Digital Forensic Investigation of Debian 4Geeks Server

This forensic investigation and penetration test of the **4Geeks Server** were conducted using the **ForensiX.sh** script, which automated all key steps, including evidence extraction, system integrity checks, and analysis. The use of this script ensured a streamlined and efficient process while maintaining the integrity of the system throughout the investigation.

Methodology

1. Environment Setup

- **Virtual Machine Deployment**: The `.ova` file for the Debian 4Geeks Server was imported into VirtualBox, and the `.vdi` file (containing the system's virtual disk) was verified for integrity before the testing began. This ensured that the system image had not been tampered with.
- **File Integrity Verification**: SHA-256 hashes of both the `.ova` and `.vdi` files were computed to confirm their authenticity. The calculated hashes were compared with those provided by the client, ensuring the image’s integrity had been maintained.

```
sha256sum 4GeeksServer.ova
sha256sum 4GeeksServer.vdi
```

2. Kali Forensics USB Boot

- **Booting into Kali Linux Forensics Mode**: The system was booted into Kali Linux from a USB drive. Kali Linux was chosen due to its **specialized tools for digital forensics** and penetration testing and USB Boot to **reduce the CPU, RAM, and Disk Space necessity**. The forensics mode of Kali ensures that the system operates in a **write-protected state**, meaning no modifications are made to the server during the investigation.

  By using this live environment, we ensured that the evidence was preserved intact and that no traces of the analysis were left behind on the server.

3. Write Block with ForensiX.sh

- **Write Blocker Setup**: The **ForensiX.sh** script was used to automate the process of mounting the server’s disk in read-only mode, ensuring a write blocker was applied. This is an essential step in digital forensics, as it prevents any accidental modification of the data during analysis.

  With the script’s automation, this step ensured the authenticity and integrity of the evidence throughout the investigation, preventing tampering or corruption while extracting critical information from the system.

4. Information Gathering and Evidence Extraction with ForensiX.sh

- **Forensic Evidence Extraction**: The **ForensiX.sh** script automated the extraction of various types of evidence from the server, including system logs, application data, user activity, and network traffic. The script facilitated a thorough and efficient extraction process while ensuring that the data collected was accurate and unaltered.

  The script generated the following reports:

  - **Full Analysis Report**: A detailed technical report with in-depth findings, including system vulnerabilities, misconfigurations, and potential security risks.
  - **Executive Summary**: A concise, high-level summary of the most critical findings for decision-makers.
  - **Security Incident Report**: A detailed analysis of any potential or confirmed security incidents discovered during the investigation.
  - **Recovery Plan**: A list of recommended actions to address the identified vulnerabilities and improve system security.

5. Data Analysis and Report Generation

- **Analysis of Extracted Data**: Once the evidence was extracted, the data was analyzed for potential security breaches, misconfigurations, outdated services, and other vulnerabilities. The **ForensiX.sh** script helped to automate the analysis and organize the findings into actionable insights.

- **Reporting**: The results of the analysis were compiled into several reports, each providing a structured overview of the findings:

  - **Full Analysis Report**: This detailed report contained a comprehensive overview of all findings from the forensic analysis, including technical observations and evidence of potential security risks.
  - **Executive Summary**: A high-level summary for non-technical stakeholders, outlining the most critical security issues.
  - **Security Incident Report**: A specific report detailing any security incidents or areas of the system that were found to be compromised.
  - **Recovery Plan**: Actionable recommendations for addressing the identified vulnerabilities and improving the system's overall security.

6. Final Verification and Integrity Checks

- **Re-verification of Integrity**: After the evidence was extracted and analyzed, the **ForensiX.sh** script also re-verified the integrity of the `.vdi` file to ensure that no tampering or alterations occurred during the analysis process. This final verification was crucial to maintaining the chain of custody and ensuring that the evidence remained unmodified.

  This step provided assurance that the evidence collected throughout the investigation was authentic and reliable, supporting any legal or security-related follow-up actions.

Conclusion

The use of the **ForensiX.sh** script allowed for an efficient and reliable forensic investigation of the Debian 4Geeks Server. The script automated critical steps, including environment setup, write blocking, evidence extraction, data analysis, and integrity checks, ensuring that the investigation was thorough and the evidence remained unaltered.

By automating the forensic process, **ForensiX.sh** minimized the risk of human error while maintaining the integrity of the server’s data. The detailed reports generated from the evidence extraction process provided a comprehensive overview of the server’s security posture, highlighting vulnerabilities, security incidents, and providing actionable recommendations to enhance the system's security.

Additionally, the **ForensiX.sh** script demonstrated an extremely low necessity for CPU, RAM, and Disk Space compared to classic open-source tools and the expensive cost of some private tools. This efficiency makes it an ideal choice for forensic investigations, ensuring thorough analysis without the need for extensive system resources or costly software.