uluagac
Cyber-Physical Systems Security Lab., Florida International University, Florida, USA
{jfran243, aacar001, aaris, suluagac}@fiu.edu

ICC 2023 - IEEE International Conference on Communications | 978-1-5386-7462-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICC45041.2023.10278764

Abstract—**Blockchain-based cryptocurrencies have transformed financial transactions and created opportunities to profit from generating new coins through cryptomining. This has led to cybercriminals stealthily using their victim's computational power and resources for their own profit. Recent trends point to an increase in cryptojacking malware targeting devices with greater processing power such as host-based docker engines for faster and greater profit. In our study, we perform a forensic analysis for detecting cryptojacking (i.e., unauthorized cryptomining) in Docker containers using honeypots. Then, we present countermeasures for securing host-based Docker containers. In addition, we propose an approach for monitoring host-based Docker containers for cryptojacking detection. To the best of our knowledge, this is the first study investigating cryptojacking detection with the use of a honeypot system. Our results reveal that host resource usage and network traffic are the key indicators of possible unauthorized cryptomining in Docker containers.**

Index Terms—**Cryptojacking, Honeypot, Network Security, Docker**

I. Introduction

Blockchain-based cryptocurrencies have transformed financial transactions. There are currently over 18,000 types of cryptocurrencies valued more than 2 trillion as of April 2022 [1]. The lucrative potential of cryptomining has led to the exploitation of this methodology through cryptojacking, by which a cybercriminal performs unauthorized cryptocurrency mining using the victim's computational power and resources for their own financial gain.

While cybercriminals perform cryptojacking attacks targeting a wide range of devices (e.g., personal computers, IoT devices [2]), the researchers spotted an increasing trend towards targeting devices with greater processing power through host-based cryptojacking [3]. For host-based cryptojacking, since the client does not go to the attacker as they do in the case of in-browser cryptojacking, the attackers need to find a way to deploy and install the malicious mining script on the victim's device. The greater the computational power, the greater and faster is the possible profit yield from mining. This usually also means a greater number of connections and processes expanding the attack surface [3]. Some examples of targeted powerful devices are servers [4], enterprise cloud infrastructures [5], [6], and Docker engines [7] to maximize profit using host-based cryptojacking.

Docker engines are widely deployed in today's enterprise computing settings from universities to big companies worldwide providing services to their employees, making host-based Docker containers a prime target for cryptojacking. In fact, Docker has become one of the top three most popular development platforms [8]. Despite the trend toward host-based cryptojacking and the wide use of Docker, the current literature has not explored the cryptojacking malware targeting Docker containers and its detection methods.

At the same time, honeypots and honeynets have become very important tools for security researchers in the past years to log and analyze how attackers perform attacks and discover their behaviors [9]. Honeypots and honeynets can provide actionable intelligence on the attackers to continuously learn from attacks and develop effective defense mechanisms [9]. A honeypot is a decoy that is used to lure attackers and deceive them into thinking they have accessed a real system [10]. and to observe and learn from their actions by gathering data about their interaction with the honeypot. Research honeypots have been a very active field of research during the last decade. However, there have been few research studies on the use of honeypot data for developing security solutions for production purposes [10]. To the best of our knowledge, no research studies have considered the use of production honeypots for the detection and mitigation of cryptojacking.

In this study, we conduct a forensic analysis of host-based cryptojacking malware in Docker containers. We utilized honeypots to collect host resources and network data. With our experiments, we found that monitoring the resource usage of a Docker host can reveal potentially unauthorized cryptomining. For example, the increased temperature of a Docker host and increased CPU and RAM loads can be attributed to a particular container and it can be a strong indicator of possible cryptomining by an intrusion or an insider. We found that further network traffic data can be used to further assure of a cryptojacking malware attack. Notable indicators to alert of cryptojacking in the network data include the identification of Stratum protocol, keywords in DNS requests, and the use of the container's ephemeral ports. We also identified countermeasures to secure Docker containers to avoid such attacks. Finally, we proposed an approach for monitoring host-based Docker containers and alerting system administrators when the indicators point to a cryptojacking attack.

Contributions: The contributions are as follows:

- We empirically identified key indicators for unauthorized cryptomining detection in a host-based Docker engine using a honeypot system. Our experiments show that both host resource usage and network traffic-based features can be used to detect unauthorized cryptomining.

- We presented potential countermeasures for securing Docker containers to prevent cryptojacking attacks.
- We proposed an approach for monitoring host-based Docker containers and alerting system administrators when the indicators point to a cryptojacking attack.

Organization: This paper is organized as follows: Section II provides the related work. Section III provides background information. Section IV describes the problem scope and threat model. Section V describes our methodology. Section VI provides a forensic analysis of the collected data. Section VII provides measures for securing host-based Docker containers. Section VIII presents an approach to monitor host-based Docker containers. Section IX proposes conclusions and future work.

II. Related Work

While there are several recent studies focusing on the detection of in-browser cryptojacking malware [11]–[13], a limited number of studies in the literature focused on detection mechanisms for host-based cryptojacking [14]–[19]. The study in [14] uses CPU as a key feature for the detection while study in [15] utilizes Hardware Performance Counters (HPC). [16], [20] focus on providing a lightweight model for detecting cryptojacking malware in low-power devices such as the Internet of Things (IoT). [17] focuses on a combination of hardware events, software events, and hardware cache events. Two studies in [18], [19] use the opcode sequences, system call invocations, and network traffic for cryptojacking detection. However, none of the aforementioned studies focused on container implementations. The study in [21] specifically targets detecting cryptojacking malware in containers. This study concentrates on monitoring Linux-kernel system calls with a machine learning-based system of anomalous pods in a Kubernetes cluster. Finally, the study in [22] focuses on docker-based honeypot systems; however, they do not focus on cryptojacking threats. To the best of our knowledge, this is the first study investigating host-based cryptojacking targeting Docker containers focusing on both device resource usage and network traffic features. Moreover, it is also the first study utilizing a honeypot system for cryptojacking detection.

III. Background

A. Host-based Cryptojacking

Cryptojacking refers to the act of performing unauthorized cryptocurrency mining using a victim's computational power and resources for financial gain. While in-browser cryptojacking is carried out through a web script, host-based crytojacking is carried out on the host system. To accomplish this, cybercriminals fraudulently embed malware in legitimate third-party applications; target identified Common Vulnerabilities and Exposures (CVEs) as in the case of Mikrotik routers; exploit poor security and install the malicious files through a web page or pop-up window [3]. Insiders can also seek to make a profit mining using the resources of the company abusing their privileges [3].

B. Stratum Protocol

Stratum is an application layer protocol created for pooled mining, in which miners pool their computing resources together. Each miner fulfills different tasks and reports them to the mining pool server. When the server sends a valid output to the cryptocurrency network, it receives the reward, takes a commission, and distributes the remaining portion according to the tasks each carried out [23].

C. Docker Containers

Docker is an open-source platform for running, developing, and distributing applications. It is lightweight, fast, highly portable, and allows for the dynamic management of workloads. The Docker platform is container-based, with containers holding everything that is needed to run an application. This is particularly useful for workflow, as this allows for sharing containers while reliably ensuring that everyone receives the same container without alteration, in a way that can be easily worked on [24].

D. Honeypots and Honeynets

A honeypot is a decoy that can be used as a first line of defense to lure attackers and deceive them into thinking they have accessed a real system. This allows the owner of the attacked system to observe and learn from attacker actions by gathering data about their interaction with the honeypot. The gathered data can be also used to develop countermeasures against attacks. Honeypots can vary greatly in the level of interaction that they allow the attacker, ranging from low-interaction honeypots that emulate particular services to high-interaction honeypots that emulate entire operating systems. They can be used in a wide variety of applications, for research or production purposes. They can be implemented with physical or virtual resources. Two or more honeypots implemented on a system form a honeynet or a honeypot system [10]. Any traffic through the honeypot is usually malicious, which allows honeypots to greatly reduce the number of false positives in comparison with anomaly-based intrusion detection systems.

IV. Problem Scope and Threat Model

In this section, we present the problem scope as well as the threat model considered for our study.

A. Problem Scope

Cryptojacking is a stealth attack which can go unnoticed for a long time. However, it can produce significant losses for victims, reducing computational efficiency of the host's system and generating operational costs. We consider a host-based Docker engine with diverse Docker containers used for enterprise purposes. Resources can be vulnerable to cryptojacking, whether it is an external cybercriminal which finds its way into the system, or an employee who abuses their access privileges to the computational power of their employer's system and uses it for their personal financial gain at their employer's expense. It is worth noting that developers tend to be targets of attack, as they have high access and even administrative privileges to carry out their work.

B. Threat Model

In this work, we consider cryptojacking attacks targeting host-based docker containers, which are commonly used in enterprise environments. Docker has become one of the top three most popular development platforms [8] and cryptojacking attacks have been identified among the most common types of attacks on Docker containers [7]. Furthermore, there is an increasing trend of cybercriminals targeting devices with greater processing power through host-based cryptojacking [3], in order to produce greater financial gain in a shorter time frame. We consider both outsider and insider threats, as insiders can also seek to make a profit mining using the resources of the company.

V. Methodology

In this section, we describe the honeypot deployment as well as host resources and network data collection processes.

A. Honeypot System Deployment

The host is an Intel Core i7-10700K Processor with 16 Cores and 32 GB RAM. We set up a high-interaction honeypot system isolated in a demilitarized zone (DMZ) with ten Docker containers: four of these used Redis container images, four used Nginx container images, and two used Ubuntu container images. Of those using Ubuntu images, one was used to run Prometheus and one was used to run Cadvisor. One of each of the Redis and Nginx containers was used to understand the baseline behavior and the other three of each were tested with three scripts mining Monero on Cudominer, Minexmr, and Xmrpool.eu mining pools. Redis and Nginx were selected because they have been identified as the two most widely used container images [25]. We selected to use Monero, as it is the most popular coin used for cryptojacking due to its strong anonymity and its efficient mining properties on CPUs [23]. The containers were allowed access to the 16 cores with unlimited swap and memory usage. On the baseline containers, a script that simulates 11,500 requests per second was run on the Nginx container and a Redis-benchmark utility that simulates 1000 user requests per second was run on the Redis container. Figure 1 presents a general overview of the honeypot system.

We used version 3.2 of the docker-compose tool [26] to define and execute the containers in our honeypot system. The containers are run with a simple command docker-compose up, and if needed they all can be down with a single command docker-compose down. Compose is configured in a predefined YAML file, where we input the services that will be running in the docker host, and the configuration they will be using such as image, container name, ports, network, volumes, commands, etc. In the docker-compose command, other files such as Dockerfile and prometheus.yml are invoked to participate in the setup process.

Mining activity was collected for a total of eight hours. The averages of the collected data for each container are presented in Table I. We stopped the mining activities and the data collection process from time to time to cool the host device since the CPU temperature passed the safe threshold of 82°Celsius.

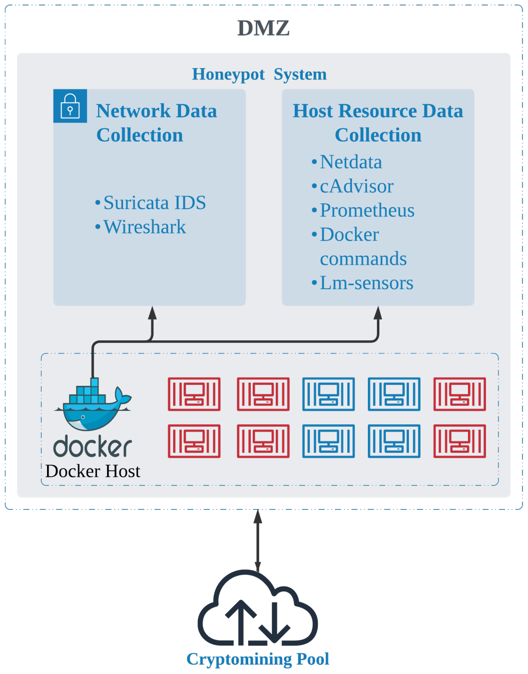


Fig. 1: Honeypot System Overview.

B. Host Resource Data Collection

We run the open-source tool cAdvisor (Container Advisor) [27] in a separate container using docker-compose to collect metrics on the usage and performance of the running containers and the Docker host. We then allow the Prometheus container to dig into the cAdvisor metrics and filter them using the Prometheus [28] expression browser. For each container and for the host, the cAdvisor metrics provide the isolation parameters, CPU usage including usage per Core and top container demand, Memory consumption, network metrics such as throughput and errors, filesystem usage, and the host /system.slice. cAdvisor can be accessed from the docker host through their web UI at http://localhost:8080. Although cAdvisor provides this interface, the Prometheus expression increases the metrics analysis. Prometheus is an open-source tool that scrapes metrics collected by the cAdvisor container. It's part of the docker-compose deployment. Prometheus allows filtering of the metrics by providing expressions such as *container_cpu_usage_seconds_total, container_tasks_state, container_memory_usage_bytes, container_fs_writes_total,* etc, that will scrape cAdvisor metrics approximately every three seconds for a duration of 122ms approximately using the labels *instance="cadvisor:8080" and job="cadvisor"*. Prometheus can be accessed from the docker host through their web interface at http://localhost:9090. Netdata [29] is a web application that provides key metrics, useful charts, and alarms to monitor the system and its components and graphically visualizes the metrics in real time. Lm-sensors [30] was used to monitor the

CPU temperature of the Docker host, and nvme-cli [31] was used to monitor Nvme ssd temperature.

C. Network Data Collection

Wireshark version 3.2.3 [32] open source network protocol analyzer was used to log the network data. Wireshark allows filtering by protocol and identifying cryptomining protocols such as Stratum. It also collects the exchanges between the cryptojacked container and the pools, timestamp, and network information such as IP address, packet length, protocol, and ports. However, it does not allow data parsing. For this purpose, Suricata version 6.0.5 [33] open source intrusion detection and intrusion prevention system (IDPS) was also used to log all network traffic, including timestamp, event type, IP addresses, packet length, source and destination port, DNS information, etc.

VI. Data Analysis

In this section, we analyze the collected data and provide notable findings for the detection of unauthorized cryptomining.

A. Host Resource Data Analysis

The data collected from the baseline container running Redis for 1000 user requests demonstrated an average of 16% CPU consumption, 5% RAM, 7Mb disk I/O, and 10% iowait. We also observed that the temperature on average of the CPU cores and the nvme were 42.25°Celsius and 36.9°Celsius respectively. In our comparison tests running mining scripts, there was a dramatic increase of more than 4 times the average container CPU compared to the baseline, with a peak approximately every 2 minutes where the container CPU consumption increases to more than 5 times the baseline average for a period of approximately one minute. During our observations, the CPU never goes lower than 55% in the cryptojacked containers, which is more than 3 times the average CPU consumption of the baseline. The RAM consumption of the containers also goes up, positioning it at an average of 27.4% which is more than 5 times the baseline average. We did not observe distinctive rate changes in the Disk I/O metrics while mining, keeping an average of 7 Mbs. However, there was an increase in the average overall iowait to 14.66%. The temperature on average of the CPU cores was 70°Celsius, which is a significant increase of 63%, and that of the nvme was 42°Celsius.

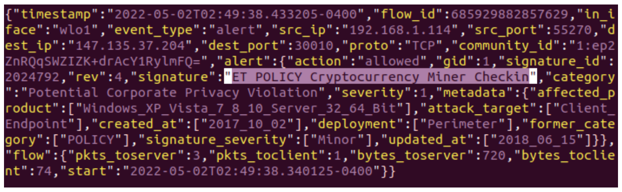
Next, we observe the data for the baseline container running Nginx. Here, the simulated 11,500 requests per second demonstrated an average of 15% CPU consumption, 1% RAM, 6Mb disk I/O, and 10% iowait. The temperature on average of the CPU cores and the nvme were 42°Celsius and 37°Celsius, respectively. During mining activity, there was a dramatic increase of more than 5 times the average CPU in comparison to the baseline, and peaks can be observed approximately every 2 minutes where the CPU increases to more than 6 times the baseline average for approximately one minute. 60% was the lowest CPU consumption measured during mining, which is more than three times the average CPU consumption of the baseline. RAM consumption increased to 30%. This is more than 30 times the baseline average. As with the Redis containers, Disk I/O metrics did not see a significant change. The average overall iowait increased to 14.33%. The temperature on average of the CPU cores saw a significant increase of 66.67% to 70°Celsius, and that of the nvme rose to 41.66°Celsius.

While increased CPU and RAM loads, as well as the increased temperature only provide hints of a possible problem such as hardware failure in a regular system, when this is identified in combination and on a Docker host where increased CPU and RAM load can be identified as coming from a particular container, this is a much clearer indication that the issue is likely an intrusion or insider mining.

B. Network Data Analysis

In all our tests, we first identify the TCP 3-way handshake between the container and the cryptomining pool server. Once communication between the cryptojacked container and the server is established and validated, the data flow begins. The server assigns the container the task to work, and once this task is solved, the client will send the results to the pool server.

We analyzed the captured network traffic from Wireshark [32] and Suricata [33] to find indicators or patterns that could be identified as possible cryptojacking activity. One of these indicators is the use of the line-based Stratum Protocol which uses plain TCP socket and JSON-RPC messages. However, when encrypted with Transport Layer Security (TLS), the packets by which the container receives jobs, sends the solutions, and the pool acknowledges the solution as "Application Data". Notably, Suricata flagged the Stratum protocol as a potential corporate privacy violation, specifically an "ET POLICY cryptocurrency miner checkin". This can be seen in Figure 2.



```
{"timestamp":"2022-05-02T02:49:38.433205-0400","flow_id":685929882857629,"in_i
face":"wlo1","event_type":"alert","src_ip":"192.168.1.114","src_port":55270,"d
est_ip":"147.135.37.204","dest_port":30010,"proto":"TCP","community_id":"1:ep2
ZnRQqSWZIZK+drAcY1RylmFQ=","alert":{"action":"allowed","gid":1,"signature_id":
2024792,"rev":4,"signature":"ET POLICY Cryptocurrency Miner Checkin","category
":"Potential Corporate Privacy Violation","severity":1,"metadata":{"affected_p
roduct":["Windows_XP_Vista_7_8_10_Server_32_64_Bit"],"attack_target":["Client_
Endpoint"],"created_at":["2017_10_02"],"deployment":["Perimeter"],"former_cate
gory":["POLICY"],"signature_severity":["Minor"],"updated_at":["2018_06_15"]}},
"flow":{"pkts_toserver":3,"pkts_toclient":1,"bytes_toserver":720,"bytes_toclie
nt":74,"start":"2022-05-02T02:49:38.340125-0400"}}
```

Fig. 2: Suricata Detection Alert.

Cudominer and Minexmr communications use TCP, while Xmrpool.eu was encrypted using TLS. Certain keywords such as monero, pool, coin, xmr, mine, hash, and cudo were identified in DNS requests under the domain name. However, the hackers could apply obfuscation techniques such as the use of a proxy to go unnoticed.

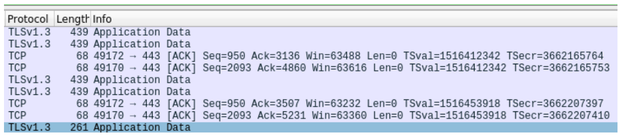
We also identified the following patterns:

- The cryptojacked containers began receiving jobs without having requested them.
- When they are not encrypted, all the packets have the ACK flag that confirms receipt of packets.
- When they are not encrypted, all the packets have the PSH flag which tells the container to process packets even if the buffer is not full.

TABLE I: Host Resource Data Collection

Tests	Crypto	CPU	CPU Peaks	RAM	Disk I/O	iowait	Hash rate	Cryptojacking	CPU Temp.	Nvme Temp
Redis	N/A	16.00%	N/A	5.00%	7Mb	10.00%	N/A	Baseline	42.25°C	36.9°C
Redis – Test 1	Monero	65.00%	75.00%	28.50%	7Mb	13.00%	5.70Kb/s	Script1	69°C	42°C
Redis – Test 2	Monero	74.00%	90.00%	26.30%	7Mb	14.00%	4.1 kh/s	Script2	71°C	42°C
Redis – Test 3	Monero	75.00%	92.00%	27.40%	7Mb	17.00%	5.2 kh/s	Script3	70°C	42°C
Nginx	N/A	15.00%	N/A	1.00%	6Mb	10.00%	N/A	Baseline	42°C	37°C
Nginx – Test 1	Monero	70.00%	85.00%	30.50%	6Mb	12.00%	4.6 kh/s	Script1	69.5°C	41°C
Nginx – Test 2	Monero	82.00%	97.00%	32.00%	6Mb	16.00%	4.8 kh/s	Script2	69°C	40°C
Nginx – Test 3	Monero	75.00%	97.7%	27.50%	6Mb	15.00%	5.2 kh/s	Script3	72°C	41°C

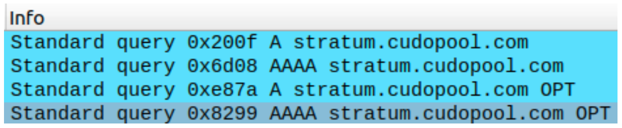
- Mining sessions are lengthy processes. During these mining sessions, ACK and PSH are the only flags that are used. FIN, RST, and SYN flags were only identified near the start or end of communications.
- All container ports used by the pools were ephemeral ports greater than 30,000.
- Between the communications for assigning jobs, sending solutions, and acknowledging receipt, there are frequent 68-byte ACK flags to maintain communication.
- For all three scripts used, similar package lengths were noted for new jobs sent from the pool to the container, solutions sent to the pool, and acknowledging receipt of the solution. For Cudominer, these lengths were consistently 403, 240, or 241, and 132 or 133, respectively. For Minexmr they were 439, 154-262, and 153-162. A screenshot of a portion of the Minxmr packet information can be seen in Figure 3. For Xmrpool.eu they were 439, 261, and 153.



```
Protocol Length Info
TLSv1.3   439 Application Data
TLSv1.3   439 Application Data
TCP        68 49172 → 443 [ACK] Seq=950 Ack=3136 Win=63488 Len=0 TSval=1516412342 TSecr=3662165764
TCP        68 49170 → 443 [ACK] Seq=2093 Ack=4860 Win=63616 Len=0 TSval=1516412342 TSecr=3662165753
TLSv1.3   439 Application Data
TLSv1.3   439 Application Data
TCP        68 49172 → 443 [ACK] Seq=950 Ack=3507 Win=63232 Len=0 TSval=1516453918 TSecr=3662207397
TCP        68 49170 → 443 [ACK] Seq=2093 Ack=5231 Win=63360 Len=0 TSval=1516453918 TSecr=3662207410
TLSv1.3   261 Application Data
```

Fig. 3: Minexmr Packet Information in Wireshark

The use of Stratum protocol and the presence of certain keywords such as monero, pool, coin, xmr, mine, hash, and cudo identified in DNS requests under domain names are the two most direct ways to detect cryptojacking with network data, as can be seen in Figure 4. However, the hackers could apply obfuscation techniques such as the use of a proxy to go unnoticed. While the use of the container's ephemeral ports by the mining pools is not guaranteed, this could be used as a notable indicator to alert that further review of the network traffic is needed.



```
Info
Standard query 0x200f A stratum.cudopool.com
Standard query 0x6d08 AAAA stratum.cudopool.com
Standard query 0xe87a A stratum.cudopool.com OPT
Standard query 0x8299 AAAA stratum.cudopool.com OPT
```

Fig. 4: Detection of the Stratum Protocol in Wireshark

VII. Docker Container Security

While the focus of our analysis is on detecting unauthorized cryptomining in a Docker host once it is in action, the first line of defense against cryptojacking threats to docker containers is to take appropriate measures to follow best practices for Docker security.

A. *Stay Up to Date*

Keeping software up to date is a key factor to mitigate the possibility of vulnerabilities in software that can be exploited by cryptojackers. It is also useful to run new software through the CVE (Common Vulnerabilities & Exposures) database [34].

B. *Resource Isolation and Management*

Namespaces isolate the processes in a container so they cannot see or affect the host system or another container. Control groups provide for resource management, making sure that each container has a fair amount of disk I/O resources, memory, and CPU. This helps to protect the containers as well as the host from denial of service attacks [24].

C. *Whitelisting/Blacklisting Rules in iptables*

When appropriate, whitelisting rules should be added to the DOCKER-USER iptables to allow only specific IP addresses or networks to access the containers and only expose certain ports [24]. Alternatively, blacklisting should be used to block connections to known cryptomining pools, remote access, or certain ports. For example, pool.*, *pool.com, *pool.org, and *xmr.* block Monero pools that are used for cryptomining.

D. *Principles of Least Privilege for Kernel Capabilities*

Instead of deploying containers on the system having root privileges, assigning containers the minimum privileges required for their processes using an allowlist is recommended [24]. This helps prevent privilege-escalation attacks and will make it much more difficult for a cybercriminal to reach the host even if they are able to gain access to a container.

E. *Image Authentication*

The Docker Engine should be configured to only run images with Docker Content Trust signature verification in order to ensure the integrity and publisher of images, as these can be sources through which cryptojackers gain access to a system [24]. In addition to this, another method to mitigate the risk of falling victim to cryptojacking malware through a malicious file is to scan new images on VirusTotal [35] and ClamAV [36]. Docker also provides a tool for scanning images for vulnerabilities [37].

VIII. Monitoring Host-based Docker Containers

There are steps that can be taken to monitor host-based docker containers to identify attacks. For example, open-source tools can be used to monitor host-based docker containers and trigger alerts for system administrators when certain indicators are present. From the Docker host, a script can run periodically to collect and log the temperature of the host. At the same time, the Docker stats can collect and log the overall CPU and RAM consumption for each container and from the docker inspection, we can get the network information of each container. If the temperature field goes more than 10°Celsius over the average temperature on the CPU core and CPU and RAM in a particular container are more than three times its average consumption, we correlate the IP address using the inspection command. An alert can be sent to the system administrator with the identified parameters for a deeper inspection in the Suricata and Wireshark logs for the container where they could observe the data to search for further indicators to confirm cryptojacking activity. A DNS record and the IP of the container can also be correlated. Netdata can trigger automatic notifications by email and to the monitor channels such as Discord and Slack, providing metrics of high consumption in specific containers.

IX. Conclusion

In this paper, we conducted a forensic analysis using honeypots in order to identify key indicators for the detection of cryptomining in host-based Docker containers. We presented countermeasures for securing host-based Docker containers and an approach for monitoring them for the detection of cryptojacking based on the indicators identified in our forensic analysis. As a future work, we are planning to develop a cryptojacking detection framework for host-based Docker containers using honeypots and machine learning.

Acknowledgement

This work was partially supported by the U.S. National Science Foundation (Awards: NSF-CAREER CNS-1453647, NSF-1663051, and NSF-2219920), CyberFlorida, Microsoft, and Google Research Grants. The views are those of the authors only.

References

[1] CoinGecko, "Cryptocurrency Prices by Market Cap," https://www.coingecko.com/, [Online; accessed 10-April-2022].

[2] E. Tekiner, A. Acar, and A. S. Uluagac, "A lightweight iot cryptojacking detection mechanism in heterogeneous smart home networks," in *Network and Distributed System Security Symposium (NDSS)*, 2022.

[3] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, and A. A. Selcuk, "Sok: cryptojacking malware," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 120–139.

[4] W. Foxley, "Crypto-Jacking Virus Infects 850,000 Servers, Hackers Run off With Millions," CoinDesk, Feb. 23, 2021. [Online]. Available: https://www.coindesk.com/crypto-jacking-virus-infects-850000-servers-hackers-on-the-run-with-millions

[5] R. Hackett, "Tesla Hackers Hacked AWS Cloud Services to Mine Monero," Fortune, Oct. 19, 2020. [Online]. Available: https://fortune.com/2018/02/20/tesla-hack-amazon-cloud-cryptocurrency-mining/

[6] C. Doman, "Team TNT: The First Crypto-Mining Worm to Steal AWS Credentials," Cado Security, Aug. 16, 2020. [Online]. Available: https://www.cadosecurity.com/post/team-tnt-the-first-crypto-mining-worm-to-steal-aws-credentials

[7] A. Sasson, "Docker Honeypot Reveals Cryptojacking as Most Common Cloud Threat," Palo Alto-Unit 42, May 27, 2021. [Online]. Available: https://unit42.paloaltonetworks.com/docker-honeypot/

[8] Stack Overflow, "Stack Overflow Developer Survey," 2020, [Online; accessed 10-April-2022]. [Online]. Available: https://insights.stackoverflow.com/survey/2020#technology-platforms-all-respondents5

[9] W. Fan, Z. Du, D. Fernandez, and V. A. Villagra, "Enabling an anatomic view to investigate honeypot systems: A survey," *IEEE Systems Journal*, vol. 12, no. 4, p. 3906–3919, Dec 2018.

[10] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, "A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems," *IEEE Communications Surveys Tutorials*, 2021.

[11] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, and A. A. Selcuk, "In-browser cryptomining for good: An untold story," in *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE, 2021, pp. 20–29.

[12] S. Bhansali, A. Aris, A. Acar, H. Oz, and A. S. Uluagac, "A first look at code obfuscation for webassembly," in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2022, pp. 140–145.

[13] F. N. Naseem, A. Aris, L. Babun, E. Tekiner, and A. S. Uluagac, "Minos: A lightweight real-time cryptojacking detection system." in *NDSS*, 2021.

[14] D. Tanana, "Behavior-based detection of cryptojacking malware," in *2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, 2020, pp. 0543–0545.

[15] G. Mani, V. Pasumarti, B. Bhargava, F. T. Vora, J. MacDonald, J. King, and J. Kobes, "Decrypto pro: Deep learning based cryptomining malware detection using performance counters," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, 2020.

[16] A. Ahmad, W. Shafiuddin, M. N. Kama, and M. M. Saudi, *A New Cryptojacking Malware Classifier Model Based on Dendritic Cell Algorithm*. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3387168.3387218

[17] A. Gangwal, S. G. Piazzetta, G. Lain, and M. Conti, "Detecting covert cryptomining using hpc," *CoRR*, vol. abs/1909.00268, 2019. [Online]. Available: http://arxiv.org/abs/1909.00268

[18] H. Darabian, S. Homayounoot, A. Dehghantanha, S. Hashemi, H. Karimipour, R. M. Parizi, and K. R. Choo, "Detecting cryptomining malware: a deep learning approach for static and dynamic analysis," *Journal of Grid Computing*, vol. 18, pp. 293–303, 2020.

[19] M. Caprolu, S. Raponi, G. Oligeri, and R. DiPietro, "Cryptomining makes noise: Detecting cryptojacking via machine learning," *Computer Communications*, vol. 171, pp. 126–139, 2021.

[20] E. Tekiner, A. Acar, and A. S. Uluagac, "A lightweight iot cryptojacking detection mechanism in heterogeneous smart home networks," in *Proc. of the ISOC Network and Distributed System Security Symposium (NDSS)*, 2022.

[21] R. R. Karn, P. Kudva, H. Huang, S. Suneja, and I. M. Elfadel, "Cryptomining detection in container clouds using system calls and explainable machine learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 674–691, 2021.

[22] J. Buzzio-Garcia, "Creation of a high-interaction honeypot system based-on docker containers," in *2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4)*, 2021, pp. 146–151.

[23] N. Russo and P. Laskov, "Detection of illicit cryptomining using network metadata," *Info. Security 2021*, vol. 11, 2021.

[24] Docker, "Docker Security," https://docs.docker.com/engine/security/, [Online; accessed 10-April-2022].

[25] Datadog, "10 Trends in Real-World Container Use," Datadog, Oct. 2021. [Online]. Available: https://www.datadoghq.com/container-report/

[26] Docker, "Overview of Docker Compose," https://docs.docker.com/compose/, [Online; accessed 10-April-2022].

[27] Google, "cAdvisor (Container Advisor)," https://github.com/google/cadvisor, [Online; accessed 10-April-2022].

[28] Prometheus Authors, "Prometheus," [Online; accessed 12-April-2022]. [Online]. Available: https://prometheus.io/

[29] Netdata, "Netdata," https://www.netdata.cloud/, [Online; accessed 10-April-2022].

[30] "lm-sensors," https://github.com/lm-sensors/lm-sensors, [Accessed 10-April-2022].

[31] N. Express, "NVMe-CLI," https://nvmexpress.org/open-source-nvme-management-utility-nvme-command-line-interface-nvme-cli/, [Online; accessed 10-April-2022].

[32] "Wireshark," https://www.wireshark.org/, [Accessed 10-April-2022].

[33] "Suricata," https://suricata.io/, [Online; accessed 10-April-2022].

[34] CVE Authors, "Security Vulnerability Datasource," [Online; accessed 6-Mayo-2022]. [Online]. Available: https://www.cve.org/

[35] "VirusTotal," https://www.virustotal.com/gui/home/upload, [Online; accessed 10-April-2022].

[36] Cisco, "ClamAV," https://www.clamav.net/, [Accessed 10-April-2022].

[37] Docker, "Vulnerability Scanning for Docker Local Images," https://docs.docker.com/engine/scan/, [Online; accessed 10-April-2022].