Forensic Analysis of Podman Container Towards Metasploit Backdoor Using Checkpointctl

Hafiidh Akbar Sya'bani[1], Chaerul Umam[2], L. Budi Handoko[3]

[1,2,3]*Informatics Department, Universitas Dian Nuswantoro, Indonesia*

[1]1111202013068@mhs.dinus.ac.id (*)

[2,3][chaerul, handoko]@dsn.dinus.ac.id

Received: 2023-12-08; Accepted: 2024-01-30; Published: 2024-02-09

Abstract— Container systems are a virtualization technology with an isolated environment. The isolated environment in a container system does not make cyber attacks impossible. In this research, containers where a cyber incident occurred, were forensically tested on the container's memory to obtain digital evidence. The forensic process uses standards from the NIST framework with collection, examination, analysis, and reporting stages. The forensic process begins by checking the container to obtain information from the container's memory. When the checkpoint procedure is executed in Podman, it is performed on one of the containers. This process produces a file in the.tar.gz format containing the container's information. After completing the checkpoint process, forensics is done by reading the checkpoint file using a *checkpointctl* tool. Forensic results showed that the container ran a malicious program as a backdoor with a PHP extension.

Keywords— Checkpoint; Container; Forensic Analysis; NIST; Podman.

I. INTRODUCTION

Information and communication technology developments have recently changed how the world interacts, communicates, and carries out its activities. The primary role of computer networks greatly influences this development. A service infrastructure with high capabilities must also handle high data and information traffic. One method to overcome this is to use a container system. The container system is a virtualization technology at the operating system level where every process or application running in each container has the same kernel [1]. Container system virtualizes applications in a lightweight manner, leading to significant improvements in cloud applications [2]. The use of an isolated container system does not preclude the possibility that cyber attacks will occur. After a cyber incident occurs, a digital forensics process is carried out to secure and analyze digital evidence and obtain evidence of an incident or digital security breach [3]. Digital evidence is information that is fragile, easy to change, volatile, and can quickly disappear if not managed carefully [4]. In this research, the forensic process was carried out on a container system where memory in containers works differently from computer memory. Generally, containers require less RAM than operating system containers or VMs [5]. From this problem, the container checkpoint method obtains information from container memory.

Checkpoints in containers are the ability to save the state of a running application so that later, it is possible to continue its execution from the last process run [6]. Container checkpoints represent container resources stored in a file with a certain data structure; the relationship between these resource structures can be reconstructed through the recovery process [7]. The container checkpoint function is generally used to migrate containers without shutting down the container. The checkpoint and restore process on Podman containers can be done using CRIU (Checkpoint and Restore In Userspace). In Podman, after the checkpoint process is carried out on one of the containers, a file in the form *tar.gz* will be obtained. This file contains the information on the container. The contents of this checkpoint file are then read using a program called *checkpointctl* so that previous processes have been executed.

II. RESEARCH METHODOLOGY

To perform forensics on a Podman container using *checkpointctl* requires tools, virtual machines, topologies, related research, and attack schemes to support the research.

In this research, Podman is used due to its functionality as a container management tool that can operate without a daemon (daemons), eliminating the need for root access. This contrasts Docker, where root access is necessary for container operations. Additionally, the experimental mode is required to use the docker checkpoint function in Docker. This experimental mode is intended for testing purposes, and the function or design may change in each release without warning or could be removed entirely in the next update.

The preference for Podman over Docker also stems from the output results of the checkpoint process. In Podman, the checkpoint process generates a single file with a tar.gz extension, allowing for direct reading using the *checkpointctl* tool. On the other hand, in Docker, as the checkpoint feature is still in the experimental stage, the output of the checkpoint process results in a directory containing a collection of JSON files.

Checkpointctl is used in this research since its function is not limited to simply reading the contents of the checkpoint file. It also allows users to access and retrieve important information regarding a container through its checkpoint file. This tool will enable users to quickly find details such as the container name, container ID, container creation time, and the history of processes running in the container. *Checkpointctl* is preferred over other forensic tools like Volatility because it can read memory within containers. Unlike the memory structure in physical or virtual machines, containers operate differently. Utilizing Volatility necessitates a kernel that matches the targeted operating system for identification. However, a challenge arises as the kernel in the container system is identical to the one employed on the host machine. This mismatch becomes problematic when the kernel on the host operating system differs from the one in the Ubuntu image within the Podman container. Due to this problem, the *Checkpointctl* tool is used.

Several key considerations shaped the decision to select NIST 800-86 over ISO 27037 for the forensic investigation. Firstly, the imperative to align with U.S. legal and regulatory standards led to the preference for NIST, a widely recognized framework in the country. Additionally, organizational commitment to cybersecurity protocols prioritizes NIST guidelines, ensuring consistency and adherence to internal standards.

The seamless integration of NIST standards with existing tools and technologies also factored prominently in the choice. The current Podman container infrastructure aligns well with NIST methodologies, making it a practical selection for the investigation. Furthermore, the nature of the case and its specific requirements favored the approach outlined in NIST 800-86 over ISO 27037. The distinct methodological features of the NIST framework align with the unique needs of investigations.

Considering skills and training in understanding and proficiency with the concepts and practices recommended by NIST 800-86 is essential. This familiarity ensured the effective implementation and analysis required for the investigation. This decision-making process emphasizes thoughtful consideration of legal, organizational, technological, and case-specific factors, emphasizing NIST 800-86's alignment with forensic investigative requirements.

A. Data Collection Method

1) Observation: The observation process was based on direct practice in daily container operations. This observation shows the process of creating images, running and shutting down containers, and the basic workings of container systems are known.

2) Consultation: Consultation is carried out with someone who is an expert in their field and a field practitioner who is experienced in conducting digital forensics. This procedure is carried out to obtain insights and information unavailable in written sources.

3) Online Resources: In this research, online sources are used to obtain data. The sources can include articles, video tutorials, software documentation, and discussion forums on related topics.

4) Related Research: Data was collected from searching journals that discuss topics related to the case being researched. Journals with related research are used as additional insight and reference in carrying out this research.

B. Related Research

Research on container forensics was carried out by Imam Riadi, Rusydi Umar, and Andi Sugandi in conducting web forensics on containers with the title "Web Forensics on Container Services Using GRR Rapid Response Framework." The container forensics process is carried out based on NIST standards. After installing the GRR server and GRR client on the victim's and attacker's computers, evidence can be found that the attacker has carried out a DDoS attack using Python code on the victim's computer, running a web server via a Docker container [8].

Another research entitled "Forensic Analysis of Docker Swarm Cluster using Grr Rapid Response Framework" by Sunardi, Imam Riadi, and Andi Sugandi shows digital evidence in the form of logs showing that container clusters running using Docker Swarm as a container orchestrator have been attacked using the DDoS method. After obtaining log files in Docker through a YAML script that has been customized to perform the role of an Artefact Collector Flow, it is possible to read the activity on the network. Based on the findings of this investigation, the Python code that the attacker employed to initiate the attack and the attacker's IP address were discovered [9].

In research conducted by Danar Cahyo Prakoso, Imam Riadi, and Yudi Prayudi with the title "Detection of Metasploit Attacks Using RAM Forensic on Proprietary Operating Systems," a forensic process was carried out on RAM or Random Access Memory using three different tools, namely FTK Imager, Dumpit, and RAM magnets. This research simulates a computer device with the Windows 10 operating system attacked with the Metasploit program. The three previous tools produce three images that can be read using Volatility. The investigation found that the Windows 10 device was running the Metasploit program named explorer.exe, which came from a USB drive connected directly to the device [10].

C. Research Limitation

In this research, problem limitations aim to maintain focus on the main objective. This research focused on finding evidence that the Metasploit backdoor had been executed on containers, with Podman as the container management tool running on the server. This research was conducted on the Fedora operating system with Podman as a container management tool. The forensic method used in this research is based on the NIST framework and consists of four steps: collection, examination, analysis, and reporting.

D. Topology & Attack Scheme

Research begins by designing a container topology. The topology consists of three virtual machines, each with its own tasks. The virtual machine used in this research was created with the Proxmox platform installed on a Dell PowerEdge R720 server computer. Proxmox is a Debian-based Linux virtualization distribution (64-bit) with support for OpenVZ and KVM, thus allowing centralized management of multiple physical servers consisting of at least one master and multiple nodes [11].

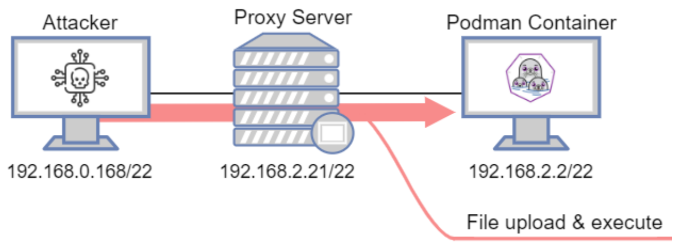


Figure 1. Virtual Machine Topology

Figure 1 shows the attacker's job: create a backdoor, upload files, and execute them on the container. Attacker specifications in Table I. Next, the proxy server is useful as the primary web server to forward attacker requests to the container. This machine uses the Ubuntu Operating System, as shown in Table II. Then, there is a server that runs a web uploader container behind a proxy server. All uploaded files will go into a directory called "uploads," the specifications for this machine are in Table III.

TABLE I
ATTACKER VIRTUAL MACHINE SPECIFICATION

Components	Specification
Processor	Two sockets and two core
Memory	3 GB
Hard disk	40 GB
Operating system	Kali Linux 2023.1
IP address	Static

TABLE II
PROXY SERVER VIRTUAL MACHINE SPECIFICATION

Components	Specification
Processor	One socket and one core
Memory	1 GB
Hard disk	14 GB
Operating system	Ubuntu Server 22.04
IP address	Static

TABLE III
PODMAN CONTAINER VIRTUAL MACHINE SPECIFICATION

Components	Specification
Processor	1 socket and 2 core
Memory	2 GB
Hard disk	28 GB
Operating system	Fedora Server 38
IP address	Static

The container will run a modified image of Ubuntu server version 22.04 installed with the Nginx web server. The container will run several main processes, including bash and a bash program with the name *dockstart.sh*, which runs the php7.4-fpm program and the nginx web server. This container will be named "ubuntu-web upload" and exposes port 3062 from the host to port 80 in the container. The Nginx web server in the container is used to run a simple file upload website. The upload feature does not apply file content restrictions so that all types of files can be uploaded to this website.

Metasploit Framework collects system vulnerability information, penetration testing, and IDS development [12]. The Metasploit framework console can scan targets, exploit security vulnerabilities, and collect data through backdoors created on the attacker's VMs [13]. The backdoor is created using Metasploit's built-in payload generator called MsfVenom. Next, this backdoor is saved with the name backdoor.php.

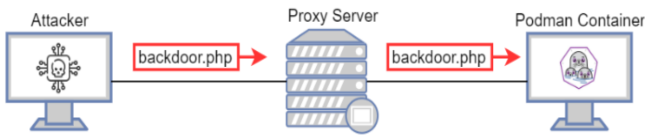
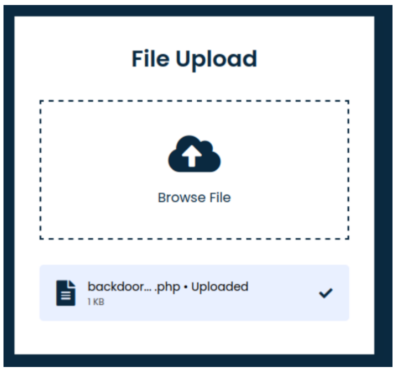


Figure 2. Attacker Upload the Backdoor Files

After the backdoor file is created, the next step is for the attacker to upload the previous backdoor.php file to the web server, as shown in Figure 2. In Figure 3, the attacker uploads the backdoor file to a web server, where this web server is run in a Podman container, and files inside the container are mounted to the host machine.


File Upload

Browse File

backdoor....php • Uploaded
1 KB

Figure 3. Attacker Upload the Backdoor File to Container

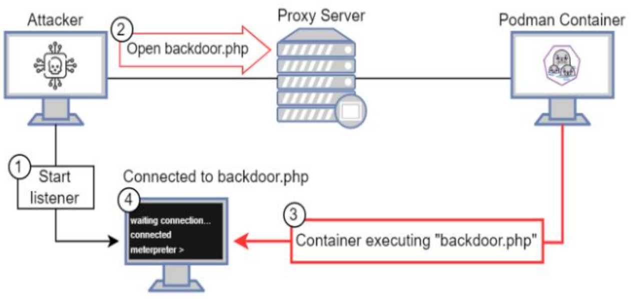
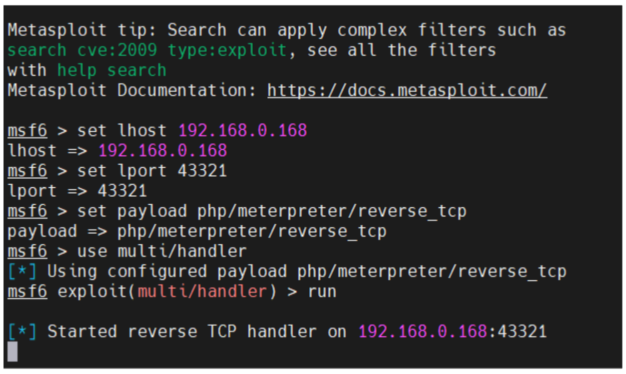


Figure 4. Attacker Executing Backdoor File

The web server then sent the file to the proxy server. Proxy servers are useful for monitoring and controlling client servers [14]. The uploaded file is then forwarded from the proxy server to the container server so that the file is stored in the container storage. Figure 4 illustrates how the attack occurs on the system once the backdoor has been uploaded.

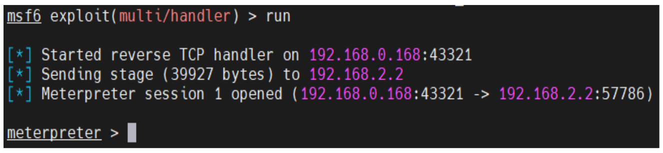
Once the file has been uploaded, in Figure 5, the attacker uses *msfconsole* as a listener for the backdoor and waits for it to be executed.



```
Metasploit tip: Search can apply complex filters such as
search cve:2009 type:exploit, see all the filters
with help search
Metasploit Documentation: https://docs.metasploit.com/

msf6 > set lhost 192.168.0.168
lhost => 192.168.0.168
msf6 > set lport 43321
lport => 43321
msf6 > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 > use multi/handler
[*] Using configured payload php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.0.168:43321
```

Figure 5. Attacker Using Msfconsole as Listener

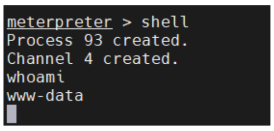
The attacker then executes the uploaded backdoor by accessing the backdoor.php file via the "uploads" directory in the browser to establish a connection between the attacker's and the victim's computers. The console will display "*meterpreter*" when the connection is established, as shown in Figure 6.



```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.0.168:43321
[*] Sending stage (39927 bytes) to 192.168.2.2
[*] Meterpreter session 1 opened (192.168.0.168:43321 -> 192.168.2.2:57786)

meterpreter >
```

Figure 6. Connection Established Between Container to Attacker

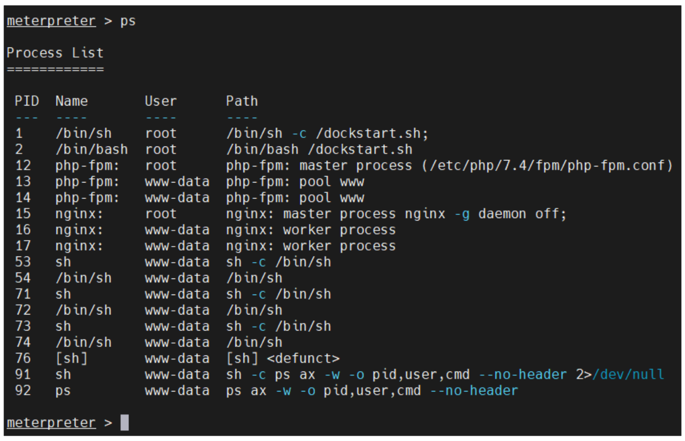
At this stage, it is necessary to determine whether the attacker has successfully entered the container. First, determine what user the attacker *logged-in*. Based on Figure 7 using the "*whoami*" command, it could be known that the current user is "www-data." Next, list the process running inside the container.



```
meterpreter > shell
Process 93 created.
Channel 4 created.
whoami
www-data
```

Figure 7. Current User

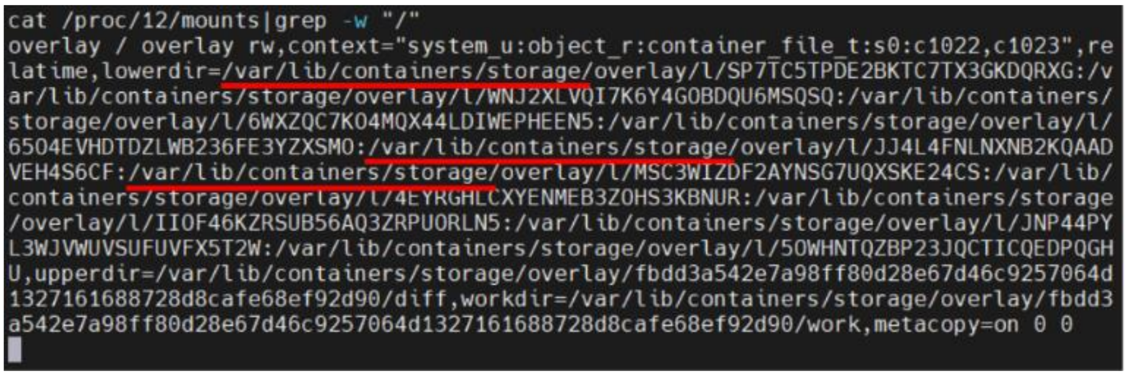
From the list in Figure 8, inspect the mount directory for "/" from a process run by user "*www-data*". If the mount directory refers to "/var/lib/containers/storage, " it runs inside the Podman container. Here, the *php-fpm* process with PID 12 is used.



```
meterpreter > ps

Process List
============

 PID  Name       User      Path
 ---  ----       ----      ----
 1    /bin/sh    root      /bin/sh -c /dockstart.sh;
 2    /bin/bash  root      /bin/bash /dockstart.sh
 12   php-fpm:   root      php-fpm: master process (/etc/php/7.4/fpm/php-fpm.conf)
 13   php-fpm:   www-data  php-fpm: pool www
 14   php-fpm:   www-data  php-fpm: pool www
 15   nginx:     root      nginx: master process nginx -g daemon off;
 16   nginx:     www-data  nginx: worker process
 17   nginx:     www-data  nginx: worker process
 53   sh         www-data  sh -c /bin/sh
 54   /bin/sh    www-data  /bin/sh
 71   sh         www-data  sh -c /bin/sh
 72   /bin/sh    www-data  /bin/sh
 73   sh         www-data  sh -c /bin/sh
 74   /bin/sh    www-data  /bin/sh
 76   [sh]       www-data  [sh] <defunct>
 91   sh         www-data  sh -c ps ax -w -o pid,user,cmd --no-header 2>/dev/null
 92   ps         www-data  ps ax -w -o pid,user,cmd --no-header

meterpreter >
```

Figure 8. Process Run Inside the Container

Based on the process inspection, as shown in Figure 9, it is known that the "/" directory is mounted in "*/var/lib/containers/storage*" a few times. Based on this, it can be determined that the executed attack took place within a Podman container.



```
cat /proc/12/mounts|grep -w "/"
overlay / overlay rw,context="system_u:object_r:container_file_t:s0:c1022,c1023",re
latime,lowerdir=/var/lib/containers/storage/overlay/l/SP7TC5TPDE2BKTC7TX3GKDQRXG:/v
ar/lib/containers/storage/overlay/l/WNJ2XLVQI7K6Y4GOBDQU6MSQSQ:/var/lib/containers/
storage/overlay/l/6WXZQC7KO4MQX44LDIWEPHEEN5:/var/lib/containers/storage/overlay/l/
6504EVHDTDZLWB236FE3YZXSMO:/var/lib/containers/storage/overlay/l/JJ4L4FNLNXNB2KQAAD
VEH4S6CF:/var/lib/containers/storage/overlay/l/MSC3WIZDF2AYNSG7UQXSKE24CS:/var/lib/
containers/storage/overlay/l/4EYRGHLCXYENMEB3ZOHS3KBNUR:/var/lib/containers/storage
/overlay/l/IIOF46KZRSUB56AQ3ZRPUORLN5:/var/lib/containers/storage/overlay/l/JNP44PY
L3WJVWUVSUFUVFX5T2W:/var/lib/containers/storage/overlay/l/5OWHNTQZBP23JQCTICQEDPQGH
U,upperdir=/var/lib/containers/storage/overlay/fbdd3a542e7a98ff80d28e67d46c9257064d
1327161688728d8cafe68ef92d90/diff,workdir=/var/lib/containers/storage/overlay/fbdd3
a542e7a98ff80d28e67d46c9257064d1327161688728d8cafe68ef92d90/work,metacopy=on 0 0
```

Figure 9. Root Mount Directory

E. Forensics

A forensic process was carried out on the container based on previous incidents. This forensic process is carried out in several stages and is based on the NIST framework. This stage comprises collection, examination, analysis, and reporting [15].

1) Collection: This stage aims to secure evidence after a cyber incident occurs. After a cyber incident occurs, evidence must be collected to help investigate the case. After the source of digital evidence, the next step is the data acquisition process. This process needs to pay attention to the data that will be acquired, whether it is classified as volatile or non-volatile. Volatile means that the data will be lost when the device is turned off or not powered by electricity. In this research, compromised Podman containers were checkpointed. Checkpointed containers will automatically have their state frozen. This checkpoint process uses commands from Podman integrated with a tool called CRIU in its implementation. The resulting checkpoint file will be made into a tar.gz file, which will later be used in the examination process, thus producing digital evidence.

2) Examination: After the data is obtained, the next process is examination. The examination process is the stage of examining data collected forensically, either automatically or manually and ensuring that the data obtained in the form of files is genuine and matches what was obtained at the scene of a computer crime [16]. At this stage, the data that has been collected is then examined to look for digital evidence. In this process, the contents of previously obtained data can be read using certain forensic tools. This data reading can also be done based on text, audio, visual graphics, or compressed files. In this study, the examination process was carried out by reading the previous container checkpoint file. The checkpoint file with the tar.gz extension that has been created is then read in its contents using a tool called *checkpointctl*. Information about the checkpoint can be found in this tool. From this reading, information about the container can be obtained, such as the container ID, the used image, the name of the container, and the process ID running in it.

3) Analysis: The analysis stage is the stage of looking at the results of the examination stage in detail to obtain digital evidence from evidence that has been carried out in the previous stages. This stage can be carried out manually or with the additional assistance of software spread in digital forensics [17]. The digital evidence obtained is examined in the analysis process, and conclusions are drawn. This process results in how the attack occurred, what caused the attack, and what impact occurred after the attack was carried out.

4) Reporting: Reporting is the stage of preparing a report on the results of forensic data analysis, including findings, methodology, and conclusions, and submitting them to the competent authorities [18]. The results of the preceding analysis process are subsequently summarized and concluded to generate a final report. The findings from the analysis process are assembled into a report that outlines the incidents that transpired, the acquired digital evidence, the analysis process, and the drawn conclusions. The report furnishes a comprehensive depiction of the forensic investigation. Additionally, the report identifies the issue and suggests several measures to minimize the probability of future incidents.

III. Result and Discussion

A. *Collection*

After the attacker uploads and executes the backdoor file, a checkpoint process is carried out on the Podman container running the web uploader. The checkpoint file is created in the form *tar.gz* with the final name "webupload_ckpt.tar.gz" and stored in the user directory at "*/home/pod*". Before the checkpoint process is carried out, the Podman container is still running. The checkpointing process is carried out using the Podman command below:

```
podman container checkpoint ubuntu-
webupload --tcp-established -e
/home/pod/webupload_ckpt.tar.gz
```

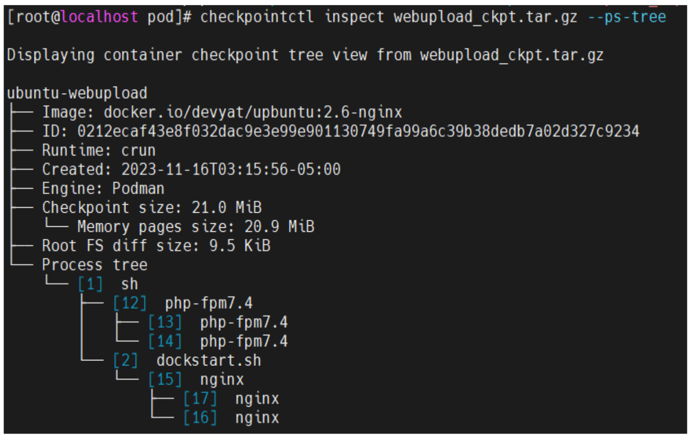
The command uses the "*--tcp-established*" option because the image has a TCP connection. This option must also be used if you want to restore the container. Based on the command above, the created checkpoint file will be saved in the "/home/pod" directory with the name "webupload_ckpt.tar.gz."

B. *Examination*

From the checkpoint file that has been created, *checkpointctl* is then used to analyze the checkpoint file using checkpointctl. The processes running in the container can be seen via the checkpoint file.

```
checkpointctl inspect webupload_ckpt.tar.gz
-ps-tree
```

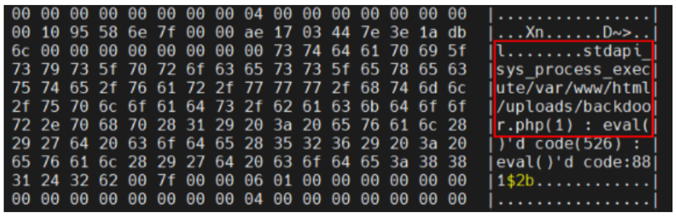
The command above inspects the checkpoint file with the name "webupload_ckpt.tar.gz." The "*-ps-tree" option briefly displays* the processes running inside the container and is displayed in a tree schema.



```
[root@localhost pod]# checkpointctl inspect webupload_ckpt.tar.gz --ps-tree

Displaying container checkpoint tree view from webupload_ckpt.tar.gz

ubuntu-webupload
├── Image: docker.io/devyat/upbuntu:2.6-nginx
├── ID: 0212ecaf43e8f032dac9e3e99e901130749fa99a6c39b38dedb7a02d327c9234
├── Runtime: crun
├── Created: 2023-11-16T03:15:56-05:00
├── Engine: Podman
├── Checkpoint size: 21.0 MiB
│   └── Memory pages size: 20.9 MiB
├── Root FS diff size: 9.5 KiB
└── Process tree
    └── [1]  sh
        ├── [12]  php-fpm7.4
        │   ├── [13]  php-fpm7.4
        │   └── [14]  php-fpm7.4
        └── [2]  dockstart.sh
            └── [15]  nginx
                ├── [17]  nginx
                └── [16]  nginx
```

Figure 10. Result of Inspect Checkpoint File

From Figure 10, the inspect command displays detailed information contained in the container. From this command, the ID number of each process can be known. The previous attack method used the PHP program so that the search would focus on the entire process "*php-fpm7.4*" with process IDs 12, 13, and 14.



```
00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  |................|
00 10 95 58 6e 7f 00 00 ae 17 03 44 7e 3e 1a db  |...Xn......D~>..|
6c 00 00 00 00 00 00 00 00 73 74 64 61 70 69 5f  |l........stdapi_|
73 79 73 5f 70 72 6f 63 65 73 73 5f 65 78 65 63  |sys_process_exec|
75 74 65 2f 76 61 72 2f 77 77 77 2f 68 74 6d 6c  |ute/var/www/html|
2f 75 70 6c 6f 61 64 73 2f 62 61 63 6b 64 6f 6f  |/uploads/backdoo|
72 2e 70 68 70 28 31 29 20 3a 20 65 76 61 6c 28  |r.php(1) : eval(|
29 27 64 20 63 6f 64 65 28 35 32 36 29 20 3a 20  |)'d code(526) : |
65 76 61 6c 28 29 27 64 20 63 6f 64 65 3a 38 38  |eval()'d code:88|
31 24 32 62 00 7f 00 00 06 01 00 00 00 00 00 00  |1$2b............|
00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  |................|
```

Figure 11. ASCII String of Metasploit Log Message in The Container

The reading process uses the *checkpointctl* command by adding the "*memparse*" command and is directed sequentially to process ID 12, 13, then 14. This reading results in the memory contents displayed in a table containing the memory address, hexadecimal value, and ASCII string. In Figure 11, the results of reading the php-fpm7.4 process with ID 13, a hexadecimal string was found, which, if translated in ASCII, become "stdapi_sys_process_execute/var/www/html/uploads/backdoor.php (1)" which is one of the log messages that commonly found in the Metasploit program.

C. Analysis

In the analysis process, digital evidence is examined, and conclusions are drawn. This process consists of how the attack occurred, what caused the attack, and what impact occurred after the attack was carried out. The examination process found that the *backdoor.php* program had been executed in the container and was running on process ID 13, namely process "php-fpm7.4."

These results show that the type of attack was a file upload attack, which utilized the upload feature on a website that does not apply a file content restriction system so that files of any type can be uploaded to the server.

A file upload attack is an attack method that exploits the vulnerability of a website in receiving files uploaded by clients. The uploaded file, once executed, can be used to launch various attacks, such as installing web shells, polluting web applications, spreading malware, and phishing [19]. The condition of unrestricted file upload refers to the absence of restrictions during the file uploading process and can let attackers retrieve information from the server [20].

The file upload vulnerability carries significant consequences, potentially resulting in code execution on the server or client side. Detection by malicious actors is highly possible, and the common occurrence of this vulnerability contributes to its elevated severity. A comprehensive evaluation of the access controls within a file upload module is essential for effectively mitigating associated risks. In the OWASP Top 10 2021, the file upload vulnerability is included in the injection category in third place with the name A03-2021: Injection. This vulnerability arises because there is no validation by the server regarding the type of file that will be received and uploaded to the server. This validation may involve filtering file names, extensions, MIME type, and size.

On the server side, attacks involve compromising the web server through the upload and execution of a web shell, enabling a range of malicious activities. Meanwhile, on the client side, uploading malicious files can expose the website to vulnerabilities such as XSS or Cross-site Content Hijacking. Additionally, uploaded files may exploit other application vulnerabilities, leading to potential client-side or server-side attacks.

Numerous risks arise and trigger vulnerabilities in libraries or applications on both client and server sides, the exploitation of real-time monitoring tools, execution of malicious code, injection of phishing pages, defacement of the website, hosting problematic files, and potential exposure of sensitive information. File uploaders may reveal internal details through error messages.

Another vulnerability on the server is that there are no access restrictions on the directories on the web server. In this case, the attacker can access the "uploads" directory, which is the directory where the files that have been uploaded are stored. Attackers can list, read, download, and execute files in the "uploads" directory. In addition, the opened file can be executed on the server. As in this research, attackers exploit this vulnerability to execute malicious programs on the server side.

This vulnerability is also called directory listing, where the contents of the current directory can be displayed in the browser. This vulnerability is included in the OWASP top 10 2021 in the Security Misconfiguration category in 5th place with the name A05:2021-Security Misconfiguration.

D. Reporting

Through analysis of the incident and the forensic results obtained, it can be reported that the container was running a malicious PHP file caused by a vulnerability in the web server running in the container. One way to overcome this incident is to implement file content restrictions where the server only accepts files of certain types. It is also possible to restrict the server to only executing files from specific directories to avoid this incident. Another precaution that can be taken is to prohibit listing directories so that attackers cannot access other directories on the server. One way to do this is by configuring the ".htaccess" file.

IV. Conclusion

Research on the forensic process involving Podman containers and *checkpointctl* reveals that creating a checkpoint file from the container is essential for conducting forensic processes. Podman's built-in commands, with the additional option "*--tcp-established*", can be utilized to create checkpoint files, which are saved with the *.tar.gz* extension. The analysis of the checkpoint file using *checkpointctl* provides details such as the container ID, container name, the image used, and the running processes within the container.

The reading of the process history on the container is displayed in the form of a table containing a hexadecimal string and translated into ASCII text. Through the contents of the process running in this container, the ASCII string "stdapi_sys_process_execute/var/www/html/uploads/backdoor.php (1)" is found, which is one of the log messages commonly found in the Metasploit program. The message also provides information that the process being executed is backdoor.php, which is located in the uploads directory, where this directory is where the files uploaded by the user are stored. From this, it can be concluded that the container has executed a file with the name backdoor.php.

Attacks on containers fall into the file upload attack type, where the attacker uploads a malicious file on a vulnerable website and then executes the file. The file upload attack method can be caused by no validation or filter applied on the client and server side, XSS, unlimited maximum file size, and not using proxy server tools. The server is also vulnerable to unrestricted directory access, specifically in the "uploads" directory, allowing attackers to view and execute files. This exploit was used to run malicious programs on the server.

To mitigate the vulnerability associated with file uploads, it is necessary to have multiple validators act as file filters before uploading to server storage. Crucial validations include checking the file name, extension, MIME type, and size. Special attention should be given to naming uploaded files, as it can impact server components. For instance, using names like "index.php" or ".htaccess" can affect server performance. To address this, files destined for server storage can be renamed by appending specific strings like dates or sequence numbers or by entirely changing the name.

In the illustration in Figure 12, files uploaded to the server will undergo an initial processing stage. This process includes a MIME-type check to validate that the uploaded file conforms to the specified conditions. The next process involves validating the file size if the file type is deemed safe. The next step can be continued if the uploaded file size is below the specified maximum limit. To ensure that the server is secure, the names of files are modified so that the user who uploads the file is unaware of the actual name stored on the server. This method is expected to prevent users from accessing the file because the file name has been scrambled, so they do not know which file is supposed to be executed.

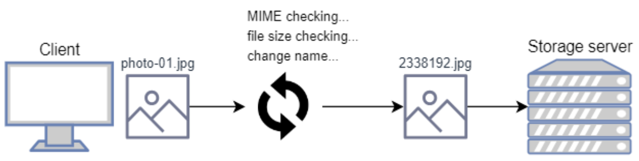


Figure 12. Validate And Change File Name Before Uploading To Server Storage

For the directory listing vulnerability, it is required to restrict visitor access to specific directories, such as the "uploads" directory. It is possible to turn off directory listing on the web server side. For example, on the Apache web server, it is necessary to remove the string "*Indexes*" from the "*Options Indexes FollowSymLinks*" line in the Apache configuration. This will disable directory listing. Another way to turn off directory listing on the Apache web server is to include the "*Options –Indexes*" line in the file called *.htaccess*. On the Nginx web server, by default, the directory listing will be disabled, but if one day the Nginx web server turns on the directory listing, then to turn it off is to change the value in "*autoindex on*" to "*autoindex off*" in the *nginx* configuration file.

Turning off directory listing is essential to prevent accidental information disclosure on a website, such as configuration files or source code, which attackers could exploit to identify another vulnerability. It also enhances security by failing malicious activities like brute force attacks and unauthorized access attempts, thereby reducing the potential attack surface. Additionally, turning off directory listing controls the website's appearance, allowing for a more professional presentation and guiding users to the intended content. This research can be developed in further study, such as further testing to determine container network traffic. This can be done to find out if the attack occurred on the network side, escape from the container, and find the attacker's IP address using other forensic tools such as Google Rapid Response and Sysdig.

V. References

[1] S. Dwiyatno, E. Rakhmat och G. Oki, ”Implementasi Virtualisasi Serv Berbasis Docker Container,” *Jurnal PROSISKO,* vol. 7, nr 2, 2020.

[2] C. Pahl, A. Brogi, J. Soldani och P. Jamshidi," Cloud Contair Technologies: A State-of-the-Art Review," *IEEE Transactions Cloud Computing,* vol. 7, nr 3, pp. 677-692, 2019.

[3] G. H. A. Kusuma och I. N. Prawiranegara, ”Analisa Digital Foren Rekaman Video CCTV dengan Menggunakan Metadata dan Hasl *Prosiding SISFOTEK,* vol. 3, nr 1, pp. 223 - 227, 2019.

[4] I. Riadi, A. Fadlil och M. I. Aulia, ”Investigasi Bukti Digital Optic Drive Menggunakan Metode National Institute of Standard a Technology (NIST),” *Jurnal RESTI (Rekayasa Sistem dan Teknol Informasi),* vol. 4, nr 5, pp. 820-828, 2020.

[5] G. Lahmann, T. McCann och W. Lloyd," Container Memory Allocati Discrepancies: An Investigation on Memory Utilization Gaps Container-Based Application Deployments," Orlando, FL, USA, 201

[6] C. Yang," Checkpoint and Restore of Micro-service in Docl Containers," i *Proceedings of the 3rd International Conference Mechatronics and Industrial Informatics*, Atlantis Press, 2015, pp. 91 918.

[7] X. Chen, J.-H. Jiang och Q. Jiang," A Method of Self-adaptive Pre-co Container," Zhangjiajie, China, 2015.

[8] I. Riadi, R. Umar och A. Sugandi," Web Forensic on Container Servic Using GRR Rapid Response Framework," *Scientific Journal Informatics,* vol. 7, nr 1, pp. 33-42, 2020.

[9] Sunardi, I. Riadi och A. Sugandi," Forensic Analysis of Docker Swa Cluster using Grr Rapid Response Framework," *International Jour of Advanced Computer Science and Applications (IJACSA),* vol. 10, 2, pp. 459-466, 2019.

[10] D. C. Prakoso, I. Riadi och Y. Prayudi," Detection of Metasploit Attac Using RAM Forensic on Proprietary Operating Systems," *Kinet Game Technology, Information System, Computer Network, Computir Electronics, and Control,* vol. 5, nr 2, pp. 155-160, 2020.

[11] O. W. Purbo, Membuat Sendiri Cloud Computing Server Menggunak Open Source, Yogyakarta: Andi, 2012.

[12] S. Raj och N. K. Walia," A Study on Metasploit Framework: A Pe Testing," i *2020 International Conference on Computatio Performance Evaluation (ComPE),* Shillong, Meghalaya, India, 202

[13] S. Thomas och B. T. K," Vulnerability Testing on Rooted Andr Phones Using Msf Venom Payloads," 2021.

[14] A. I. R och F. Marisa, ”Membangun Proxy Serversebagai Penyari Konten Dan Manajemen Akses Jaringan Internet Pada PT. Indomari Surabaya,” *Jurnal Teknologi dan Manajemen Informatika,* vol. 3, nr pp. 172-177, 2017.

[15] K. Kent, S. Chevalier, T. Grance och H. Dang," Guide to Integrati Forensic Techniques into Incident Response," *National Institute Standards and Technology,* pp. doi: 10.6028/nist.sp.800-86, 2006.

[16] I. Riadi, U. Rusydi och I. M. Nasrulloh," ANALISIS FORENS DIGITAL PADA FROZEN SOLID STATE DRIVE DENGA METODE NATIONAL INSTITUTE OF JUSTICE (NIJ)," *Elin (Electronics, Informatics, and Vocational Education),* vol. 3, nr 1, 70-82, 2018.

[17] N. S. och I. Riadi, ”Analisis Forensik Smartphone Andr Menggunakan Metode NIST dan Tool MOBILedit Forensic Expres *Jurnal Informatika Universitas Pamulang,* vol. 5, nr 1, pp. 89-94, 20

[18] D. Royadi, M. Asfi och A. Sevtiana, "Implementasi Metode Stanc NIST Dalam Analisis Data Forensik Studi Kasus Penipuan Sal Transfer Mencatut Nama Wabup Pada SMP Ar-rohman Krangkeng *LOFIAN: Jurnal Teknologi Informasi Dan Komunikasi*, vol. 3, nr 1, p 12-19, 2023.

[19] J. Huang, Y. Li, J. Zhang och R. Dai," UChecker: Automatica Detecting PHP-Based," Portland, OR, USA, 2019.

[20] I. Riadi och E. I. Aristanto," An Analysis of Vulnerability Web Agai Attack Unrestricted Image File Upload," *Computer Engineering Applications Journal*, vol. 5, nr 1, pp. 19-28, 2016.