## INTRODUCTION AND PROJECT OVERVIEW

Wordpress is a content management and publishing system used to publish and update websites. It accounts for approximately 43% of the websites in existence on the public internet [1]. It is estimated that sites using Wordpress are attacked 4.7 million times a year [2, p. 22] with a vast majority of these attacks resulting from human error during configuration of the site.

Amazon Web Services is a cloud platform that allows users to provision and scale computing resources such as web servers without owning any hardware [3]. It can be used to deploy compute resources that host Wordpress websites. These are relatively insecure by default both from an infrastructure and web perspective.

This project seeks to:

- configure computing resources on a cloud platform,
- host and secure a Wordpress website and,
- evaluate the security of the website and its host.

Fig. 1 below shows an architecture of the technology stack of the project assets:

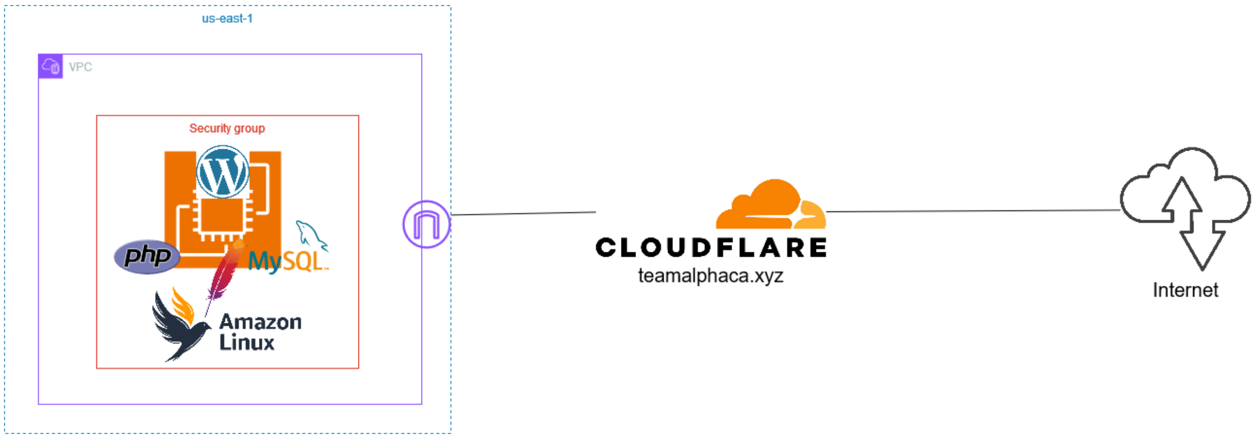


Fig. 1. Architecture of the technology stack used for the project.

The architecture shown above details the technology stack that was used for the project. The web server was an Amazon Web Services Elastic Compute instance (EC2) with eight gigabytes of memory and thirty gigabytes of storage. The LAMP stack (Linux, Apache, MySQL, PHP) was utilized for the web server with Cloudflare used as a DNS provider and

Content Delivery Network. The remaining sections of this report will detail efforts to evaluate the security measures deployed on the EC2 instance and Wordpress website.

## APPROACH AND PROJECT PLANNING

A defence-in-depth approach was used for the project, combining more than one strategy for securing the assets to create redundancy while maintaining the highest level of security and performance possible. Table 1 below shows the project plan and division of tasks:

| Phase | Task | Dependencies | Person Responsible | Subtasks |
|---|---|---|---|---|
| **Planning** | 1. Define project benchmarks | None | All team members | Research benchmarks for testing implemented security measures. |
| **Planning** | 2. Define project scope | None | All team members | Read and understand project brief to determine all deliverables. |
| **Planning** | 3. Outline deliverables and divide tasks among team members | Task 2 | Boluwarin | Provisioning and securing the IaaS (Boluwarin)<br><br>Provisioning and securing the LAMP stack and the Wordpress website (Oreoluwa)<br><br>Evaluating the security (Paul) |
| **Setup** | 4. IaaS setup | Task 3 | Boluwarin | Setup and configure EC2 instance.<br><br>Allocate static Elastic public IP address.<br><br>Create user accounts for all team members with appropriate permissions.<br><br>Configure DNS records mapping domain `teamalphaca.xyz` and elastic IP address. |

| | | | | |
|---|---|---|---|---|
| | | | | Setup backups of EC2 instance with snapshots |
| **Setup** | 5. Configuration of LAMP stack and Wordpress | Task 4 | Oreoluwa | Confirm of SSH access into EC2 instance<br><br>Install LAMP stack technologies from relevant repositories.<br><br>Install and configure Wordpress website.<br><br>Setup user accounts on Wordpress website |
| **Setup** | 6. Install SSL certificate | Task 5 | Boluwarin | Issue SSL certificate on certificate authority provider.<br><br>Generate and install Certificate Authority SSL certificate on web server.<br><br>Ensure Cloudflare and Apache web server TLS settings are aligned |
| **Post-setup** | 7. Secure the EC2 instance | Task 4, task 6 | Boluwarin | Disable password access for SSH.<br><br>Configure security group settings and open relevant ports and close ports not in use.<br><br>Utilize public key authentication for SSH access for all users.<br><br>Disable root login for SSH access.<br>Configure public key authentication SSH access for all team members.<br><br>Enable monitoring of important events such as sudo usage |

| | | | | |
|---|---|---|---|---|
| **Post-setup** | 8. Secure the Wordpress site | Task 5 | | Install Wordpress security plugins.<br><br>Configure multi-factor authentication.<br><br>Ensure all website pages are redirecting to https.<br><br>Ensure no sensitive information is exposed in the publicly available directories of the website.<br><br>Disable file editor on Wordpress administrator interface |
| **Post-setup** | 9. Testing | Task 8 | Paul | Ensure passwordless and public key SSH access to the EC2 instance.<br><br>Verify Wordpress site multi-factor authentication functionality.<br><br>Ensure important events are logged within monitoring solution |
| **Document** | 10. Record demonstration video | Task 8 | All team members | Display efforts to harden security for all assets |
| **Document** | 11. Write report | Task 8 | All team members | |

Table 1. Project plan

## SELECTION OF TOOLS, METHODOLOGIES, FRAMEWORKS AND BENCHMARKS

### Tools

The security tools selected ensure every layer of resources are protected from attack. used are detailed in Table 2 below:

| Tool | Category | Justification | Default provision | Hardening effort |
|---|---|---|---|---|
| **Rublon MFA** | Identity and Access Management | Ensures only users with proper authorization can access Wordpress administration panel | Username and password pair for authentication | This further hardens the Wordpress website by adding an extra layer of authentication to the administrative panel of the website |
| **Cloudflare** | DDoS protection | Ensures web server resources are used for legitimate purposes | Nil | This further protects the webserver against multiple malicious requests with the intent of taking it down |
| **Cloudflare Turnstile** | Bot Management | Protection against malicious bots by ensuring suspicious requests are given a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) challenge to solve before they can access the web page | Nil | This further hardens the Wordpress site by ensuring only humans access the website |
| **WPScan** | Application Security | Vulnerability management for core files, plugins, and themes | Nil | This hardens the Wordpress site by managing any potential vulnerabilities that may be present in themes, plugins, or |

| | | | | |
|---|---|---|---|---|
| | | | | core files in a specific version |
| **ZeroSSL** | SSL/TLS encryption | Certificate Authority for issuing signed SSL certificates | Apache self-signed SSL certificate | This ensures traffic between a client browser and the server is sent over an encrypted tunnel |
| **Snapshots** | Backup and recovery | Backups enable easy and quick restoration to an okay state before a site is compromised or brought down | Nil | This enables recovery of the instance in the event of a disaster |
| **Datadog** | Monitoring | Ensuring uptime of all the resources needed for the website to function | Nil | Monitoring of vital system metrics and logs to ensure compliance with security policies |

Table 2. Tools used.

## Frameworks and Benchmark

In this study we used established frameworks and benchmarks to ensure effective implementation for IaaS and web application. Framework provides platform to facilitate deployment, and configuration.  The key frameworks used in this study are AWS cloud services, and LAMP stack used for hosting a scalable and reliable WordPress application.

## LAMP stack

This framework encompasses Linux, Apache, PHP and MySQL to provide an environment to host WordPress in the EC2 instance.  The reason for selecting this Framework is cost effective since its open source, provide developers with flexibility and reliability with various operating systems, and easy maintenance due to large community [9].

## AWS Cloud Service

AWS cloud services use the Amazon Web to provide resources to support cloud computing. WordPress application was hosted by the AWS EC2 instance leveraging the powerful services and infrastructure provided by AWS.

## Cloudflare

Cloudflare is not only one of the powerful web hosting service providers, but it also provides security products especially Content Delivery Network (CDN), and web optimization. Its plays a vital role in preventing DDoS attack by improving security, performance, and web availability for global content distribution. We used Cloudflare provide extra layer of protection by mitigating bots protecting user interaction.

For benchmark, we measured effectiveness of each tool used by using alternative tools and comparing the outcome. For example, we used Nmap was used to enumerate open ports and services to complement the Amazon Inspector that provided comparable results. Similarly different vulnerability tools were also used to complement each other.

## TECHNICAL TESTING APPROACH

After setting up security features for the EC2 instance and WordPress, we used different tool to evaluate the security features implemented. The purpose of testing was to outline any the efficiency of the IaaS platform and WordPress, security posture and identify vulnerabilities that attackers could exploit and implementing appropriate measures to mitigate and minimise attack surface. This process involved testing both the host and the web application security. The process involved initial scanning, vulnerability assessment, network analysis, web vulnerability scanning, and conducting dynamic web application security testing.

i. Nmap Scanning

We conducted a deep Nmap scan to establish open ports and services running on our EC2 instance. This step was focused on collecting information about the EC2. The information of our interest were the open ports, services and their versions, domain names, and IP addresses. The service versions were crucial in identifying vulnerabilities specific to the services.

ii. Vulnerability Assessment

The objective of this testing was to conduct a comprehensive vulnerability assessment of both the web application and EC2 instance to identify security vulnerabilities. Nessus was used to perform this scan because it has the capability to outline system misconfigurations, potential exploits and outdated software which poses security risk. The tool provided the severity ratings and recommendations to the identified vulnerabilities.

iii. Network analysis.

Network monitoring and analysis was conducted using Wireshark to monitor inbound and outbound traffic to the EC2 instance. This was achieved by capturing the network traffic using Wireshark to trace suspicious activities in the network. The aim of performing network analysis on the EC2 was to detect malicious activities and potential intrusions that could result in data theft.

iv. WordPress vulnerability assessment

Burp Suite was used to conduct WordPress vulnerability assessment. Burpsuite is a crucial tool in evaluating web application security as it has tools to perform web penetration testing [4]. In this exercise we used the Intruder option in Burpsuite to conduct a brute force attack using the user and password payloads. The objective was to try and gain access by using common usernames and matching it with a list of passwords using the rockyou.txt file.

v. Dynamic Application Security Testing (DAST)

DAST is a web application analysis technique using the user interface to identify security flaws using simulated attacks [5]. This type of attack analyses the application like an actor would do. This attack is performed real-time. Zed Attack Proxy (ZAP) was used to conduct automated attack on the WordPress web application to establish security risks such as insecure authentication, cross-site scripting (XSS), and SQL injection. We performed the attack entering our WordPress URL https://teamalphaca.xyz and pressing the attack button. The attack generates alerts with risk rating, description, and solutions, among others.

vi. Amazon Inspector report

Amazon inspector was used to assess EC2 instance network reachability. This was achieved by using the "Assessment-Template-Default-All-Rules". The objective of this assessment was the "Network Reachability –1.1" rules package. This was done by evaluating the EC2 instance security configurations by enumerating open ports and services accessible over the internet.

vii. MFA Verification

Manual test was done to confirm MFA and password protected login. The access to the web application requires a username and password for access and after that an MFA to authenticate users and grant access to the site.

## FINDINGS AND RISK RATINGS

This section will outline the finding from the various tests performed and the risk rating for identified vulnerabilities. The finding will be outlined based on the different tests performed.

The Dynamic Application Security Testing (DAST) using ZAP identified vulnerabilities with risk rating as **medium**. The vulnerabilities are listed below.

i) Absence of Anti-CSRF Tokens

This security flaw was identified in https://teamalphaca.xyz/un-sdg-8/ and has a medium risk rating. With absence of CSRF tokens, the WordPress is susceptible to cross-site request forgery attack (CWE-352) [6]. This kind of attack intends to trick a user to submit malicious request to the server then take control over the user by inheriting their privileges and identify and performing malicious activities on their behalf. If the user is admin or has privileges, the attacker will take control of the system since They will perform any operations as the user.

This security weakness can be addressed by;

- Using vetted framework during architecture and design phase which prevents such weakness from occurring. For example, OWASP CSRFGuard which is an anti-CSRF package.
- The implementation phase should ensure the web application is not susceptible to cross-site scripting issues by securing the wen from attacker-controlled script.

ii) Missing Anti-clickjacking Header

This vulnerability was established at http://teamalphaca.xyz with a **medium** risk rating. The response captured by ZAP lacks the Content-Security –Policy that offers protection against clickjacking attacks. With clickjacking users are tricked to click an invisible webpage element which could result in malware download, online product purchase, money transfer or exposure of personal identifiable information [7]. This can be addressed by implementing Content-Security –Policy.

Secondly, the use of a strong password that incorporates lowercase and uppercase letters, symbols and numbers on our WordPress website improved its security [8]. The use of brute force attack in Burpsuite intruder was not successful as the passwords assigned to users and admin were complex to break. Furthermore, the use of MFA that requires more information than password alone, provided an additional layer of security.

Thirdly, vulnerability assessment conducted using Nessus identified the following vulnerabilities:

i) HTTP TRACE/TRACK method allowed in the remote server. This vulnerability has severity rate of medium. This method is used in web server debugging and are prone to cross-Site-Tracing (XST) attacks. If these methods are allowed, sensitive information have potential of exposure, and this can be resolved by disabling the HTTP TRACETRACK methods in the EC2 host.

ii) Nessus picked an Outdated OpenSSL 3.0.0<3.15 with critical severity score. The vulnerability has CVE-2022-3602 and CVE-2022-3786 that has the potential to cause denial of service (DoS) attack. An attacker can deploy a malicious email using the CVE-2022-3786 to cause buffer overflow causing the system to crash and giving them control over the affected system. This was addressed by upgrading to OpenSSL version 3.0.15.

iii) Nessus also picked outdated version 2.4.x of Apache httpd installed in the EC2 instance with medium risk rating. Versions before 2.4.62 have vulnerabilities tracked as CVE-2024-39884 that discloses the source code such rendering PHP as plaintext by ignoring the content-type configurations and **C**VE-2024-40898 that leaks NTLM hashes to malicious server and requests. This vulnerability identified was solved by upgrading to version 2.4.62.

Finally, the Amazon Inspector Identifies open ports 20 (FTP), 21 (FTP), 22 (SSH), 80(HTTP), and 443 (HTTPS) as reachable via internet. The Inspector advice security group (g-0a63ac0692c96cdda) modification to restrict access to these ports via internet to minimize attack surface.

## CHALLENGES AND LIMITATIONS

### Challenges

While there have been concerted efforts made on the Wordpress site and the EC2 instance, they may remain vulnerable to brute force attacks by zero-day threats conducted by a resolute threat actor. The EC2 instance will be more vulnerable to this due to SSH access not being routable through a virtual private network due to the requirement of a physical network device. This is identified in the Amazon Inspector classic vulnerability scan.

### Limitations

- Restrictions on AWS platform to create IAM roles and utilize other security features.
- Hard limit of 4 hours on all created resources
- Lack of scripting experience amongst team members to automate other tasks.

## CONCLUSIONS

By following the defence in depth principles from the beginning, the compute resources, technology stack and the Wordpress website were properly secured against known threats. All vulnerabilities identified in the initial scan after securing the assets were addressed and there were no critical or high-risk vulnerabilities identified in subsequent scans, displaying a good effort in securing the assets. This enabled a complete achievement of the objectives set out in the beginning of the project. Multiple scanning tools were utilized to ensure replication of results across different platforms and balance out any weaknesses in each scanning tool.

This project displays the sustainable inexpensive security for important systems to ensure the principle of confidentiality, integrity and availability of information contained on important systems such as Wordpress which is very widely used. It has also shown the importance of provisioning and building security into all important assets.

## REFLECTION AND INDIVIDUAL CONTRIBUTIONS

### Paul

Participating in this project has been an amazing opportunity for me to sharpen my practical knowledge and enhancing my teamwork abilities. Throughout the project I have learned the importance of effective communication, project planning, and team contribution. The use of ClickUp to manage our project has been opportunity to foster effective collaboration as a team. One of the key lessons I have learned is the importance of having effective security measure in place for cloud infrastructure. Paul was responsible for technical testing, setting up ClickUp, writing findings, framework, and benchmark section.

### Boluwarin

This project was extremely fun and challenging to work on and taught me a lot about cloud technologies, networking and Wordpress. Boluwarin was responsible for drafting the project plan, leading team meetings, creating and securing the IaaS assets and writing the introduction and conclusions of the report.

### Ore

Having to set up an AWS EC2 instance with LAMP stack, and a WordPress environment with my team members was a challenging yet rewarding project. I led the configuration and hardening of the LAMP stack, ensuring a seamless WordPress installation. I was also responsible for the hardening of the Wordpress site.  This project emphasizes the importance of integrating security from the start and highlighted the continuous effort needed to maintain a secure and reliable web server and site through the various security measures implemented by the team.

## REFERENCES

[1] “About – WordPress.org.” Accessed: Jul. 26, 2024. [Online]. Available: https://wordpress.org/about/
[2] “Top 22 WordPress Statistics: Defining Trends and Insights for 2024.” Accessed: Jul. 26, 2024. [Online]. Available: https://www.hostinger.com/tutorials/wordpress-statistics#Top_10_WordPress_Stats_You_Should_Know
[3] “What is AWS? - Cloud Computing with AWS - Amazon Web Services.” Accessed: Jul. 26, 2024. [Online]. Available: https://aws.amazon.com/what-is-aws/

[4] 0x4C3DD, “Security Testing of Web Applications Using Burp Suite,” *Medium*, Jun. 17, 2023. https://medium.com/@aka.0x4C3DD/security-testing-of-web-applications-using-burp-suite-f01e161a3325#:~:text=Burp%20Suite%20offers%20various%20features (accessed Jul. 28, 2024).

[5] opentext, “What is Dynamic Application Security Testing (DAST),” *OpenText*. https://www.opentext.com/what-is/dast#:~:text=Dynamic%20Application%20Security%20Testing%20(DAST)%20is%20the%20process%20of%20analyzing (accessed Jul. 28, 2024).

[6] MITRE, “CWE - CWE-352: Cross-Site Request Forgery (CSRF) (3.4.1),” *Mitre.org*, 2009. https://cwe.mitre.org/data/definitions/352.html (accessed Jul. 28, 2024).

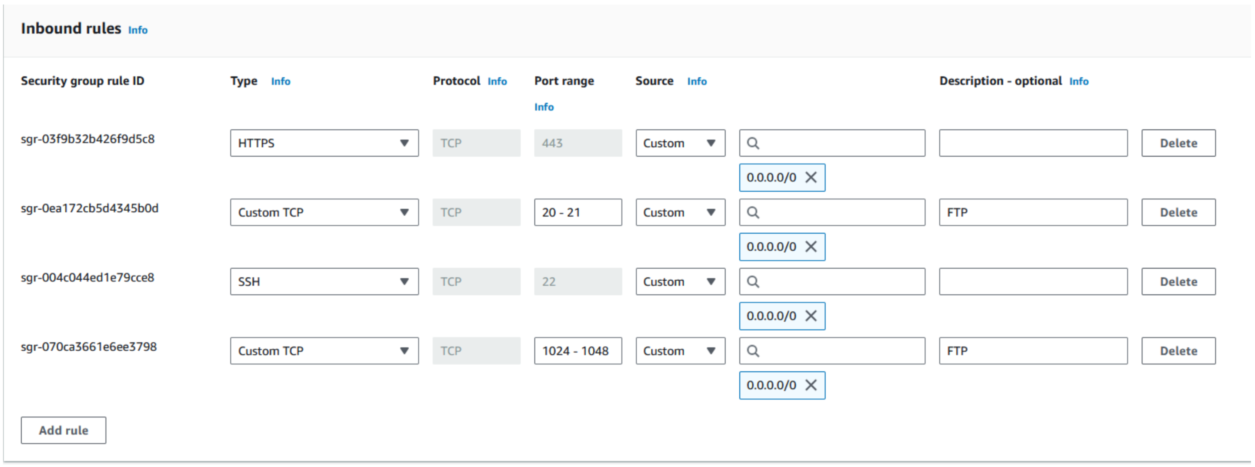
[7] “What is Clickjacking | Attack Example | X-Frame-Options Pros & Cons | Imperva,” *Iperva*. https://www.imperva.com/learn/application-security/clickjacking/#:~:text=Clickjacking%20is%20an%20attack%20that (accessed Jul. 28, 2024).
[8] Walden University, “Cybersecurity 101: Why choosing a secure password is so important,” *Walden University*, 2024. https://www.waldenu.edu/programs/information-technology/resource/cybersecurity-101-why-choosing-a-secure-password-in-so-important#:~:text=Strong%20passwords%20are%20of%20the (accessed Jul. 28, 2024).
[9] AWS, “What is LAMP Stack? - LAMP Stack - AWS,” *Amazon Web Services, Inc.* https://aws.amazon.com/what-is/lamp-stack/ (accessed Jul. 28, 2024).

# Appendices

## Appendix A

The steps below were followed to secure the cloud compute resources on the Amazon Web Services cloud platform:

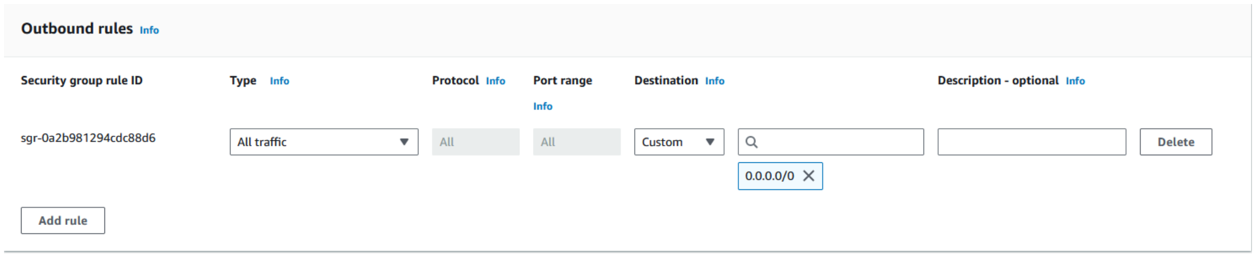
1. After the instance was created, ports 20, 21, 80, 443, 1024-1048 were opened in the security group's inbound rules in addition to port 22 for SSH access that was open by default to ensure the needed protocols for Wordpress operation namely HTTP, HTTPS, and FTP were allowed. The only outbound rule was configured to allow all traffic out of the security group. Fig. A1 and Fig A2 show the inbound and outbound rules as configured:



**Inbound rules** Info

| Security group rule ID | Type | Protocol | Port range | Source | | Description - optional |
|---|---|---|---|---|---|---|
| sgr-03f9b32b426f9d5c8 | HTTPS | TCP | 443 | Custom | 0.0.0.0/0 | |
| sgr-0ea172cb5d4345b0d | Custom TCP | TCP | 20 - 21 | Custom | 0.0.0.0/0 | FTP |
| sgr-004c044ed1e79cce8 | SSH | TCP | 22 | Custom | 0.0.0.0/0 | |
| sgr-070ca3661e6ee3798 | Custom TCP | TCP | 1024 - 1048 | Custom | 0.0.0.0/0 | FTP |

Fig. A1. Inbound rules



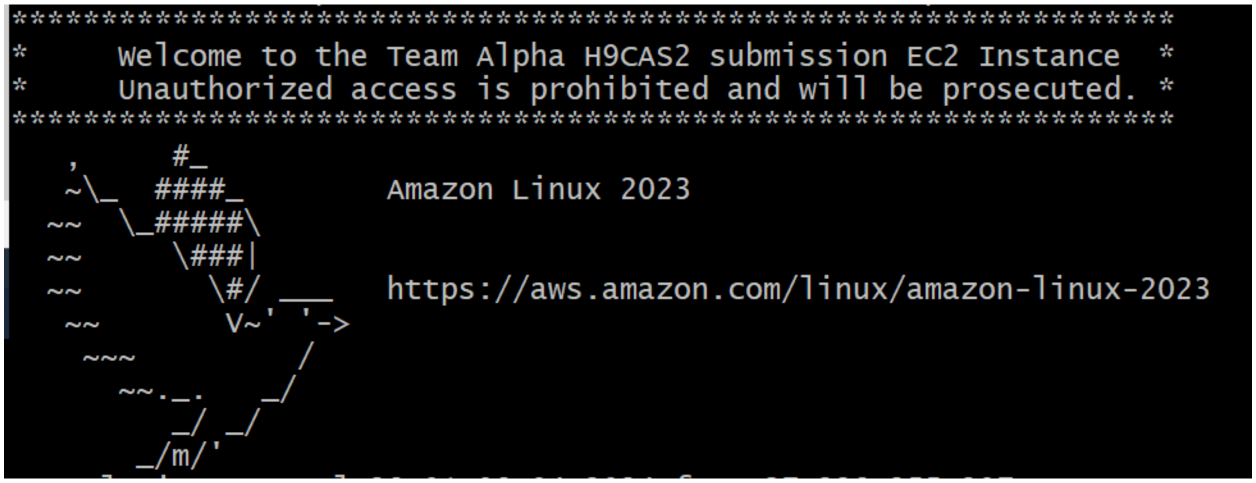
**Outbound rules** Info

| Security group rule ID | Type | Protocol | Port range | Destination | | Description - optional |
|---|---|---|---|---|---|---|
| sgr-0a2b981294cdc88d6 | All traffic | All | All | Custom | 0.0.0.0/0 | |

Fig. A2 Outbound rules

2. SSH access was established to the instance with the default created ec2-user account. User accounts were created for all team members with super user privileges to enable them to conduct their assigned tasks. Root login, and password authentication was disabled, and public key authentication enabled by default on creation of the EC2 instance. Private keys and their public key equivalent were generated and placed in the appropriate files (`/home/user/.ssh/authorized_keys`)

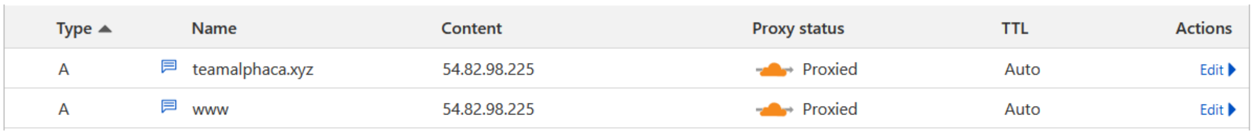
and directories of each user to enable authentication. A message of the day (MOTD) banner warning unauthorized users against attempting to connect to the instance through SSH was also configured. This is shown in Fig. A3 below:



```
****************************************************************
*     Welcome to the Team Alpha H9CAS2 submission EC2 Instance  *
*     Unauthorized access is prohibited and will be prosecuted. *
****************************************************************
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

Fig. A3. Banner displaying on SSH access attempt.

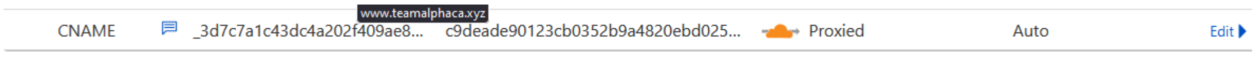
3. The domain name teamalphaca.xyz was obtained for the project. Cloudflare was selected as the DNS provider, proxy service. Two A records were created, pointing to the public Elastic IP address allocated to the computing resource provisioned on AWS. All requests to the website were set to be proxied through Cloudflare's network so the public IP address of the cloud compute resources is not exposed. Fig. A4 shows the A records:



| Type ▲ | Name | Content | Proxy status | TTL | Actions |
|---|---|---|---|---|---|
| A | teamalphaca.xyz | 54.82.98.225 | Proxied | Auto | Edit ▶ |
| A | www | 54.82.98.225 | Proxied | Auto | Edit ▶ |

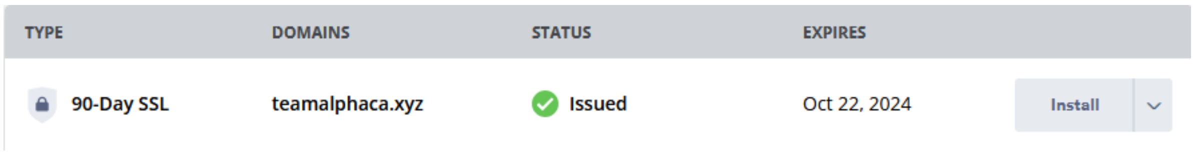
Fig. A4. DNS A records

4. ZeroSSL was used as a Certificate Authority to issue a 90-day SSL certificate for the domain teamalphaca.xyz. A private key file was generated on the server and used to generate a Certificate Signing Request (CSR) which was then used to generate a primary certificate and certificate chain for the domain. The primary certificate and certificate chain were then placed in the `/etc/pki/tls/certs` directory on the server and the private key placed in the `/etc/pki/tls/private` directory. The Apache SSL configuration file in the `/etc/httpd/conf.d/ssl.conf` was then edited to reflect the changes and the Apache service restarted to load the latest changes.



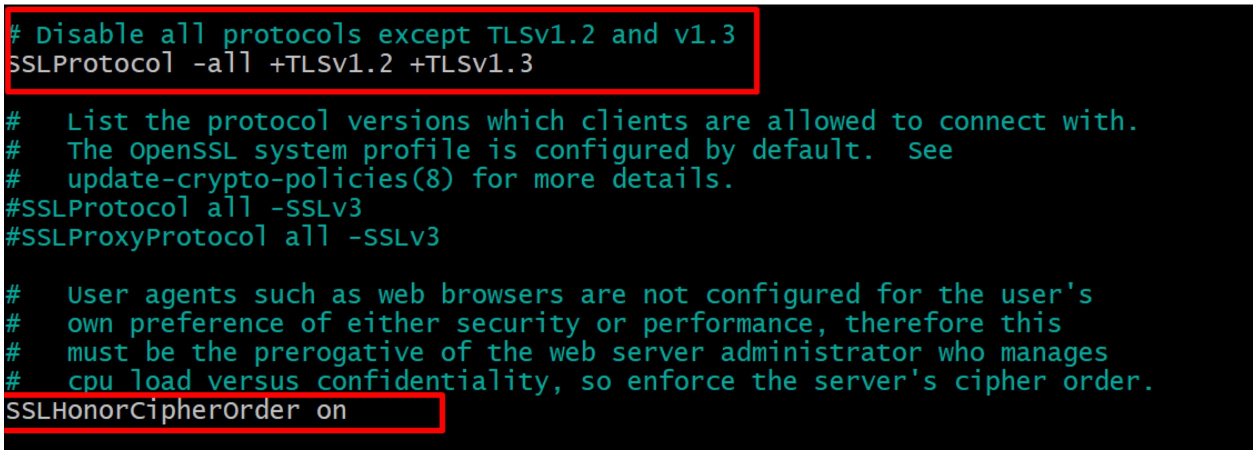
| | | | | | |
|---|---|---|---|---|---|
| CNAME | _3d7c7a1c43dc4a202f409ae8... | c9deade90123cb0352b9a4820ebd025... | Proxied | Auto | Edit |

Fig. A4 CNAME record



| TYPE | DOMAINS | STATUS | EXPIRES | |
|---|---|---|---|---|
| 90-Day SSL | teamalphaca.xyz | Issued | Oct 22, 2024 | Install |

Fig. A5 SSL certificate issued on ZeroSSL.

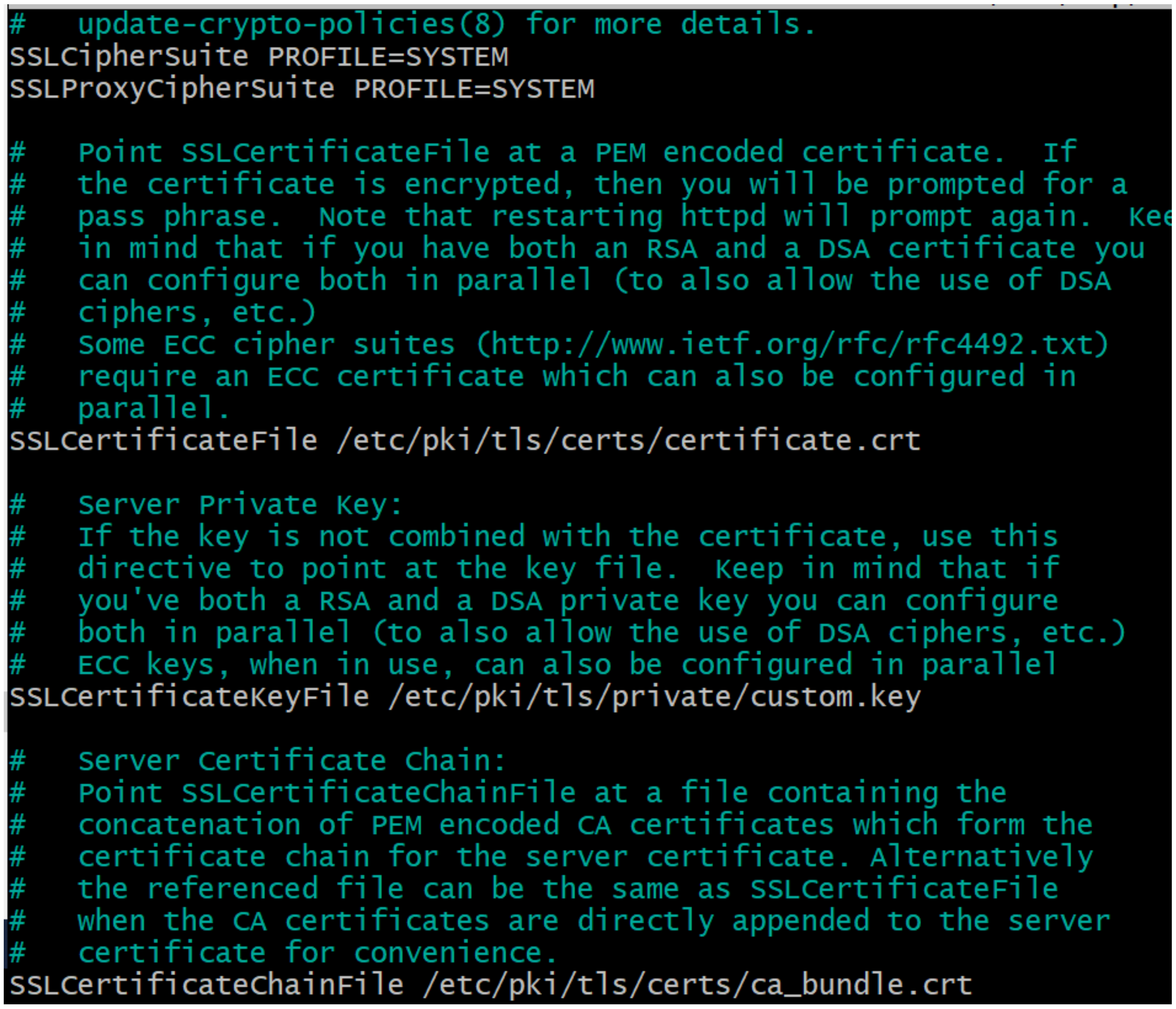


```
# Disable all protocols except TLSv1.2 and v1.3
SSLProtocol -all +TLSv1.2 +TLSv1.3

#   List the protocol versions which clients are allowed to connect with.
#   The OpenSSL system profile is configured by default.  See
#   update-crypto-policies(8) for more details.
#SSLProtocol all -SSLv3
#SSLProxyProtocol all -SSLv3

#   User agents such as web browsers are not configured for the user's
#   own preference of either security or performance, therefore this
#   must be the prerogative of the web server administrator who manages
#   cpu load versus confidentiality, so enforce the server's cipher order.
SSLHonorCipherOrder on
```

Fig. A6. Apache `ssl.conf` file showing only TLS v1.2 and v1.3 used, and all other protocols denied.



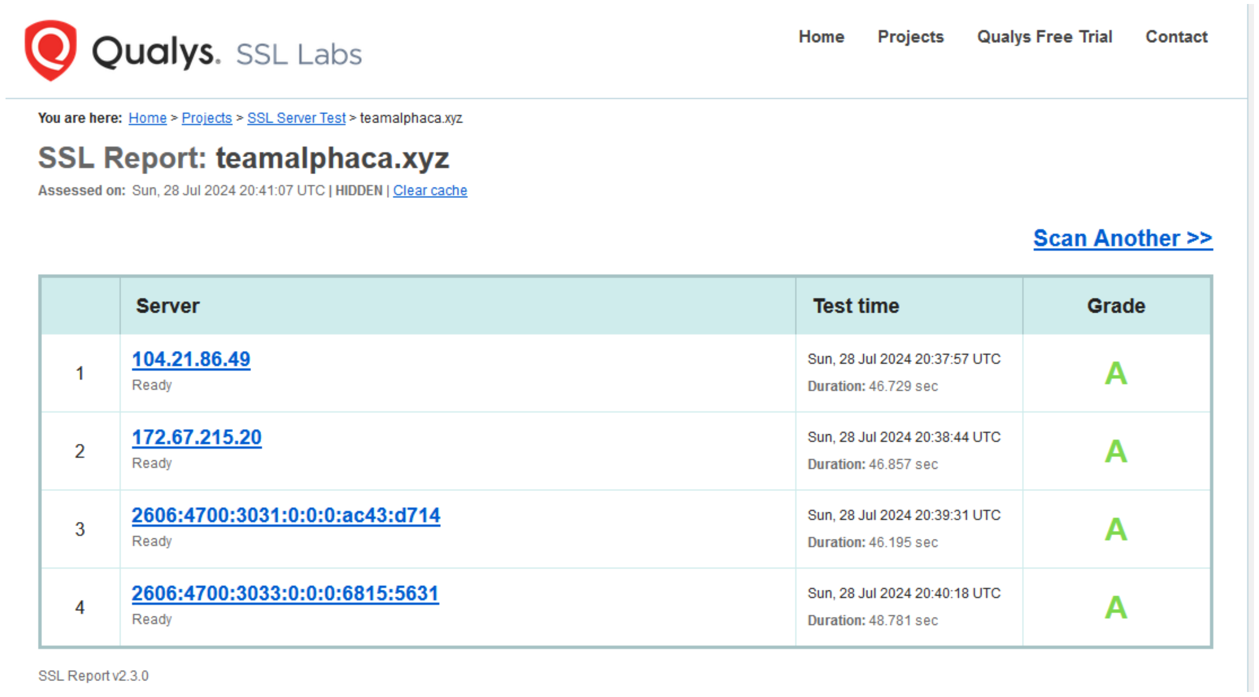
```
#   update-crypto-policies(8) for more details.
SSLCipherSuite PROFILE=SYSTEM
SSLProxyCipherSuite PROFILE=SYSTEM

#   Point SSLCertificateFile at a PEM encoded certificate.  If
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that restarting httpd will prompt again.  Kee
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
#   Some ECC cipher suites (http://www.ietf.org/rfc/rfc4492.txt)
#   require an ECC certificate which can also be configured in
#   parallel.
SSLCertificateFile /etc/pki/tls/certs/certificate.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
#   ECC keys, when in use, can also be configured in parallel
SSLCertificateKeyFile /etc/pki/tls/private/custom.key

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convenience.
SSLCertificateChainFile /etc/pki/tls/certs/ca_bundle.crt
```

Fig. A7. Apache `ssl.conf` file showing paths to the certificate, chain certificate and private key.



You are here: Home > Projects > SSL Server Test > teamalphaca.xyz

SSL Report: teamalphaca.xyz

Assessed on: Sun, 28 Jul 2024 20:41:07 UTC | HIDDEN | Clear cache

Scan Another >>

| | Server | Test time | Grade |
|---|---|---|---|
| 1 | 104.21.86.49<br>Ready | Sun, 28 Jul 2024 20:37:57 UTC<br>Duration: 46.729 sec | A |
| 2 | 172.67.215.20<br>Ready | Sun, 28 Jul 2024 20:38:44 UTC<br>Duration: 46.857 sec | A |
| 3 | 2606:4700:3031:0:0:0:ac43:d714<br>Ready | Sun, 28 Jul 2024 20:39:31 UTC<br>Duration: 46.195 sec | A |
| 4 | 2606:4700:3033:0:0:0:6815:5631<br>Ready | Sun, 28 Jul 2024 20:40:18 UTC<br>Duration: 48.781 sec | A |

SSL Report v2.3.0

Fig. A8. SSL report using Qualys SSL labs service.

5. Datadog was utilized for system metrics, log monitoring and Cloudflare monitoring. The Datadog agent was configured on the cloud compute resource and the web server to collect the required metrics. The dashboards are shown in Fig. A9 and A10 below. Logs for ssh access to the compute resource, super user (sudo) events and Apache web server events.

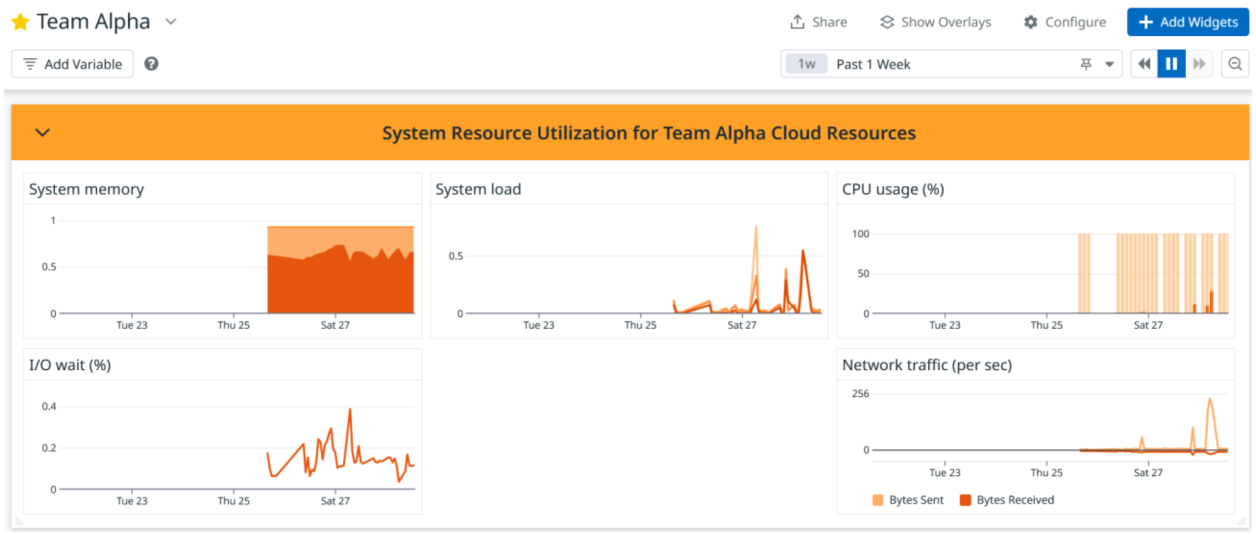
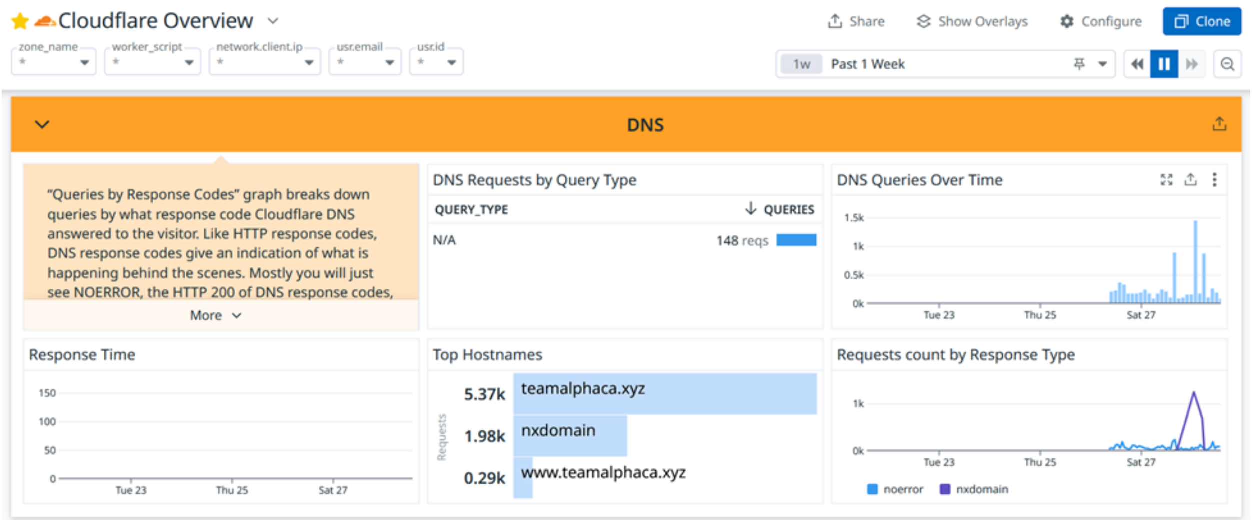


Fig. A9. System resource utilization dashboard showing utilization for 1 week.



"Queries by Response Codes" graph breaks down queries by what response code Cloudflare DNS answered to the visitor. Like HTTP response codes, DNS response codes give an indication of what is happening behind the scenes. Mostly you will just see NOERROR, the HTTP 200 of DNS response codes,

More

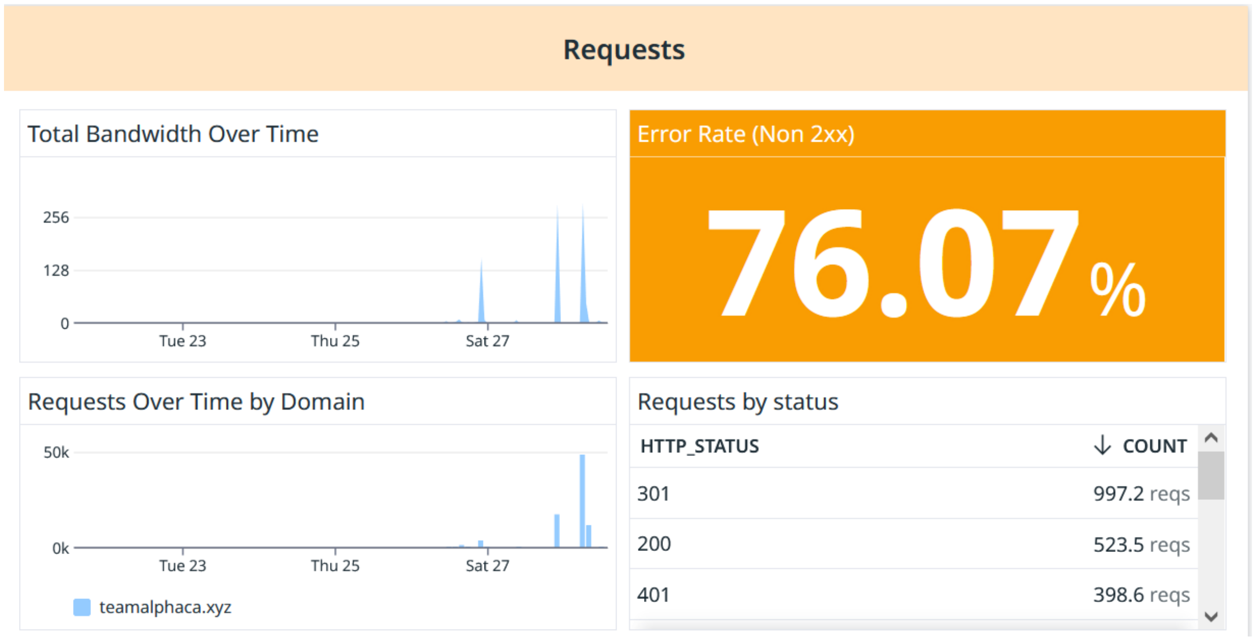
DNS Requests by Query Type

| QUERY_TYPE | QUERIES |
|---|---|
| N/A | 148 reqs |

Top Hostnames

| | |
|---|---|
| 5.37k | teamalphaca.xyz |
| 1.98k | nxdomain |
| 0.29k | www.teamalphaca.xyz |

Fig. A10. DNS queries for the teamalphaca.xyz website



Error Rate (Non 2xx)

76.07%

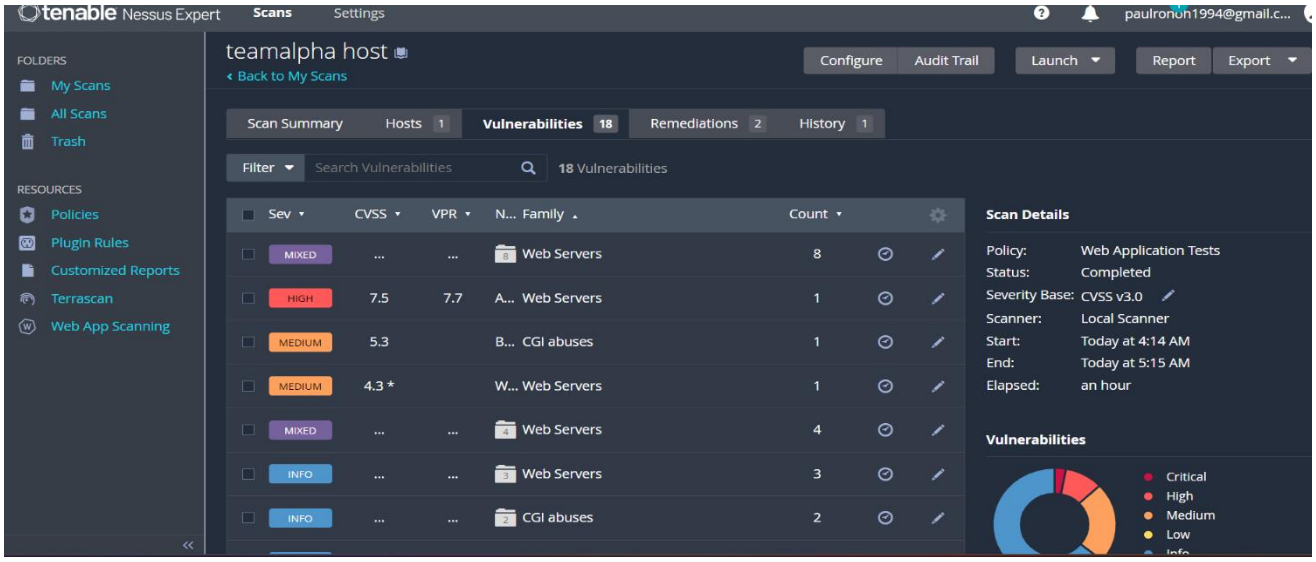
Requests by status

| HTTP_STATUS | COUNT |
|---|---|
| 301 | 997.2 reqs |
| 200 | 523.5 reqs |
| 401 | 398.6 reqs |

Fig. A11. Request status summary and error rate.

## Appendix B

Nessus was used to scan the web application and the EC2 instance for vulnerabilities. This indicated various vulnerabilities with severity rating, description, and solutions.



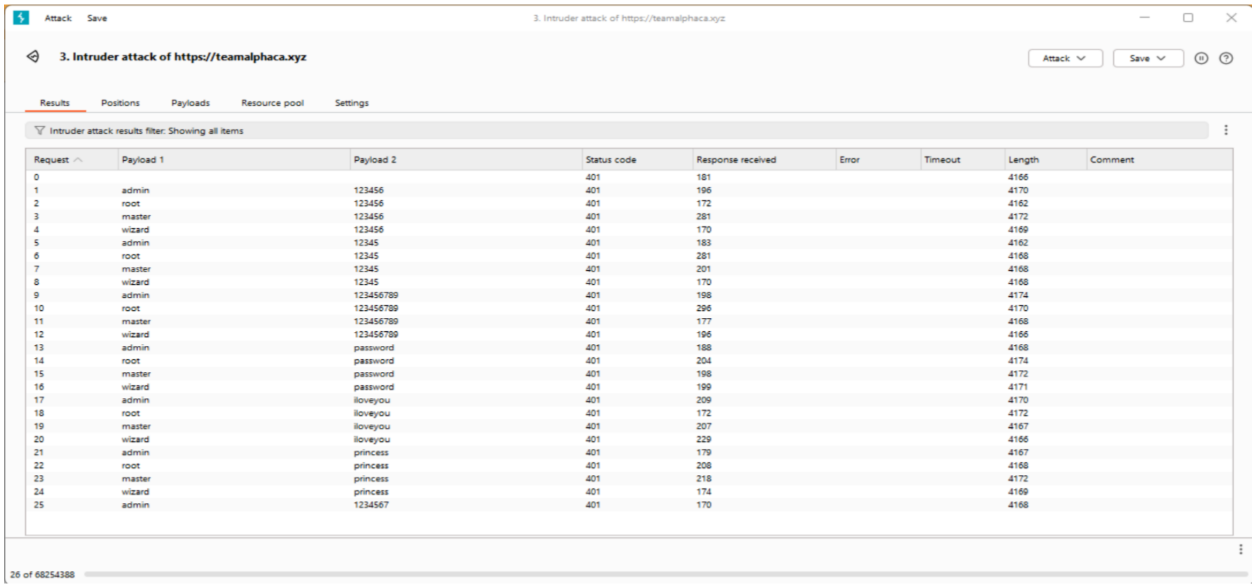
| Sev | CVSS | VPR | N... Family | Count |
|---|---|---|---|---|
| MIXED | ... | ... | Web Servers | 8 |
| HIGH | 7.5 | 7.7 | A... Web Servers | 1 |
| MEDIUM | 5.3 | | B... CGI abuses | 1 |
| MEDIUM | 4.3 * | | W... Web Servers | 1 |
| MIXED | ... | ... | Web Servers | 4 |
| INFO | ... | ... | Web Servers | 3 |
| INFO | ... | ... | CGI abuses | 2 |

**Scan Details**

| | |
|---|---|
| Policy: | Web Application Tests |
| Status: | Completed |
| Severity Base: | CVSS v3.0 |
| Scanner: | Local Scanner |
| Start: | Today at 4:14 AM |
| End: | Today at 5:15 AM |
| Elapsed: | an hour |

Figure C1: Nessus vulnerability scan.

Burp suite was used to perform Brute force attack using the Burb intruder with 2 payloads, the username and password. The status code 401 indicates no response to client request and unsuccessful Brute force attack.

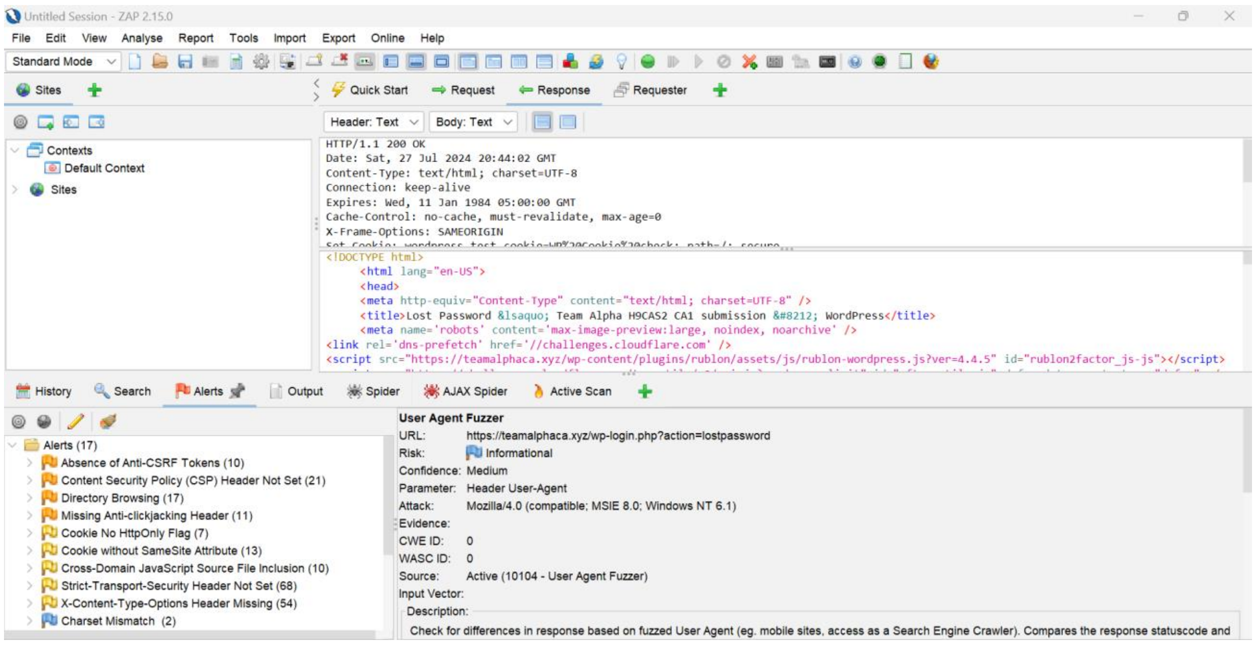

Attack
Save
3. Intruder attack of https://teamalphaca.xyz
3. Intruder attack of https://teamalphaca.xyz
Attack
Save
Results
Positions
Payloads
Resource pool
Settings
Intruder attack results filter: Showing all items

| Request | Payload 1 | Payload 2 | Status code | Response received | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|---|---|
| 0 | | | 401 | 181 | | | 4166 | |
| 1 | admin | 123456 | 401 | 196 | | | 4170 | |
| 2 | root | 123456 | 401 | 172 | | | 4162 | |
| 3 | master | 123456 | 401 | 281 | | | 4172 | |
| 4 | wizard | 123456 | 401 | 170 | | | 4169 | |
| 5 | admin | 12345 | 401 | 183 | | | 4162 | |
| 6 | root | 12345 | 401 | 281 | | | 4168 | |
| 7 | master | 12345 | 401 | 201 | | | 4168 | |
| 8 | wizard | 12345 | 401 | 170 | | | 4168 | |
| 9 | admin | 123456789 | 401 | 198 | | | 4174 | |
| 10 | root | 123456789 | 401 | 296 | | | 4170 | |
| 11 | master | 123456789 | 401 | 177 | | | 4168 | |
| 12 | wizard | 123456789 | 401 | 196 | | | 4166 | |
| 13 | admin | password | 401 | 188 | | | 4168 | |
| 14 | root | password | 401 | 204 | | | 4174 | |
| 15 | master | password | 401 | 198 | | | 4172 | |
| 16 | wizard | password | 401 | 199 | | | 4171 | |
| 17 | admin | iloveyou | 401 | 209 | | | 4170 | |
| 18 | root | iloveyou | 401 | 172 | | | 4172 | |
| 19 | master | iloveyou | 401 | 207 | | | 4167 | |
| 20 | wizard | iloveyou | 401 | 229 | | | 4166 | |
| 21 | admin | princess | 401 | 179 | | | 4167 | |
| 22 | root | princess | 401 | 208 | | | 4168 | |
| 23 | master | princess | 401 | 218 | | | 4172 | |
| 24 | wizard | princess | 401 | 174 | | | 4169 | |
| 25 | admin | 1234567 | 401 | 170 | | | 4168 | |

26 of 68254388

Fig C2: Burpsuite Intruder results

OWASP ZAP tool was used to carry out automated attacks on the WordPress web application to establish security risks such as insecure authentication, cross-site scripting (XSS), and SQL injection.



```
HTTP/1.1 200 OK
Date: Sat, 27 Jul 2024 20:44:02 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
X-Frame-Options: SAMEORIGIN
```

```
<!DOCTYPE html>
     <html lang="en-US">
     <head>
     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
     <title>Lost Password &lsaquo; Team Alpha H9CAS2 CA1 submission &#8212; WordPress</title>
     <meta name='robots' content='max-image-preview:large, noindex, noarchive' />
<link rel='dns-prefetch' href='//challenges.cloudflare.com' />
<script src="https://teamalphaca.xyz/wp-content/plugins/rublon/assets/js/rublon-wordpress.js?ver=4.4.5" id="rublon2factor_js-js"></script>
```

- Alerts (17)
  - Absence of Anti-CSRF Tokens (10)
  - Content Security Policy (CSP) Header Not Set (21)
  - Directory Browsing (17)
  - Missing Anti-clickjacking Header (11)
  - Cookie No HttpOnly Flag (7)
  - Cookie without SameSite Attribute (13)
  - Cross-Domain JavaScript Source File Inclusion (10)
  - Strict-Transport-Security Header Not Set (68)
  - X-Content-Type-Options Header Missing (54)
  - Charset Mismatch (2)

**User Agent Fuzzer**

| | |
|---|---|
| URL: | https://teamalphaca.xyz/wp-login.php?action=lostpassword |
| Risk: | Informational |
| Confidence: | Medium |
| Parameter: | Header User-Agent |
| Attack: | Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1) |
| Evidence: | |
| CWE ID: | 0 |
| WASC ID: | 0 |
| Source: | Active (10104 - User Agent Fuzzer) |
| Input Vector: | |

Description:

Check for differences in response based on fuzzed User Agent (eg. mobile sites, access as a Search Engine Crawler). Compares the response statuscode and

Fig C3: ZAP Scan