Pentest Report Capstone Team-1

By Capstone-Team-1

Executive Summary

Overview

The objectives of the penetration test and source code review were to identify vulnerabilities and weaknesses in the application's design and implementation. To assess the effectiveness of existing security controls and to identify areas where additional controls may be necessary. Next, to identify potential attack vectors and determine the level of effort required to exploit them. Finally, to provide recommendations for remediation of identified vulnerabilities

Key Findings

- CVE-2020-14145
- CVE-2021-42560
- CVE-2021-42561
- CVE-2021-28041
- CVE-2021-41617
- File Mismanagement/Security

Infrastructure Security

- Virtual Machines were not encrypted using a Customer Managed or KMS issued encryption key.
- Virtual Private Logs were not enabled for auditing and incident response and remediation activities
- There are multiple servers with Public IP and access from the internet which is a major risk that must be managed.
- Virtual Machine Integrity Check and Monitoring was not enabled.
- Project wide SSH keys must be disabled.

DNS Server Security

- Version disclosure for open ports helps attackers to execute sophisticated attacks.
- Multiple CVE’s were applicable to SSH version found.
- No domain issues were found.

Static Code Analysis

- GitHub token committed in the source code
- Clear text logging of sensitive information
- No CSRF protection on authentication endpoints

Dependencies Analysis

- Vulnerable JWT package (jsonwebtoken)

Introduction

Test Objectives:

The objectives of a white box penetration test and source code review are:

1. To identify vulnerabilities and weaknesses in the application's design and implementation.
2. To assess the effectiveness of existing security controls and to identify areas where additional controls may be necessary.
3. To identify potential attack vectors and determine the level of effort required to exploit them.
4. To provide recommendations for remediation of identified vulnerabilities

Benefits of a White Box Penetration Test and Source Code Review:

1. Early detection of security vulnerabilities, which can save time and money in the long run by avoiding costly security breaches.
2. Improved security posture of the application
3. Identification of areas for improvement in security controls and procedures.
4. Compliance with industry standards.
5. Increased confidence in the security of the application.:)

Test Environment:

The capstone project involves a NodeJs web application and a private npm package, which is a typical web application environment. In addition, Caldera and a DNS Bind9 server were used to simulate a real-world attack scenario.

Infrastructure: The infrastructure of the PenTest environment would consist of the server(s) hosting the NodeJs web application, the private npm package repository server, Caldera server, and the DNS Bind9 server. Additionally, the local system where the Terraform scripts were executed was also used to run tfsec tool to detect security shortcomings in infrastructure scripts.

Web Application Components: The web application components in the test environment include the frontend user interface, backend server-side code. There are two application servers: Production, and Development web-server.

Out-of-Scope

The Verdaccio server and private package were out of scope, because the system are not public facing.

Findings and Recommendations

Attacker Server

For the Attacker Server, this was tested by seeing what CVEs Caldera has, as well as what Caldera could find within the server, and what could be taken as well, if an agent within Caldera was successfully created.

Ability to Execute Shell Commands

CVE: CVE-2021-42561

Description: When activated, the Human plugin passes the unsanitized name parameter to a python "os.system" function. This allows attackers to use shell metacharacters in order to escape the current command and execute arbitrary shell commands.

Ability to Execute Shell Commands

CVE: CVE-2021-42560

Description: The Debrief plugin receives base64 encoded "SVG" parameters when generating a PDF document. These SVG documents are parsed in an unsafe manner and can be leveraged for XXE attacks

Over 80 CVEs were found for Caldera, these two held the highest score for potential damage that could be caused and therefore were included within the report.

Using Caldera on the Attacker Server gave the ability to find various files and the file types. As well as location to these files within the server. There are no forms of protection to any documents and all are very accessible and not protected. Nor do there seem to be backups of any documents. All information is stored poorly and with no regards to potential of data being stolen or lost.

For best practices all files should be kept encrypted with least privilege access for files. As well as backups being made periodically with each backup having separate encryptions as well. Overall, a complete overhaul must be made in regards to file management, location and security.

DNS Server Findings

Port Scanning on Public Facing IP Address : 104.154.138.4

- Two ports are open : SSH (22) and DNS (53).

```
┌──(root㉿kali)-[/home/kali]
└─# nmap 104.154.138.48
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-11 21:36 EDT
Nmap scan report for 48.138.154.104.bc.googleusercontent.com (104.154.138.48)
Host is up (0.0042s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT   STATE SERVICE
22/tcp open  ssh
53/tcp open  domain

Nmap done: 1 IP address (1 host up) scanned in 56.89 seconds
```

Scanning for Vulnerabilities using NMAP Scripts:

- With the help of NMAP scripts, we can identify the versions of each port.
    - 22 : OpenSSH 8.2p1 Ubuntu
    - 53 : ISC BIND 9.61.1
- Version Disclosure can help attackers to write and execute more sophisticated attacks.
- We can also perform OS banner grabbing to identify the OS we are using. In this case, Linux.

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sV 104.154.138.48
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-11 21:40 EDT
Nmap scan report for 48.138.154.104.bc.googleusercontent.com (104.154.138.48)
Host is up (0.0053s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
53/tcp open  domain  ISC BIND 9.16.1 (Ubuntu Linux)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 75.57 seconds
```

Applicable CVE's

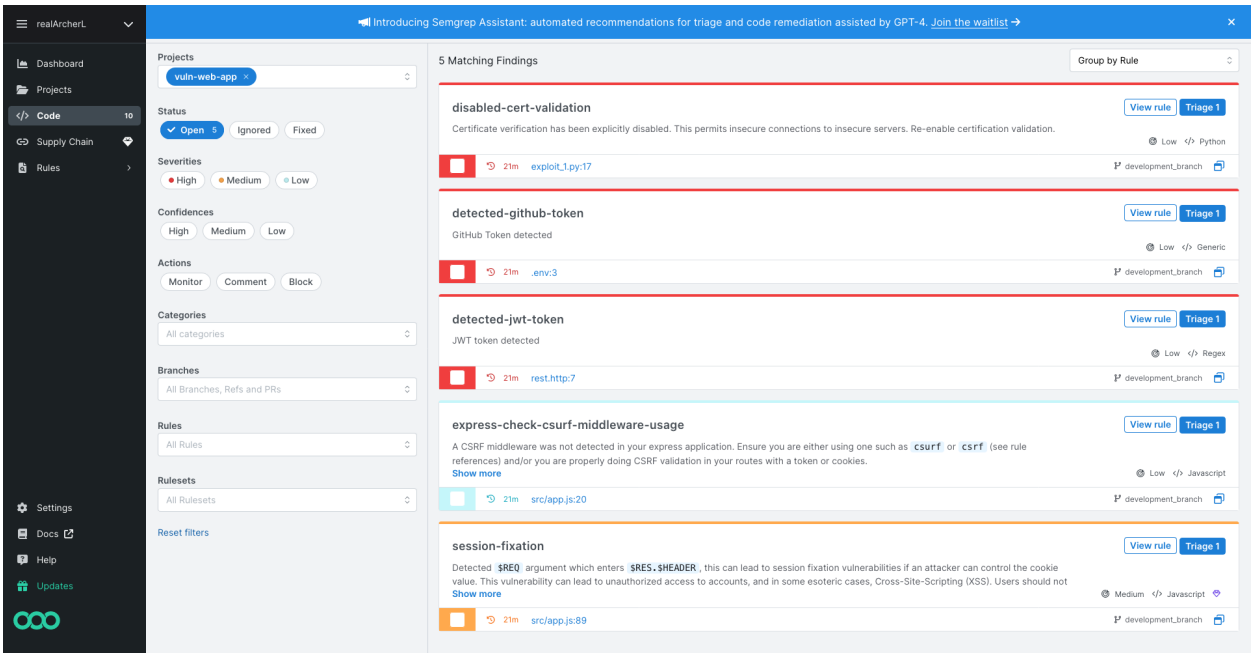
- **Use of a broken or risky cryptographic algorithm**
    - **CVE-ID:** CVE-2020-14145
    - **Description:** A remote attacker can launch a Man-in-the-Middle (MitM) attack using the vulnerability, which occurs in the openssh client during algorithm negotiation because of an apparent disagreement.
- **Double Free**

    - **CVE-ID:** CVE-2021-28041
    - **Description:** A boundary mistake in ssh-agent is the cause of the vulnerability. A remote attacker has the ability to mislead the victim into connecting to a server on which they have root rights, send the ssh client specially constructed data, cause a double free error, and run arbitrary code on the target system
- **Improper Privilege Management**
    - **CVE-ID:** CVE-2021-41617
    - **Description:** When specific non-default configurations are utilized, the vulnerability occurs due to poor privilege control in sshd since supplemental groups are not initialized as intended. If the setting specifies running the command as a separate user, helper programs for AuthorizedKeysCommand and AuthorizedPrincipalsCommand may run with rights associated with group memberships of the sshd process. The system's privileges can be increased for a local user.

Static Code Analysis Findings

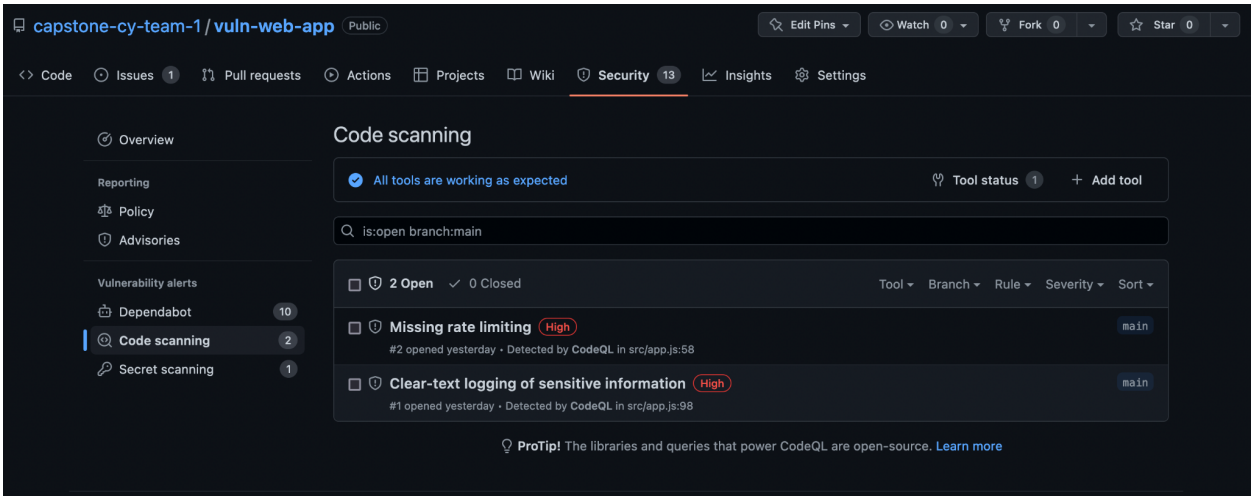
For the static code analysis of our Node.js application. We used Semgrep and CodeQL. These industry-leading tools allowed us to scan the application's codebase for potential security issues, including those related to using the Handlebars templating engine and authentication mechanisms. Semgrep and CodeQL provided detailed insights into the codebase, helping us uncover potential vulnerabilities and insecure coding practices.
The tools CodeQL and Semgrep identified various security issues, of which 2 were common finds. The results are as follows:

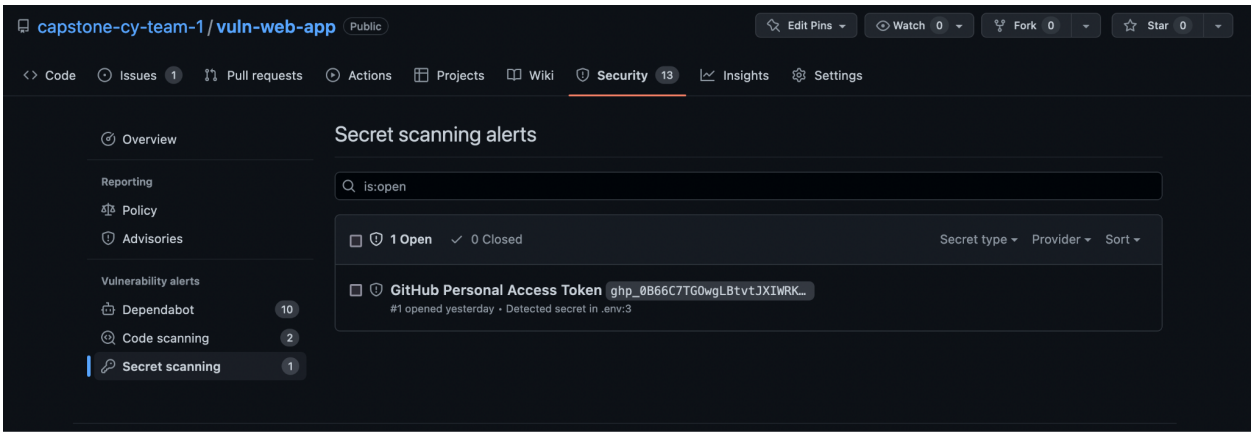
SemGrep



The tool ran on the source code for **vulnerable-web-app**. SemGrep found 3 **High, 1 Medium,** and **1 Low** severity issue in the code base. The .json results are provided in the zip folder along with this report.

CodeQL



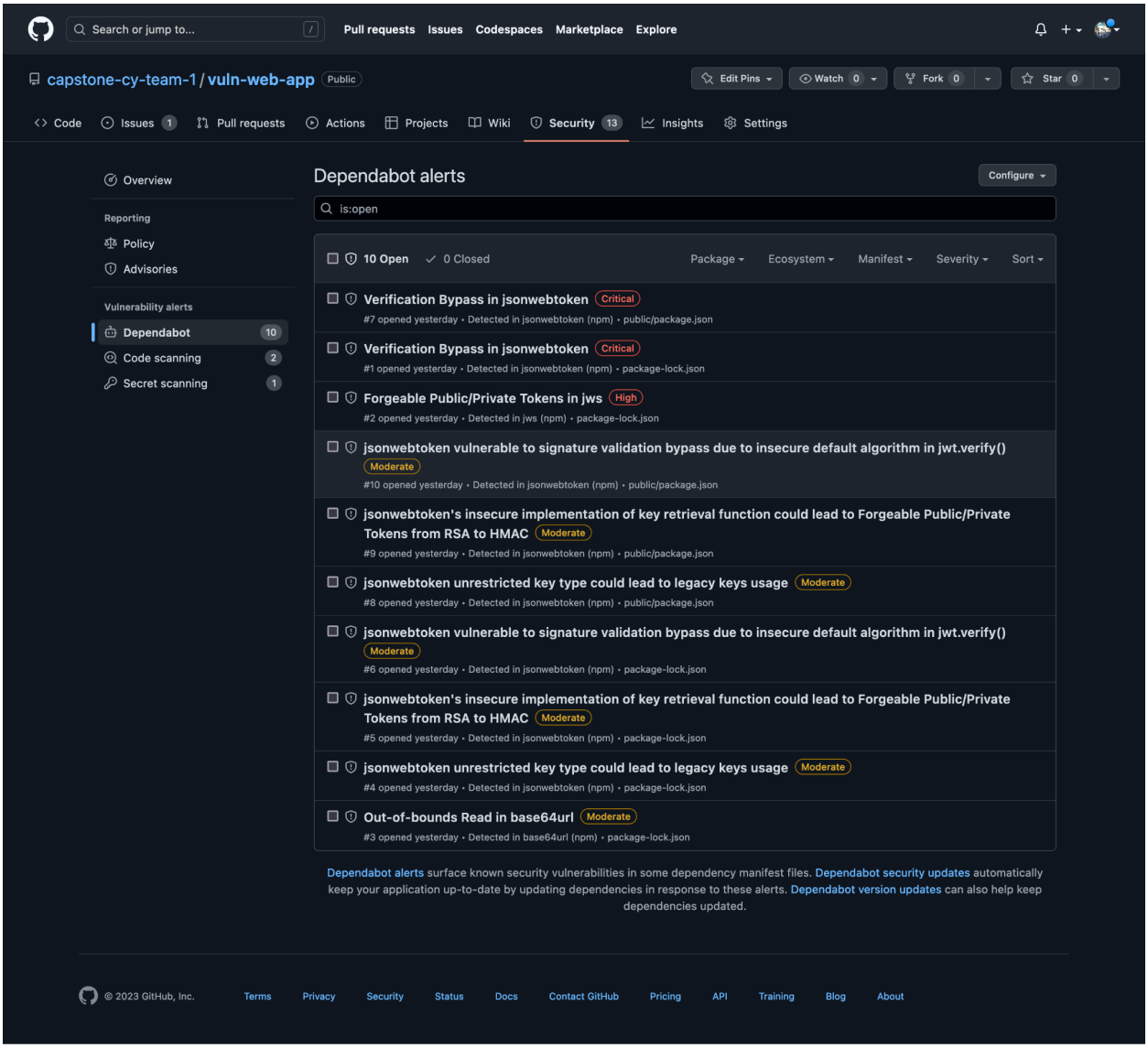


CodeQL provides us with **2 High** severity issues in the codebase. It also identified the GitHub token which was committed to the repository. The results for the same can be viewed on the GitHub repository here: https://github.com/capstone-cy-team-1/vuln-web-app/security/code-scanning

Dependency Analysis

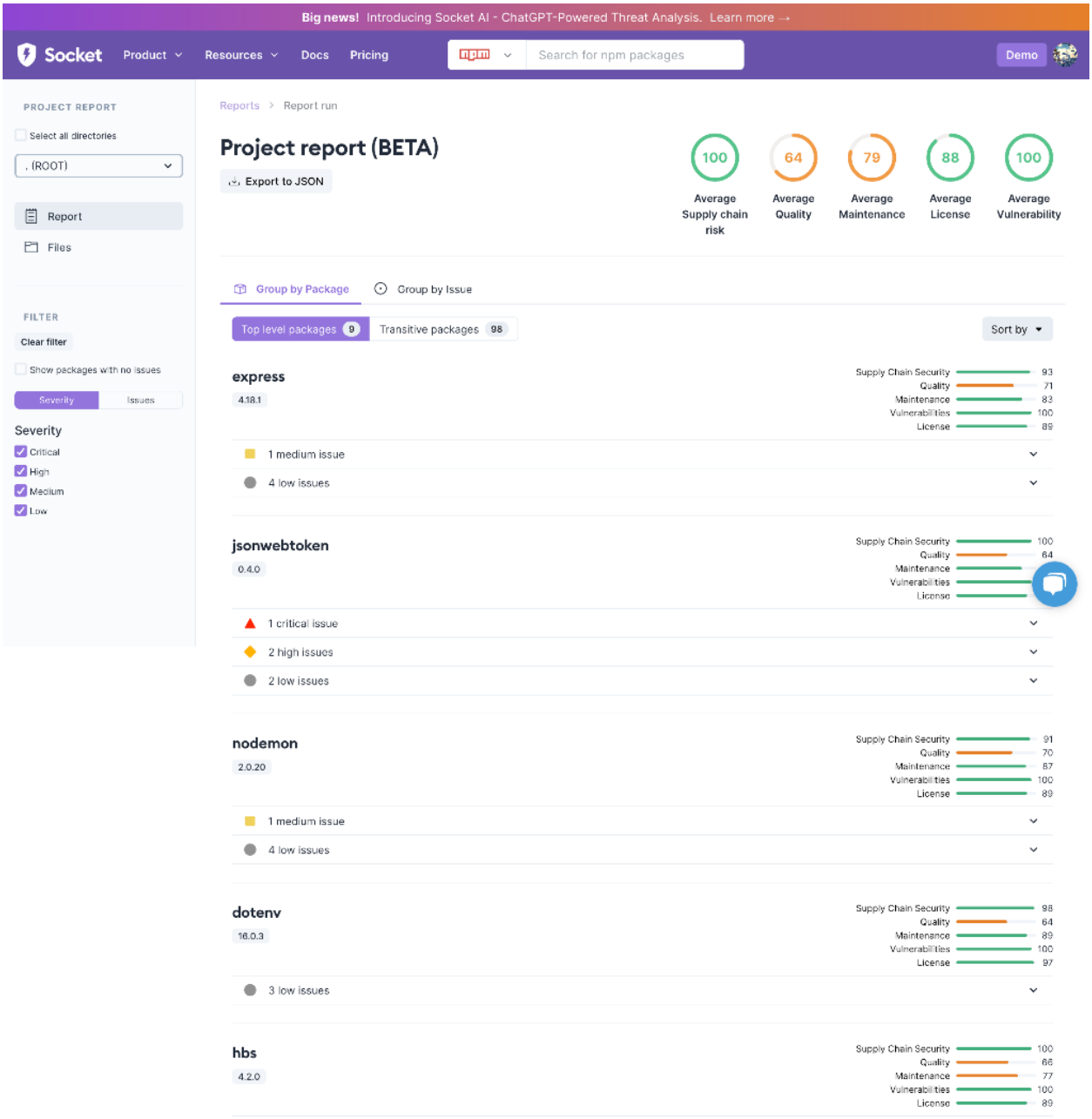
To assess the security of our application's dependencies, including our private package, we utilized Dependabot and Socket. These tools enabled us to identify outdated or vulnerable packages within our project and recommended updating to more secure versions. By analyzing our dependencies, we were able to gauge the potential impact of third-party libraries on the overall security of our application.

Dependabot



Dependabot identified **10 Security issues** related to dependencies used by the package. **2 Critical, 1 High,** and **7 Medium** issues. The scan results can be viewed here: https://github.com/capstone-cy-team-1/vuln-web-app/security/dependabot

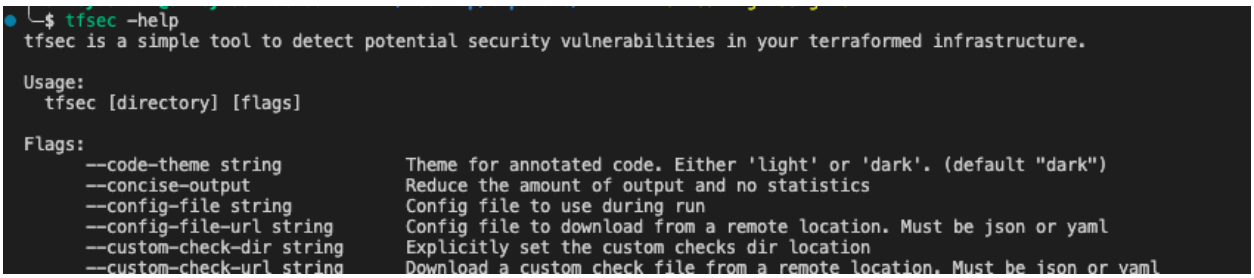
Socket



Socket was able to identify several supply chain security issues, not only known issues in the dependencies but also provide us with insights.

Infrastructure Security Findings

The Infrastructure is implemented on Google Cloud Platform and all the resources were created and managed using Terraform Infrastructure as a Code. Some of the main components in the Infrastructure are - Victim Web Server, Victim Dev and Internal package repository server; Attacker Caldera and DNS server. Two VPC's each for attacker and victim and subnets are also implemented with necessary firewall rules, NAT an Internet gateways.



As part of the InfraSec testing activity, TFSEC, a tool to identify security shortcomings in the infrastructure that is specifially set up using Terraform was used and the following findings emerged:

There were 1 critical, 3 high, 15 medium and 8 low findings that was discovered, as specified in the below snapshot:

```
timings
──────────────────────────────────────────
disk i/o                4.4469ms
parsing                 11.134189ms
adaptation              9.588229ms
checks                  34.529157ms
total                   59.698475ms

counts
──────────────────────────────────────────
modules downloaded      0
modules processed       1
blocks processed        24
files read              5

results
──────────────────────────────────────────
passed                  29
ignored                 0
critical                1
high                    3
medium                  15
low                     8

29 passed, 27 potential problem(s) detected.
```

- The critical finding was since a firewall rule allowed ingresstraffic from the public internet. This can be accepted as it is part of the project usecase for the webserver and caldera server to receive traffic from the internet. The below snapshot provides information on this finding:

```
Result #1 CRITICAL Firewall rule allows ingress traffic from multiple addresses on the public internet.

  firewall.tf:11

    1    resource "google_compute_firewall" "attacker_allow_ssh" {
    .
   11  [   source_ranges = ["0.0.0.0/0"]
   ..
   13    }

          ID google-compute-no-public-ingress
      Impact The port is exposed for ingress from the internet
  Resolution Set a more restrictive cidr range
```

- The three high findings are of similar nature to the critical finding. The issue was with web server, Caldera and DNS servers were allocated a public IP. Yet again, the project usecase required these servers to have public IPs, hence this findings may also be accepted. The below snapshot provides information on this finding for one of the servers:

```
Result #2 HIGH Instance has a public IP allocated.

  attack_compute.tf:18-19

    1    resource "google_compute_instance" "attacker_caldera_server" {
    .
   18  ┌     access_config {
   19  └     }
   ..
   31    }

          ID google-compute-no-public-ip
      Impact Direct exposure of an instance to the public internet
  Resolution Remove public IP
```

- The next finding was a medium level shortcoming. The compute instances did not specifically block use of project-level SSH keys. This meant that if a project level SSH key was leaked, the attacker would have access to all the servers over SSH. The below snapshot provides information on this finding for one of the servers:

```
Result #11 MEDIUM Instance allows use of project-level SSH keys.

  victim_compute.tf:1-29

   1 ┌ resource "google_compute_instance" "victim_dev_server" {
   2 │   name                      = "victim-dev-server"
   3 │   machine_type              = "e2-micro"
   4 │   zone                      = "us-central1-a"
   5 │   tags                      = ["victim-allow-outbound-tcp", "victim-allow-ssh", "victim-allow-inbound-tcp"]
   6 │   allow_stopping_for_update = true
   7 │
   8 │   boot_disk {
   9 └     initialize_params {
  ..

          ID google-compute-no-project-wide-ssh-keys
      Impact Compromise of a single key pair compromises all instances
  Resolution Disable project-wide SSH keys
```

This finding was remediated by adding the below attribute to the Terraform resource of all the compute servers:

```
block-project-ssh-keys = true
```

- The next finding was again of Medium level that specific that a Virtual Machine Integrity check was not enabled. The feature if enabled would detect any changes made to the VM that are not authorized and by malicious actors and would trigger alerts and remediation efforts as specified. The below snapshot provides information on this finding for one of the servers:

```
Result #7 MEDIUM Instance does not have shielded VM integrity monitoring enabled.

  attack_compute.tf:1-31

   1 ┌ resource "google_compute_instance" "attacker_caldera_server" {
   2 │   name                      = "attacker-caldera-server"
   3 │   machine_type              = "e2-micro"
   4 │   zone                      = "us-central1-a"
   5 │   tags                      = ["attacker-allow-caldera", "attacker-allow-outbound-tcp", "attacker-allow-ssh"]
   6 │   allow_stopping_for_update = true
   7 │
   8 │   boot_disk {
   9 └     initialize_params {
  ..

          ID google-compute-enable-shielded-vm-im
      Impact No visibility of VM instance boot state.
  Resolution Enable Shielded VM Integrity Monitoring
```

This finding was remediated by adding the below attribute to the Terraform resource of all the compute servers:

```
shielded_instance_config {
  enable_vtpm                 = true
  enable_integrity_monitoring = true
}
```

- The next finding, which was also Medium in nature required Virtual Machine integrity monitoring to be enabled which was actively monitor the resources state and data integrity. The below snapshot provides information on this finding for one of the servers:

```
Result #9 MEDIUM Instance does not have VTPM for shielded VMs enabled.

  attack_compute.tf:33-62

   33 ┌ resource "google_compute_instance" "attacker_dns_server" {
   34 │   name                      = "attacker-dns-server"
   35 │   machine_type              = "e2-micro"
   36 │   zone                      = "us-central1-a"
   37 │   tags                      = ["attacker-allow-dns", "attacker-allow-outbound-tcp", "attacker-allow-ssh"]
   38 │   allow_stopping_for_update = true
   39 │
   40 │   boot_disk {
   41 └     initialize_params {
   ..

         ID google-compute-enable-shielded-vm-vtpm
     Impact Unable to prevent unwanted system state modification
 Resolution Enable Shielded VM VTPM
```

This finding was remediated by adding the below attribute to the Terraform resource of all the compute servers:

```
shielded_instance_config {
  enable_vtpm                 = true
  enable_integrity_monitoring = true
}
```

- The next finding, which was of Low level required the usage of either a Cryptographic key to encrypt the Virtual machines disk. Either a customer provided key or a key that can be generated using Google Cloud Platforms Key Management Service can be used to remediate this finding. The below snapshot provides information on this finding for one of the servers:

```
Result #5 LOW Instance disk encryption does not use a customer managed key.

  attack_compute.tf:45-49

  38    resource "google_compute_instance" "attacker_dns_server" {
  ..
  45      boot_disk {
  46        initialize_params {
  47          image = "ubuntu-os-cloud/ubuntu-2004-lts"
  48        }
  49      }
  ..
  72    }

         ID google-compute-vm-disk-encryption-customer-key
     Impact Using unmanaged keys does not allow for proper management
 Resolution Use managed keys
```

This was remediated by created a new GCP KMS Key ring and creating a new Cryptographic key and then using that key to encrypt the Disk of the VMs for added protection of the resources.

```
resource "google_kms_key_ring" "capstone_keyring" {
  name     = "capstone-keyring"
  location = "us-central1"
}

resource "google_kms_crypto_key" "capstone_key" {
  name            = "capstone-key"
  key_ring        = google_kms_key_ring.capstone_keyring.id
  rotation_period = "864000s"

  lifecycle {
    prevent_destroy = true
  }
}
```

```
boot_disk {
  initialize_params {
    image = "ubuntu-os-cloud/ubuntu-2004-lts"
  }
  kms_key_self_link = google_kms_crypto_key.capstone_key.id
}
```

- Logs are important for troubleshooting, auditing and Incident Response activities. The next finding was related to logging of the VPC flow not being enabled. The below snapshot provides information on this finding for one of the VPCs:

```
Result #7 LOW Subnetwork does not have VPC flow logs enabled.

  network.tf:11-18

  11    resource "google_compute_subnetwork" "victim_public_subnet" {
  12      name          = "victim-public-subnet"
  13      ip_cidr_range = "10.0.1.0/24"
  14      region        = "us-central1"
  15      network       = google_compute_network.victim_vpc.self_link
  16
  17      private_ip_google_access = true
  18    }

        ID google-compute-enable-vpc-flow-logs
    Impact Limited auditing capability and awareness
Resolution Enable VPC flow logs
```

The finding was remediated by enabling VPC logs for both the VPCs as shown below:

```
log_config {
  aggregation_interval = "INTERVAL_10_MIN"
  flow_sampling        = 0.5
  metadata             = "INCLUDE_ALL_METADATA"
}
```

- Once all the findings were remediated, the report was generated once again and the following status was achieved with 0 Medium and Low level findings. As mentioned earlier, the Critical and 3 High level findings were accepted as part of the project requirement.

```
timings
─────────────────────────────────────
disk i/o             5.351786ms
parsing              8.883128ms
adaptation           854.472µs
checks               34.935357ms
total                50.024743ms

counts
─────────────────────────────────────
modules downloaded   0
modules processed    1
blocks processed     26
files read           6

results
─────────────────────────────────────
passed               53
ignored              0
critical             1
high                 3
medium               0
low                  0

53 passed, 4 potential problem(s) detected.

shreyasnair@Shreyass-MacBook-Air ~/Desktop/capst
```

Recommendations

Static Code Analysis

SemGrep

The disable-cert-validation can be ignored since the website was built to communicate over HTTP. The sample HTTP requests file contained an invalid JWT which can also be ignored, but developers should be careful about committing authentication cookies in the future.
The GitHub token which was committed should also be revoked, and pre-hooks should be configured to prevent users from committing secrets. The application is vulnerable to CSRF attacks and hence a defensive capability should be added to the code.

CodeQL and Secret Scanning

Based on the issues identified, the users should code the ability to rate limit to the source code, and should also not the clear text logging of the inputs.

Dependency Analysis

All the vulnerable and outdated dependencies should be updated and merged into the main branch's package.json. This will prevent supply chain attacks. Developers though should be careful when updating these packages since these might break the functionality of the website.

DNS Server Security

The mitigation for all applicable CVEs is available in the recent version of SSH that has been deployed.

Conclusion

In summary, the white box penetration test and source code review for the capstone project would involve assessing the security of the NodeJs web application and private npm package, identifying vulnerabilities and weaknesses in their design and implementation, and providing recommendations for remediation. The test environment would consist of the servers hosting the application, the application components, Caldera, and a DNS Bind9 server. As well as, incorporating a source code review into our penetration test is a critical step towards improving the overall security of our web application.