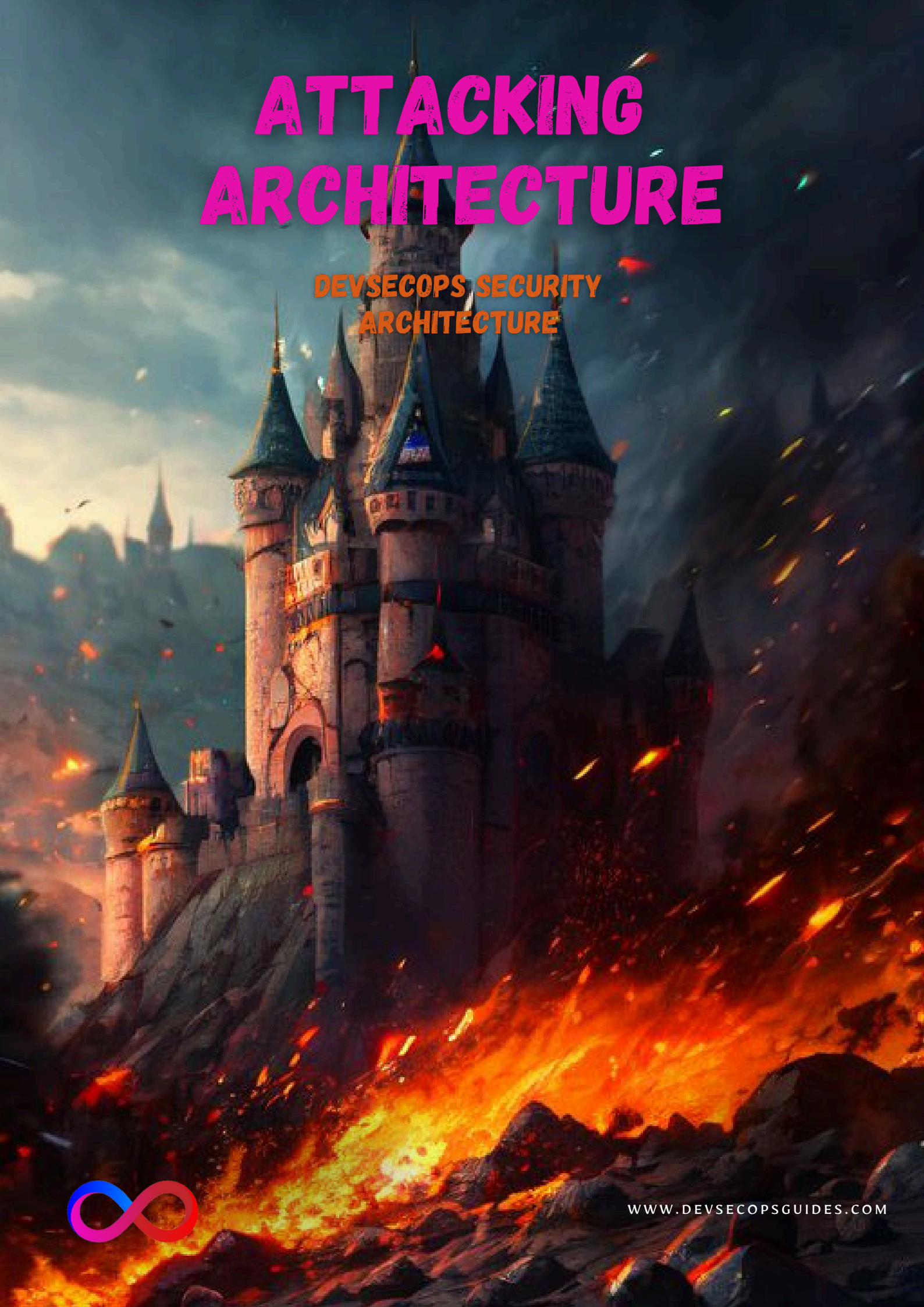
ATTACKING ARCHITECTURE

DEVSECOPS SECURITY ARCHITECTURE



WWW.DEVSECOPSGUIDES.COM

DevSecOps Security Architecture

• Jun 3, 2024 • 19 min read

Table of contents

Integrating a source code management system like GitLab with an identity management system like FreeIPA

Step 1: Install GitLab on CentOS 7

Honeypot Network and Services in DevSecOps Security Architecture

Flume log collection

Kafka Knowledge System

Zookeeper Knowledge System

ElastAlert ES Alarm Tool

Elastic Knowledge System

- Viewing ElasticSearch Cluster:
- Kibana Installation and Configuration:
- Timelion Configuration:
- Logstash Configuration:
- Nginx HTTP Auth Basic Configuration:

Real IP address Detection

- Nginx Configuration on proxy_server_1:
- Nginx Configuration on proxy_server_2:
- Nginx Configuration on web_server:
- Nginx Configuration on proxy_server_1:

- Nginx Configuration on proxy_server_2:
- Nginx Configuration on proxy_server_1:
- Nginx Configuration on proxy_server_2:

Nginx configuration log format

- Initial Nginx Configuration:
- Configure in JSON format and add fields:
- Nginx log field meaning:
- Print request_body:
- Print response_body:
- Complete Nginx log configuration:
- Nginx logs directly output to Logstash:
- Logstash Configuration:

Container security tools

- Anchore Engine: Image scanning tool
- Usage:
- Clair: Image scanning tool
- Trivy: Image scanning tool
- Usage:
- Docker Bench: Container Security Baseline Detection Tool
- Docker Scan: Image scanning tool
- Tool Comparison Table:

osquery operating system detection and analysis

- Osquery Table Queries:
- osquery + ELK + Kolide Fleet
- Pushing Hosts to Fleet Server using Launcher:
- Pushing Hosts to Fleet Server using Osquery:

jumpserver open source bastion server

- Install Jumpserver:
- Install SSH Server and WebSocket Server (Coco):
- Configure automatic startup:

wazuh Host Intrusion Detection System

- Wazuh Server Installation:
- Wazuh API Installation:
- Wazuh Client Installation:
- Configure Filebeat:
- Configure Logstash:

Bro Network Security Monitoring

- Compile and install from source:
- BroControl:
- Bro Command-Line:
- Analyzing SSH login logs:

GitHub Information Leak Monitoring

Application layer denial of service attacks

- SlowLoris Mode:
- SlowPost Mode:
- SlowRead Mode:

Slowloris

- Usage:

Resources

Show less

In the rapidly evolving landscape of cybersecurity, DevSecOps Security Architecture emerges as a critical framework that integrates security practices within the DevOps process, ensuring that security is a shared responsibility throughout the entire development lifecycle. This holistic approach encompasses various facets, including account security, application security, big data protection, and CAPTCHA security. By embedding security measures such as code audits, data security protocols, and honeypots into the development pipeline, DevSecOps enhances the resilience of applications against potential threats. Identity and Access Management (IAM), GitLab integration with FreeIPA, Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), and comprehensive infrastructure security protocols further bolster this architecture, safeguarding both the development environment and the deployed applications.

Moreover, DevSecOps Security Architecture places significant emphasis on continuous monitoring and analysis through log analysis, ensuring real-time threat detection and response. It integrates a wide array of security tools, from web application firewalls (WAF) to vulnerability management systems, facilitating proactive defense strategies. The architecture also supports specialized security branches such as network security, miscellaneous security considerations, and the dynamic interplay of Red vs. Blue team exercises, which simulate attack and defense scenarios to enhance readiness. Supplemented by in-depth study notes and security documentation, this comprehensive approach not only addresses immediate security concerns but also fosters a culture of continuous learning and improvement within the organization, ensuring robust protection against an ever-growing array of cyber threats.

Integrating a source code management system like GitLab with an identity management system like FreeIPA

Integrating a source code management system like GitLab with an identity management system like FreeIPA can significantly streamline authentication and authorization processes. Below are the steps to set up GitLab on CentOS 7 and

configure it to connect with FreeIPA for LDAP authentication, including the necessary commands and configurations.

Step 1: Install GitLab on CentOS 7

Update and install dependencies:

```
sudo yum update -y
sudo yum install -y curl policycoreutils-python openssh-server perl
sudo systemctl enable sshd
sudo systemctl start sshd
sudo yum install -y postfix
sudo systemctl enable postfix
sudo systemctl start postfix
```

Add GitLab repository and install GitLab:

```
curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ee/script.rpm.sh | sudo bash
sudo EXTERNAL_URL="http://gitlab.example.com" yum install -y gitlab-ee
```

Configure and start GitLab:

```
sudo gitlab-ctl reconfigure
```

Install FreeIPA server:

```
sudo yum install -y ipa-server ipa-server-dns
```

Set up FreeIPA server:

```
COPY
sudo ipa-server-install --setup-dns
```

Start FreeIPA services:

```
COPY
sudo systemctl start ipa
sudo systemctl enable ipa
```

Modify the GitLab configuration file: Edit the `/etc/gitlab/gitlab.rb` file with the following LDAP settings:

```
COPY
gitlab_rails['ldap_enabled'] = true

gitlab_rails['ldap_servers'] = YAML.load <<-'EOS'
  main:
    label: 'LDAP'
    host: 'ipa.bloodzer0.com'
    port: 389
    uid: 'uid'
    bind_dn: 'uid=admin,cn=users,cn=compat,dc=bloodzer0,dc=com'
    password: 'password2'
    encryption: 'plain'
    active_directory: true
    allow_username_or_email_login: false
    lowercase_usernames: false
```

```
    block_auto_created_users: false
    base: 'cn=users,cn=compat,dc=bloodzer0,dc=com'
    user_filter: ''
EOS
```

Reconfigure and restart GitLab:

```
sudo gitlab-ctl reconfigure
sudo gitlab-ctl restart
```

Add a group and a user in FreeIPA:

```
ipa group-add gitlab_user
ipa user-add john --first=John --last=Doe --password
ipa group-add-member gitlab_user --users=john
```

Modify the GitLab configuration file again to restrict access to specific LDAP group members: Edit the `/etc/gitlab/gitlab.rb` file with the updated user filter:

```
gitlab_rails['ldap_enabled'] = true

gitlab_rails['ldap_servers'] = YAML.load <<-'EOS'
main:
    label: 'LDAP'
    host: 'ipa.bloodzer0.com'
    port: 389
    uid: 'uid'
    bind_dn: 'uid=admin,cn=users,cn=accounts,dc=bloodzer0,dc=com'
    password: 'password2'
```

```
    encryption: 'plain'
    active_directory: true
    allow_username_or_email_login: false
    lowercase_usernames: false
    block_auto_created_users: false
    base: 'cn=users,cn=accounts,dc=bloodzer0,dc=com'
    user_filter:
(memberOf=cn=gitlab_user,cn=groups,cn=accounts,dc=bloodzer0,dc=com)
EOS
```

Reconfigure and restart GitLab:

COPY

```
sudo gitlab-ctl reconfigure
sudo gitlab-ctl restart
```

Honeypot Network and Services in DevSecOps Security Architecture

In a DevSecOps security architecture, honeypots play a crucial role by acting as decoy systems designed to attract and analyze potential attackers. By setting up honeypots, organizations can gather valuable intelligence on attack vectors and methods, thereby improving their overall security posture. One commonly used honeypot for SSH services is Kippo, which simulates a vulnerable SSH server to lure attackers. Below are detailed steps to install and configure Kippo on CentOS 7.

First, ensure that all necessary dependencies are installed:

COPY

```
# Install EPEL repository and required packages
sudo yum install epel-release -y
sudo yum install python2-pip python-devel gcc -y
```

```
# Install Python libraries
sudo pip install Twisted==15.2.0  # Specific version to avoid issues
sudo pip install pycrypto zope.interface cryptography pyasn1
```

Create a dedicated user for Kippo:

```
COPY
sudo useradd kippo
su - kippo
```

Download and configure Kippo:

```
COPY
# Download Kippo from GitHub
git clone https://github.com/desaster/kippo.git
cd kippo

# Copy the default configuration file
cp kippo.cfg.dist kippo.cfg

# Modify configuration as needed
nano kippo.cfg

# Sample changes to kippo.cfg:
# (Edit 'hostname' to make it more enticing to attackers)
# hostname = honeypot

# Start Kippo (cannot be executed as root)
./start.sh
```

Verify Kippo is running: After starting Kippo, it will listen on port 2222:

```
# Verify by connecting to the honeypot
ssh root@localhost -p 2222

# Monitor the log to see interaction
tail -f ~/kippo/log/kippo.log
```

Step 3: Advanced Configuration - Port Forwarding

To make the honeypot more realistic, forward traffic from port 22 to Kippo's port 2222:

COPY

```
# Add an iptables rule to redirect traffic from port 22 to 2222
sudo iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-
port 2222

# Test the setup by connecting to port 22
ssh root@localhost
```

Flume log collection

Apache Flume is a robust, reliable, and highly available service for efficiently collecting, aggregating, and moving large amounts of log data from many different sources to a centralized data store. In a DevSecOps security architecture, Flume can be used to collect logs from various sources, such as application servers, and transport them to a centralized logging system for analysis and monitoring. Below are the steps to install and configure Flume on CentOS 7, along with case studies demonstrating its use.

1. **Install Java:**

```
# Extract Java JDK
tar -xf jdk-8u191-linux-x64.tar.gz -C /app/

# Set Java environment variables
vim /etc/profile

# Add the following lines to the file
export JAVA_HOME=/app/jdk1.8.0_191
export PATH=$JAVA_HOME/bin:$PATH

# Load the new environment variables
source /etc/profile

# Verify Java installation
java -version
```

Download and Install Flume:

COPY

```
# Extract Flume
tar -xf apache-flume-1.8.0-bin.tar.gz -C /app/

# Set Flume environment variables
vim /etc/profile

# Add the following lines to the file
export FLUME_HOME=/app/apache-flume-1.8.0-bin
export PATH=$FLUME_HOME/bin:$PATH

# Load the new environment variables
source /etc/profile

# Copy and configure flume-env.sh
cp /app/apache-flume-1.8.0-bin/conf/flume-env.sh.template /app/apache-
flume-1.8.0-bin/conf/flume-env.sh
```

```
vim /app/apache-flume-1.8.0-bin/conf/flume-env.sh

# Add the following line to set Java home
export JAVA_HOME=/app/jdk1.8.0_191

# Verify Flume installation
flume-ng version
```

Step 2: Flume Case Study - Collect Data from a Network Port and Output to Console

1. **Create Configuration File:**

COPY

```
vim /app/apache-flume-1.8.0-bin/conf/example-1.conf

# Add the following configuration
# a1: agent name
# r1: source name
# c1: channel name
# k1: sink name

# Name the components on this agent
a1.sources = r1
a1.channels = c1
a1.sinks = k1

# Describe/configure the source
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444

# Describe the sink
a1.sinks.k1.type = logger

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
```

```
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

Start Flume Agent:

```
flume-ng agent --name a1 --conf /app/apache-flume-
1.8.0-bin/conf --conf-file /app/apache-flume-1.8.0-
bin/conf/example-1.conf -Dflume.root.logger=INFO,console
```

Step 3: Real-Time File Monitoring Output to Console

1. **Create Configuration File:**

```
vim /app/apache-flume-1.8.0-bin/conf/example-2.conf

# Add the following configuration
# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = exec
a1.sources.r1.command = tail -F /var/log/messages
a1.sources.r1.shell = /bin/sh -c

# Describe the sink
a1.sinks.k1.type = logger
```

```
# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

Start Flume Agent:

COPY

```
flume-ng agent --name a1 --conf /app/apache-flume-
1.8.0-bin/conf --conf-file /app/apache-flume-1.8.0-
bin/conf/example-2.conf -Dflume.root.logger=INFO,console
```

Step 4: Transfer Logs from Server1 to Console of Server2

1. **Create Configuration File on Server1:**

COPY

```
vim /app/apache-flume-1.8.0-bin/conf/exec-memory-avro.conf

# Add the following configuration
exec-memory-avro.sources = exec-source
exec-memory-avro.sinks = avro-sink
exec-memory-avro.channels = memory-channel

exec-memory-avro.sources.exec-source.type = exec
exec-memory-avro.sources.exec-source.command = tail -F
/var/log/messages
exec-memory-avro.sources.exec-source.shell = /bin/sh -c
```

```
exec-memory-avro.sinks.avro-sink.type = avro
exec-memory-avro.sinks.avro-sink.hostname = localhost
exec-memory-avro.sinks.avro-sink.port = 44444

exec-memory-avro.channels.memory-channel.type = memory
exec-memory-avro.channels.memory-channel.capacity = 1000
exec-memory-avro.channels.memory-channel.transactionCapacity = 100

exec-memory-avro.sources.exec-source.channels = memory-channel
exec-memory-avro.sinks.avro-sink.channel = memory-channel
```

Create Configuration File on Server2:

COPY

```
vim /app/apache-flume-1.8.0-bin/conf/avro-memory-logger.conf

# Add the following configuration
avro-memory-logger.sources = avro-source
avro-memory-logger.sinks = logger-sink
avro-memory-logger.channels = memory-channel

avro-memory-logger.sources.avro-source.type = avro
avro-memory-logger.sources.avro-source.bind = localhost
avro-memory-logger.sources.avro-source.port = 44444

avro-memory-logger.sinks.logger-sink.type = logger

avro-memory-logger.channels.memory-channel.type = memory
avro-memory-logger.channels.memory-channel.capacity = 1000
avro-memory-logger.channels.memory-channel.transactionCapacity = 100

avro-memory-logger.sources.avro-source.channels = memory-channel
avro-memory-logger.sinks.logger-sink.channel = memory-channel
```

Start Flume Agent on Server2 (must be executed first):

```
flume-ng agent --name avro-memory-logger --conf /app/apache-flume-1.8.0-bin/conf --conf-file /app/apache-flume-1.8.0-bin/conf/avro-memory-logger.conf -Dflume.root.logger=INFO,console
```

Start Flume Agent on Server1:

```
flume-ng agent --name exec-memory-avro --conf /app/apache-flume-1.8.0-bin/conf --conf-file /app/apache-flume-1.8.0-bin/conf/exec-memory-avro.conf -Dflume.root.logger=INFO,console
```

Kafka Knowledge System

Apache Kafka is a distributed streaming platform capable of handling high-throughput, fault-tolerant, and real-time data streaming. Here's a guide to installing and configuring Kafka on CentOS 7.

Step 1: Install Java Environment

1. **Install Java:**

```
yum install java-1.8.0-openjdk -y
```

Step 2: Download and Extract Kafka

1. **Download Kafka:**

```
wget http://mirror.bit.edu.cn/apache/kafka/2.3.0/kafka_2.11-2.3.0.tgz
```

Extract Kafka:

```
COPY
tar -xf kafka_2.11-2.3.0.tgz
```

Move Kafka to a reasonable location:

```
COPY
mv kafka_2.11-2.3.0 /opt/kafka2.11
```

Step 3: Configure System Environment Variables

1. **Set environment variables:**

```
COPY
echo "export KAFKA_HOME=/opt/kafka2.11" >> /etc/profile
echo "export PATH=\$PATH:\$KAFKA_HOME/bin" >> /etc/profile
source /etc/profile
```

Step 4: Start Kafka

1. **Start Zookeeper (assuming Zookeeper is running at 10.10.10.19:2181):**

```
COPY
# Start Kafka server
kafka-server-start.sh -daemon /opt/kafka2.11/config/server.properties
```

Verify Kafka is running:

```
jps -m
```

Step 5: Configure Kafka

1. **Edit Kafka configuration:**

```
vim /opt/kafka2.11/config/server.properties
```

Update the following properties:

```
broker.id=0
listeners=PLAINTEXT://PLAINTEXT:9092
advertised.listeners=PLAINTEXT://kafka-server:9092
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
log.dirs=/tmp/kafka-logs
num.partitions=1
zookeeper.connect=10.10.10.19:2181
zookeeper.connection.timeout.ms=6000
```

Step 6: Run Kafka in Standalone Mode

1. **Start Kafka server**

```
kafka-server-start.sh -daemon /opt/kafka2.11/config/server.properties
```

Step 7: Run Kafka in Cluster Mode

1. **Single Node, Multiple Brokers:**
    - Duplicate the `server.properties` file and modify the `broker.id` for each instance

```
cp /opt/kafka2.11/config/server.properties
/opt/kafka2.11/config/server_1.properties
cp /opt/kafka2.11/config/server.properties
/opt/kafka2.11/config/server_2.properties

vim /opt/kafka2.11/config/server_1.properties
# Update broker.id and log directories
broker.id=1
log.dirs=/tmp/kafka-logs-1

vim /opt/kafka2.11/config/server_2.properties
# Update broker.id and log directories
broker.id=2
log.dirs=/tmp/kafka-logs-2

# Start multiple brokers
kafka-server-start.sh -daemon
/opt/kafka2.11/config/server_1.properties
kafka-server-start.sh -daemon
/opt/kafka2.11/config/server_2.properties
```

2. **Multiple Nodes, Multiple Brokers:**
    - Repeat the above steps on multiple servers.

Step 8: Configure Log Retention

1. **Edit the configuration file to manage log retention:**

```
vim /opt/kafka2.11/config/server.properties
```

Set log retention policies:

```
log.retention.hours=1
log.retention.minutes=60
log.retention.ms=3600000
log.retention.bytes=1073741824
```

Step 9: Using Kafka

1. **Create a topic:**

```
kafka-topics.sh --zookeeper 10.10.10.19:2181 --create --
--replication-factor 1 --partitions 1 --topic topic_name
```

List topics:

```
kafka-topics.sh --zookeeper 10.10.10.19:2181 --list
```

View topic details:

```
kafka-topics.sh --zookeeper 10.10.10.19:2181 -
-describe --topic topic_name
```

Modify topic partition:

```
kafka-topics.sh --zookeeper 10.10.10.19:2181 -
-alter --partitions 20 --topic topic_name
```

Delete a topic:

```
kafka-topics.sh --zookeeper 10.10.10.19:2181 -
-delete --topic topic_name
```

Start a consumer:

```
kafka-console-consumer.sh --zookeeper
10.10.10.19:2181 --topic topic_name
```

Start a producer:

```
kafka-console-producer.sh --broker-
list 10.10.10.19:9092 --topic topic_name
```

Consume from the beginning:

```
kafka-console-consumer.sh --zookeeper
10.10.10.19:2181 --topic topic_name --from-beginning
```

Pitfalls

1. **Error message when deleting topic:**
    - **Solution: Enable topic deletion**

```
vim /opt/kafka2.11/config/server.properties
# Add the following line
delete.topic.enable=true
```

Zookeeper Knowledge System

Apache Zookeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. Here's a guide to installing and configuring Zookeeper on CentOS 7.

Step 1: Install Java Environment

```
yum install java-1.8.0-openjdk.x86_64
java-1.8.0-openjdk-devel.x86_64 -y
```

Move Zookeeper to a reasonable location:

```
mv apache-zookeeper-3.5.5-bin /opt/zookeeper3.5.5
```

Configure System Environment Variables

```
echo "export ZOOKEEPER_HOME=/opt/zookeeper3.5.5" >> /etc/profile
echo "export PATH=\$PATH:\$ZOOKEEPER_HOME/bin" >> /etc/profile
source /etc/profile
```

Configure Zookeeper

1. **Copy the sample configuration file:**

```
cp /opt/zookeeper3.5.5/conf/zoo_sample.cfg
/opt/zookeeper3.5.5/conf/zoo.cfg
```

Edit the configuration file:

```
vim /opt/zookeeper3.5.5/conf/zoo.cfg
```

Update the following properties:

```
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/tmp/zookeeper
clientPort=2181
```

Start Zookeeper

1. **Start Zookeeper server:**

```
zkServer.sh start
```

Use Zookeeper Client

1. **Start Zookeeper client:**

```
zkCli.sh
```

- **View help:**
  `h`
- **List root directory nodes:**
  `ls /`
- **View root node status:**
  `stat /`
- **Get root node data and details:**
  `get /`
- **Exit the client:**
  `quit`

ElastAlert ES Alarm Tool

ElastAlert is a simple framework for alerting on anomalies, spikes, or other patterns of interest from data in Elasticsearch. Below is a step-by-step guide to installing and configuring ElastAlert on CentOS 7.

Clone and install ElastAlert:

```
git clone https://github.com/Yelp/elastalert.git && cd elastalert
pip install -r requirements.txt
python setup.py install
```

Create ElastAlert index:

```
elastalert-create-index
```

Test rule file:

```
elastalert-test-rule rule.yaml
```

Start monitoring and alerting:

```
python -m elastalert.elastalert --verbose --
rule /root/elastalert/example_rules/rule.yaml
```

Create configuration directory and copy sample configuration:

```
mkdir /etc/elastalert
cp /root/elastalert/config.yaml.example /etc/elastalert/config.yaml
```

Create rule directory and copy example rule:

```
mkdir /etc/elastalert/rules
cp /root/elastalert/example_rules/example_frequency.yaml
/etc/elastalert/rules/rule.yaml
```

Create ElastAlert service file:

```
vim /etc/systemd/system/elastalert.service
```

Download DingTalk plugin:

```
git clone https://github.com/xuyaoqiang/elastalert-dingtalk-plugin.git
```

Copy plugin to ElastAlert directory:

```
cp -r elastalert-dingtalk-plugin/elastalert_modules/ /etc/elastalert/
```

Update rule configuration:

```
vim /etc/elastalert/rules/rule.yaml
```

Enable Alerts:

```
python -m elastalert.elastalert --verbose --config
/etc/elastalert/config.yaml --rule /etc/elastalert/rules/rule.yaml
```

Configure Service Startup

1. **Edit service file:**

```
vim /etc/systemd/system/elastalert.service
```

2. **Optimize the alarm format:**

```
alert_text: |
    kibana_url: "https://hostname:5601/app/kibana"
    alarm_reason: "1 minute: login.php accessed at least 10 times"
    alarm_name: {}
    request_uri: {}
    request_ip: {}
    response_status: {}
```

Elastic Knowledge System

1. Import the private key.
2. Install Elasticsearch RPM.
3. Enable and start Elasticsearch service.
4. Modify the Elasticsearch configuration file `/etc/elasticsearch/elasticsearch.yml`.
5. Adjust basic configuration options such as cluster name, node name, data and log paths, network settings, etc.

6. Configure JVM options in `/etc/elasticsearch/jvm.options`.

```
# Import private key
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch

# Install Elasticsearch RPM
rpm -ivh elasticsearch-7.2.0-x86_64.rpm

# Enable and start Elasticsearch service
systemctl enable elasticsearch.service
systemctl start elasticsearch.service

# Modify Elasticsearch configuration
# /etc/elasticsearch/elasticsearch.yml
```

Viewing ElasticSearch Cluster:

```
# Check cluster overview and health
curl http://10.10.10.15:9200/
curl http://10.10.10.15:9200/_cat/health?v
```

Kibana Installation and Configuration:

```
# Install Kibana RPM
rpm -ivh kibana-7.2.0-x86_64.rpm

# Enable and start Kibana service
systemctl enable kibana.service
systemctl start kibana.service
```

```
# Modify Kibana configuration
# /etc/kibana/kibana.yml
```

Timelion Configuration:

COPY

```
# Timelion usage example
# In Kibana Dev Tools
.es(index=wazuh-alerts-3.x*,q='rule.description: "PAM: Login session
opened."').label(登录),.es(index=wazuh-alerts-3.x*,q='rule.description:
"PAM: Login session closed."').label(退出)
```

Logstash Configuration:

COPY

```
# Install Logstash RPM
rpm -ivh logstash-7.2.0.rpm

# Modify Logstash configuration
# /etc/logstash/logstash.yml
# /etc/logstash/conf.d/*.conf

# Test Logstash configuration
/usr/share/logstash/bin/logstash -e 'input { stdin {} } output {
stdout {} }'
```

Nginx HTTP Auth Basic Configuration:

COPY

```
# Install Nginx and HTTP Auth Basic module
yum install epel-release -y
yum install nginx.x86_64 httpd-tools.x86_64 -y
```

```
# Configure Nginx with authentication
# /etc/nginx/nginx.conf
```

Real IP address Detection

Use X-Forwarded-For + realip module:

Nginx Configuration on proxy_server_1:

COPY

```
location / {
    proxy_pass http://10.10.10.17;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
```

Nginx Configuration on proxy_server_2:

COPY

```
location / {
    proxy_pass http://10.10.10.18;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
```

Nginx Configuration on web_server:

COPY

```
set_real_ip_from 10.10.10.16;
set_real_ip_from 10.10.10.17;
```

```
real_ip_header X-Forwarded-For;
real_ip_recursive on;
```

Nginx Configuration on proxy_server_1:

```
location / {
    proxy_pass http://10.10.10.17;
    proxy_set_header X-Forwarded-For $remote_addr;
}
```

Nginx Configuration on proxy_server_2:

```
location / {
    proxy_pass http://10.10.10.18;
}
```

Nginx Configuration on proxy_server_1:

```
location / {
    proxy_pass http://10.10.10.17;
    proxy_set_header X-Real-IP $remote_addr;
}
```

Nginx Configuration on proxy_server_2:

```
location / {
    proxy_pass http://10.10.10.18;
```

```
}
```

Nginx configuration log format

Initial Nginx Configuration:

COPY

```
log_format  main  '$remote_addr -
$remote_user [$time_local] "$request" '
                '$status $body_bytes_sent "$http_referer" '
                '"$http_user_agent" "$http_x_forwarded_for"';
```

Configure in JSON format and add fields:

COPY

```
log_format  main escape=json  '{ "@timestamp": "$time_local", '
                        '"remote_addr": "$remote_addr",'
                        '"remote_port": "$remote_port",'
                        '"scheme": "$scheme",'
                        '"request_uri": "$request_uri",'
                        '"request_method": "$request_method",'
                        '"request_time": "$request_time",'
                        '"request_length": "$request_length",'
                        '"response_status": "$status",'
                        '"body_bytes_sent": "$body_bytes_sent",'
                        '"http_referer": "$http_referer",'
                        '"http_user_agent": "$http_user_agent",'
                        '"http_x_forwarded_for":
"$http_x_forwarded_for",'
                        '"upstream_addr": "$upstream_addr",'
                        '"upstream_response_time":
"$upstream_response_time"}';
```

Nginx log field meaning:

Fields	Meaning	Example
body_bytes_sent	Response body bytes	3650
remote_addr	Client Address	10.10.10.1
remote_user	Client authentication username	admin
request	Request URI and protocol	GET /favicon.ico HTTP/1.1
request_length	Request length	571
request_method	Request method	GET
request_time	Request processing time	0.000
response_status	Return status code	404
time_local	Timestamp	16/Jun/2019:23:29:50 -0400
http_x_forwarded_for	XFF Information	192.168.1.1
...	...	...

Print request_body:

```
log_format main $request_body;
```

Print response_body:

```
log_format main $response_body;
```

Complete Nginx log configuration:

```
log_format  main escape=json  '{ "@timestamp": "$time_local", '
                    '"remote_addr": "$remote_addr",'
                    '"remote_port": "$remote_port",'
                    '"scheme": "$scheme",'
                    '"request_uri": "$request_uri",'
                    '"request_method": "$request_method",'
                    '"request_time": "$request_time",'
                    '"request_length": "$request_length",'
                    '"response_status": "$status",'
                    '"body_bytes_sent": "$body_bytes_sent",'
                    '"http_referer": "$http_referer",'
                    '"http_user_agent": "$http_user_agent",'
                    '"http_x_forwarded_for": "$http_x_forwarded_for",'
                    '"upstream_addr": "$upstream_addr",'
                    '"upstream_response_time":
"$upstream_response_time",'
                    '"request_body": "$request_body", '
                    '"response_body": "$response_body" }';
```

Nginx logs directly output to Logstash:

```
log_format  logstash  '$remote_addr -
$remote_user [$time_local] "$request" '
                  '$status $body_bytes_sent "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for"';

access_log
syslog:server=127.0.0.1:514,nohostname,tag=nginx_access_log logstash;
```

Logstash Configuration:

```
input {
    udp {
        host => "127.0.0.1"
        port => 514
    }
}
output {
    stdout {}
}
```

Container security tools

Anchore Engine: Image scanning tool

COPY

```
mkdir anchore
cd anchore
docker pull docker.io/anchore/anchore-engine:latest
docker create --name ae docker.io/anchore/anchore-engine:latest
docker cp ae:/docker-compose.yaml docker-compose.yaml
docker rm ae
docker-compose pull
docker-compose up -d
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python get-pip.py
pip install anchorecli
```

Usage:

COPY

```
anchore-cli --u admin --p foobar system status
anchore-cli --u admin --p foobar system feeds list
anchore-cli --u admin --p foobar image add
```

```
docker.io/library/nginx:latest
anchore-cli --u admin --p foobar image list
anchore-cli --u admin --p foobar image get
docker.io/library/nginx:latest
anchore-cli --u admin --p foobar image vuln
docker.io/library/nginx:latest os
anchore-cli --u admin --p foobar account list
anchore-cli --u admin --p foobar account user setpassword New_Password
anchore-cli --u admin --p foobar registry add registry_name user pass
```

Clair: Image scanning tool

COPY

```
mkdir clair
cd clair
curl -L
https://raw.githubusercontent.com/coreos/clair/master/contrib/compose/
docker-compose.yml -o docker-compose.yml
mkdir clair_config
curl -L
https://raw.githubusercontent.com/coreos/clair/master/config.yaml.samp
le -o ./clair_config/config.yaml
# Modify config.yaml
vim ./clair_config/config.yaml
docker-compose pull
docker-compose up -d
```

Trivy: Image scanning tool

COPY

```
rpm -ivh trivy_0.0.15_Linux-64bit.rpm
```

Usage:

```
trivy docker.io/library/nginx:latest
```

Docker Bench: Container Security Baseline Detection Tool

```
git clone https://github.com/docker/docker-bench-security.git && cd docker-bench-security
bash docker-bench-security.sh
```

Docker Scan: Image scanning tool

```
pip3 install dockerscan
dockerscan -h
```

Tool Comparison Table:

Comparison Items/Tools	Anchore	Clair	Trivy
Install	docker-compose	docker-compose	yum
Language	Python	Go	Go
Scanning Method	CVE vulnerability library scanning	CVE vulnerability library scanning	CVE vulnerability scanning
Scan Speed	A few minutes	A few minutes	A few minutes
Cross-platform	Good cross-platform performance	Cit	Cit

osquery operating system detection and analysis

Installation and Usage

```
# Install yum repository
rpm -ivh https://osquery-
packages.s3.amazonaws.com/centos7/noarch/osquery-s3-centos7-repo-1-
0.0.noarch.rpm

# Install osquery
yum install osquery.x86_64 -y

# Configure
cp /usr/share/osquery/osquery.example.conf /etc/osquery/osquery.conf

# Start
systemctl start osqueryd
```

COPY

Osquery Table Queries:

```
# Top 10 processes using the most memory
SELECT pid, name, uid, resident_size FROM processes ORDER BY
resident_size DESC LIMIT 10;

# Top 10 most active processes
SELECT COUNT(pid) AS total, name FROM processes GROUP BY name ORDER BY
total DESC LIMIT 10;

# View network port processes
SELECT DISTINCT process.name, listening.port, listening.address,
process.pid FROM processes AS process JOIN listening_ports AS
listening ON process.pid = listening.pid;
```

COPY

```
# View loaded kernel modules
SELECT name FROM kernel_modules;
```

osquery + ELK + Kolide Fleet

COPY

```
# Install mysql
wget https://repo.mysql.com//mysql80-community-release-el7-
1.noarch.rpm
rpm -ivh mysql80-community-release-el7-1.noarch.rpm
yum install mysql-community-server.x86_64 mysql-community-
client.x86_64 -y
# Create database
create database kolide;

# Install redis
yum install epel-release -y
yum install redis.x86_64 -y
# Start service
systemctl start redis.service

# Install Fleet
wget https://dl.kolide.co/bin/fleet_latest.zip
unzip fleet_latest.zip 'linux/*' -d fleet
cp fleet/linux/fleet /usr/bin/fleet
cp fleet/linux/fleetctl /usr/bin/fleetctl

# Connect to the database
/usr/bin/fleet prepare db --mysql_address=127.0.0.1:3306 --
mysql_database=kolide --mysql_username=root --mysql_password=password

# Configure certificate
openssl genrsa -out /etc/pki/tls/private/server.key 4096
openssl req -new -key /etc/pki/tls/private/server.key -out
/etc/pki/tls/certs/server.csr
openssl x509 -req -days 366 -in /etc/pki/tls/certs/server.csr -signkey
```

```
/etc/pki/tls/private/server.key -out /etc/pki/tls/certs/server.cert

# Start Fleet Server
mkdir /var/log/kolide
/usr/bin/fleet serve \
    --mysql_address=127.0.0.1:3306 \
    --mysql_database=kolide \
    --mysql_username=root \
    --mysql_password=password \
    --redis_address=127.0.0.1:6379 \
    --server_cert=/etc/pki/tls/certs/server.cert \
    --server_key=/etc/pki/tls/private/server.key \
    --logging_json \
    --osquery_result_log_file=/var/log/kolide/osquery_result \
    --osquery_status_log_file=/var/log/kolide/osquery_status \
        --auth_jwt_key=zJ+TKjgtGqCFX6XcF5SmDDsy4BCSReLH
```

Pushing Hosts to Fleet Server using Launcher:

COPY

```
# Download and unzip launcher
wget
https://github.com/kolide/launcher/releases/download/0.5.0/launcher_0.
5.0.zip
unzip launcher_0.5.0.zip

# Run launcher
cd linux/
./launcher --hostname=localhost:8080 --
enroll_secret=OB+ltcnAmEqykZXNthWNRv4qQMh9Rp0b --insecure
```

Pushing Hosts to Fleet Server using Osquery:

```
# Create enrollment secret
echo 'OB+ltcnAmEqykZXNthWNRv4qQMh9Rp0b' > /var/osquery/enroll_secret

# Configure certificate
mv 10.10.10.5_8080.pem /var/osquery/server.pem

# Run osqueryd
/usr/bin/osqueryd \
    --enroll_secret_path=/var/osquery/enroll_secret \
    --tls_server_certs=/var/osquery/server.pem \
    --tls_hostname=10.10.10.5:8080 \
    --host_identifier=hostname \
    --enroll_tls_endpoint=/api/v1/osquery/enroll \
    --config_plugin=tls \
    --config_tls_endpoint=/api/v1/osquery/config \
    --config_tls_refresh=10 \
    --disable_distributed=false \
    --distributed_plugin=tls \
    --distributed_interval=3 \
    --distributed_tls_max_attempts=3 \
    --distributed_tls_read_endpoint=/api/v1/osquery/distributed/read \
    --distributed_tls_write_endpoint=/api/v1/osquery/distributed/write
\
    --logger_plugin=tls \
    --logger_tls_endpoint=/api/v1/osquery/log \
    --logger_tls_period=10
```

jumpserver open source bastion server

Install Jumpserver:

COPY

```
git clone https://github.com/jumpserver/jumpserver.git
```

```
cd /opt/jumpserver/requirements
yum install $(cat rpm_requirements.txt) -y
pip install --upgrade pip setuptools
pip install -r requirements.txt

# Install Redis
yum install redis.x86_64 -y
systemctl start redis.service

# Install MySQL
yum install mariadb-devel.x86_64 mariadb-libs.x86_64 mariadb-
server.x86_64 mariadb.x86_64 -y

# Create database
mysql -uroot -p
create database jumpserver default charset 'utf8';
exit

# Modify configuration file /opt/jumpserver/config.py as per official
documentation

# Initialize database
cd /opt/jumpserver/utils/
bash make_migrations.sh

# Run Jumpserver
cd /opt/jumpserver
./jms start all # Run in background with -d parameter
```

Install SSH Server and WebSocket Server (Coco):

COPY

```
cd /opt
git clone https://github.com/jumpserver/coco.git

cd /opt/coco/requirements/
```

```
yum install $(cat rpm_requirements.txt) -y
pip install -r requirements.txt

# Modify configuration file /opt/coco/conf.py as per official
documentation

# Start Coco
cd /opt/coco/
./cocod start # Run in background with -d parameter
```

Configure automatic startup:

COPY

```
sed -i "s/START_TIMEOUT = 15/START_TIMEOUT = 40/g" /opt/jumpserver/jms

# Create systemd service for Jumpserver
vim /usr/lib/systemd/system/jms.service
# Add service configuration as per official documentation

# Create systemd service for Coco
vim /usr/lib/systemd/system/coco.service
# Add service configuration as per official documentation

# Enable and start the services
systemctl enable jms.service
systemctl enable coco.service
systemctl start jms.service
systemctl start coco.service
```

wazuh Host Intrusion Detection System

Wazuh Server Installation:

```
rpm -ivh wazuh-manager-3.3.1-1.x86_64.rpm
systemctl start wazuh-manager.service
systemctl status wazuh-manager.service
```

Wazuh API Installation:

```
wget -qO- https://rpm.nodesource.com/setup_8.x | bash -
yum install nodejs.x86_64
rpm -ivh wazuh-api-3.3.1-1.x86_64.rpm
systemctl start wazuh-api.service
systemctl status wazuh-api.service
```

Wazuh Client Installation:

```
rpm -ivh wazuh-agent-3.3.1-1.x86_64.rpm
vim /var/ossec/etc/ossec.conf # Modify configuration file
/var/ossec/bin/manage_agents # Import key
/var/ossec/bin/ossec-control start # Start service
```

Configure Filebeat:

```
vim /etc/filebeat/filebeat.yml
```

Filebeat.yml content:

```
filebeat.inputs:
- type: log
  paths:
    - "/var/ossec/logs/alerts/alerts.json"
  document_type: json
  json.message_key: log
  json.keys_under_root: true
  json.overwrite_keys: true

output.logstash:
  hosts: ["localhost:5044"]
```

Configure Logstash:

COPY

```
vim /etc/logstash/conf.d/wazuh.conf

input {
    beats {
        port => 5044
        codec => "json_lines"
    }
}

filter {
    if [data][srcip] {
        mutate {
            add_field => [ "@src_ip", "%{[data][srcip]}" ]
        }
    }
    if [data][aws][sourceIPAddress] {
        mutate {
            add_field => [ "@src_ip", "%{[data][aws]
[sourceIPAddress]}" ]
        }
```

```
    }
}

filter {
    geoip {
        source => "@src_ip"
        target => "GeoLocation"
        fields => ["city_name", "country_name", "region_name",
"location"]
    }
    date {
        match => ["timestamp", "ISO8601"]
        target => "@timestamp"
    }
    mutate {
        remove_field => [ "timestamp", "beat", "input_type", "tags",
"count", "@version", "log", "offset", "type", "@src_ip", "host"]
    }
}

output {
    elasticsearch {
        hosts => ["localhost:9200"]
        index => "wazuh-alerts-3.x-%{+YYYY.MM.dd}"
        document_type => "wazuh"
    }
}
```

Bro Network Security Monitoring

Compile and install from source:

COPY

```
wget https://www.bro.org/downloads/bro-2.5.5.tar.gz
tar -xf bro-2.5.5.tar.gz && cd bro-2.5.5
```

```
./configure
make && make install
```

BroControl:

```
vim /usr/local/bro/etc/node.cfg # Configure network interfaces
broctl # Start Bro
```

Bro Command-Line:

```
bro -i ens33 # Start Bro with specified interface
bro -r *.pcap # Analyze a pcap file
```

Analyzing SSH login logs:

View SSH logs in Bro:

```
cat /usr/local/bro/logs/current/ssh.log
```

Edit the Logstash configuration file for SSH logs:

```
wget https://raw.githubusercontent.com/fakrul/bro-
elk/master/bro-ssh_log.conf
```

GitHub Information Leak Monitoring

1. Install required dependencies:

```
wget -qO- https://rpm.nodesource.com/setup_10.x | bash
yum install nodejs.x86_64 nodejs-npmlog.noarch -y
npm config set registry https://registry.npm.taobao.org
```

2. Install GitHub-Monitor: Follow the official installation documentation. Additionally, ensure that npm installation uses a reliable source.
3. Access the frontend: Once installed, you can access the frontend using the designated port.

Application layer denial of service attacks

Configure and install SlowHTTPTest:

```
./configure
autoreconf -ivf
make && make install
```

SlowHTTPTest offers various modes for conducting application layer DoS attacks. Here are some common use cases:

SlowLoris Mode:

```
slowhttptest -c 1000 -H -g -o my_header_stats -
i 10 -r 200 -t GET -u URL -x 24 -p 3
```

SlowPost Mode:

```
slowhttptest -c 3000 -B -g -o my_body_stats -
i 110 -r 200 -s 8192 -t POST -u URL -x 10 -p 3
```

SlowRead Mode:

```
slowhttptest -c 8000 -X -r 200 -w 512 -
y 1024 -n 5 -z 32 -k 3 -u URL -p 3
```

Slowloris

```
pip3 install slowloris
git clone https://github.com/gkbrk/slowloris.git && cd slowloris
```

Usage:

```
slowloris example.com
python3 slowloris.py -p 443 --https IP # Attack HTTPS on port 443
```

Resources

- https://bloodzer0.github.io/ossa/iam/gitlab_freeipa/

Subscribe to our newsletter

Read articles from **DevSecOpsGuides** directly inside your inbox. Subscribe to the newsletter, and don't miss out.

reza.rashidi.business@gmail.com SUBSCRIBE

Devops DevSecOps Security architecture Cloud infrastructure

Written by

RR **Reza Rashidi**

Published on

DevSecOpsGuides

MORE ARTICLES

RR **Reza Rashidi**

Attacking Secrets

RR **Reza Rashidi**

Attacking .NET

A Secrets and Vault Manager is a critical tool in modern IT infrastructure, designed to securely sto...

Attacking .NET applications often involves exploiting weaknesses in the code or the runtime environm...

RR **Reza Rashidi**

Attacking Rust

"Attacking Rust" delves into the intricacies of identifying and mitigating security vulnerabilities ...