PHASE 2: DATA COLLECTION & ANALYSIS

1. System/Network Reconnaissance

   Given the highly secure network setup in the PNexus Web Application environment, a full network reconnaissance was not applicable. However, a local application-level reconnaissance scan was conducted using ZAP to assess the security posture of the system. The scan targeted the application on http://127.0.0.1, simulating real-world attack scenarios and attack surfaces that could be exposed through the web application. This approach ensured a comprehensive identification of web-based vulnerabilities, despite the limited scope of the assessment at the network level.

   The local reconnaissance scan focused on evaluating potential security risks associated with the application infrastructure. While the network itself was secured via the GlobalProtect VPN, which restricts access to authenticated users and adds a layer of security, it was still important to analyze the web application for vulnerabilities that could be exploited even when protected by the VPN.

   The PNexus Web Application, developed using CodeIgniter 3 on the back-end, was assessed for potential exposure to SQL injection vulnerabilities, due to the lack of built-in ORM support. Additionally, the front-end components, including jQuery, HTML5, and Bootstrap 4, were evaluated for common vulnerabilities like cross-site scripting (XSS) and cross-site request forgery (CSRF), especially since the application lacks proper configuration to mitigate these risks.

   On the security mechanisms such as CAPTCHA and Two-Factor Authentication (2FA) were not implemented, which significantly increases the risk of bot attacks and account compromises. Although the system integrates with Active Directory for user authentication, the integration with the web application was also assessed for potential weaknesses. Given that the application operates over HTTP, an insecure protocol, this raised concerns about the integrity and confidentiality of the data transmitted.

   The scan results indicated that while the network setup itself provided strong protection, the application's vulnerabilities, including lack of encryption, insecure authentication mechanisms, and missing security headers, were clear areas for immediate improvement.

   Note: The scan was performed in a development or staging environment (not production) to evaluate the application's web security posture in isolation from the network-level security mechanisms.

2. Use of Security Tools for Data Gathering

A. Application-Level Vulnerabilities

A security assessment was conducted using OWASP ZAP (Zed Attack Proxy) v2.16.1 as the primary tool for data gathering. ZAP was configured to perform both passive and active scans on the application running in an isolated server environment. The scanning process was designed to identify potential vulnerabilities across various endpoints of the application. This comprehensive scan included a detailed analysis of HTTP headers, JavaScript libraries, form handling mechanisms, hidden files, session and cookie management, and response metadata. The scan specifically targeted key application endpoints, such as login forms, APIs, and session management features. Several vulnerabilities were detected during the scan, and they were subsequently categorized based on their severity level.

B. Deployment Environment Vulnerabilities

Wireshark was used to analyze the network traffic in the deployment environment. It helped identify vulnerabilities such as sensitive data being transmitted without encryption that could be intercepted. It also detected the use of outdated protocols like old SSL/TLS versions, which are insecure.

NMAP (Network Mapper) was employed to conduct a thorough network discovery and vulnerability assessment of the deployment environment. NMAP was used to map out open ports, services, and active hosts within the network. By performing a variety of scans, including TCP Connect, SYN Scan, and OS detection, NMAP provided valuable insights into the network's exposed attack surface.

3. Risk and Vulnerability Analysis

A. Infrastructure Level Analysis

The deployment environment of the PNexus Web Application was also assessed to identify potential infrastructure-level risks. This evaluation focused on configuration weaknesses, absence of recommended production-grade practices, and the overall security posture of the application hosting environment.

#	Vulnerability	Description	Risk Level	Impact	Recommendation
1	CVE-2024-38476 (Apache HTTPD)	Apache HTTPD 2.4.58 vulnerable to remote code execution and other attacks.	Critical	Potential for remote code execution and service disruption if exploited.	Update Apache HTTPD to the latest stable version. Apply available security patches.

2	CVE-2024-38474 (Apache HTTPD)	Apache HTTPD 2.4.58 vulnerable to denial-of-service (DoS) attacks.	High	Could lead to service outages and disruption in server availability.	Update Apache HTTPD to the latest stable version. Ensure that DoS protections are implemented.
3	Slowloris DoS Vulnerability	Apache HTTPD vulnerable to Slowloris Denial of Service attack.	High	Can cause a server crash or service denial by exhausting server resources.	Configure the server to mitigate Slowloris attacks (e.g., increase connection timeouts, use a WAF).
4	CVE-2024-38475 (Apache HTTPD)	Apache HTTPD 2.4.58 vulnerable to privilege escalation.	High	Attackers may escalate privileges and gain unauthorized access to the server.	Update Apache HTTPD to the latest stable version.
5	CVE-2024-39573 (Apache HTTPD)	Apache HTTPD vulnerable to multiple exploits affecting version 2.4.58.	High	May lead to data leakage, unauthorized access, and potential privilege escalation.	Apply all patches available for Apache HTTPD and review security configurations.
6	CVE-2024-40898 (Apache HTTPD)	Vulnerability allowing DoS or remote code execution in Apache HTTPD.	High	Server could be taken offline or used to perform malicious activities.	Update Apache HTTPD and verify any related security configurations.
7	CVE-2024-27316 (Apache HTTPD)	Apache HTTPD vulnerability related to improper input validation.	High	Exploitation could lead to application-level vulnerabilities such as arbitrary code execution.	Update Apache HTTPD and implement strong input validation mechanisms.
8	CVE-2007-6750 (Slowloris DOS attack)	Potential for Slowloris DoS attack, which can exhaust HTTP server resources by keeping connections open.	Medium	Affects server availability and may lead to service denial.	Implement mitigation techniques like rate-limiting, connection timeouts, or using a WAF to block slow requests.
9	CVE-2017-1001000 (Grafana)	Unauthenticated access vulnerability in Grafana (if accessible publicly).	High	Unauthorized users may gain access to sensitive data or administrative functions.	Ensure Grafana is properly secured behind authentication and only exposed to trusted networks.
10	CVE-2010-0738 (JMX Console)	Authentication vulnerability in the JMX console, allowing unauthorized access.	High	Unauthorized users may gain access to management functions of the Java server.	Secure JMX console access by enforcing strong authentication and limiting exposure.
11	PostgreSQL Default Version (9.6.0 or later)	Potential vulnerabilities in older PostgreSQL versions.	Medium	Potential for database compromise if vulnerabilities exist in the version being used.	Ensure that PostgreSQL is updated to the latest stable version. Implement strong database access control.
12	MSRPC (Port 135)	Microsoft RPC service exposed, vulnerable to remote exploits.	Critical	Can allow attackers to exploit known vulnerabilities and gain unauthorized access to systems.	Disable MSRPC if not required. If necessary, restrict access to trusted IPs and patch vulnerabilities.

13	RDP (Port 3389)	Remote Desktop Protocol exposed, vulnerable to brute force and other attacks.	Critical	Attackers can exploit RDP for unauthorized access to the host machine.	Disable RDP if not needed. Use strong passwords and multi-factor authentication if RDP must be enabled.
14	Microsoft DS (Port 445)	SMBv1 vulnerabilities on open Microsoft DS port, potentially exposing the system to WannaCry-style exploits.	Critical	Remote code execution and full system compromise can occur if SMB vulnerabilities are exploited.	Disable SMBv1, restrict access to port 445, and apply security patches to mitigate SMB vulnerabilities.

The table helps prioritize remediation efforts, starting with the most critical vulnerabilities (such as RDP and SMB issues) that require urgent attention to prevent potential exploitation. It provides a structured way of assessing and addressing the security weaknesses found in the system during the scan.

B. Application-Level Analysis

Figure 1: the number of alerts for each level of risk and confidence

		Confidence			
		High	Medium	Low	Total
Risk	High	0 (0.0%)	1 (6.2%)	0 (0.0%)	1 (6.2%)
	Medium	1 (6.2%)	2 (12.5%)	2 (12.5%)	5 (31.2%)
	Low	1 (6.2%)	5 (31.2%)	1 (6.2%)	7 (43.8%)
	Informational	0 (0.0%)	3 (18.8%)	0 (0.0%)	3 (18.8%)
	Total	2 (12.5%)	11 (68.8%)	3 (18.8%)	16 (100%)

The table provides a distribution of alerts categorized by Risk and Confidence levels. It shows the number of alerts for each combination of these two categories and represents the proportion of each combination in relation to the total alerts. The data helps to analyze where the system has uncertainty in its risk assessments and how confident it is in identifying those risks.

A key observation is that Low Confidence is prevalent across all risk categories. Of the total 16 alerts, 11 (68.8%) are classified as Low Confidence. This suggests that the system is uncertain about the risk levels of most alerts, particularly for Low Risk and Medium Risk categories. This could indicate that the alerts in these categories are harder to classify with high certainty, which may point to a need for further refinement in the risk-assessment algorithms or additional data for better accuracy.

The High Risk category, with only 1 alert (6.2%) in the entire dataset. Interestingly, there are no High Confidence or User Confirmed High Risk alerts, further highlighting the uncertainty around high-risk alerts. This suggests that either high-risk scenarios are less frequent, or the system struggles to classify high-risk situations with confidence, making them harder to verify.

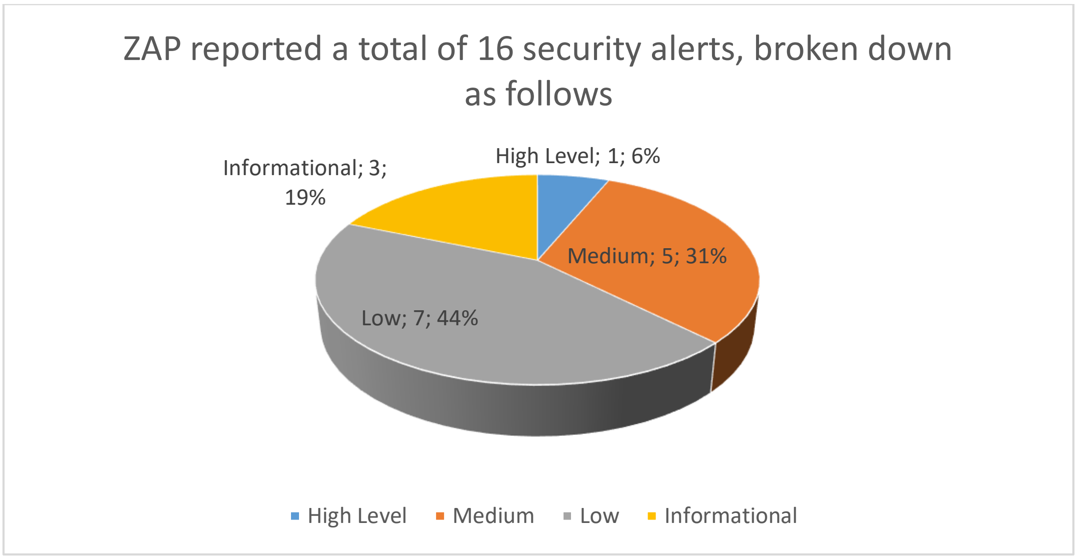
Both the Medium Risk and Low Risk categories have similar distributions. While the Medium Risk category has a mix of Low Confidence (12.5%) and User Confirmed (12.5%) alerts, the Low Risk category contains a larger portion of Low Confidence alerts (31.2%). Despite this, Low Risk still represents the largest number of alerts in the dataset (7 alerts or 43.8%). This indicates that the system frequently identifies low-risk situations but often with low certainty, potentially due to the nature of the data or the complexity of the risk classification process.

Finally, Informational alerts make up 18.8% of the total, with all three of them categorized under Low Confidence. Informational alerts typically don’t indicate an immediate risk but might provide valuable context or background information. The low confidence in these alerts suggests that they may be harder to categorize or verify, yet they still represent a meaningful portion of the overall alert system.

In summary, the table shows that Low Confidence is a dominant factor across the alert categories, particularly in Low Risk and Medium Risk situations. This shows a potential area for improvement in the alert system’s confidence level. It might be beneficial to further assess and refine the risk classification algorithms to increase the system's certainty in identifying and verifying risks. Additionally, investigating the cause of low-confidence alerts, especially for Low Risk items, could help improve the system’s overall performance.

Note: The percentages in brackets represent the count as a percentage of the total number of alerts included in the report, rounded to one decimal place.

Figure 2: ZAP reported a total of 16 security alerts, broken down as follows:



The security scan of the PNexus Web Application, conducted using ZAP, identified a total of 16 vulnerabilities, which span various levels of severity and confidence. These vulnerabilities are categorized into high, medium, low, and informational risks, as outlined below:

Figure 3: Vulnerabilities classified as high, medium, low, or informational risks.

Risk Level	Count	Description	Recommendation
High	1	- Vulnerable JavaScript Library Detected (File: moment.min.js) - Risk: Contains known security issues that may be exploited by attackers. - Info: The use of an outdated version of the moment.js library exposes the application to known security vulnerabilities that could potentially be exploited by attackers.	It is recommended to replace this library with the latest, secure version to mitigate the risk.
Medium	5	• Missing Content Security Policy (CSP) Header: A missing CSP header can allow attackers to inject malicious content into the application. Implementing a CSP header will mitigate the risk of XSS attacks by restricting the sources from which content can be loaded. • Missing Anti-clickjacking Header (X-Frame-Options): The absence of the X-Frame-Options header leaves the application vulnerable to clickjacking attacks, where an attacker could trick users into clicking on hidden or invisible buttons. Setting this header to "DENY" or "SAMEORIGIN" will prevent such attacks. • Missing Anti-CSRF Tokens: The lack of anti-CSRF tokens in the application increases the risk of cross-site request forgery (CSRF)	Implement the missing security headers and anti-CSRF tokens. Investigate and address the hidden file and outdated library.

		attacks, where an attacker can trick authenticated users into making unwanted requests. Implementing anti-CSRF tokens will protect against these types of attacks. • Hidden File Detected (.hg): The detection of a hidden file (.hg) suggests that version control data might be exposed to unauthorized access. It is essential to remove any unnecessary or sensitive files from the public directory. • Outdated JavaScript Library (bootstrap.bundle.min.js): An outdated version of Bootstrap exposes the application to potential vulnerabilities. Updating to the latest version of the library will ensure that known security issues are addressed	
Low	7	• Information Disclosure via Headers and Timestamps: The application is disclosing potentially sensitive information, such as server version numbers, in HTTP headers and timestamps. Removing this information will make it harder for attackers to identify the underlying server technology. • Debug/Error Messages Visible in Responses: Debugging or error messages exposed in the application responses could provide attackers with information about the system that can be exploited. These messages should be suppressed in the production environment. • Cross-domain Script Inclusion: Cross-domain script inclusion vulnerabilities could allow an attacker to inject malicious scripts into the application. Review and secure the implementation to prevent cross-origin resource sharing (CORS) issues.	Review and remove sensitive information in headers and responses. Investigate and address the cross-domain script inclusion.
Informational	3	• Suspicious Comments Found in Source Code: The source code contains suspicious comments that could provide useful information to potential attackers. These comments should be removed or sanitized to prevent leakage of sensitive information. • Modern Web Application Detected: The application uses modern web technologies, which may require periodic review to ensure they adhere to current security best practices. Although this finding is informational, ongoing assessment of emerging threats is recommended.	Review and remove any sensitive or unnecessary comments. (No specific action needed for the other informational findings).

		• Session Management Behavior Identified: The application's session management behavior was identified, which could potentially be optimized to ensure secure handling of user sessions. No immediate action is required unless further vulnerabilities in session management are discovered.	

4. Comparison with Industry Security Standards

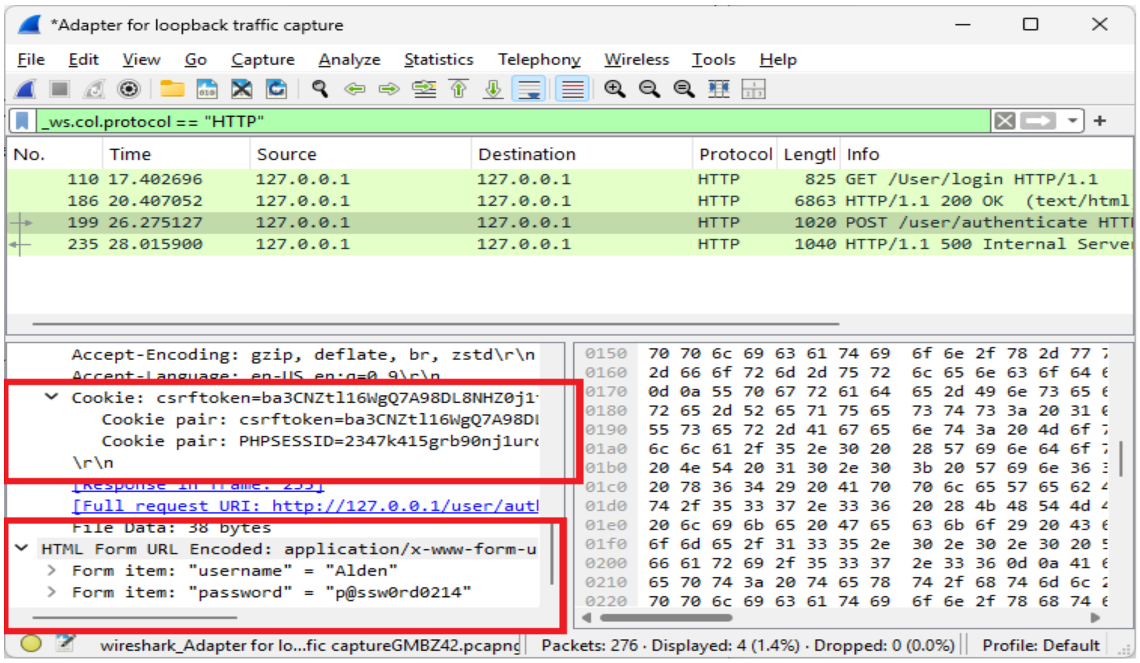
A. Infrastructure Level

The table provides a detailed risk assessment for vulnerabilities found on a system through an Nmap scan. It lists 14 vulnerabilities related to different services like Apache HTTPD, Microsoft RPC, and PostgreSQL.

#	Vulnerability	Description	Industry Security Standard
1	**CVE-2024-38476 (Apache HTTPD)**	Remote code execution vulnerability in Apache HTTPD 2.4.58.	**NIST SP 800-53 (System and Communications Protection - SC)**: Ensures secure configuration and patch management practices to prevent exploits like these.
2	**CVE-2024-38474 (Apache HTTPD)**	Remote code execution vulnerability in Apache HTTPD 2.4.58.	**OWASP Top 10 (A9 - Using Components with Known Vulnerabilities)**: Using vulnerable components like outdated versions of HTTPD increases security risks.
3	**CVE-2024-38475 (Apache HTTPD)**	Remote code execution vulnerability in Apache HTTPD 2.4.58.	**NIST SP 800-53 (Access Control - AC)**: Ensure strong access controls and update systems promptly to mitigate risks from known vulnerabilities.
4	**Slowloris DOS (Port 80)**	**Slowloris** attack can be used to keep many HTTP connections open and starve the server of resources, causing a Denial of Service (DoS).	**ISO/IEC 27001:2022 (A.14.2.5 - Protection from Malicious Code)**: Ensure effective protection against DoS attacks by implementing server resource limitations.
5	**MSRPC (Port 135)**	Microsoft RPC service exposed to the network, commonly exploited in remote attacks.	**NIST SP 800-53 (System and Communications Protection - SC)**: Ensure proper network segmentation and access control to prevent unauthorized access.
6	**CVE-2024-40898 (Apache HTTPD)**	Vulnerability in Apache HTTPD 2.4.58 that could allow for remote code execution.	**CIS Control 3.1**: Regularly apply patches to mitigate risks associated with known vulnerabilities.
7	**CVE-2024-39573 (Apache HTTPD)**	Vulnerability in Apache HTTPD 2.4.58.	**ISO/IEC 27001:2022 (A.12.6.1 - Management of Technical Vulnerabilities)**: Implement a patch

			management process to minimize exposure to critical vulnerabilities.
8	**CVE-2024-38477 (Apache HTTPD)**	Vulnerability in Apache HTTPD 2.4.58 that could allow an attacker to execute arbitrary code.	**NIST SP 800-53 (Risk Assessment - RA)**: Assess and mitigate vulnerabilities in all deployed services and software.
9	**CVE-2024-27316 (Apache HTTPD)**	Vulnerability in Apache HTTPD 2.4.58 that could allow a remote attacker to execute arbitrary code.	**CIS Control 4.8**: Regular patching of software components to minimize risk from known vulnerabilities.
10	**CVE-2024-38476, CVE-2024-38475 (Apache HTTPD)**	Multiple vulnerabilities identified in Apache HTTPD 2.4.58.	**ISO/IEC 27001:2022 (A.10.1.1 - Cryptographic Controls)**: Apply encryption and security patches to mitigate the impact of vulnerabilities like these.
11	**RDP (Port 3389)**	Remote Desktop Protocol exposed, vulnerable to brute-force and credential stuffing attacks.	**OWASP Top 10 (A8 - Insecure Deserialization)**: Ensure secure access to sensitive services like RDP, with strong authentication mechanisms.
12	**PostgreSQL DB (Port 5432)**	Exposed **PostgreSQL** database potentially allowing unauthorized access.	**NIST SP 800-53 (Access Control - AC)**: Implement strict access controls for databases and sensitive services.
13	**Grafana (Port 3000)**	Exposed Grafana service without proper authentication.	**OWASP Top 10 (A2 - Broken Authentication)**: Proper authentication is required for accessing sensitive administrative interfaces like Grafana.
14	**TRACE Enabled (Port 80)**	The **TRACE** HTTP method is enabled, potentially exposing sensitive data and increasing vulnerability to Cross-Site Tracing (XST) attacks.	**OWASP Top 10 (A5 - Security Misconfiguration)**: Misconfiguration of web services like HTTP methods creates unnecessary attack vectors.

Figure 5: Screenshot showing unencrypted packet of user credential.


*Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
_ws.col.protocol == "HTTP"

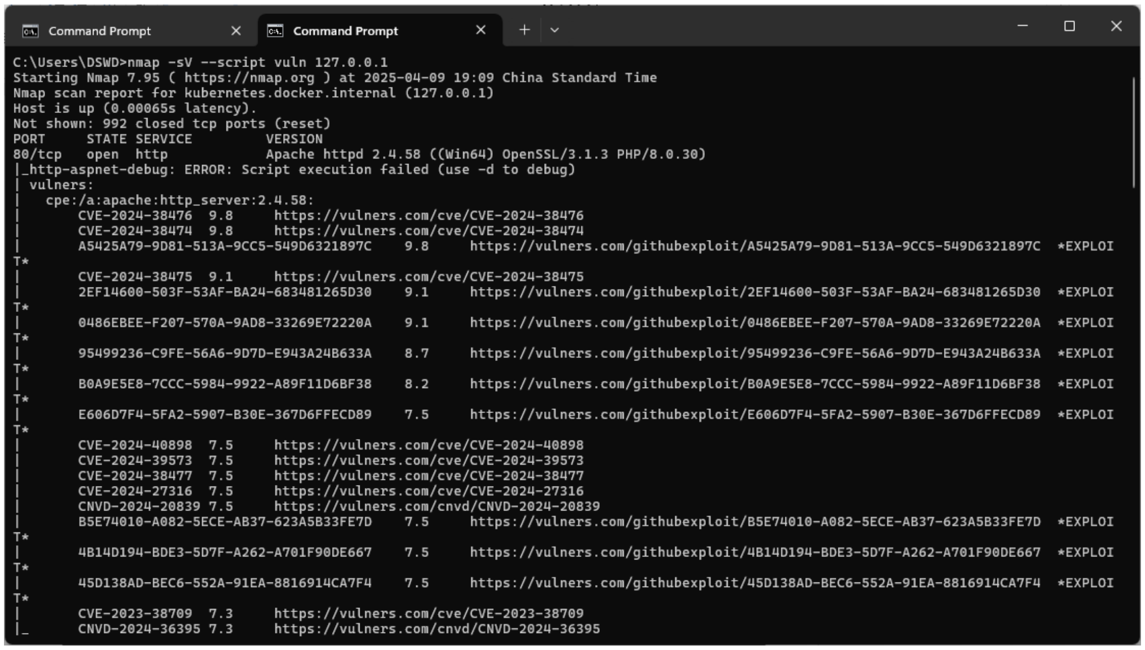
```
No.    Time         Source       Destination   Protocol Lengtl Info
   110 17.402696    127.0.0.1    127.0.0.1     HTTP      825 GET /User/login HTTP/1.1
   186 20.407052    127.0.0.1    127.0.0.1     HTTP     6863 HTTP/1.1 200 OK  (text/html
   199 26.275127    127.0.0.1    127.0.0.1     HTTP     1020 POST /user/authenticate HTT
   235 28.015900    127.0.0.1    127.0.0.1     HTTP     1040 HTTP/1.1 500 Internal Serve
```

```
    Accept-Encoding: gzip, deflate, br, zstd\r\n
    Accept-Language: en-US en;q=0.9\r\n
  ⌄ Cookie: csrftoken=ba3CNZtl16WgQ7A98DL8NHZ0j1
      Cookie pair: csrftoken=ba3CNZtl16WgQ7A98DI
      Cookie pair: PHPSESSID=2347k415grb90nj1ur
    \r\n
    [Response in frame: 235]
    [Full request URI: http://127.0.0.1/user/aut
    File Data: 38 bytes
⌄ HTML Form URL Encoded: application/x-www-form-u
  > Form item: "username" = "Alden"
  > Form item: "password" = "p@ssw0rd0214"
```

```
0150  70 70 6c 69 63 61 74 69  6f 6e 2f 78 2d 77
0160  2d 66 6f 72 6d 2d 75 72  6c 65 6e 63 6f 64
0170  0d 0a 55 70 67 72 61 64  65 2d 49 6e 73 65
0180  72 65 2d 52 65 71 75 65  73 74 73 3a 20 31
0190  55 73 65 72 2d 41 67 65  6e 74 3a 20 4d 6f
01a0  6c 6c 61 2f 35 2e 30 20  28 57 69 6e 64 6f
01b0  20 4e 54 20 31 30 2e 30  3b 20 57 69 6e 36
01c0  20 78 36 34 29 20 41 70  70 6c 65 57 65 62
01d0  74 2f 35 33 37 2e 33 36  20 28 4b 48 54 4d
01e0  20 6c 69 6b 65 20 47 65  63 6b 6f 29 20 43
01f0  6f 6d 65 2f 31 33 35 2e  30 2e 30 2e 30 20
0200  66 61 72 69 2f 35 33 37  2e 33 36 0d 0a 41
0210  65 70 74 3a 20 74 65 78  74 2f 68 74 6d 6c
0220  70 70 6c 69 63 61 74 69  6f 6e 2f 78 68 74
```

```
wireshark_Adapter for lo...fic captureGMBZ42.pcapng | Packets: 276 · Displayed: 4 (1.4%) · Dropped: 0 (0.0%) | Profile: Default
```

Figure 6: Screenshot of server vulnerability scan using NMAP



```
C:\Users\DSWD>nmap -sV --script vuln 127.0.0.1
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-09 19:09 China Standard Time
Nmap scan report for kubernetes.docker.internal (127.0.0.1)
Host is up (0.00065s latency).
Not shown: 992 closed tcp ports (reset)
PORT      STATE SERVICE         VERSION
80/tcp    open  http            Apache httpd 2.4.58 ((Win64) OpenSSL/3.1.3 PHP/8.0.30)
|_http-aspnet-debug: ERROR: Script execution failed (use -d to debug)
| vulners:
|   cpe:/a:apache:http_server:2.4.58:
|       CVE-2024-38476  9.8     https://vulners.com/cve/CVE-2024-38476
|       CVE-2024-38474  9.8     https://vulners.com/cve/CVE-2024-38474
|       A5425A79-9D81-513A-9CC5-549D6321897C    9.8     https://vulners.com/githubexploit/A5425A79-9D81-513A-9CC5-549D6321897C  *EXPLOI
T*
|       CVE-2024-38475  9.1     https://vulners.com/cve/CVE-2024-38475
|       2EF14600-503F-53AF-BA24-683481265D30    9.1     https://vulners.com/githubexploit/2EF14600-503F-53AF-BA24-683481265D30  *EXPLOI
T*
|       0486EBEE-F207-570A-9AD8-33269E72220A    9.1     https://vulners.com/githubexploit/0486EBEE-F207-570A-9AD8-33269E72220A  *EXPLOI
T*
|       95499236-C9FE-56A6-9D7D-E943A24B633A    8.7     https://vulners.com/githubexploit/95499236-C9FE-56A6-9D7D-E943A24B633A  *EXPLOI
T*
|       B0A9E5E8-7CCC-5984-9922-A89F11D6BF38    8.2     https://vulners.com/githubexploit/B0A9E5E8-7CCC-5984-9922-A89F11D6BF38  *EXPLOI
T*
|       E606D7F4-5FA2-5907-B30E-367D6FFECD89    7.5     https://vulners.com/githubexploit/E606D7F4-5FA2-5907-B30E-367D6FFECD89  *EXPLOI
T*
|       CVE-2024-40898  7.5     https://vulners.com/cve/CVE-2024-40898
|       CVE-2024-39573  7.5     https://vulners.com/cve/CVE-2024-39573
|       CVE-2024-38477  7.5     https://vulners.com/cve/CVE-2024-38477
|       CVE-2024-27316  7.5     https://vulners.com/cve/CVE-2024-27316
|       CNVD-2024-20839 7.5     https://vulners.com/cnvd/CNVD-2024-20839
|       B5E74010-A082-5ECE-AB37-623A5B33FE7D    7.5     https://vulners.com/githubexploit/B5E74010-A082-5ECE-AB37-623A5B33FE7D  *EXPLOI
T*
|       4B14D194-BDE3-5D7F-A262-A701F90DE667    7.5     https://vulners.com/githubexploit/4B14D194-BDE3-5D7F-A262-A701F90DE667  *EXPLOI
T*
|       45D138AD-BEC6-552A-91EA-8816914CA7F4    7.5     https://vulners.com/githubexploit/45D138AD-BEC6-552A-91EA-8816914CA7F4  *EXPLOI
T*
|       CVE-2023-38709  7.3     https://vulners.com/cve/CVE-2023-38709
|_      CNVD-2024-36395 7.3     https://vulners.com/cnvd/CNVD-2024-36395
```

B. Application Level

The identified vulnerabilities were assessed in relation to widely recognized cybersecurity frameworks and standards to determine their severity, prevalence, and recommended remediation strategies. Notably, many findings align with critical areas outlined in the OWASP Top 10 (2021), Common Weakness Enumeration (CWE), and Web Application Security Consortium (WASC) Threat Classification.

One of the most critical findings—the use of vulnerable and outdated JavaScript libraries such as moment.min.js and bootstrap.bundle.min.js—falls under OWASP A06:2021 –

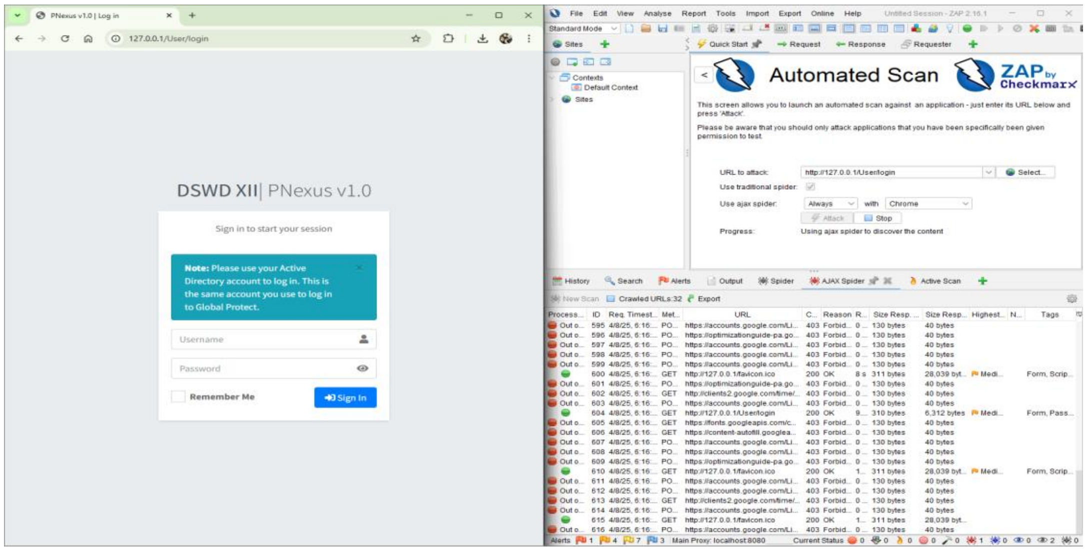
Vulnerable and Outdated Components. Using known-vulnerable components in a production environment can lead to severe exploits, including remote code execution or cross-site scripting. This issue also maps to CWE-1104: Use of Unmaintained Third-Party Components.

The table below shows how the findings relate to industry security standards:

Vulnerability	Related OWASP Top 10	CWE	WASC
Vulnerable JS Library	A06: Vulnerable Components	CWE-1104	WASC-20: Improper Input Handling
Missing CSP Header	A05: Security Misconfiguration	CWE-693	WASC-15: Application Misconfiguration
Missing X-Frame-Options	A05: Security Misconfiguration	CWE-1021	WASC-15
Missing Anti-CSRF Tokens	A01: Broken Access Control	CWE-352: Cross-Site Request Forgery	WASC-9: CSRF
Hidden .hg Directory	A05: Security Misconfiguration	CWE-200: Exposure of Sensitive Info	WASC-13: Info Leakage
Suspicious Comments/Debug Info	A03: Injection / A05: Misconfiguration	CWE-615: Debug Info Exposure	WASC-13

The analysis clearly shows that multiple findings correspond to major categories of application vulnerabilities commonly exploited in real-world attacks. These gaps not only pose technical risks but also reduce compliance with standards such as ISO 27001:2013 (Annex A.14 – System Acquisition, Development, and Maintenance) and NIST SP 800-53 (System and Information Integrity SI-10, SI-2). Aligning the system with industry standards requires addressing these vulnerabilities through component updates, HTTP header configuration, and secure development lifecycle (SDLC) improvements. Regular vulnerability assessments and automated dependency checks should be integrated into the development and deployment pipeline to maintain compliance and reduce risk exposure over time.

Figure 6: Screenshot showing automated vulnerability scan using Owasp Zap



DSWD XII| PNexus v1.0

Sign in to start your session

Note: Please use your Active Directory account to log in. This is the same account you use to log in to Global Protect.

Username

Password

Remember Me

Sign In

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://127.0.0.1/User/login

Use traditional spider: ✓

Use ajax spider: Always with Chrome

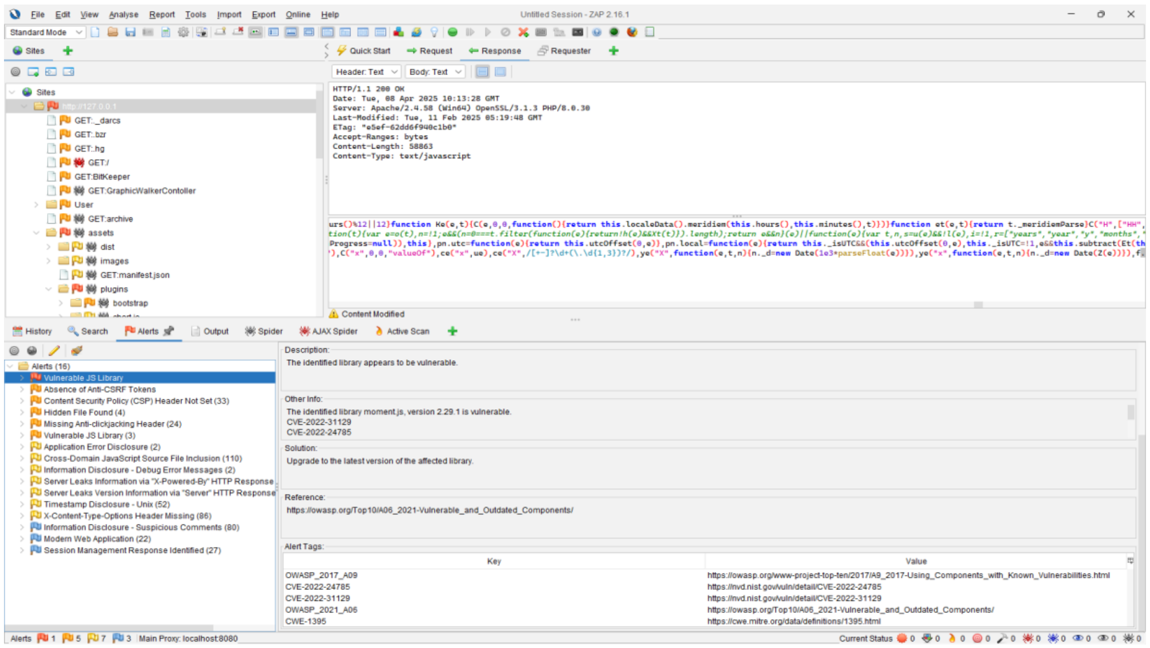
Progress: Using ajax spider to discover the content

Crawled URLs:32

```
Process...  ID   Req. Timest...  Met...  URL                                 C...  Reason     R...  Size Resp. ...  Size Resp...  Highest...  N...  Tags
Out o...    595  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    596  4/8/25, 6:16:...  PO...   https://optimizationguide-pa.go...  403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    597  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    598  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    599  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
            600  4/8/25, 6:16:...  GET     http://127.0.0.1/favicon.ico        200   OK         8 s    311 bytes       28,039 byt...  Medi...           Form, Scrip...
Out o...    601  4/8/25, 6:16:...  PO...   https://optimizationguide-pa.go...  403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    602  4/8/25, 6:16:...  GET     http://clients2.google.com/time/... 403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    603  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
            604  4/8/25, 6:16:...  GET     http://127.0.0.1/User/login         200   OK         9...   310 bytes       6,312 bytes    Medi...           Form, Pass...
Out o...    605  4/8/25, 6:16:...  GET     https://fonts.googleapis.com/c...   403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    606  4/8/25, 6:16:...  GET     https://content-autofill.googlea... 403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    607  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    608  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    609  4/8/25, 6:16:...  PO...   https://optimizationguide-pa.go...  403   Forbid...  0 ...  130 bytes       40 bytes
            610  4/8/25, 6:16:...  GET     http://127.0.0.1/favicon.ico        200   OK         1...   311 bytes       28,039 byt...  Medi...           Form, Scrip...
Out o...    611  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    612  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    613  4/8/25, 6:16:...  GET     http://clients2.google.com/time/... 403   Forbid...  0 ...  130 bytes       40 bytes
Out o...    614  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
            615  4/8/25, 6:16:...  GET     http://127.0.0.1/favicon.ico        200   OK         1...   311 bytes       28,039 byt...
Out o...    616  4/8/25, 6:16:...  PO...   https://accounts.google.com/Li...   403   Forbid...  0 ...  130 bytes       40 bytes
```

Alerts 1 4 7 3 Main Proxy: localhost:8080 Current Status 0 0 0 0 0 1 0 0 2 0

Figure 7: Screenshot showing high-risk vulnerability assessment result.



Untitled Session - ZAP 2.16.1

```
HTTP/1.1 200 OK
Date: Tue, 08 Apr 2025 10:13:28 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30
Last-Modified: Tue, 11 Feb 2025 05:19:48 GMT
ETag: "e5ef-62dd6f940c1b0"
Accept-Ranges: bytes
Content-Length: 58863
Content-Type: text/javascript
```

```
urs()%12||12}function Ke(e,t){C(e,0,0,function(){return this.localeData().meridiem(this.hours(),this.minutes(),t)})}function et(e,t){return t._meridiemParse}C("H",["HH",
tion(t){var e=o(t),n=!1;e&&(n=0===t.filter(function(e){return!h(e)&&Xt(t)}).length);return e&&n}(e)||function(e){var t,n,s=u(e)&&!l(e),i=!1,r=["years","year","y","months",
Progress=null)),this},pn.utc=function(e){return this.utcOffset(0,e)},pn.local=function(e){return this._isUTC&&(this.utcOffset(0,e),this._isUTC=!1,e&&this.subtract(Et(th
'),C("x",0,0,"valueOf"),ce("x",ue),ce("X",/[+-]?\d+(\.\d{1,3})?/),ye("X",function(e,t,n){n._d=new Date(1e3*parseFloat(e))}),ye("x",function(e,t,n){n._d=new Date(Z(e))}),f
```

Content Modified

Sites
- http://127.0.0.1
  - GET:_darcs
  - GET:.bzr
  - GET:.hg
  - GET:/
  - GET:BitKeeper
  - GET:GraphicWalkerContoller
  - User
  - GET:archive
  - assets
    - dist
    - images
    - GET:manifest.json
    - plugins
      - bootstrap

Alerts (16)
- Vulnerable JS Library
- Absence of Anti-CSRF Tokens
- Content Security Policy (CSP) Header Not Set (33)
- Hidden File Found (4)
- Missing Anti-clickjacking Header (24)
- Vulnerable JS Library (3)
- Application Error Disclosure (2)
- Cross-Domain JavaScript Source File Inclusion (110)
- Information Disclosure - Debug Error Messages (2)
- Server Leaks Information via "X-Powered-By" HTTP Response
- Server Leaks Version Information via "Server" HTTP Response
- Timestamp Disclosure - Unix (52)
- X-Content-Type-Options Header Missing (86)
- Information Disclosure - Suspicious Comments (80)
- Modern Web Application (22)
- Session Management Response Identified (27)

Description:
The identified library appears to be vulnerable.

Other Info:
The identified library moment.js, version 2.29.1 is vulnerable.
CVE-2022-31129
CVE-2022-24785

Solution:
Upgrade to the latest version of the affected library.

Reference:
https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

Alert Tags:

Key	Value
OWASP_2017_A09	https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities.html
CVE-2022-24785	https://nvd.nist.gov/vuln/detail/CVE-2022-24785
CVE-2022-31129	https://nvd.nist.gov/vuln/detail/CVE-2022-31129
OWASP_2021_A06	https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/
CWE-1395	https://cwe.mitre.org/data/definitions/1395.html

Alerts 1 5 7 3 Main Proxy: localhost:8080 Current Status 0 0 0 0 0 0 0 0 0 0