|  |  |
| --- | --- |
| Name: | BHUMIKA GUPTA |
| Lab User ID: | 23SEK3324\_U13 |
| Date: | January 10, 2024 |
| Application Name: | dvwps: Damn Vulnerable WordPress Site |

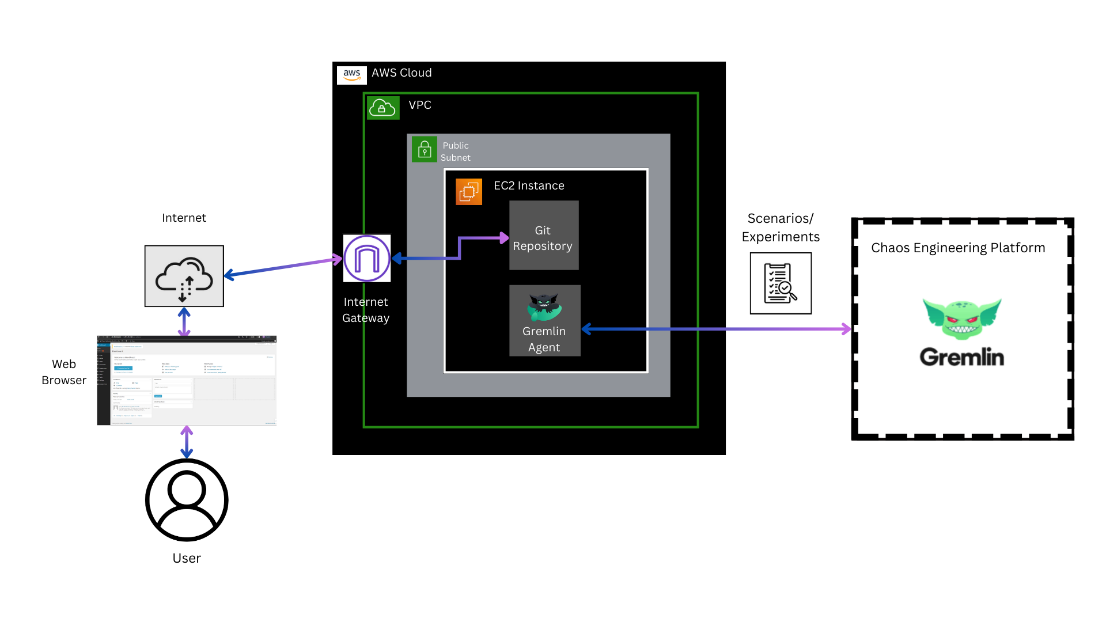
**Follow the below guidelines:**





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)



**Fig: System Architecture**

**System architecture** is a conceptual model that describes the structure and behavior of multiple components and subsystems like multiple software applications, network devices, hardware, and even other machinery of a system.

**Explanation of the above System Architecture-**

1. AWS Cloud: The entire infrastructure is hosted on AWS.
2. VPC (Virtual Private Cloud): The VPC serves as an isolated network within the AWS Cloud. It allows you to logically isolate resources and control network settings.
3. Public Subnet: Within the VPC, there is a public subnet. Public subnets are accessible from the internet and typically host resources like web servers.
4. EC2 Instance: An EC2 instance is launched in the public subnet.
   * + **Scenario 1:** *Web Application Deployment*
       - This instance hosts the web application.
       - Git Repository: There is a Git repository associated with this setup for version control of the WordPress code or configurations.
       - Internet Gateway: The VPC is connected to an Internet Gateway. This allows the EC2 instance to communicate with the internet, enabling users to access the web application.
       - Web Browser (User): Users access the vulnerable WordPress site through a web browser. The entire setup is designed to facilitate learning about penetration testing by allowing users to interact with a deliberately vulnerable environment.
     + **Scenario 2:** *Gremlin Chaos Engineering Experiment*
       - This instance hosts the Gremlin Agent.
       - Gremlin Agent: The Gremlin Agent is a software component installed on the EC2 instance. It facilitates chaos engineering experiments by injecting faults into the system.
       - Gremlin Chaos Engineering Platform: The Gremlin Agent connects to the Gremlin Chaos Engineering Platform. This platform allows you to perform controlled experiments, such as shutting down the EC2 instance.

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)

**Steady State** – The steady state is the stable and expected state where the system functions smoothly without disruptions.

**System’s Normal Behavior -** It involves the consistent interactions between various components, resulting in reliable performance and desired outcomes. Measurable outputs and observed behaviors during steady state operations define the system's normal behavior

So the steady state and measurable outcome of our system is as follows:-

1. **Web Application Deployment:** Users can successfully access the vulnerable WordPress site hosted on the EC2 instance.
   * Measurable Outputs:
     + The Docker container with the vulnerable WordPress setup operates without unexpected disruptions.
     + Git repository maintains stable version control for the WordPress code and configurations.
2. **Gremlin Chaos Engineering Experiment:** The Gremlin Agent operates without disruptions. Controlled chaos experiments, such as shutting down the EC2 instance, do not occur under normal circumstances.
   * Measurable Outputs:
     + Stable connectivity between the Gremlin Agent and the Gremlin Chaos Engineering Platform.
     + No unexpected interruptions or failures during Gremlin experiments.

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**

New plugins were added to the WordPress setup

Sudden increase in web traffic

**Unknown**

**Unknown**

**Known**

Unexpected system update

Internet gateway configuration modifications

These hypothesis provide a structured approach to defining and understanding the expected and unexpected behaviors of the dvwps: Damn Vulnerable WordPress Site.

By categorizing scenarios into knowns and unknowns, and actions into intentional and unintentional, the hypothesis aid in identifying, responding to, and improving the security and robustness of the system.

**Knowns-Knowns Hypothesis:**

* If there is a sudden increase in web traffic, the system will handle the load efficiently, and users will experience minimal latency.
* Monitoring tools will show an increase in resource utilization within acceptable thresholds.

**Known-Unknown Hypothesis:**

* If a new plugin is added to the WordPress setup, the system will continue to function without critical vulnerabilities.
* Regular security scans will detect and flag any potential vulnerabilities introduced by the new plugin.

**Unknown-Known Hypothesis:**

* If the internet gateway configuration is modified, the web application will remain accessible to users without interruptions.
* Network monitoring tools will show stable connectivity between the EC2 instance and the internet.

**Unknown-Unknown Hypothesis:**

* If an unexpected system update is applied, the system will recover gracefully, and users won't experience service disruptions.
* System logs will show successful update installations without error spikes in user access or application errors.

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

**Objective:**

The objective of this report is to document the implementation and analysis of a combined penetration testing and chaos engineering approach in an AWS environment. The setup includes a vulnerable WordPress site for penetration testing and a Gremlin chaos engineering experiment to assess system resilience.

**Context:**

In the AWS cloud environment, a Virtual Private Cloud (VPC) has been configured with a public subnet. An EC2 instance within this VPC hosts a Docker container with a deliberately vulnerable WordPress version and plugins, facilitating a controlled environment for penetration testing.

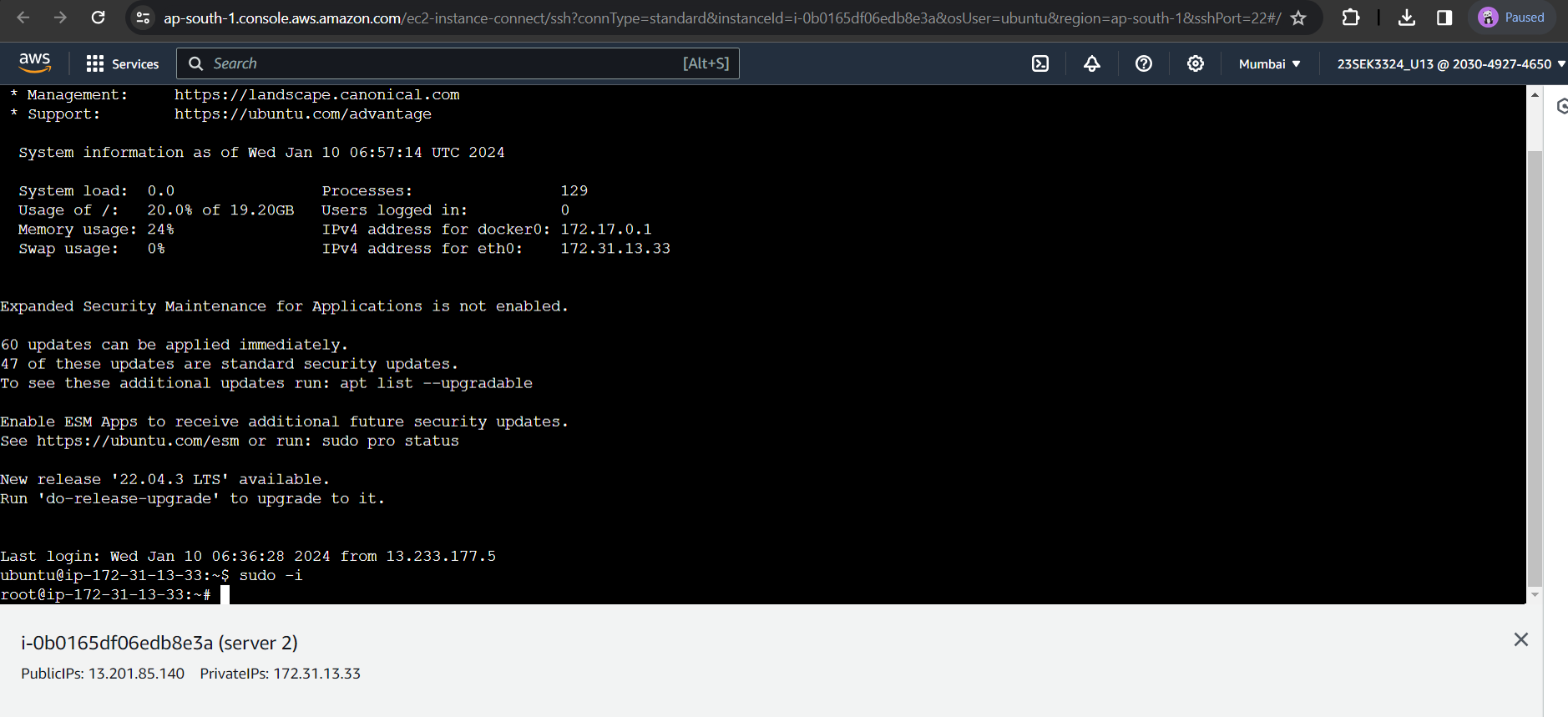
Also the EC2 instance in the same VPC serves as a host for the Gremlin agent, connected to the Gremlin Chaos Engineering Platform for planned experiments aimed at testing system resilience.

Additionally, vulnerability scanning using Trivy was implemented to identify potential security vulnerabilities within the application.

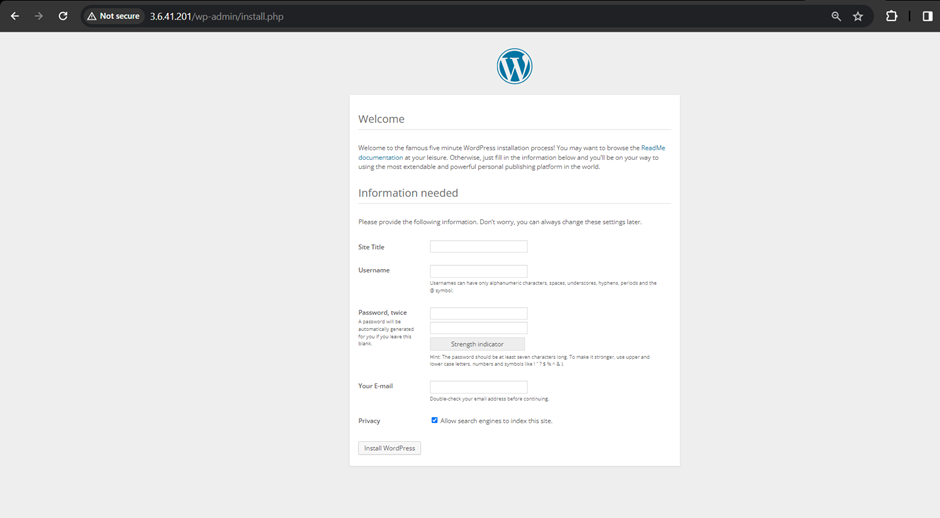
This experiment is organized into four main sections:

**Preparation:**

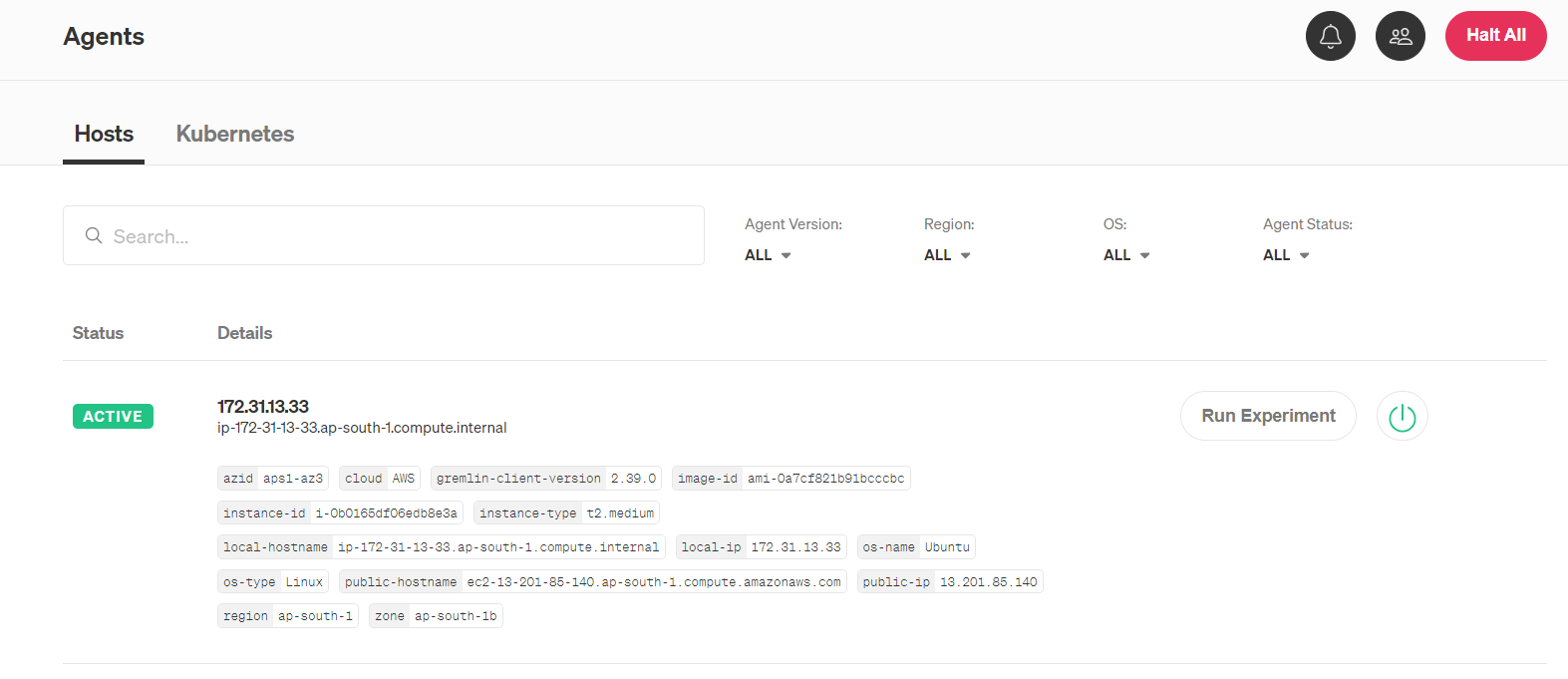
1. **AWS Infrastructure Setup:**
   * Launched an EC2 instance within an AWS Virtual Private Cloud (VPC).
   * Configured a public subnet to enable internet connectivity.
   * Connected the VPC to an Internet Gateway, allowing communication with the internet.



1. **Application Deployment:**
   * Installed a Git repository containing the dvwps: Damn Vulnerable WordPress Site web application on the EC2 instance.
   * Ensured the web application is accessible via a web browser over the internet.



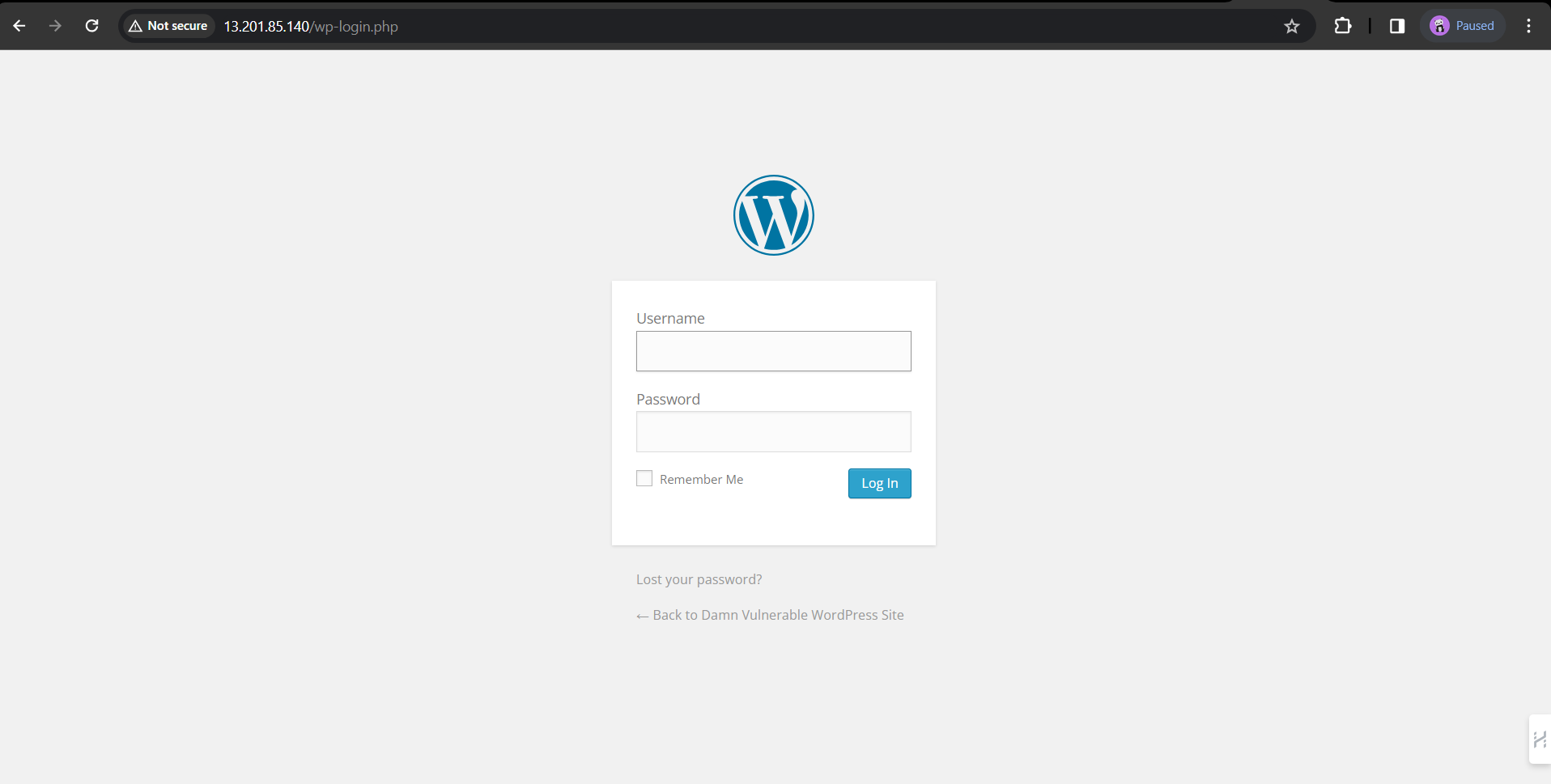
1. **Chaos Engineering Setup:**
   * Deployed a Gremlin Agent on a separate EC2 instance within the same public subnet.
   * Established connectivity between the Gremlin Agent and the Gremlin Chaos Engineering Platform.

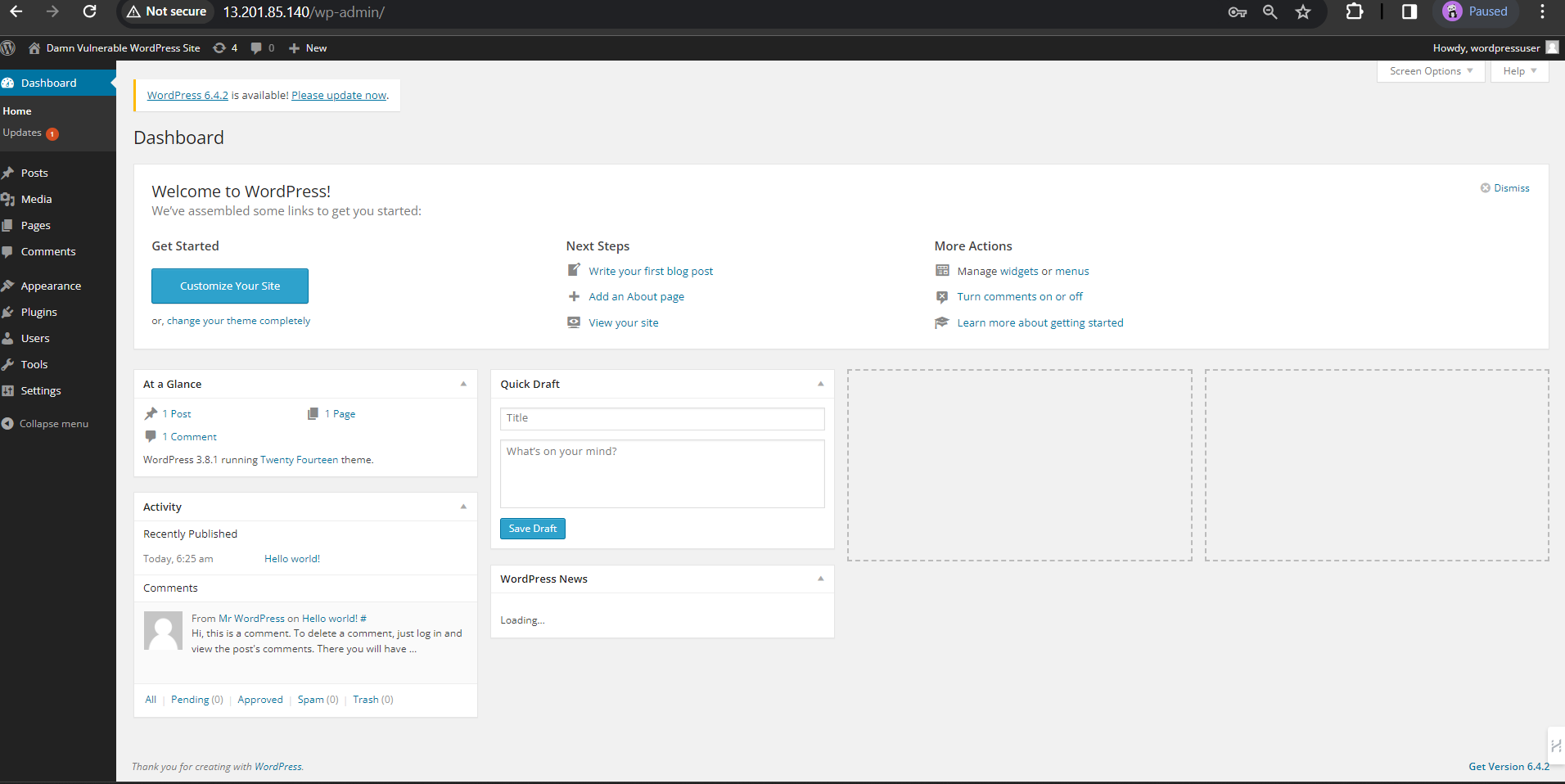


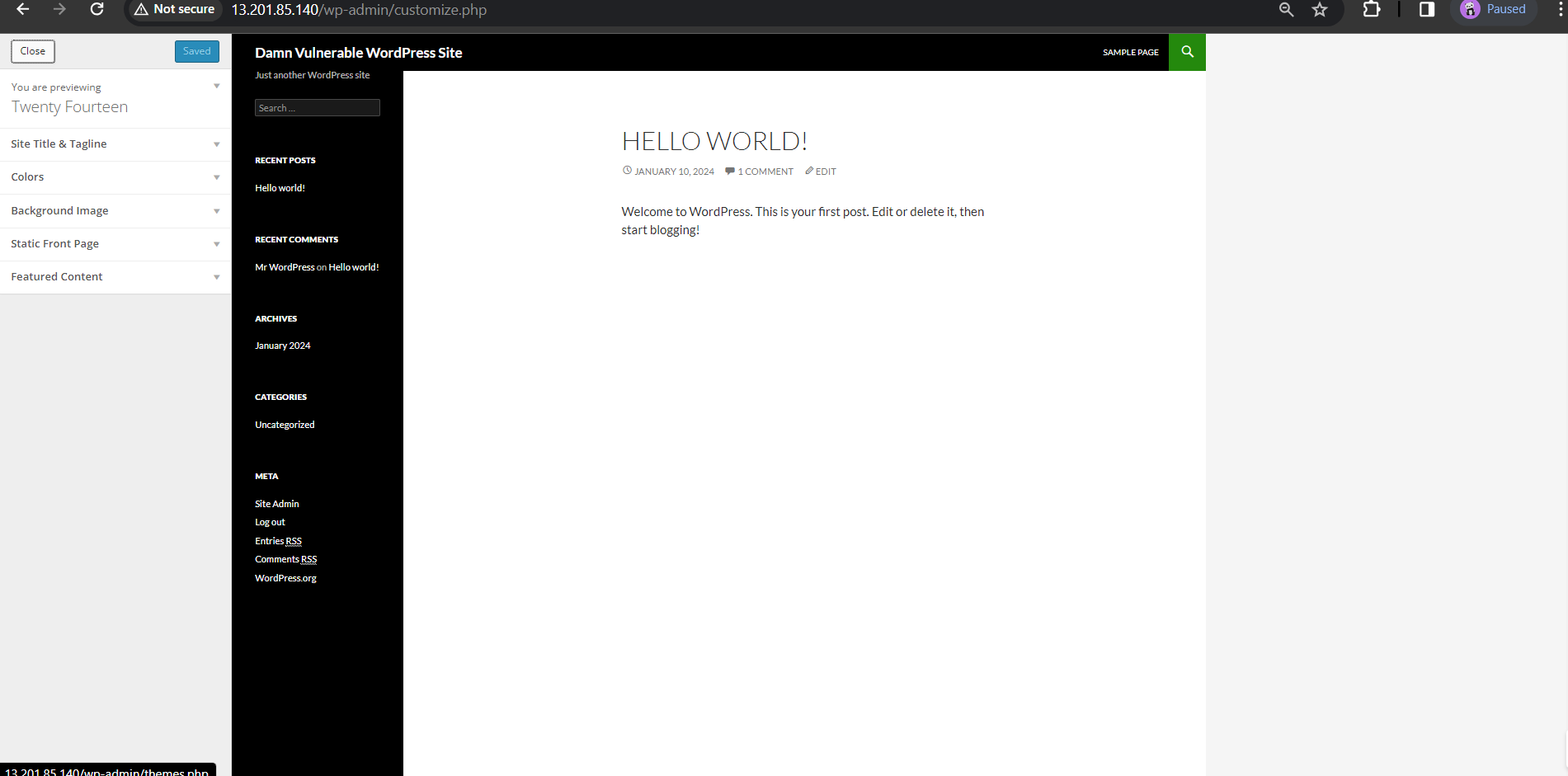
1. **Vulnerability Scanning:**
   * Install Trivy on the EC2 instance hosting the vulnerable WordPress setup.
   * Configure Trivy to scan the Docker container regularly for vulnerabilities.

**Implementation:**

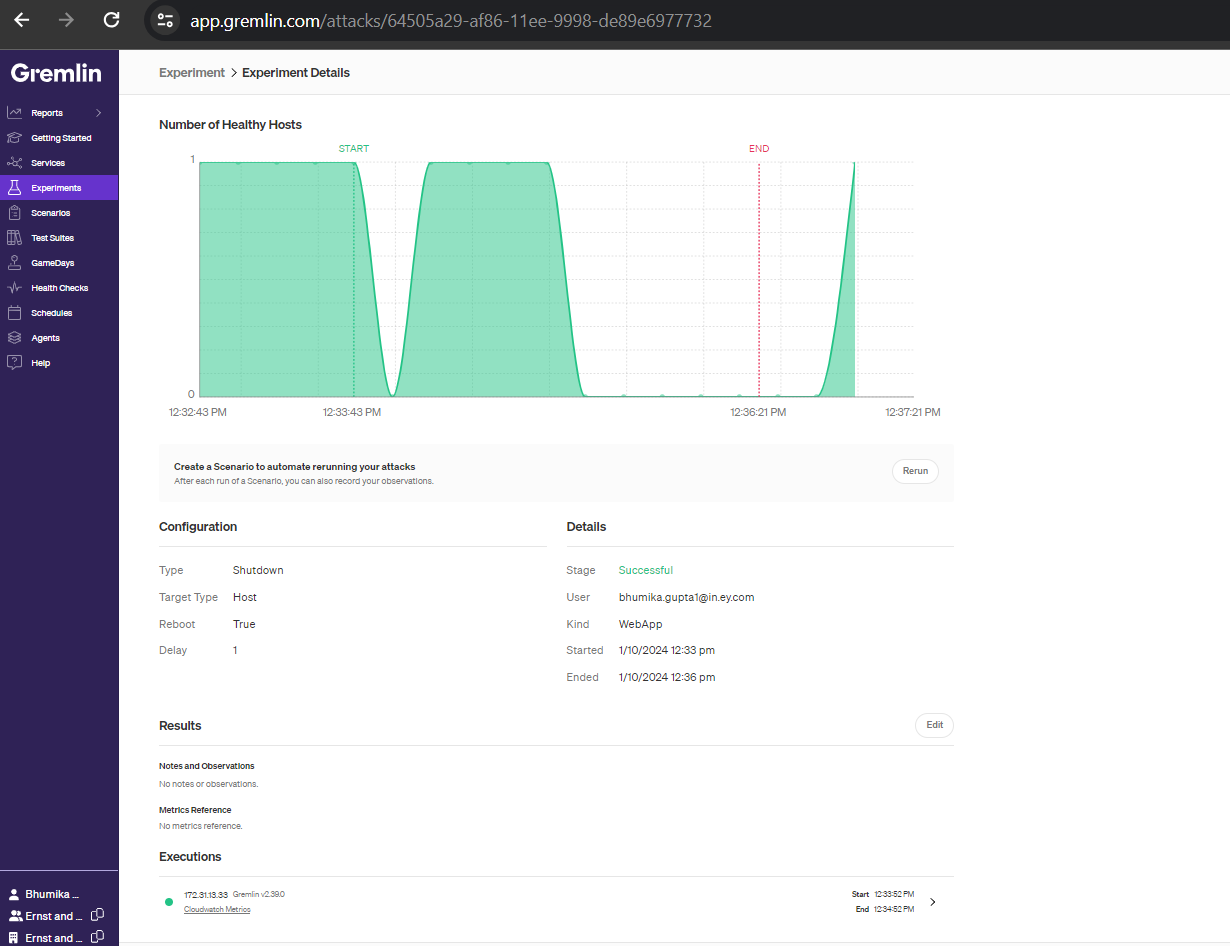
1. **Web Application Usage:**
   * Ensure the Docker container with the vulnerable WordPress setup is running on the EC2 instance.
   * Monitor the Git repository for version control and manage changes to the WordPress code and configurations.



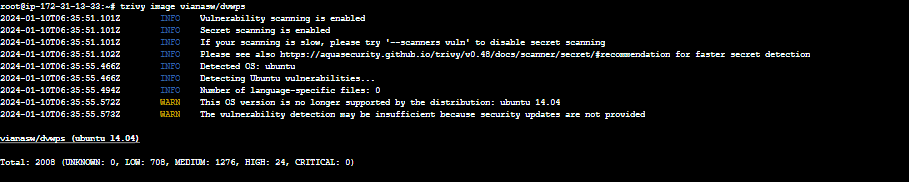




1. **Chaos Engineering Experiment:**
   * Initiated a controlled chaos experiment using the Gremlin Chaos Engineering Platform.
   * Experiment focused on inducing a shutdown scenario to assess the system's resilience.



1. **Vulnerability Mitigation:**
   * Schedule regular Trivy scans to assess the security posture of the Docker container.
   * Implement remediation measures for identified vulnerabilities, such as updating WordPress versions or patching plugins.



|  |  |  |
| --- | --- | --- |
| **Vulnerability** | **Severity** | **Weakness** |
| CVE-2015-3183 | MEDIUM | The product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly. |
| CVE-2018-2773 | LOW | Successful attacks of this vulnerability can result in unauthorized ability to cause a hang or frequently repeatable crash (complete DOS) of MySQL Server. |
| [CVE-2018-2767](https://nvd.nist.gov/vuln/detail/CVE-2018-2767) | LOW | Difficult to exploit vulnerability allows low privileged attacker with network access via multiple protocols to compromise MySQL Server. Successful attacks of this vulnerability can result in unauthorized read access to a subset of MySQ |
| CVE-2018-2755 | LOW | Successful attacks require human interaction from a person other than the attacker and while the vulnerability is in MySQL Server, attacks may significantly impact additional products. Successful attacks of this vulnerability can result in takeover of MySQL Server. |

**Observation:**

1. **Web Application Deployment:**
   * Observe user access to the vulnerable WordPress site through a web browser.
   * Monitor the EC2 instance for stability and resource utilization during normal usage.
2. **Chaos Engineering Configuration:**
   * Observe user access to the vulnerable WordPress site through a web browser.
   * Monitor the EC2 instance for stability and resource utilization during normal usage.
3. **Vulnerability Scanning:**
   * Observe user access to the vulnerable WordPress site through a web browser.
   * Monitor the EC2 instance for stability and resource utilization during normal usage.

**Analysis:**

1. **Web Application Stability:**
   * Observe user access to the vulnerable WordPress site through a web browser.
   * Monitor the EC2 instance for stability and resource utilization during normal usage.
2. **Chaos Experiment Outcomes:**
   * Observe user access to the vulnerable WordPress site through a web browser.
   * Monitor the EC2 instance for stability and resource utilization during normal usage.
3. **Vulnerability Analysis:**
   * Observe user access to the vulnerable WordPress site through a web browser.
   * Monitor the EC2 instance for stability and resource utilization during normal usage.