DIONES, JOHN CEDRICK N. **12//5/2023**
CPE 243 - CPE32S1 **Dr. Jonathan Taylar**

FINAL EXAM

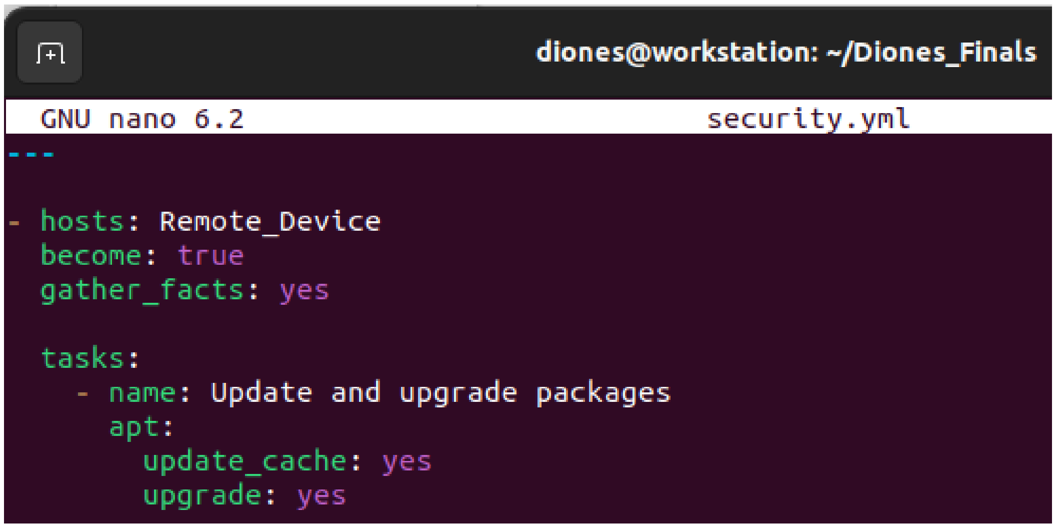
“Squadzip” is a platform designed for field sales teams to streamline their operations and improve productivity. **Squadzip** is an investor of **Kickstart Ventures and Plug and Play Tech Center**. It’s a member of **Amazon Partner Network (APN)**, the global partner program for **AWS**. There IT infrastructure contains the following:

- Hosting and Storage
- Access and Control
- Support and Maintenance
- Data Deletion

My aim is to enhance the security posture of your remote device with my comprehensive playbook, ensuring robust defense measures. Implementing measures such as firewall configuration, restricted SSH access, key-based authentication, session timeout settings, intrusion detection with Snort, automatic security updates, strong password policies, fail2ban protection, auditd for auditing, and optimized log rotation and retention. The playbook follows industry best practices to fortify the system against potential threats and vulnerabilities, providing a resilient and secure computing environment.

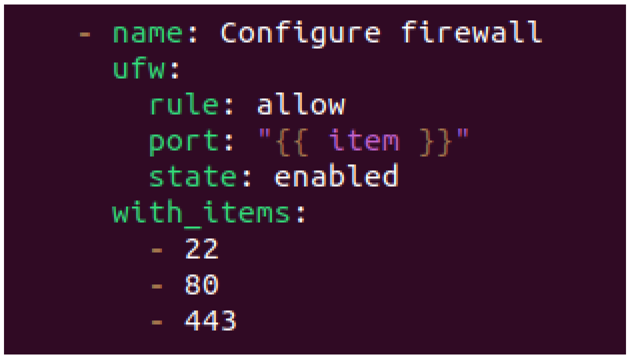
PLAYBOOK INPUT

security.yml



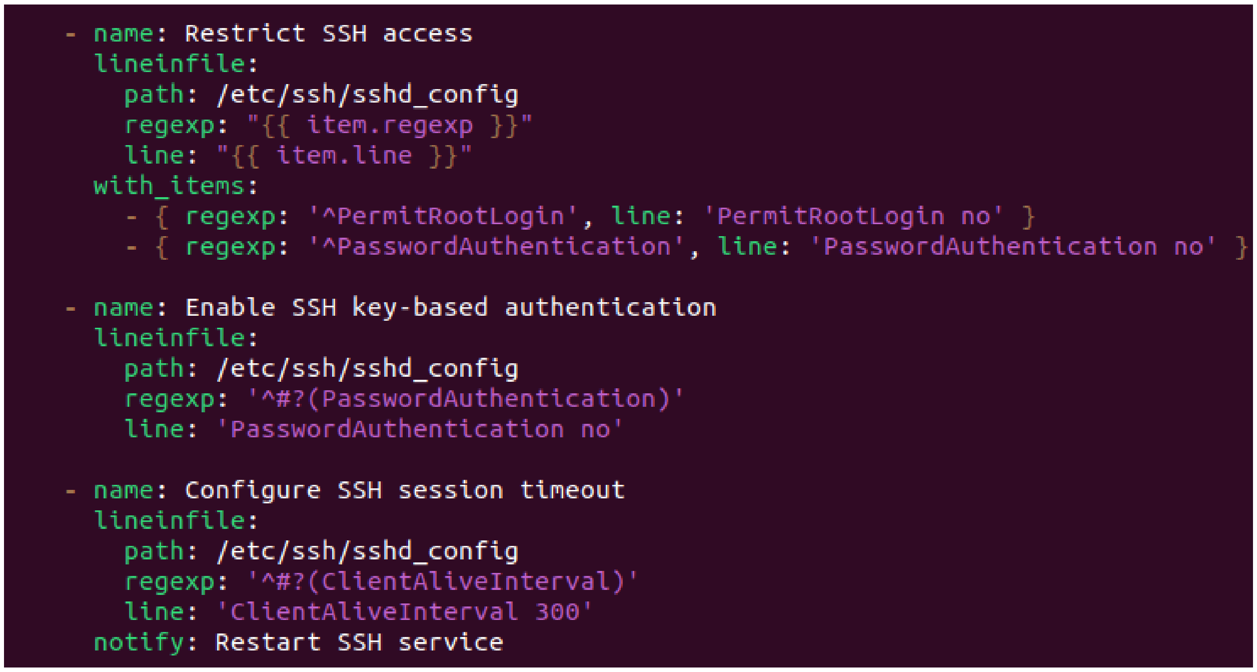
```
diones@workstation: ~/Diones_Finals
 GNU nano 6.2                              security.yml
---

- hosts: Remote_Device
  become: true
  gather_facts: yes

  tasks:
    - name: Update and upgrade packages
      apt:
        update_cache: yes
        upgrade: yes
```

Ensures that the system packages are up-to-date with the latest security patches and updates. Regularly updating and upgrading packages is a crucial security practice to address vulnerabilities and improve system stability.



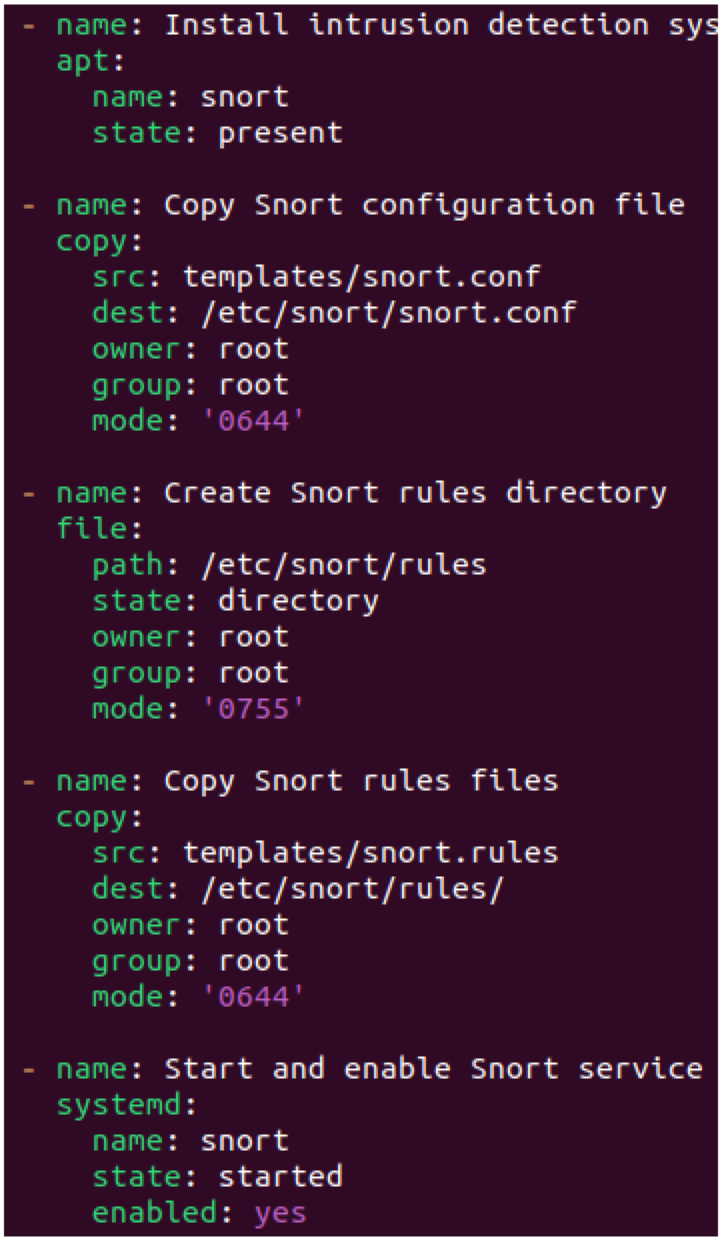
```
  - name: Configure firewall
    ufw:
      rule: allow
      port: "{{ item }}"
      state: enabled
    with_items:
      - 22
      - 80
      - 443
```

Opens specific ports (22, 80, 443) in the firewall to allow SSH, HTTP, and HTTPS traffic. Restricting open ports to only necessary services reduces the attack surface and enhances security.



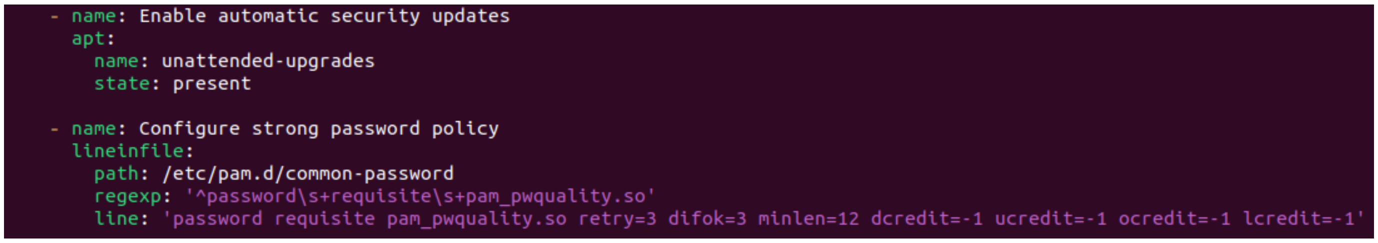
```
  - name: Restrict SSH access
    lineinfile:
      path: /etc/ssh/sshd_config
      regexp: "{{ item.regexp }}"
      line: "{{ item.line }}"
    with_items:
      - { regexp: '^PermitRootLogin', line: 'PermitRootLogin no' }
      - { regexp: '^PasswordAuthentication', line: 'PasswordAuthentication no' }

  - name: Enable SSH key-based authentication
    lineinfile:
      path: /etc/ssh/sshd_config
      regexp: '^#?(PasswordAuthentication)'
      line: 'PasswordAuthentication no'

  - name: Configure SSH session timeout
    lineinfile:
      path: /etc/ssh/sshd_config
      regexp: '^#?(ClientAliveInterval)'
      line: 'ClientAliveInterval 300'
    notify: Restart SSH service
```

These tasks collectively aim to enhance the security of the SSH service by disabling root login, enforcing key-based authentication, and configuring a session timeout to automatically disconnect inactive sessions after a specified period. Disabling root login and password authentication enhances security by requiring users to authenticate using more secure methods like SSH keys. SSH key-based authentication is considered more secure than password authentication, providing an additional layer of protection. Configuring a session timeout improves security by automatically disconnecting idle SSH sessions after a specified period.



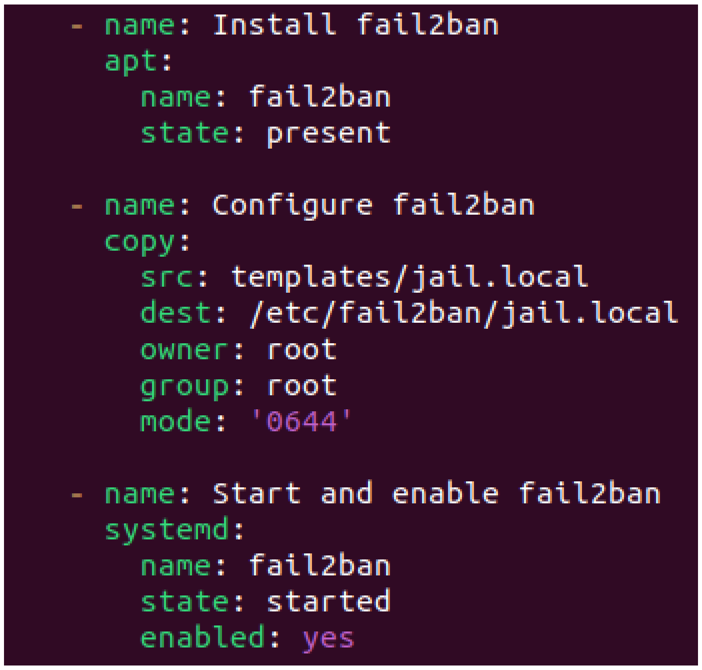
```
- name: Install intrusion detection sys
  apt:
    name: snort
    state: present

- name: Copy Snort configuration file
  copy:
    src: templates/snort.conf
    dest: /etc/snort/snort.conf
    owner: root
    group: root
    mode: '0644'

- name: Create Snort rules directory
  file:
    path: /etc/snort/rules
    state: directory
    owner: root
    group: root
    mode: '0755'

- name: Copy Snort rules files
  copy:
    src: templates/snort.rules
    dest: /etc/snort/rules/
    owner: root
    group: root
    mode: '0644'

- name: Start and enable Snort service
  systemd:
    name: snort
    state: started
    enabled: yes
```

This segment sets up the Snort intrusion detection system on a target system by installing the necessary package, configuring its main settings and rules, and ensuring that the Snort service is started and enabled for automatic startup. Configuring Snort with the appropriate rules is essential for effective intrusion detection and prevention.



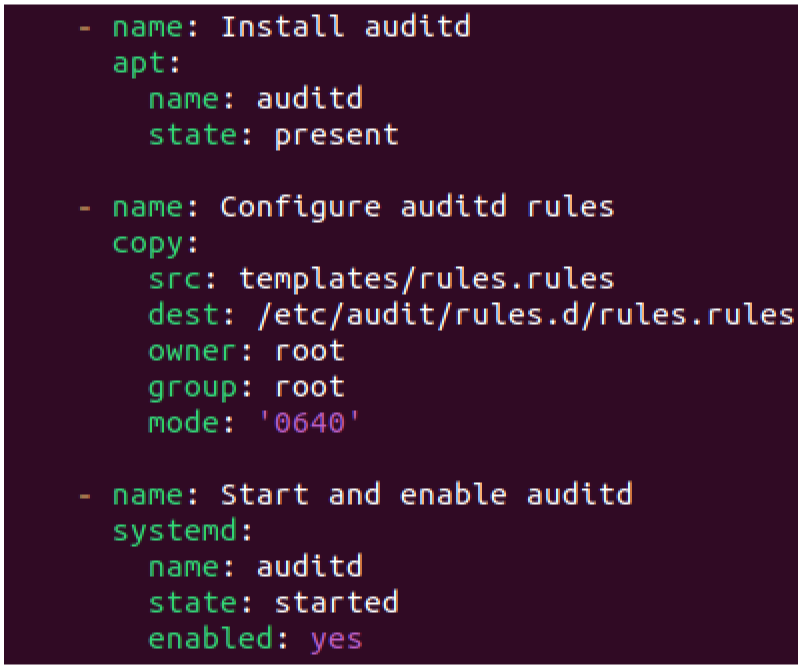
```
  - name: Enable automatic security updates
    apt:
      name: unattended-upgrades
      state: present

  - name: Configure strong password policy
    lineinfile:
      path: /etc/pam.d/common-password
      regexp: '^password\s+requisite\s+pam_pwquality.so'
      line: 'password requisite pam_pwquality.so retry=3 difok=3 minlen=12 dcredit=-1 ucredit=-1 ocredit=-1 lcredit=-1'
```

These tasks contribute to improving the security posture of the target system by automating the application of security updates and enforcing a strong password policy to enhance the resilience of user authentication mechanisms. Automatic security updates help keep the system protected against known vulnerabilities. Enforcing strong password policies enhances security by requiring users to use complex and secure passwords.



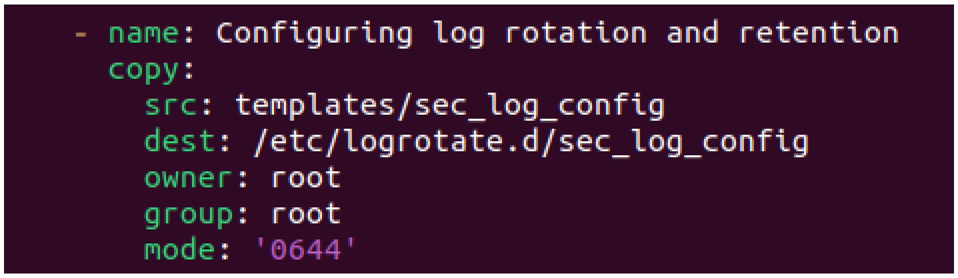
```
  - name: Install fail2ban
    apt:
      name: fail2ban
      state: present

  - name: Configure fail2ban
    copy:
      src: templates/jail.local
      dest: /etc/fail2ban/jail.local
      owner: root
      group: root
      mode: '0644'

  - name: Start and enable fail2ban
    systemd:
      name: fail2ban
      state: started
      enabled: yes
```

These tasks collectively install Fail2Ban, provide custom configuration through a local file, and then start and enable the Fail2Ban service to enhance the security of the system by mitigating potential threats through log monitoring and IP blocking. Fail2Ban helps protect against brute-force attacks by monitoring and blocking suspicious activities.



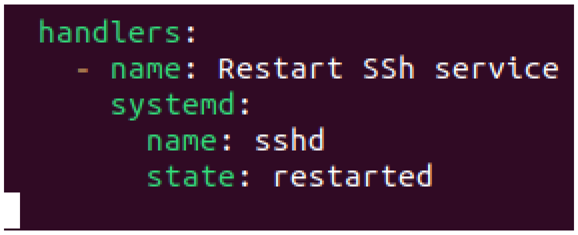
```yaml
  - name: Install auditd
    apt:
      name: auditd
      state: present

  - name: Configure auditd rules
    copy:
      src: templates/rules.rules
      dest: /etc/audit/rules.d/rules.rules
      owner: root
      group: root
      mode: '0640'

  - name: Start and enable auditd
    systemd:
      name: auditd
      state: started
      enabled: yes
```

These tasks collectively ensure that the auditd package is installed, the audit rules are configured, and the auditd service is started and enabled on the remote system. This helps in monitoring and recording security-related events on the system for later analysis and auditing. Auditd provides auditing capabilities to track security-relevant events.



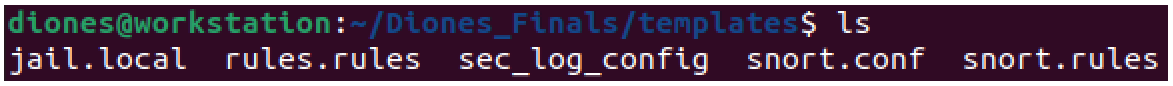
```yaml
  - name: Configuring log rotation and retention
    copy:
      src: templates/sec_log_config
      dest: /etc/logrotate.d/sec_log_config
      owner: root
      group: root
      mode: '0644'
```

This Ansible task is responsible for deploying a log rotation and retention configuration file named "sec_log_config" from the local machine's "templates" directory to the remote machine's "/etc/logrotate.d/" directory. The configuration file likely contains settings to manage the rotation and retention of a specific log file, enhancing the system's log management practices.

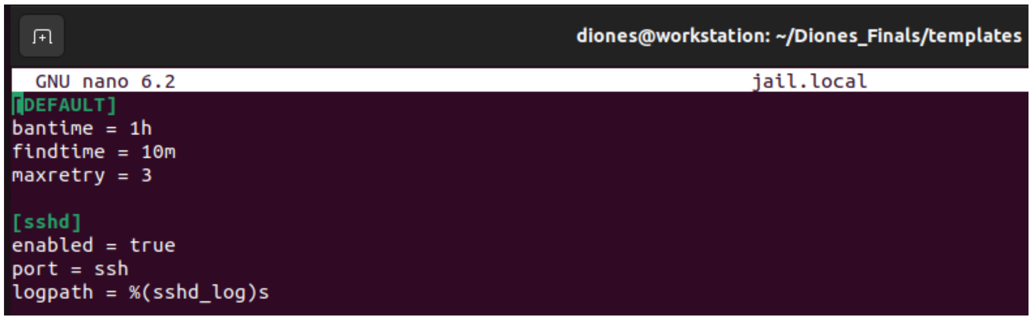


```
  handlers:
    - name: Restart SSh service
      systemd:
        name: sshd
        state: restarted
```

This handler ensures that the SSH service is restarted when certain tasks in the playbook make changes that require the service to be reloaded for the changes to take effect. It's a way to manage service restarts efficiently based on the playbook's execution flow.

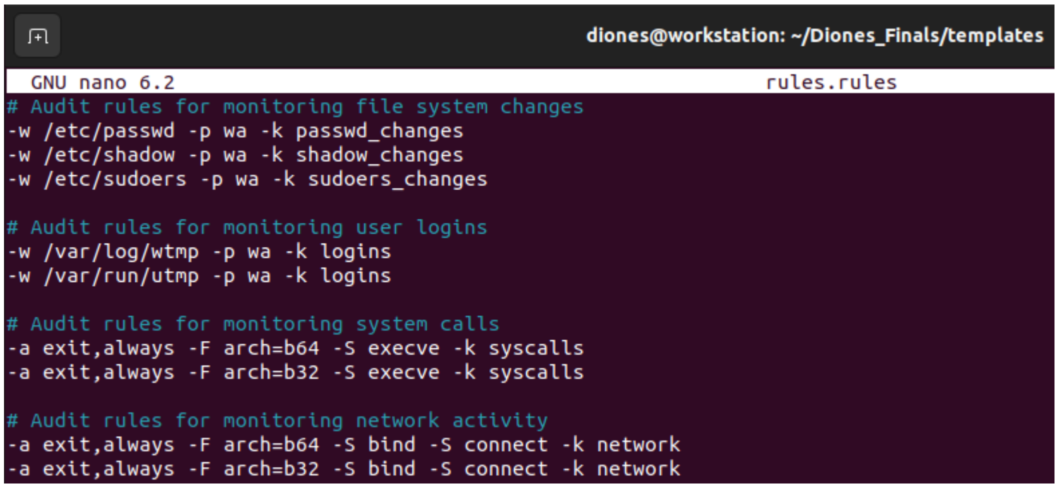


```
diones@workstation:~/Diones_Finals/templates$ ls
jail.local  rules.rules  sec_log_config  snort.conf  snort.rules
```



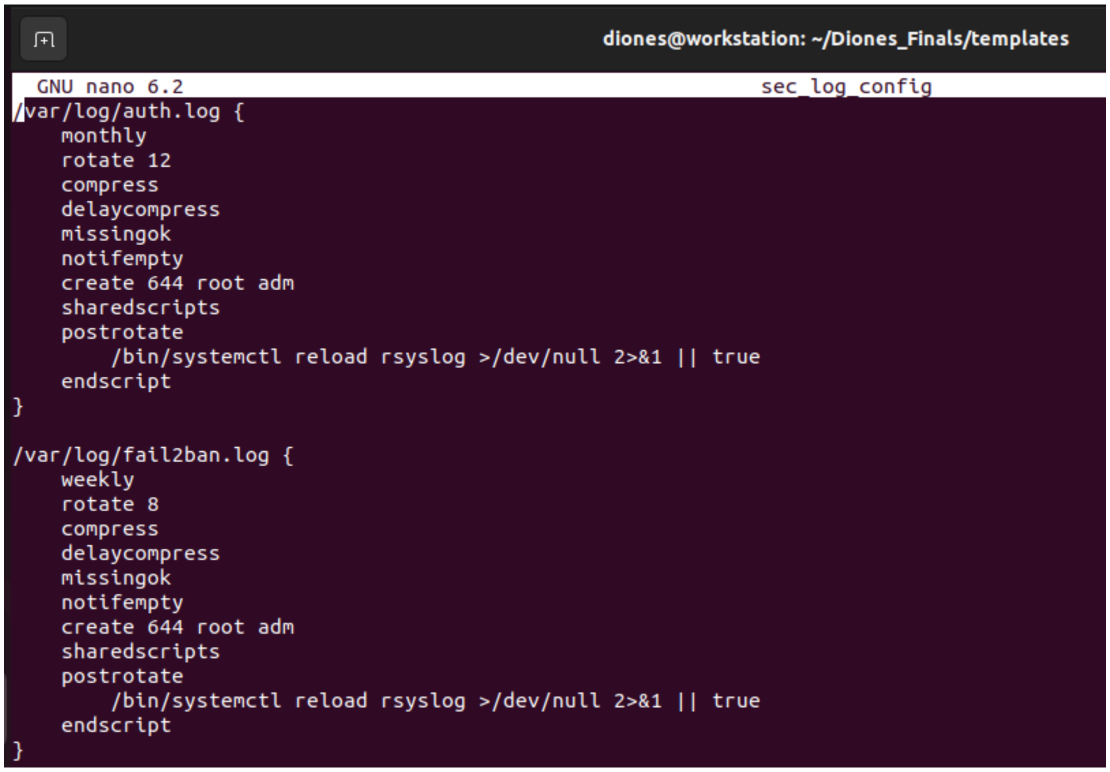
```
diones@workstation: ~/Diones_Finals/templates
 GNU nano 6.2                                   jail.local
[DEFAULT]
bantime = 1h
findtime = 10m
maxretry = 3

[sshd]
enabled = true
port = ssh
logpath = %(sshd_log)s
```

This configuration helps protect against brute-force attacks targeting SSH by temporarily banning IP addresses exhibiting suspicious login behavior. It's a proactive security measure to mitigate the risk of unauthorized access to the system.



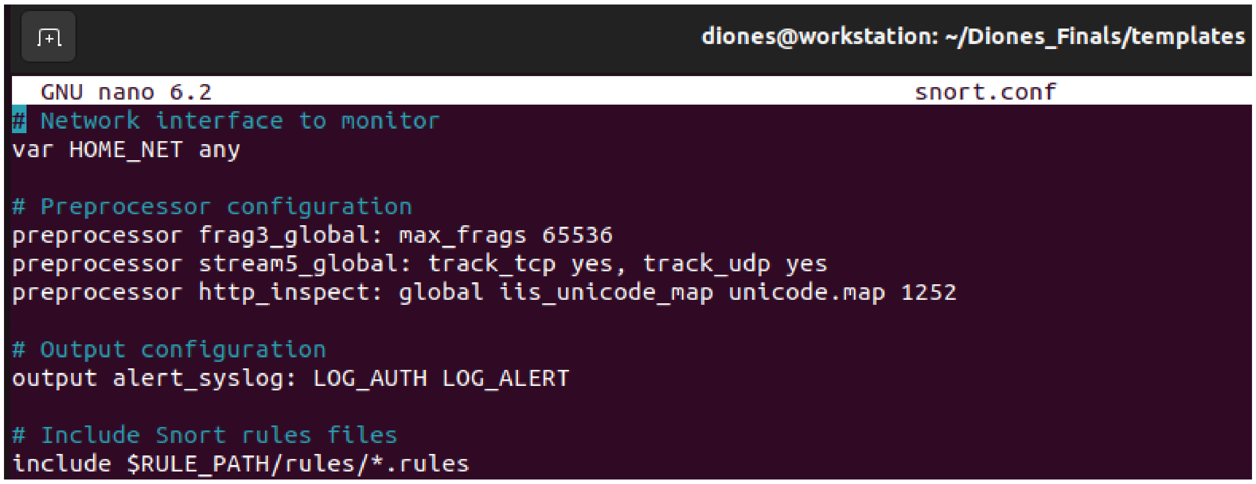
```
diones@workstation: ~/Diones_Finals/templates
 GNU nano 6.2                                          rules.rules
# Audit rules for monitoring file system changes
-w /etc/passwd -p wa -k passwd_changes
-w /etc/shadow -p wa -k shadow_changes
-w /etc/sudoers -p wa -k sudoers_changes

# Audit rules for monitoring user logins
-w /var/log/wtmp -p wa -k logins
-w /var/run/utmp -p wa -k logins

# Audit rules for monitoring system calls
-a exit,always -F arch=b64 -S execve -k syscalls
-a exit,always -F arch=b32 -S execve -k syscalls

# Audit rules for monitoring network activity
-a exit,always -F arch=b64 -S bind -S connect -k network
-a exit,always -F arch=b32 -S bind -S connect -k network
```

These audit rules help monitor and log specific events related to file system changes, user logins, system calls, and network activity on a Linux system. The associated keys (passwd_changes, shadow_changes, sudoers_changes, logins, syscalls, network) are useful for identifying and searching for these events in the audit logs.



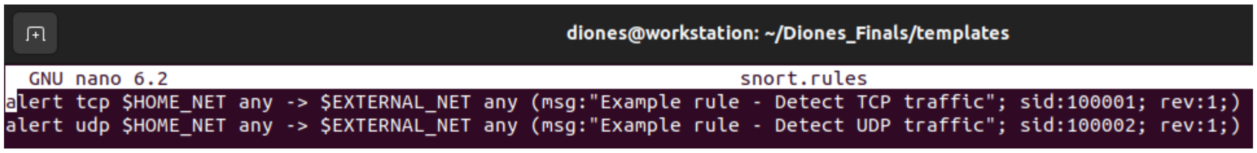
```
diones@workstation: ~/Diones_Finals/templates
  GNU nano 6.2                                            sec_log_config
/var/log/auth.log {
    monthly
    rotate 12
    compress
    delaycompress
    missingok
    notifempty
    create 644 root adm
    sharedscripts
    postrotate
        /bin/systemctl reload rsyslog >/dev/null 2>&1 || true
    endscript
}

/var/log/fail2ban.log {
    weekly
    rotate 8
    compress
    delaycompress
    missingok
    notifempty
    create 644 root adm
    sharedscripts
    postrotate
        /bin/systemctl reload rsyslog >/dev/null 2>&1 || true
    endscript
}
```

Both configurations follow the same structure and utilize common options provided by the logrotate tool. These configurations ensure that the specified log files are rotated regularly, compressed, and the log rotation is coordinated with the rsyslog service through a postrotate command. The notifempty option prevents unnecessary rotations when the log files are empty.



```
diones@workstation: ~/Diones_Finals/templates
  GNU nano 6.2                                              snort.conf
# Network interface to monitor
var HOME_NET any

# Preprocessor configuration
preprocessor frag3_global: max_frags 65536
preprocessor stream5_global: track_tcp yes, track_udp yes
preprocessor http_inspect: global iis_unicode_map unicode.map 1252

# Output configuration
output alert_syslog: LOG_AUTH LOG_ALERT

# Include Snort rules files
include $RULE_PATH/rules/*.rules
```

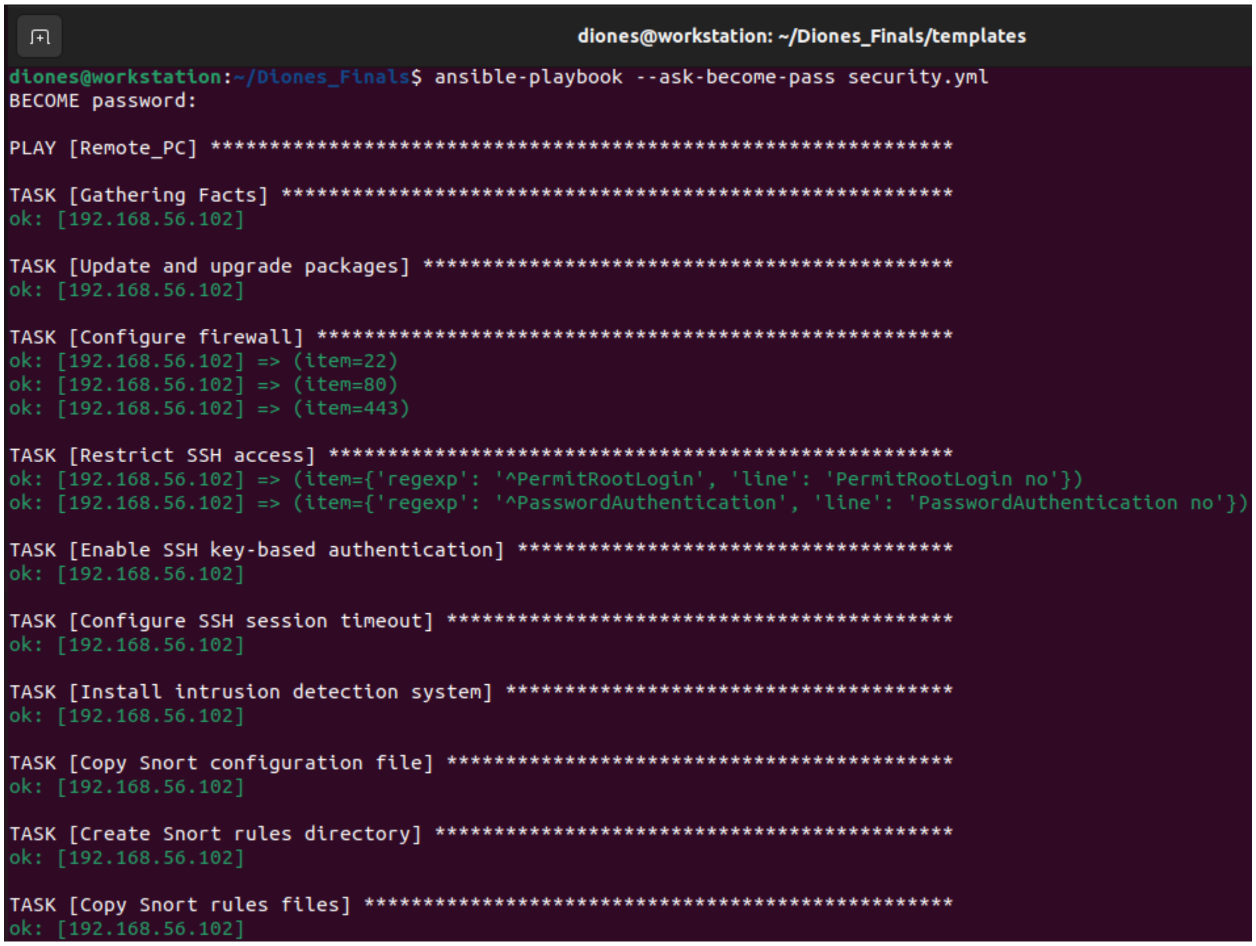
This ‘snort.conf’ file sets up Snort to monitor network traffic on any interface, configures preprocessors to handle various aspects of the traffic, specifies an output configuration to log alerts to syslog, and includes Snort rules files for detection.



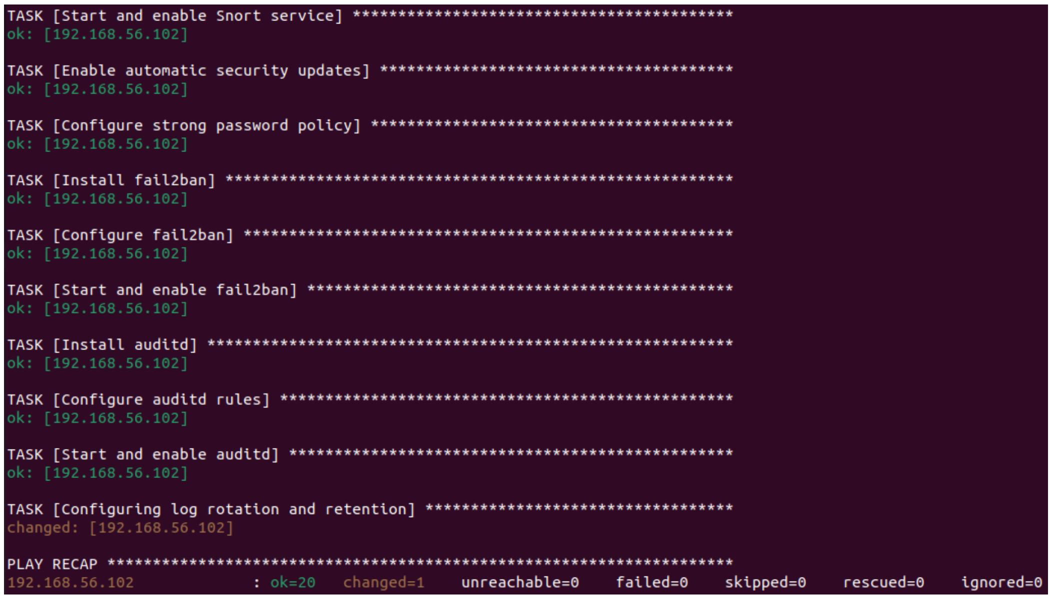
```
dioness@workstation: ~/Diones_Finals/templates
 GNU nano 6.2                                          snort.rules
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Example rule - Detect TCP traffic"; sid:100001; rev:1;)
alert udp $HOME_NET any -> $EXTERNAL_NET any (msg:"Example rule - Detect UDP traffic"; sid:100002; rev:1;)
```

These rules are simple examples demonstrating how Snort rules are structured. The rules, as provided, will generate alerts for any TCP and UDP traffic from the internal network to any external network. These are basic templates, and in a real-world scenario, rules would be more specific and tailored to the types of threats or activities you want to detect on your network.

PLAYBOOK OUTPUT

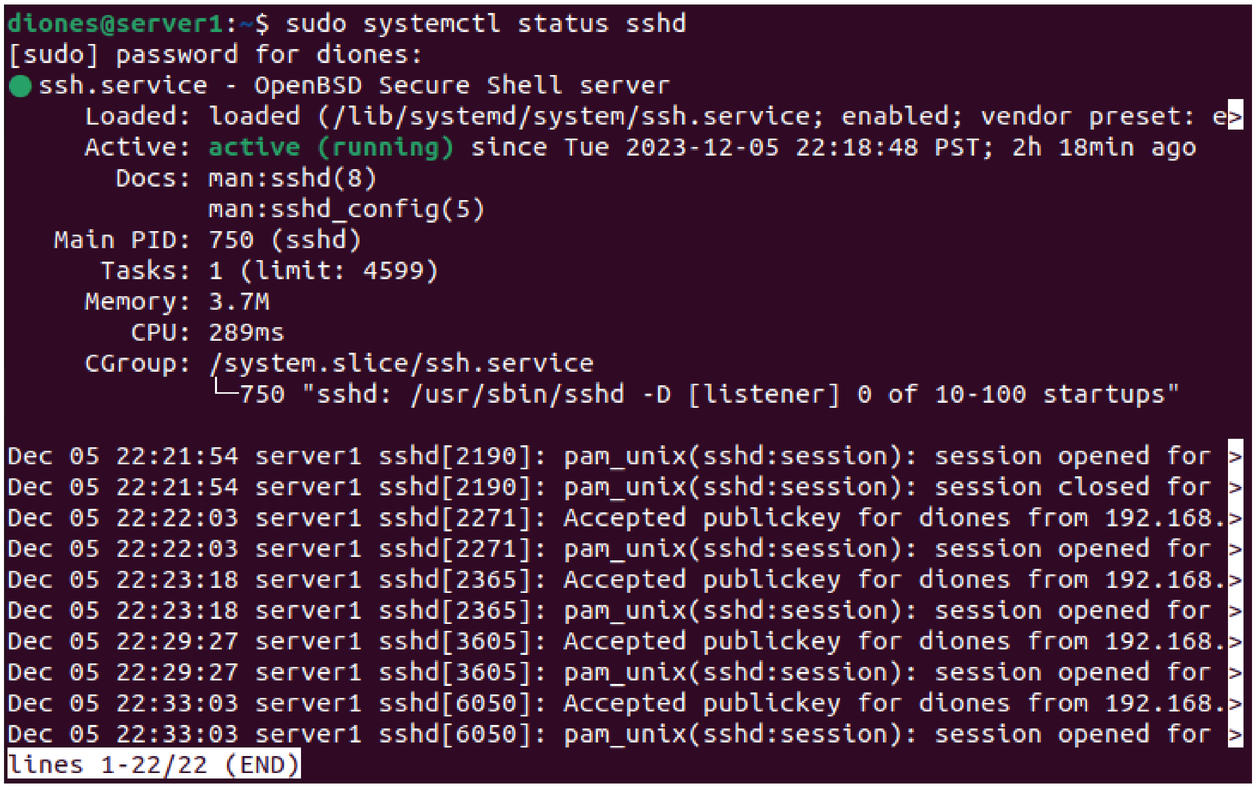


```
dioness@workstation: ~/Diones_Finals/templates
diones@workstation:~/Diones_Finals$ ansible-playbook --ask-become-pass security.yml
BECOME password:

PLAY [Remote_PC] ***************************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.102]

TASK [Update and upgrade packages] *********************************************
ok: [192.168.56.102]

TASK [Configure firewall] ******************************************************
ok: [192.168.56.102] => (item=22)
ok: [192.168.56.102] => (item=80)
ok: [192.168.56.102] => (item=443)

TASK [Restrict SSH access] *****************************************************
ok: [192.168.56.102] => (item={'regexp': '^PermitRootLogin', 'line': 'PermitRootLogin no'})
ok: [192.168.56.102] => (item={'regexp': '^PasswordAuthentication', 'line': 'PasswordAuthentication no'})

TASK [Enable SSH key-based authentication] *************************************
ok: [192.168.56.102]

TASK [Configure SSH session timeout] *******************************************
ok: [192.168.56.102]

TASK [Install intrusion detection system] **************************************
ok: [192.168.56.102]

TASK [Copy Snort configuration file] *******************************************
ok: [192.168.56.102]

TASK [Create Snort rules directory] ********************************************
ok: [192.168.56.102]

TASK [Copy Snort rules files] **************************************************
ok: [192.168.56.102]
```



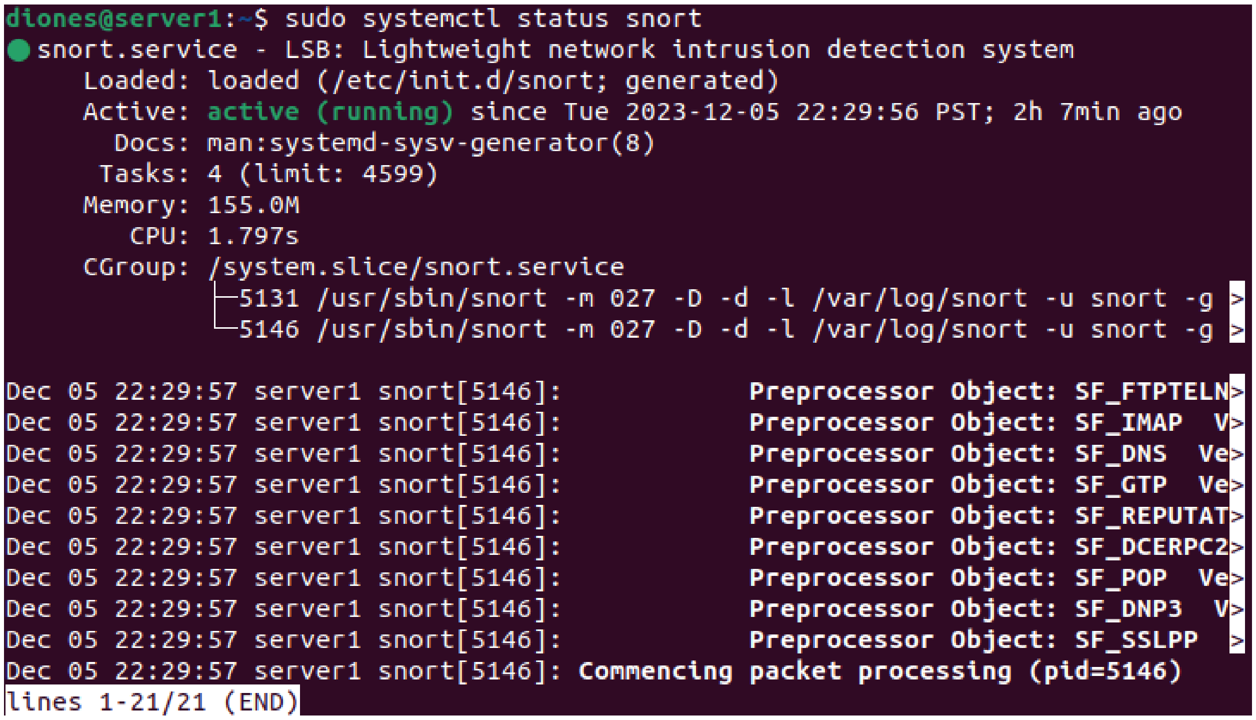
```
TASK [Start and enable Snort service] *******************************************
ok: [192.168.56.102]

TASK [Enable automatic security updates] ****************************************
ok: [192.168.56.102]

TASK [Configure strong password policy] *****************************************
ok: [192.168.56.102]

TASK [Install fail2ban] *********************************************************
ok: [192.168.56.102]

TASK [Configure fail2ban] *******************************************************
ok: [192.168.56.102]

TASK [Start and enable fail2ban] ************************************************
ok: [192.168.56.102]

TASK [Install auditd] ***********************************************************
ok: [192.168.56.102]

TASK [Configure auditd rules] ***************************************************
ok: [192.168.56.102]

TASK [Start and enable auditd] **************************************************
ok: [192.168.56.102]

TASK [Configuring log rotation and retention] ***********************************
changed: [192.168.56.102]

PLAY RECAP **********************************************************************
192.168.56.102             : ok=20   changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

SSHD STATUS



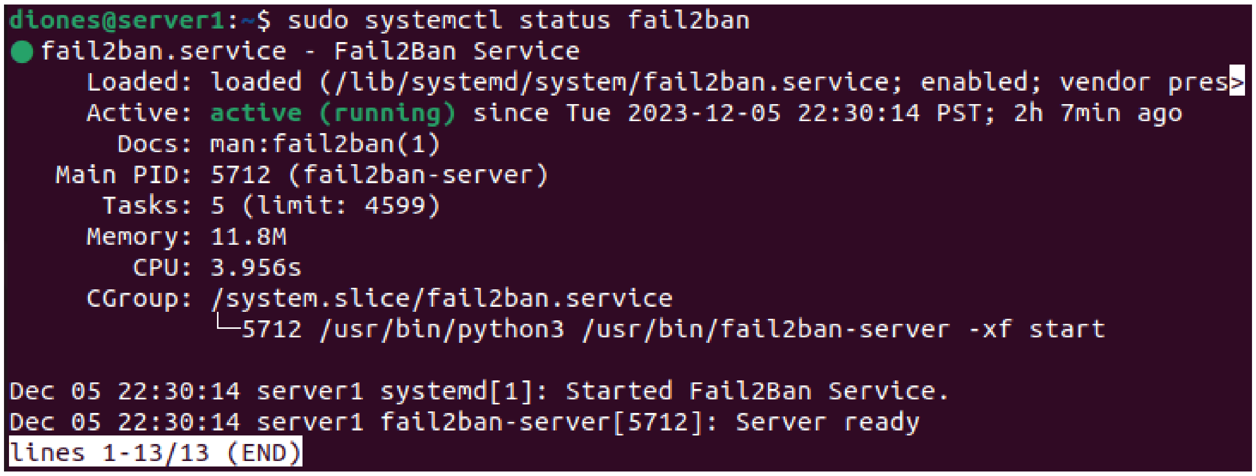
```
diones@server1:~$ sudo systemctl status sshd
[sudo] password for diones:
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: e>
     Active: active (running) since Tue 2023-12-05 22:18:48 PST; 2h 18min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 750 (sshd)
      Tasks: 1 (limit: 4599)
     Memory: 3.7M
        CPU: 289ms
     CGroup: /system.slice/ssh.service
             └─750 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Dec 05 22:21:54 server1 sshd[2190]: pam_unix(sshd:session): session opened for >
Dec 05 22:21:54 server1 sshd[2190]: pam_unix(sshd:session): session closed for >
Dec 05 22:22:03 server1 sshd[2271]: Accepted publickey for diones from 192.168.>
Dec 05 22:22:03 server1 sshd[2271]: pam_unix(sshd:session): session opened for >
Dec 05 22:23:18 server1 sshd[2365]: Accepted publickey for diones from 192.168.>
Dec 05 22:23:18 server1 sshd[2365]: pam_unix(sshd:session): session opened for >
Dec 05 22:29:27 server1 sshd[3605]: Accepted publickey for diones from 192.168.>
Dec 05 22:29:27 server1 sshd[3605]: pam_unix(sshd:session): session opened for >
Dec 05 22:33:03 server1 sshd[6050]: Accepted publickey for diones from 192.168.>
Dec 05 22:33:03 server1 sshd[6050]: pam_unix(sshd:session): session opened for >
lines 1-22/22 (END)
```

SNORT STATUS



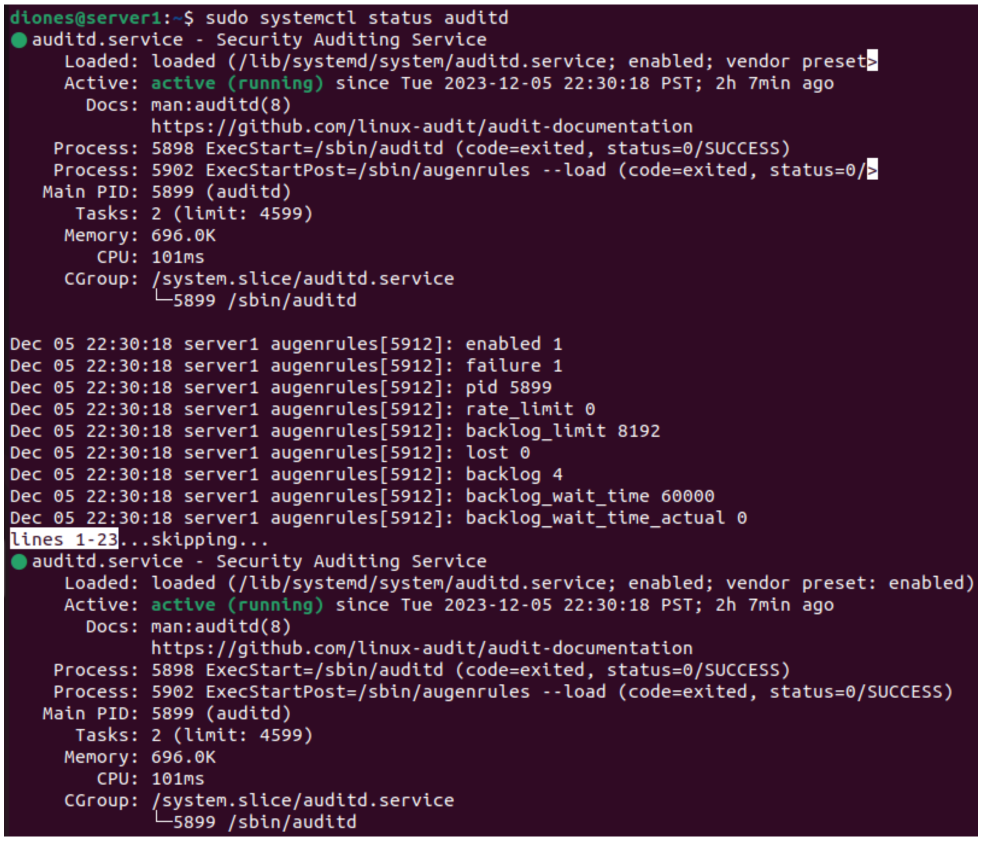
```
diones@server1:~$ sudo systemctl status snort
● snort.service - LSB: Lightweight network intrusion detection system
     Loaded: loaded (/etc/init.d/snort; generated)
     Active: active (running) since Tue 2023-12-05 22:29:56 PST; 2h 7min ago
       Docs: man:systemd-sysv-generator(8)
      Tasks: 4 (limit: 4599)
     Memory: 155.0M
        CPU: 1.797s
     CGroup: /system.slice/snort.service
             ├─5131 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g >
             └─5146 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g >

Dec 05 22:29:57 server1 snort[5146]:           Preprocessor Object: SF_FTPTELN>
Dec 05 22:29:57 server1 snort[5146]:           Preprocessor Object: SF_IMAP  V>
Dec 05 22:29:57 server1 snort[5146]:           Preprocessor Object: SF_DNS  Ve>
Dec 05 22:29:57 server1 snort[5146]:           Preprocessor Object: SF_GTP  Ve>
Dec 05 22:29:57 server1 snort[5146]:           Preprocessor Object: SF_REPUTAT>
Dec 05 22:29:57 server1 snort[5146]:           Preprocessor Object: SF_DCERPC2>
Dec 05 22:29:57 server1 snort[5146]:           Preprocessor Object: SF_POP  Ve>
Dec 05 22:29:57 server1 snort[5146]:           Preprocessor Object: SF_DNP3  V>
Dec 05 22:29:57 server1 snort[5146]:           Preprocessor Object: SF_SSLPP  >
Dec 05 22:29:57 server1 snort[5146]: Commencing packet processing (pid=5146)
lines 1-21/21 (END)
```

FAIL2BAN STATUS

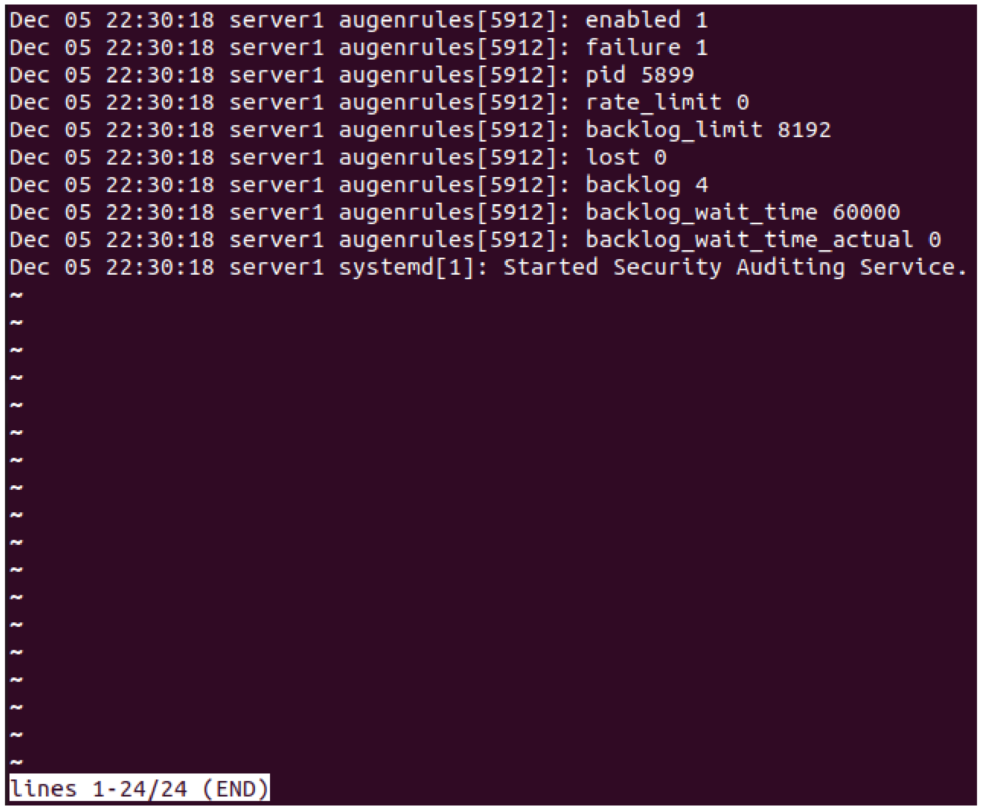


```
diones@server1:~$ sudo systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
     Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; vendor pres>
     Active: active (running) since Tue 2023-12-05 22:30:14 PST; 2h 7min ago
       Docs: man:fail2ban(1)
   Main PID: 5712 (fail2ban-server)
      Tasks: 5 (limit: 4599)
     Memory: 11.8M
        CPU: 3.956s
     CGroup: /system.slice/fail2ban.service
             └─5712 /usr/bin/python3 /usr/bin/fail2ban-server -xf start

Dec 05 22:30:14 server1 systemd[1]: Started Fail2Ban Service.
Dec 05 22:30:14 server1 fail2ban-server[5712]: Server ready
lines 1-13/13 (END)
```

AUDITD STATUS



```
diones@server1:~$ sudo systemctl status auditd
● auditd.service - Security Auditing Service
     Loaded: loaded (/lib/systemd/system/auditd.service; enabled; vendor preset>
     Active: active (running) since Tue 2023-12-05 22:30:18 PST; 2h 7min ago
       Docs: man:auditd(8)
             https://github.com/linux-audit/audit-documentation
    Process: 5898 ExecStart=/sbin/auditd (code=exited, status=0/SUCCESS)
    Process: 5902 ExecStartPost=/sbin/augenrules --load (code=exited, status=0/>
   Main PID: 5899 (auditd)
      Tasks: 2 (limit: 4599)
     Memory: 696.0K
        CPU: 101ms
     CGroup: /system.slice/auditd.service
             └─5899 /sbin/auditd

Dec 05 22:30:18 server1 augenrules[5912]: enabled 1
Dec 05 22:30:18 server1 augenrules[5912]: failure 1
Dec 05 22:30:18 server1 augenrules[5912]: pid 5899
Dec 05 22:30:18 server1 augenrules[5912]: rate_limit 0
Dec 05 22:30:18 server1 augenrules[5912]: backlog_limit 8192
Dec 05 22:30:18 server1 augenrules[5912]: lost 0
Dec 05 22:30:18 server1 augenrules[5912]: backlog 4
Dec 05 22:30:18 server1 augenrules[5912]: backlog_wait_time 60000
Dec 05 22:30:18 server1 augenrules[5912]: backlog_wait_time_actual 0
lines 1-23...skipping...
● auditd.service - Security Auditing Service
     Loaded: loaded (/lib/systemd/system/auditd.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-12-05 22:30:18 PST; 2h 7min ago
       Docs: man:auditd(8)
             https://github.com/linux-audit/audit-documentation
    Process: 5898 ExecStart=/sbin/auditd (code=exited, status=0/SUCCESS)
    Process: 5902 ExecStartPost=/sbin/augenrules --load (code=exited, status=0/SUCCESS)
   Main PID: 5899 (auditd)
      Tasks: 2 (limit: 4599)
     Memory: 696.0K
        CPU: 101ms
     CGroup: /system.slice/auditd.service
             └─5899 /sbin/auditd
```



```
Dec 05 22:30:18 server1 augenrules[5912]: enabled 1
Dec 05 22:30:18 server1 augenrules[5912]: failure 1
Dec 05 22:30:18 server1 augenrules[5912]: pid 5899
Dec 05 22:30:18 server1 augenrules[5912]: rate_limit 0
Dec 05 22:30:18 server1 augenrules[5912]: backlog_limit 8192
Dec 05 22:30:18 server1 augenrules[5912]: lost 0
Dec 05 22:30:18 server1 augenrules[5912]: backlog 4
Dec 05 22:30:18 server1 augenrules[5912]: backlog_wait_time 60000
Dec 05 22:30:18 server1 augenrules[5912]: backlog_wait_time_actual 0
Dec 05 22:30:18 server1 systemd[1]: Started Security Auditing Service.
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
lines 1-24/24 (END)
```

CONCLUSION AND LEARNINGS:

Security Awareness: This activity emphasizes the importance of security measures to protect infrastructure.

Configuration Management: Understanding and implementing configuration changes using Ansible for security enhancement.

Multi-Layered Security: Employing a multi-layered security approach with measures such as firewalls, intrusion detection, and access controls.

Automation: Utilizing automation tools like Ansible for efficient and consistent security policy enforcement.

Continuous Improvement: Regularly updating and adapting security measures to address evolving threat

Documentation: Importance of documenting security configurations and policies for future reference and auditing.

Incident Response: Implementing fail2ban for automated response to security incidents, demonstrating a proactive security stance.

This security plan aims to create a robust defense against potential threats, combining network security, access controls, intrusion detection, and proactive measures to ensure the integrity and availability of the infrastructure.