

Project Report on

**To solve the problem of managing and hardening servers concurrently**

**Submitted by**

**Neha Rote (240344223021)**

**Harsh Raj (240344223011)**

**Gauri Gagare (240344223009)**

**Prachi Tadse (240344223028)**

Under the guidance of

**Mr. Sandeep Walvekar**

**In partial fulfillment of the award of** **Post Graduate Diploma in**

**IT Infrastructure, Systems and Security**

**(PG-DITISS)**



**Sunbeam Institute of Information Technology,**

**Pune (Maharashtra)**

**PG-DITISS -2024**

# DECLARATION

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed**.**

Place: Pune

Date:

**Neha Rote Harsh Raj Gauri Gagare Prachi Tadse**

**(240344223021) (240344223011) (240344223009) (240344223028)**

# CERTIFICATE

This is to certify that the project report entitled **“To solve the problem of managing and hardening servers concurrently”** submitted by **Neha Rote** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

**Mr. Sandeep Walvekar Mr. Vishal Salunkhe**

Guide Course Coordinator

**Mr. Nitin Kudale**

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

# CERTIFICATE

This is to certify that the project report entitled **“To solve the problem of managing and hardening servers concurrently”**, submitted by **Harsh Raj** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

**Mr. Sandeep Walvekar Mr. Vishal Salunkhe**

Guide Course Coordinator

**Mr. Nitin Kudale**

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

# CERTIFICATE

This is to certify that the project report entitled **“To solve the problem of managing and hardening servers concurrently”**, submitted by **Gauri Gagare** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

**Mr. Sandeep Walvekar Mr. Vishal Salunkhe**

Guide Course Coordinator

**Mr. Nitin Kudale**

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

# CERTIFICATE

This is to certify that the project report entitled **“To solve the problem of managing and hardening servers concurrently”**, submitted by **Prachi Tadse** is the bonafide work completed under our supervision and guidance in partial fulfillment for the award of Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

**Mr. Sandeep Walvekar Mr. Vishal Salunkhe**

Guide Course Coordinator

**Mr. Nitin Kudale**

CEO

Sunbeam Institute of Information Technology

Pune (M.S.) – 411057

# APPROVAL CERTIFICATE

This Project II report entitled **“To solve the problem of managing and hardening servers concurrently”** by **Neha Rote (240344223021)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner:

**(Signature)**

**(Name)**

# APPROVAL CERTIFICATE

This Project II report entitled **“To solve the problem of managing and hardening servers concurrently”** by **Harsh Raj (240344223011)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner:

**(Signature)**

**(Name)**

# APPROVAL CERTIFICATE

This Project II report entitled **“To solve the problem of managing and hardening servers concurrently”** by **Gauri Gagare (240344223009)**  is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner:

**(Signature)**

**(Name)**

# APPROVAL CERTIFICATE

This Project II report entitled **“To solve the problem of managing and hardening servers concurrently”** by **Prachi Tadse (240344223028)** is approved for Post Graduate Diploma in IT Infrastructure, Systems and Security (PG-DITISS) of Sunbeam Institute of Information Technology, Pune (M.S.).

Place: Pune

Date:

Examiner:

**(Signature)**

**(Name)**

# CONTENTS

|  |  |
| --- | --- |
| **TITLE** | **PAGE NO** |
| **Declaration** |  |
| **Certificate** |  |
| **Approval Certificate** |  |
| **Abstract** | i |
| **1.INTRODUCION** | 1 |
| 1.1 Applications | 2 |
| 1.2 Organization and Project Plan | 5 |
| **2. LITERATURE SURVEY** | 6 |
| Paper 1 | 6 |
| Paper 2 | 6 |
| Paper 3 | 6 |
| **3. SYSTEM DEVELOPMENT AND DESIGN** | 7 |
| 3.1 Proposed System | 7 |
| 3.2 Flow Chart | 8 |
| 3.3 Technology used | 9 |
| 3.3.1 AWS EC2 | 9 |
| 3.3.2 Git | 10 |
| 3.3.3 Terraform | 11 |
| 3.3.4 CIS Benchmark | 14 |
| 3.3.5 Jenkins | 17 |
| 3.3.6 Aws Inspector | 22 |
| 3.3.7 Ansible & Packer | 23 |
| 3.4 Workflow | 29 |
| **4. PROJECT OUTPUT** | 30 |
| **5. CONCLUSION** | 33 |
| 5.1 Conclusion | 34 |
| 5.2 Future Scope | 34 |
| **REFERENCES** | 3t5 |

# ABSTRACT

In today's fast-paced tech world, managing and securing servers is more important than ever. This project aims to tackle both tasks at the same time, making it easier to keep servers running smoothly and securely.

Our solution combines automation, real-time monitoring, and advanced security techniques to create a powerful system for managing and hardening servers. By automating routine tasks and security updates, we reduce human errors and allow IT staff to focus on more important work. Real-time monitoring keeps a constant watch on server performance and security, helping to quickly spot and fix any issues.

Our project focuses on several key components to enhance server management and security. First, we leverage automation by using scripts and tools to handle tasks such as server setup, applying updates, and ensuring compliance with security standards. This approach saves time and guarantees consistency in server operations. Additionally, we implement real-time monitoring tools that continuously track server health, performance, and security. This enables immediate detection and response to any potential problems, ensuring servers run smoothly and securely.

Moreover, we prioritize security hardening by following best practices to fortify server defenses. This involves managing configurations, controlling access, and regularly checking for vulnerabilities to prevent potential threats. To streamline the management process, we also develop a centralized management dashboard. This user-friendly interface allows administrators to manage and monitor servers, view alerts, and check logs all in one place. By integrating these components, our project aims to create a robust and efficient system for concurrent server management and hardening.

  Our goal is to build a seamless and scalable system that makes server management and security easier and more effective. This solution not only strengthens the security and reliability of servers but also improves the efficiency of IT operations. By doing so, we help ensure that servers remain safe from threats while performing at their best.

# INTRODUCTION

The project aims to enhance the security and manageability of servers by implementing a comprehensive and automated solution for concurrent server hardening and management. By leveraging a combination of industry-standard tools and technologies, this project ensures that all servers meet rigorous security standards while maintaining operational efficiency. The tools and technologies used in this project include CIS benchmarks, network hardening techniques, Terraform, Packer, AWS EC2, AWS Inspector, Git, and Jenkins.

AWS EC2: Amazon Elastic Compute Cloud (EC2) provides scalable computing capacity in the cloud, allowing users to run applications on virtual servers. EC2 offers flexibility in choosing instance types, configurations, and scaling, making it easier to deploy and manage servers efficiently.

Git: Git is a distributed version control system that enables teams to track changes in source code during software development. By using Git, the project ensures version control for infrastructure code, allowing for easy collaboration, change tracking, and rollback if necessary.

Terraform: Terraform is an open-source infrastructure as code (IaC) tool that allows for the automated provisioning and management of server infrastructure. With Terraform, server configurations can be defined in code, enabling consistent and repeatable deployments across different environments.

CIS Benchmarks: The Center for Internet Security (CIS) benchmarks provide globally recognized best practices for securing IT systems and data. These benchmarks offer detailed configuration guidelines to help organizations harden their servers against common vulnerabilities and threats.

AWS Inspector: AWS Inspector is a security assessment service that helps improve the security and compliance of applications deployed on AWS. It performs automated security assessments to identify vulnerabilities and deviations from best practices, providing detailed reports and recommendations for remediation.

Packer: Packer is an open-source tool used for creating identical machine images for multiple platforms from a single source configuration. Packer automates the creation of server images, ensuring that all servers are provisioned with the same base configuration, thereby reducing configuration drift.

Network Hardening Techniques: These techniques involve securing the network infrastructure by implementing measures such as firewalls, intrusion detection systems, and network segmentation. By reducing the attack surface and controlling access, network hardening helps protect servers from unauthorized access and attacks.

By integrating these tools and technologies, the project aims to create a robust solution for managing and hardening servers concurrently. This approach not only ensures compliance with best practices but also reduces the risk of security breaches and enhances operational efficiency. The ability to harden multiple servers simultaneously and maintain consistent security standards across the infrastructure is crucial in today's fast-paced and ever-evolving IT landscape.

## Applications

* **Enterprise IT Infrastructure:** Large organizations with extensive IT infrastructures can benefit from automated server management and hardening to ensure consistent security standards across all servers. This reduces the risk of security breaches and enhances operational efficiency, allowing IT teams to focus on strategic initiatives.
* **Cloud Service Providers:** Cloud service providers can use this solution to manage and secure the servers that host their clients' applications and data. By maintaining high security standards and operational efficiency, providers can offer more reliable and secure services to their customers.
* **DevOps Environments:** In DevOps environments, the integration of tools like Terraform, Packer, Git, and Jenkins facilitates continuous integration and continuous deployment (CI/CD). This ensures that server configurations are consistently applied and updated, reducing configuration drift and enabling rapid deployment of changes.
* **E-commerce Platforms:** E-commerce platforms require robust security measures to protect sensitive customer data and ensure the availability of their services. Automated server hardening and management help maintain high security standards and optimize server performance, supporting a seamless user experience.
* **Financial Institutions:** Financial institutions handle sensitive financial data and are frequent targets of cyberattacks. Implementing automated server management and hardening enhances the security and compliance of their IT infrastructure, safeguarding critical data and operations.
* **Healthcare Providers:** Healthcare providers must protect patient data and comply with stringent regulatory requirements. Automated server management and hardening ensure that servers are configured securely and compliant with regulations like HIPAA, minimizing the risk of data breaches.
* **Government Agencies:** Government agencies can use this solution to manage and secure their IT infrastructures, ensuring that all servers comply with security standards and regulations. This enhances the security of public sector data and services, protecting them from cyber threats.
* **Telecommunications:** Telecommunications companies rely on robust and secure IT infrastructures to provide uninterrupted services to their customers. Automated server management and hardening help maintain the reliability and security of their networks, supporting consistent service delivery.
* **Education Institutions:** Educational institutions can benefit from automated server management to ensure that their IT resources are secure and efficiently managed. This supports the delivery of online learning platforms and the protection of student and faculty data.
* **Small and Medium Enterprises (SMEs):** SMEs often lack the resources to manage and secure their IT infrastructure manually. Automated solutions provide an affordable and effective way to maintain high security standards and operational efficiency, enabling them to compete with larger organizations.

## 1.2 Project Plan

**Table: Activities Details**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **ACTIVITY** | **WEEK** | | | |
| **1** | **2** | **3** | **4** |
| 1 | Project group formation |  |  |  |  |
| 2 | Project work to be started in respective labs |  |  |  |  |
| 3 | First review with PPT presentation |  |  |  |  |
| 4 | Design Use-Case view as per project |  |  |  |  |
| 5 | Design Block diagram as per project |  |  |  |  |
| 6 | Second review with PPT presentation |  |  |  |  |
| 7 | Selection |  |  |  |  |
| 8 | Final review with PPT presentation |  |  |  |  |
| 9 | Implementation coding as per project |  |  |  |  |
| 10 | Testing, Troubleshooting with different techniques |  |  |  |  |
| 11 | Created Soft copy of project and then final hard copy |  |  |  |  |

# LITERATURE SURVEY

## **Paper 1**: - A Qualitative Study of DevOps Usage in Practice

**Author:** Floris Erich, C. Amrit & M. Daneva

**Description:** Organizations are introducing agile and lean software development techniques in operations to increase the pace of their software development process and to improve the quality of their software. They use the term DevOps, a portmanteau of development and operations, as an umbrella term to describe their efforts. In this paper we describe the ways in which organizations implement DevOps and the outcomes they experience. We first summarize the results of a Systematic Literature Review that we performed to discover what researchers have written about DevOps. We then describe the results of an exploratory interview-based study involving six organizations of various sizes that are active in various industries. As part of our findings, we observed that all organizations were positive about their experiences and only minor problems were encountered while adopting DevOps.

## **Paper 2:** **- Automating Security Hardening and Compliance in Cloud Environments**

**Author:** Smith et al. (2019)

**Description**: This study explores the use of automation tools to enforce security policies and compliance in cloud environments. The authors demonstrate how tools like Terraform and Packer can be used to automate server provisioning and configuration, ensuring that security benchmarks are consistently applied across all servers.

## **Paper 3: -** A Comprehensive Approach to Server Management Using CI/CD Pipelines

## Author: Johnson and Miller (2020)

**Description:** Johnson and Miller discuss the implementation of continuous integration and continuous deployment (CI/CD) pipelines using Jenkins and Git. They illustrate how these tools can automate the deployment of security updates and configuration changes, reducing the risk of human error and enhancing the efficiency of server management.

# SYSTEM DEVELOPMENT AND DESIGN

## 3.1 Proposed System

The proposed system aims to address the concurrent challenges of managing and hardening servers by integrating a suite of automated tools and technologies, thereby enhancing security, operational efficiency, and compliance. It leverages Terraform for infrastructure provisioning, Packer for creating standardized server images, and CIS benchmarks for enforcing security best practices. Network hardening techniques and AWS Inspector are employed for continuous security assessment and vulnerability management. Real-time monitoring and alerting are facilitated through tools like CloudWatch and Prometheus, ensuring immediate issue detection and response. Git and Jenkins streamline version control and automate the CI/CD pipeline, promoting seamless updates and consistent server configurations. A centralized management dashboard provides a user-friendly interface for monitoring server status, alerts, and compliance reports, while comprehensive logging ensures auditability and regulatory compliance. This integrated approach not only fortifies server security and optimizes management processes but also scales efficiently across various IT environments, making it a robust solution for modern organizations.

## 3.2 Flow chart





**Figure: Flowchart**

## 3.3 Technology used

### 3.3.1 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service provided by Amazon it allows you to rent virtual servers in the cloud, known as instances, to run your applications and workloads. EC2 provides a scalable and flexible infrastructure that enables you to quickly deploy and manage virtual servers without the need to invest in physical hardware.

**Key features**

* Instances: These are virtual servers that you can launch and manage in the cloud. You can choose from a wide range of instance types optimized for different use cases, such as general-purpose computing, memory-intensive tasks, high-performance computing, and more.
* Elasticity and Scaling: EC2 allows you to easily scale your infrastructure up or down based on your workload's demands. You can create multiple instances or change the instance type to handle varying levels of traffic.
* Amazon Machine Images (AMIs): AMIs are pre-configured templates that contain an operating system and often additional software needed to launch an instance. You can use AWS-provided AMIs or create your own custom AMIs.
* Security Groups: Security groups act as virtual firewalls that control inbound and outbound traffic to your instances. You can define rules to allow or deny specific types of traffic.
* Cost Management: EC2 offers various pricing options, including On-Demand Instances, Reserved Instances, and Spot Instances, which provide flexibility in managing costs based on your usage patterns.

### 3.3.2 Git

Git is a distributed version control system (VCS) designed to manage source code history and facilitate collaborative software development.

**Key features of Git:**

* Distributed Architecture: Unlike centralized version control systems, Git is distributed. Each developer has a complete copy of the repository, including its entire history. This allows for offline work, faster operations, and improved resilience.
* Branching and Merging: Git makes it easy to create branches, which are separate lines of development. Developers can work on features, bug fixes, or experiments in their own branches without affecting the main codebase. Merging branches back together is relatively simple and allows for collaborative development.
* Commit History: Git maintains a detailed history of changes to the codebase. Each change is represented by a commit, which includes information about who made the change, when it was made, and what was changed. This commit history provides a clear view of the evolution of the project.
* Fast and Efficient: Git is designed for speed and efficiency. Most operations are local, as the repository resides on the developer's machine. This results in rapid commits, branching, and merging.
* Collaboration: Git enables effective collaboration among developers. Multiple developers can work on different branches simultaneously, and changes can be shared by pushing them to a remote repository. Pull requests or merge requests facilitate the process of reviewing and integrating changes from different contributors.

### 3.3.3 Terraform

Terraform, developed by HashiCorp, is an open-source infrastructure as code (IaC) tool designed to automate the provisioning, management, and scaling of infrastructure resources. Using HCL (HashiCorp Configuration Language), and optionally JSON, Terraform allows users to define infrastructure resources in code, facilitating consistent deployments across diverse environments. It supports a broad range of platforms, including major cloud providers like AWS, Azure, and Google Cloud, as well as on-premises environments and various other infrastructure services.

**Key features of Terraform:**

* Declarative Configuration:
  + Users define their infrastructure requirements in a declarative configuration language (HCL). This means you describe the desired state of the infrastructure, and Terraform figures out how to achieve that state.
* Infrastructure as Code (IaC):
  + Terraform allows infrastructure to be managed as code, which can be versioned, stored in version control systems (e.g., Git), and reviewed. This approach promotes repeatability, consistency, and collaboration.
* Provisioning and Management:
  + Terraform automates the provisioning and management of infrastructure resources. It supports creating, updating, and deleting resources across various providers using a single configuration file.
* State Management:
  + Terraform maintains a state file that tracks the current state of the infrastructure. This file is used to determine what changes need to be made to reach the desired state specified in the configuration files.
* Execution Plans:
  + Before applying changes, Terraform generates an execution plan that shows what actions will be taken to achieve the desired state. This allows users to review changes and avoid unintended modifications.
* Modular Design:
  + Terraform configurations can be organized into reusable modules, which encapsulate common infrastructure patterns and best practices. Modules can be shared and reused across different projects, promoting consistency and efficiency.
* Provider Plugins:
  + Terraform uses provider plugins to interact with different infrastructure platforms and services. Each provider is responsible for managing the lifecycle of resources on its respective platform (e.g., AWS, Azure, Google Cloud).
* Resource Graph:
  + Terraform builds a dependency graph of the infrastructure resources defined in the configuration. This graph helps Terraform understand the relationships between resources and optimize the order in which they are created or updated.
* Multi-Cloud and Hybrid Cloud Support:
  + Terraform can manage resources across multiple cloud providers and on-premises environments from a single configuration. This is useful for organizations using a multi-cloud or hybrid cloud strategy.
* Community and Ecosystem:
  + Terraform has a strong community and ecosystem with numerous modules, providers, and extensions available in the Terraform Registry. This community support helps users find solutions and best practices for various use cases.
* **Configuration Files**

provider "aws" {

region = "ap-south-1"

}

resource "aws\_instance" "intro" {

ami = "ami-0ad21ae1d0696ad58"

instance\_type = "t2.micro"

availability\_zone = "ap-south-1a"

key\_name = "kkk"

vpc\_security\_group\_ids = ["sg-03dbba0c29ba8dca5"]

tags = {

Name = "ubuntu-instance"

}

}

### 3.3.4 CIS Benchmark

The CIS (Center for Internet Security) Benchmarks are a set of best practices for securely configuring IT systems, software, networks, and cloud infrastructure. These benchmarks provide detailed, step-by-step guidance on how to secure various types of systems and are widely used by organizations to enhance their cybersecurity posture

**Key Features**

* Detailed Security Guidelines:
  + CIS Benchmarks offer specific configuration settings and practices to enhance the security of systems. This includes guidance on settings like password policies, user permissions, and network security configurations.
* Configuration Hardening:
  + The benchmarks provide recommendations for hardening systems by disabling unnecessary services, enforcing secure configurations, and implementing access controls. This helps reduce the attack surface and prevent unauthorized access.
* Checklists and Actionable Steps:
  + Each benchmark includes detailed checklists with actionable steps to implement the recommended security measures. These checklists make it easier for organizations to apply and verify security controls.
* Platform-Specific Benchmarks:
  + CIS Benchmarks are available for a variety of platforms, including major operating systems (Windows, Linux), cloud providers (AWS, Azure, Google Cloud), and network devices (firewalls, routers). This allows for consistent security practices across different technologies.
* Automated Assessment Tools:
  + The benchmarks are often integrated with automated assessment tools and scripts that help organizations evaluate their systems' compliance with the recommended configurations. These tools streamline the process of applying and verifying security controls.
* Regular Updates:
  + CIS Benchmarks are regularly updated to address emerging threats and vulnerabilities. The updates ensure that the guidelines remain relevant and effective in the face of evolving security challenges.
* Community Contributions:
  + The benchmarks are developed with input from a wide range of cybersecurity professionals, vendors, and organizations. This collaborative approach ensures that the guidelines reflect industry best practices and real-world threats.
* Compliance and Reporting:
  + Adherence to CIS Benchmarks can help organizations achieve compliance with various regulatory requirements and industry standards. They provide a foundation for generating compliance reports and conducting security audits.
* Training and Resources:
  + CIS offers additional resources, including training materials and documentation, to support organizations in implementing and maintaining the benchmarks. These resources help organizations understand and apply the guidelines effectively.

**Configuration Files**

vulnerability.sh\_oscap

#!/bin/bash

# Ensure the script is run as root

if [ "$(id -u)" -ne "0" ]; then

echo "This script must be run as root" 1>&2

exit 1

fi

# Path to the SCAP data stream file

SCAP\_DS="/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml"

# Path to the profile ID

PROFILE\_ID="xccdf\_org.ssgproject.content\_profile\_cis\_server\_l1"

# Paths for results and report

RESULTS\_FILE="arf.xml"

REPORT\_FILE="report.html"

# Create a test vulnerability:

# Example: Disable firewall (this should generally be enabled in secure systems)

echo "Disabling firewall as a test vulnerability..."

systemctl stop firewalld

systemctl disable firewalld

# Example: Allow root login via SSH (generally should be disabled for security)

echo "Allowing root login via SSH..."

sed -i 's/^PermitRootLogin .\*$/PermitRootLogin yes/' /etc/ssh/sshd\_config

systemctl restart sshd

# Run OpenSCAP evaluation

echo "Running OpenSCAP evaluation..."

oscap xccdf eval --profile $PROFILE\_ID --results-arf $RESULTS\_FILE --report $REPORT\_FILE $SCAP\_DS

# Output result

echo "OpenSCAP evaluation completed. Check $REPORT\_FILE for details."

# Clean up: Optionally revert changes after testing

# Uncomment the following lines if you want to restore the system to its original state

# echo "Reverting changes..."

# systemctl enable firewalld

# systemctl start firewalld

# sed -i 's/^PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd\_config

# systemctl restart sshd

### 3.3.5 Jenkins

Jenkins is an open-source automation server that facilitates the continuous integration and continuous delivery (CI/CD) of software projects. It helps automate various tasks related to building, testing, and deploying applications, making the development and release process more efficient and reliable.

**Key features of Jenkins:**

* Continuous Integration:
* Jenkins automates the process of integrating code changes from multiple contributors into a shared repository. It triggers builds whenever code is committed, allowing developers to identify and fix integration issues early.
* Automated Builds:
* Jenkins can automatically build projects from source code repositories. It supports various build tools, languages, and platforms, making it versatile for different types of projects.
* Extensibility:
* Jenkins can be extended through a wide range of plugins that provide additional functionalities. Plugins are available for source code management, build tools, testing frameworks, and deployment options.
* Pipeline as Code:
* Jenkins uses a domain-specific language called Groovy to define build pipelines as code. This enables you to define complex workflows that include build, test, and deployment stages in a version-controlled script.
* Continuous Delivery:
* Jenkins supports continuous delivery by automating the deployment process after successful builds. It can deploy applications to different environments, such as development, staging, and production.
* Distributed Builds:
* Jenkins can distribute builds across multiple machines, allowing for parallel builds and improved build performance. This is particularly useful for large and resource-intensive projects.

**Configuration File**

pipeline {

agent any

environment {

AWS\_REGION = 'us-west-2' //

TERRAFORM\_DIR = 'terraform'

ANSIBLE\_DIR = 'ansible'

PACKER\_DIR = 'packer'

}

stages {

stage('Checkout Code') {

steps {

checkout scm

}

}

stage('Terraform Apply') {

steps {

script {

dir(TERRAFORM\_DIR) {

sh 'terraform init'

sh 'terraform apply -auto-approve'

}

}

}

}

stage('Wait for EC2 Instance') {

steps {

script {

sleep(time: 60, unit: 'SECONDS')

}

}

}

stage('Run AWS Inspector') {

steps {

script {

sh '''

aws inspector start-assessment-run --assessment-template-arn arn:aws:inspector:ap-south-1:123456789012:template/0-abcdefg

'''

}

}

}

stage('Run OpenSCAP') {

steps {

script {

sh '''

oscap xccdf eval --profile xccdf\_org.ssgproject.content\_profile\_anssi\_bp28\_enhanced --results-arf arf.xml --report report.html /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml

'''

}

}

}

stage('Apply Ansible Remediation') {

steps {

script {

dir(ANSIBLE\_DIR) {

sh 'ansible-playbook -i inventory playbook.yml'

}

}

}

}

stage('Create AMI with Packer') {

steps {

script {

dir(PACKER\_DIR) {

sh 'packer build packer-template.json'

}

}

}

}

}

post {

always {

echo 'Cleaning up...'

}

success {

echo 'Pipeline succeeded!'

}

failure {

echo 'Pipeline failed!'

}

}

}

### 3.3.6 Aws Inspector

AWS Inspector is a security assessment service offered by Amazon Web Services (AWS) that helps improve the security and compliance of applications deployed on AWS. It automatically assesses applications for vulnerabilities and deviations from best practices. Here’s an overview of AWS Inspector and its key features:

**Key Features:**

* Automated Security Assessments:
* AWS Inspector automatically performs security assessments of your applications and infrastructure. It scans for vulnerabilities, misconfigurations, and deviations from security best practices without manual intervention.
* Pre-Built Assessment Rules:
* The service includes pre-built assessment rules based on industry standards and best practices. These rules cover a wide range of security issues, such as missing patches, insecure configurations, and common vulnerabilities.
* Custom Assessment Rules:
* Users can create and apply custom assessment rules tailored to their specific security needs. This flexibility allows for targeted assessments based on unique application requirements.
* Detailed Assessment Reports:
* AWS Inspector provides detailed reports that include findings from the security assessments. These reports highlight vulnerabilities, misconfigurations, and potential risks, and offer actionable recommendations for remediation.
* Integration with AWS Security Services:
* AWS Inspector integrates with other AWS security services such as AWS CloudTrail (for logging and monitoring), AWS Config (for compliance tracking), and AWS Security Hub (for centralized security management). This integration provides a comprehensive view of your security posture.
* Continuous Monitoring:
* The service supports continuous monitoring of your applications and infrastructure. It can be configured to run assessments on a scheduled basis or triggered by specific events, ensuring ongoing security vigilance.
* Compliance Frameworks:
* AWS Inspector helps assess compliance with various security frameworks and standards, such as CIS benchmarks and PCI DSS. It provides insights into how well your configurations align with these standards.
* Integration with CI/CD Pipelines:
* AWS Inspector can be integrated into continuous integration and continuous deployment (CI/CD) pipelines to automatically assess security during the development and deployment phases. This helps identify and address security issues early in the software lifecycle.
* Scalability:
* The service is designed to scale with your infrastructure, handling assessments for a large number of EC2 instances and other resources as your environment grows.
* Cost-Effective:
* AWS Inspector operates on a pay-as-you-go pricing model, where you pay only for the assessments performed. This makes it a cost-effective solution for maintaining security and compliance.

### 3.3.7 Ansible & Packer

**Ansible** is an open-source automation tool used for IT tasks such as configuration management, application deployment, orchestration, and provisioning. It is designed to be simple to use, reliable, and highly extensible, making it popular in DevOps environments.

**Key Features**

* Agentless Architecture:
* Ansible does not require any agent software to be installed on the managed nodes. It uses SSH for communication, making it easier to manage and reducing overhead.
* Simple YAML Syntax:
* Ansible playbooks, the configuration, deployment, and orchestration language forAnsible, are written in YAML, which is easy to read and write.
* Idempotency:
* Ansible ensures that repeated application of a playbook will have the same effect as running it once. This means you can run the same playbook multiple times without changing the final state of the system.
* Modular Architecture:
* Ansible uses modules, which are reusable, standalone scripts that can be used to perform tasks. There are hundreds of built-in modules, and you can also write custom ones.

**Configuration File**

remediation.yml

- name: Remediate Vulnerabilities

hosts: your\_ec2\_instance

become: yes

tasks:

- name: Update system packages

yum:

name: "\*"

state: latest

- name: Stop and disable httpd service

service:

name: httpd

state: stopped

enabled: no

- name: Stop and disable vsftpd service

service:

name: vsftpd

state: stopped

enabled: no

- name: Remove httpd and vsftpd packages

yum:

name:

- httpd

- vsftpd

state: absent

- name: Remove git package

yum:

name: git

state: absent

- name: Remove DVWA directory

file:

path: /var/www/html/DVWA

state: absent

- name: Close firewall ports

firewalld:

port:

- 80/tcp

- 21/tcp

state: absent

permanent: yes

- name: Reload firewall

command: firewall-cmd --reload

- name: Optional cleanup (uncomment if needed)

yum:

name: "{{ item }}"

state: absent

loop:

- <any-dependency-to-remove>

**Packer** is an open-source tool developed by HashiCorp used for creating identical machine images for multiple platforms from a single source configuration. It simplifies the process of provisioning and maintaining server images, ensuring consistency across different environments.

**Key Features**

* Multi-Platform Support:
  + Packer can create images for various platforms from a single configuration file. This includes cloud environments (e.g., AWS AMIs, Azure VM images), virtualization platforms (e.g., VMware, VirtualBox), and container formats (e.g., Docker).
* Configuration as Code:
  + Packer uses a declarative configuration file (written in JSON or HCL) to define the desired state of the machine image. This configuration file specifies the base image, provisioning scripts, and post-processing steps.
* Provisioners:
  + Provisioners are used to install and configure software on the machine image. Packer supports various provisioners such as Shell scripts, Ansible, Chef, and Puppet. This allows for the automation of software installation and configuration.
* Post-Processors:
  + Post-processors modify or convert the machine image after it has been created. For example, they can be used to compress the image, upload it to a cloud provider, or convert it into a different format.
* Immutable Infrastructure:
  + Packer supports the concept of immutable infrastructure by creating images that are pre-configured and tested. This reduces the need for manual configuration and updates on live systems, as changes are applied by creating new images.
* Parallel Builds:
  + Packer can build multiple images in parallel, which speeds up the image creation process. This is particularly useful for creating images for different platforms or regions simultaneously.
* Version Control Integration:
  + Packer configurations can be versioned and managed using version control systems like Git. This enables teams to track changes, collaborate on image configurations, and ensure consistency across different environments.
* Automated Image Updates:
  + With Packer, you can automate the creation of updated images, which helps in maintaining consistent environments with the latest software and security patches.
* Extensible and Modular:
  + Packer is highly extensible through plugins, allowing users to create custom provisioners, post-processors, and builders. This modular approach supports a wide range of use cases and integrations.
* Integration with CI/CD Pipelines:
  + Packer can be integrated into continuous integration and continuous deployment (CI/CD) pipelines to automate the creation and deployment of images as part of the software delivery process.

**Configuration File**

**template.json**

{

"builders": [

{

"type": "amazon-ebs",

"region": "ap-south-1",

"source\_ami": "i-041fcb678dad9e4d8",

"instance\_type": "t2.micro",

"ssh\_username": "ec2-user",

"ami\_name": "rhel-test",

"ami\_description": "AMI created with Packer",

"tags": {

"Name": "packer-builder"

},

"run\_tags": {

"Name": "packer-builder"

}

}

]

}

# 3.4 Workflow

To Solve the problem of managing hardening Server Concurrently. (using tools and technology like terraform Packer, Jenkins, (Is benchmark etaws. inspector nessuse etc).

In this project we focused on automating and streamlining the process of managing and hardening servers concurrently to ensure robust security and compliance across distributed environment. a

first we designed and implemented an automated CI/CD pipeline using terraform to provision and manage ecz instances. This pipeline ensures consistent, Scalable, and repeatable infrastructure deployment in Aws. after that we created a Bash Script that automates the generation of dummy vulnerability by iterrating through predefined patterns of common security issues, such as weak Outdated software Versions. This and script simulates vulnerabilities for testing and validation purposes in Security tools and environment also we are using Git and Jenkins, enabling a seamless workflow for code updates and ensuring all changes were throughy tested before deployment. That then we run identify Security scan pipeline to vulnerabilities before applying remediation steps, and the generated report is automatically pushed to an 53 bucket for further analysis and storage Next we auto male the appn of security fixes using shell script, which ensures compliance with security benchmarks. by directly running the necessary remediation steps. Amazon inspector genarates finding by implementing specific fixes which are applied.

This pipeline re-ran Security scans to Ensure that all vulnerabilities were resolved and provided Comprehensive Finding reports for final review at last we developed pipeline using packer automate the creation of hardened EC2 images this pipeline integtates security benchmarks and to ensure the AMI Configuration were compliant with industry standards before deployment.

This project resulted in a more security, Scalable and efficient server mangement process, Significantly reducing the risk of security breaches and ensuring compliance with industry regulations.

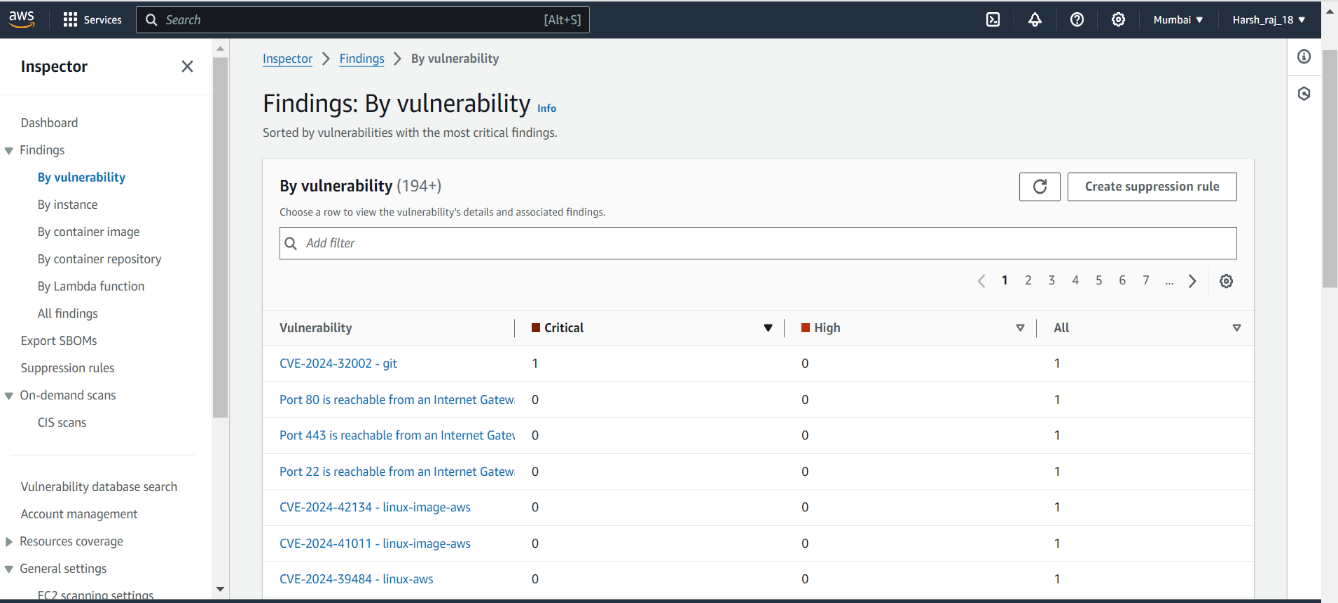
# Project Output

## AWS EC2

A screenshot of a computer

Description automatically generated

## AWS Inspector

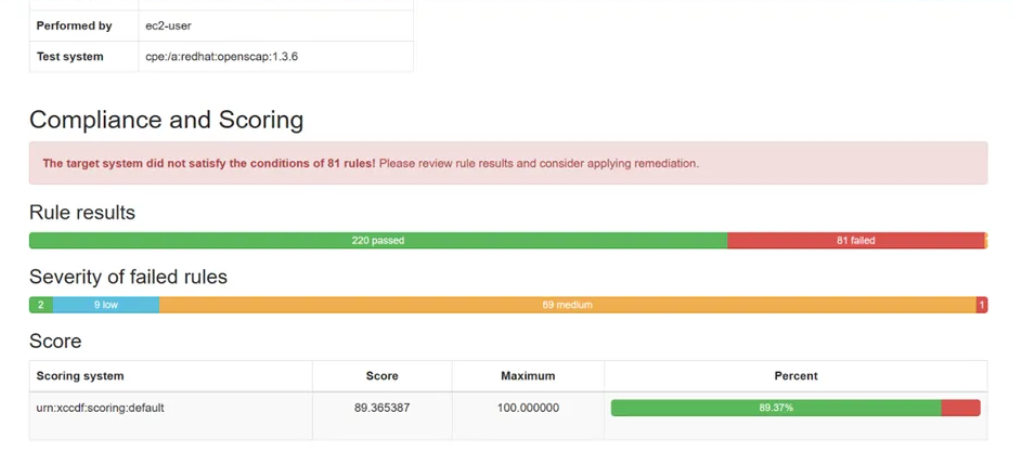


A screenshot of a computer

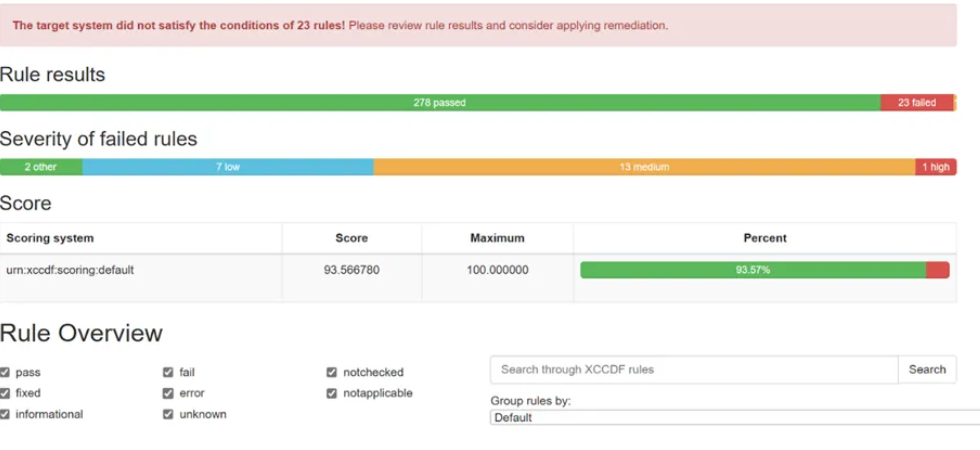
Description automatically generated

## Openscap

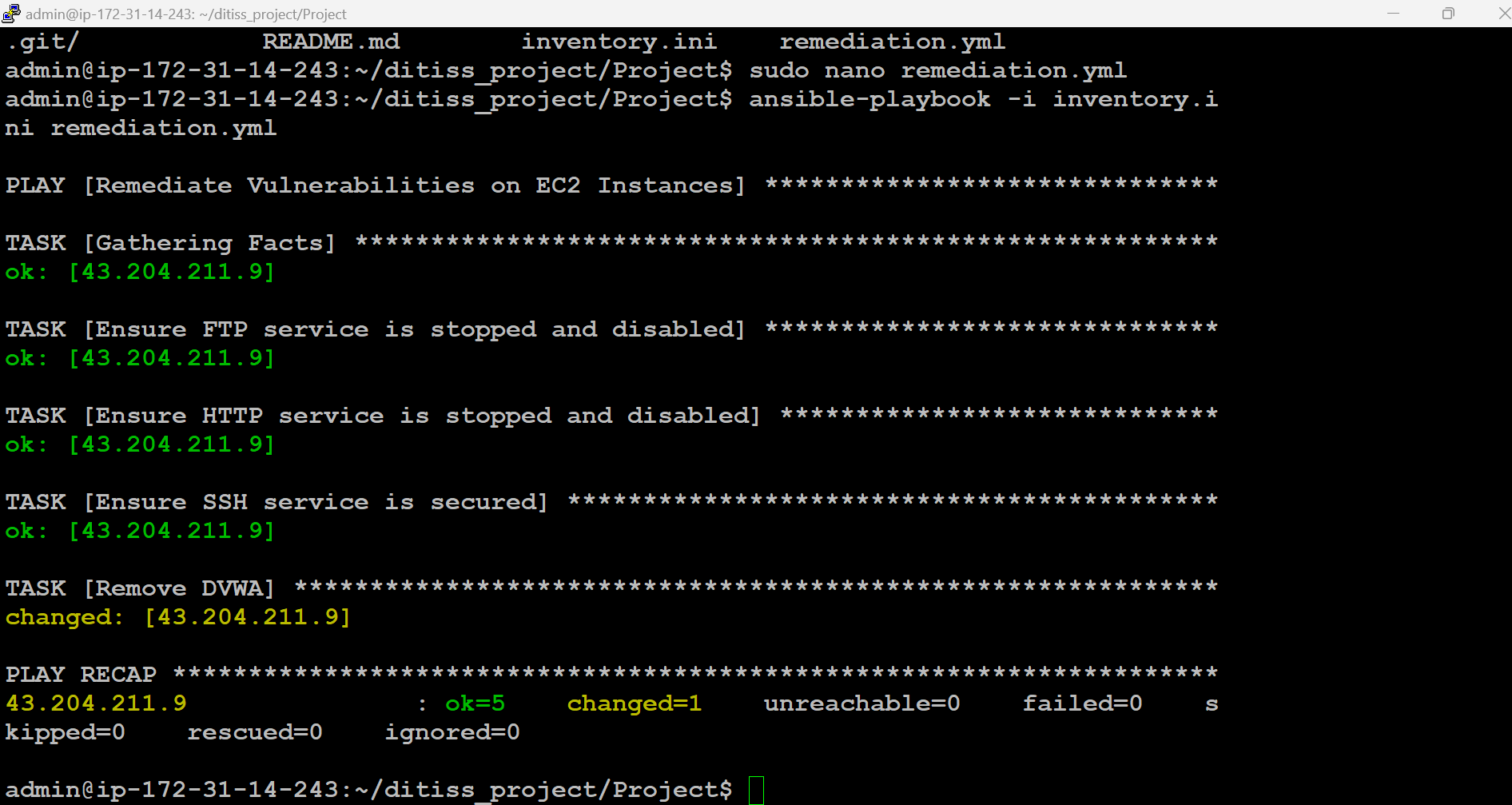
## 1.1 Before



**1.2 after**

****

## Ansible



A screen shot of a computer

Description automatically generated

# 5. CONCLUSION

## 5.1 Conclusion

Hence ,we have successfully addresses the dual challenge of managing and hardening servers concurrently through a robust and automated approach. By utilizing Terraform for infrastructure provisioning, we ensure that servers are consistently and efficiently deployed across multiple environments. Packer complements this by creating standardized machine images, reducing configuration drift and ensuring uniformity. The integration of AWS Inspector enhances security by performing continuous assessments, identifying vulnerabilities, and ensuring compliance with industry standards.

## 5.2 Future Scope

The future scope of this project includes several key enhancements to address evolving technological and organizational needs. Expanding support to additional cloud providers, virtualization platforms, and container orchestration systems will increase the solution’s versatility. Integrating advanced security features like threat intelligence, anomaly detection, and machine learning-based analytics can enhance the system's ability to counter sophisticated threats. Further integration with DevOps tools such as Kubernetes and service meshes will support modern application deployments. Improving compliance reporting with more detailed and customizable features will aid in meeting specific regulatory requirements. Automated remediation workflows could streamline responses to vulnerabilities and misconfigurations. Enhancing the centralized management dashboard with advanced visualization tools and user-friendly interfaces will improve user experience. Optimizing performance and scalability will ensure the system adapts to dynamic IT environments, while exploring integrations with emerging technologies like edge computing and serverless architectures will keep the solution relevant. Additionally, adding features for better team collaboration and comprehensive training and support resources will enhance implementation and management, ensuring the project remains robust, scalable, and effective.

# REFERENCES

**Paper 1:** - A Qualitative Study of DevOps Usage in Practice

**Author:** Floris Erich, C. Amrit & M. Daneva.

## Paper 2: **Automating Security Hardening and Compliance in Cloud Environments**

**Author:** Smith et al. (2019).

## Paper 3: - **-** A Comprehensive Approach to Server Management Using CI/CD Pipelines Author: Johnson and Miller (2020).