|  |  |
| --- | --- |
| Name: | **Subhasish Mukherjee** |
| Lab User ID: | **23SEK3324\_U05** |
| Date: | 10/01/24 |
| Application Name: | DataDog – Vulnarable Java Application |

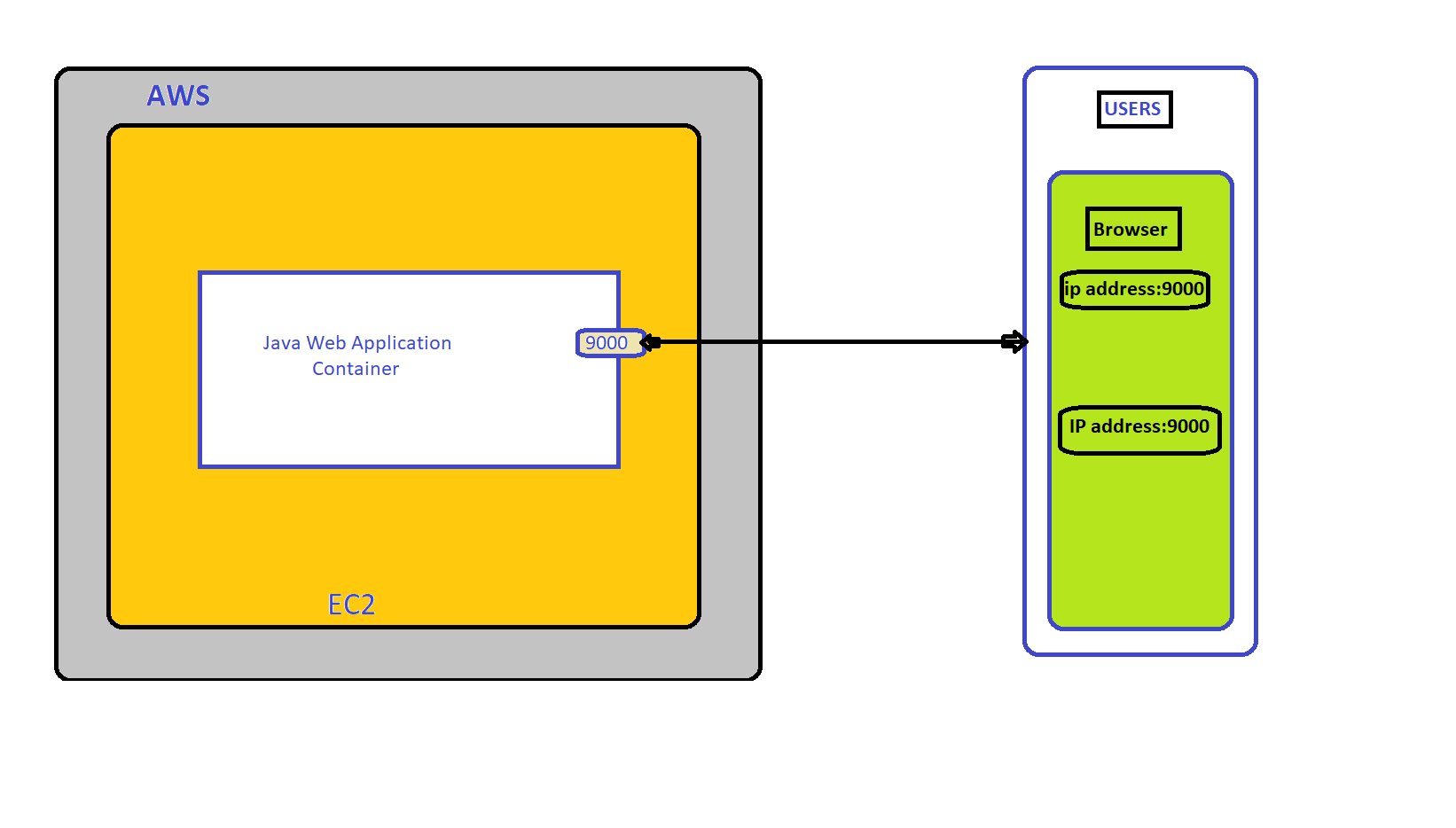
**Follow the below guidelines:**





System Architecture:

(Understand the system and document the physical and logical architecture of the system, use the shapes and icons to capture the system architecture)



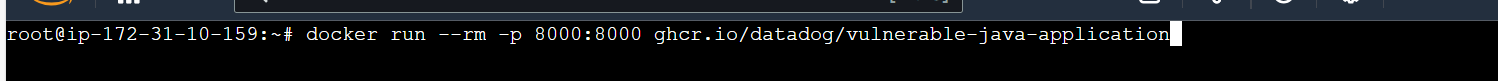
Docker Container

IPaddress:8000

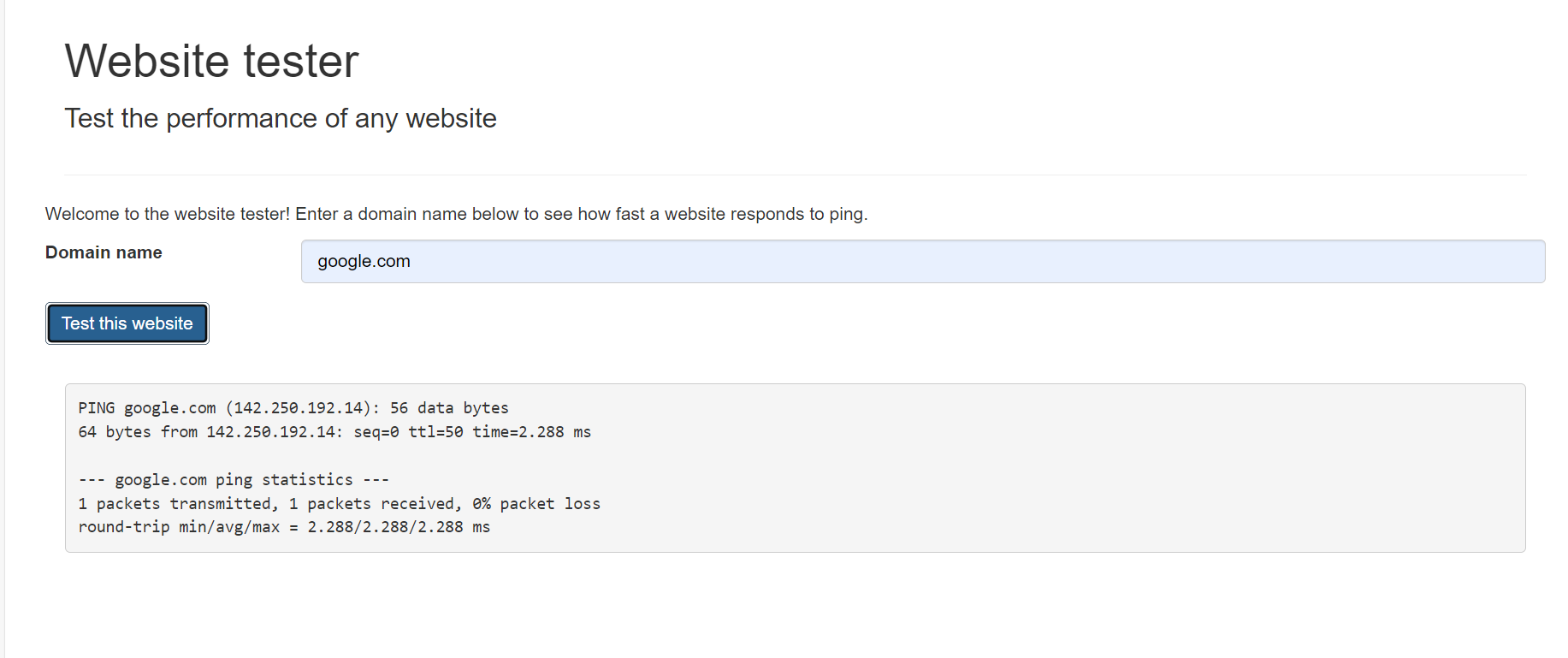
80000

**Web App**

**Command:**



After performing this command we can access the application in the wesite. Use public IP address and open port 8000. < IP address:8000 >



The Java application you have allows users to perform live website tests through a user-friendly interface. The main feature involves a domain name search bar where users can input the URL of a live website. Upon entering a domain and clicking the "test the website" button, the application initiates a ping to the specified domain.

Ping is a network utility that sends a message to the target domain and measures the time it takes for a response, providing valuable information about the website's connectivity and responsiveness.

For example, if you input "google.com" and initiate the test, the application sends a ping to Google's servers and displays the ping result. This information is crucial for assessing the health and performance of a live application. The application's user-friendly design simplifies the testing process, making it accessible even for users with limited technical knowledge.

This functionality aids in troubleshooting network issues, identifying potential latency problems, and ensuring that the live application is responsive and accessible. Overall, the Java application serves as a valuable tool for website administrators and developers to assess the status of live applications.

Define system’s normal behavior:

(Define the steady state of the system is defined, thereby defining some measurable outputs which can indicate the system’s normal behavior)

The Java application serves as a versatile tool for testing live websites through its user interface (UI). The UI features a domain name search bar, enabling users to input the desired live domain for testing. For instance, entering "google.com" into the search bar and subsequently clicking the "test the website" button initiates a process wherein the application pings the specified website.

The term "ping" refers to a network utility tool that sends Internet Control Message Protocol (ICMP) Echo Request messages to a target host (in this case, google.com). The application, acting as a network diagnostic tool, sends out these packets to the target website to assess its responsiveness and measure the round-trip time for the data packets to travel from the source to the destination and back.

Upon clicking the "test the website" button, the application establishes a connection with the provided domain by sending a series of ICMP packets. The website then responds, and the application captures and analyzes the response time and other relevant network metrics. This information is then presented to the user, offering insights into the website's performance and connectivity status.

Such a tool is invaluable for web developers, network administrators, and anyone interested in assessing the live status of a website. It provides a quick and convenient means to troubleshoot network issues, gauge website responsiveness, and identify potential connectivity problems. This Java application's ability to test live applications by pinging specified domains enhances the efficiency of website maintenance and ensures optimal performance for end-users.

Hypothesis:

(During an experiment, we need a hypothesis for comparing to a stable control group, and the same applies here too. If there is a reasonable expectation for a particular action according to which we will change the steady state of a system, then the first thing to do is to fix the system so that we accommodate for the action that will potentially have that effect on the system. For eg: "If one of our database servers fails, our service will automatically switch to a backup server, and users will not experience any downtime or data loss.")



**Known**

If the application experiences any kind of external attack, then its defense mechanisms will effectively mitigate the threat, preventing unauthorized access or data compromise.

If we induce a CPU outage by simulating high load, then the system will automatically scale up to handle increased demand, maintaining performance and availability.

**Unknown**

If a SQL injection attack is executed on the vulnerable Java application, then the application's security measures will may not prevent or mitigate the unauthorized database access attempt. Causing any kind of security issues.

Engineers simulate real-world unpredictability by randomly manipulating network bandwidth or introducing intermittent latency in various system components to test and ensure robustness under diverse and challenging conditions.

**Unknown**

**Known**

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

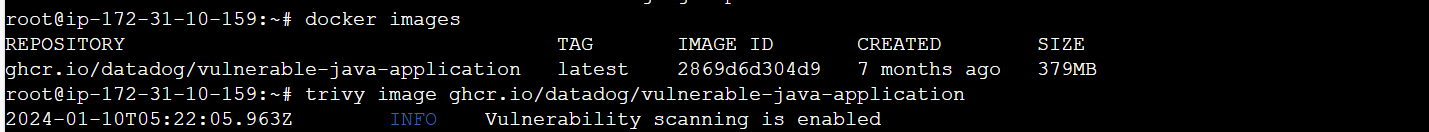
**Using some tools for security analysis on this Java Web Testing** application.

1. **TRIVY**
2. **SNYK**
3. **NUCLEI**
4. **OWASAP ZAPs**

**Observation:**

**TRIVY: Performing TRIVY**

**Command:**



**Output:**



A screenshot of a computer program

Description automatically generated

**By performing this TRIVY, I have got several security issues.**

**Explaining some vulnerabilities with solution:**

Before July 26, 2021**, libfetch, as utilized in apk-tools, xbps,** and similar products, had a vulnerability. Numeric strings mishandling in FTP and HTTP protocols led to an out-of-bounds read due to insufficient checks in the FTP passive mode implementation, where strtol was used to parse numbers. Premature line ending could occur, causing a for-loop to check for the 0 terminator one byte too late.

**Solution:** Address the libfetch vulnerability, update the library to a version released after July 26, 2021. This ensures that numeric string parsing in FTP and HTTP protocols is handled securely, preventing potential out-of-bounds reads and addressing the premature line ending issue. Regularly check for and apply software updates to maintain system security.

A **use-after-free vulnerability in Busybox's awk applet allows a crafted awk pattern** to trigger a denial-of-service condition and potentially execute arbitrary code. Exploiting the flaw in the getvar\_i function can lead to unintended consequences, compromising the stability and security of the system running Busybox.

**Solution:** To mitigate the use-after-free vulnerability in Busybox's awk applet, users should promptly update their software to the latest version provided by the vendor. This helps to address the identified issue, incorporating security patches and preventing potential denial-of-service attacks and code execution risks. Regularly monitoring and applying software updates is crucial for maintaining a secure and resilient system.

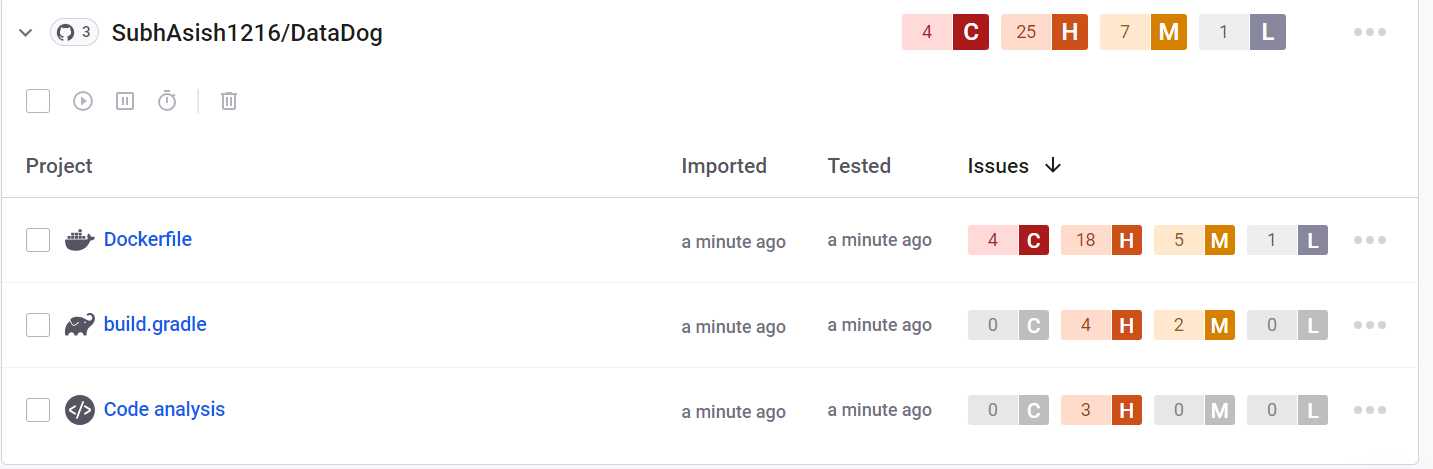
**Pivotal Spring Framework, up to version 5.3.16, is susceptible** to a potential remote code execution risk when used for Java deserialization of untrusted data. The vendor asserts that untrusted data usage is not intended, and the product behavior won't change as some users depend on deserialization of trusted data. Authentication may be necessary, and the impact depends on the specific product's implementation of the library.

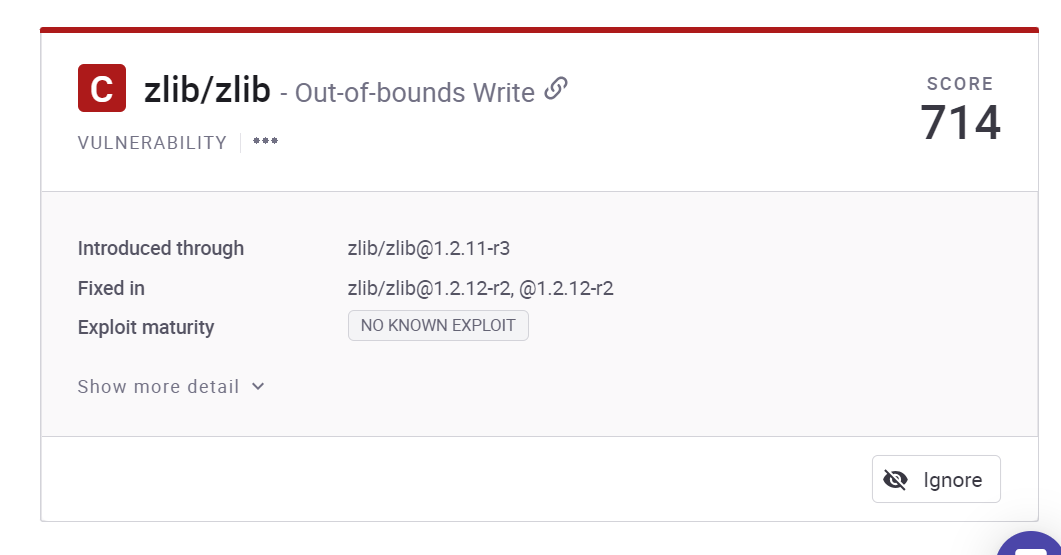
**Solution**: Pivotal Spring Framework, up to version 5.3.16, is susceptible to a potential remote code execution risk when used for Java deserialization of untrusted data. The vendor asserts that untrusted data usage is not intended, and the product behavior won't change as some users depend on deserialization of trusted data. Authentication may be necessary, and the impact depends on the specific product's implementation of the library.

**SnakeYAML's vulnerability in the Constructor() class** allows deserialization of arbitrary types, posing a remote code execution risk when processing attacker-supplied YAML content. To mitigate this, users are advised to employ SnakeYAML's SafeConstructor for parsing untrusted content, restricting deserialization. Upgrading to version 2.0 or later is strongly recommended to ensure the security of YAML processing in applications.

**Solution**: To address the vulnerability in SnakeYAML, users should adopt SnakeYAML's SafeConstructor when parsing untrusted content. This constrains deserialization, mitigating the risk of remote code execution. Additionally, upgrading to SnakeYAML version 2.0 or later is strongly recommended to ensure the implementation of the latest security enhancements and patches. These measures collectively enhance the security posture of applications utilizing SnakeYAML for YAML processing.

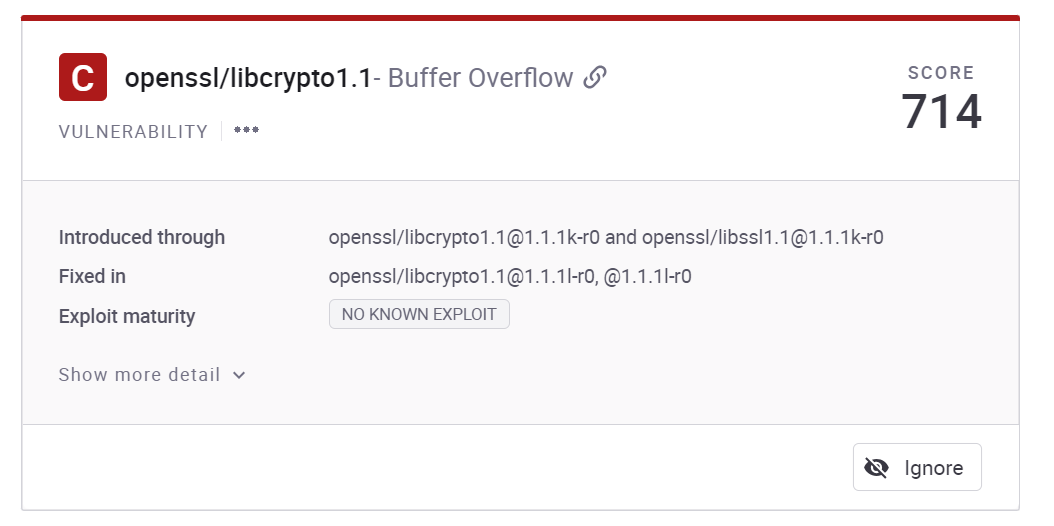
**SNYK: Performing SNYK**





The zlib library is susceptible to an out-of-bounds write vulnerability, potentially triggered during data processing. Exploiting this flaw could lead to unauthorized memory access, enabling attackers to manipulate data beyond the intended boundaries. To mitigate this risk, users are advised to apply relevant patches or updates provided by zlib, enhancing the security of applications utilizing the library for compression and decompression operations.

**Solution**: To address the zlib Out-of-bounds Write vulnerability, users should promptly update the zlib library to the latest version provided by the maintainers. Applying patches or security updates is crucial to fix the identified flaw and enhance the security of applications utilizing zlib for data compression and decompression operations. Regularly monitoring and applying software updates helps ensure a resilient and secure software environment.

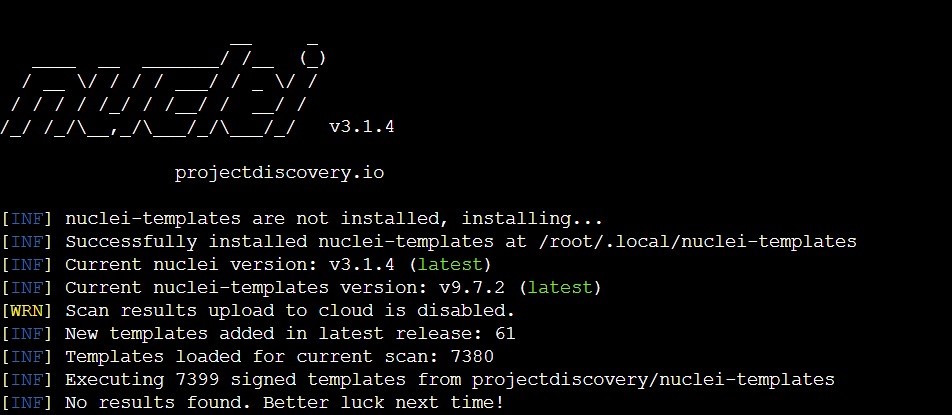


A **Buffer Overflow vulnerability exists in OpenSSL's libcrypto1.1 library**, potentially allowing attackers to overwrite adjacent memory areas. Exploiting this flaw could lead to unauthorized code execution or system crashes. Users are strongly advised to update OpenSSL to the latest version released by the maintainers to mitigate this security risk and ensure the secure operation of applications relying on cryptographic functions provided by libcrypto1.1.

**Solution:** To address the Buffer Overflow vulnerability in OpenSSL's libcrypto1.1, users should promptly update their OpenSSL installation to the latest version provided by the maintainers. Applying patches and security updates is essential for resolving identified vulnerabilities, enhancing the security of cryptographic functions, and preventing potential unauthorized code execution or system instability. Regularly monitoring and updating cryptographic libraries contribute to a more secure software environment.

**NUCLEI: Performing NUCLEI**

After performing HORUSEC, got some issues in it.



A screen shot of a computer program

Description automatically generated

The **"http-missing-security-headers:referrer-policy"** vulnerability refers to the absence of a Referrer-Policy security header in an HTTP response. This header instructs browsers on how to handle and share referrer information when navigating to a different site. The absence of this header may expose sensitive information. To enhance security, websites should implement the Referrer-Policy header with an appropriate directive, specifying how referrer data should be handled.

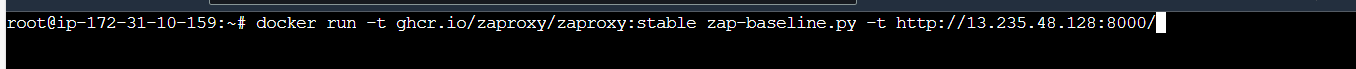
**Solution**: To address the "http-missing-security-headers:referrer-policy" vulnerability, website administrators should implement the Referrer-Policy header in their HTTP responses. This header should be configured with an appropriate directive based on the desired level of security. For example, setting "Referrer-Policy: strict-origin-when-cross-origin" ensures a more secure handling of referrer information. Regularly auditing and updating security headers contributes to a robust web application security posture.

The **"http-missing-security-headers:cross-origin-opener-policy"** vulnerability indicates the absence of the Cross-Origin Opener Policy (COOP) security header in an HTTP response. COOP defines how a document in one browsing context can interact with another. Without this header, websites may be susceptible to security risks. Implementing the COOP header with the appropriate directives enhances web application security by controlling interactions between different browsing contexts.

**Solution**: To address the "http-missing-security-headers:cross-origin-opener-policy" vulnerability, website administrators should implement the Cross-Origin Opener Policy (COOP) security header in their HTTP responses. This header should be configured with the appropriate directives based on the desired security level for cross-origin interactions. Regularly auditing and updating security headers contributes to a more robust web application security posture, mitigating potential security risks associated with cross-origin interactions.

**OWASP ZAP:** Analyzing the application by OWASP ZAP.

**Command:**



By using this command, we are performing zap base line command on our live application.

**Analysis and result:**

A computer screen shot of a black screen

Description automatically generated

The "**Missing Anti-clickjacking Header" issue (ID: 10020) in**dicates the absence of a proper anti-clickjacking header in the HTTP response. Clickjacking involves tricking users into clicking on malicious elements unknowingly. Implementing headers like X-Frame-Options with appropriate settings, such as "DENY" or "SAMEORIGIN," helps prevent unauthorized embedding of a web page within an iframe, mitigating the risk of clickjacking attacks.

Solution: To address the "Missing Anti-clickjacking Header" issue, implement an appropriate anti-clickjacking header in the HTTP response, such as X-Frame-Options with values like "DENY" or "SAMEORIGIN." This helps prevent unauthorized embedding of the web page within iframes, enhancing security and mitigating the risk of clickjacking attacks. Regularly update and audit security headers for robust protection.

The **"X-Content-Type-Options Header Missing" issue (ID: 10021**) indicates that the HTTP response lacks the X-Content-Type-Options header. This header prevents browsers from interpreting files as a different MIME type than declared by the server, reducing the risk of MIME-sniffing attacks. Implementing this header with the value "nosniff" ensures browsers adhere strictly to the declared content type, enhancing security.

Solution: To address the "X-Content-Type-Options Header Missing" issue (ID: 10021), include the X-Content-Type-Options header in the HTTP response with the value "nosniff." This ensures browsers strictly adhere to the declared content type, preventing MIME-sniffing attacks. Regularly audit and update security headers to maintain a robust defense against potential security vulnerabilities.

The "**Content Security Policy (CSP) Header Not Set" issue (ID: 10038**) indicates the absence of a Content Security Policy header in the HTTP response. CSP defines rules for browser interactions, mitigating various types of attacks, including cross-site scripting. Implementing CSP headers helps restrict which resources the browser can load, enhancing web application security by preventing unauthorized script execution and minimizing the impact of potential security threats.

Solution: To address the "Content Security Policy (CSP) Header Not Set" issue (ID: 10038), include a Content Security Policy header in the HTTP response. Define and enforce policies specifying trusted sources for scripts, styles, and other resources. Regularly review and update the CSP directives to adapt to evolving security requirements, ensuring a robust defense against cross-site scripting and related threats in web applications.

The **"Storable and Cacheable Content" issue (ID: 10049) signifies that** certain content is both storable and cacheable. While caching enhances performance, it might expose sensitive information if improperly stored. To resolve, carefully assess the sensitivity of data, implement proper cache-control headers, and use encryption where necessary, ensuring a balance between performance optimization and data security.

**Solution**: To mitigate the "Storable and Cacheable Content" issue (ID: 10049), carefully assess data sensitivity. Implement appropriate cache-control headers, ensuring that sensitive information is not stored or cached inadvertently. Utilize encryption for sensitive content. Regularly review and update caching policies to align with security requirements, achieving a balance between performance optimization and safeguarding sensitive data.