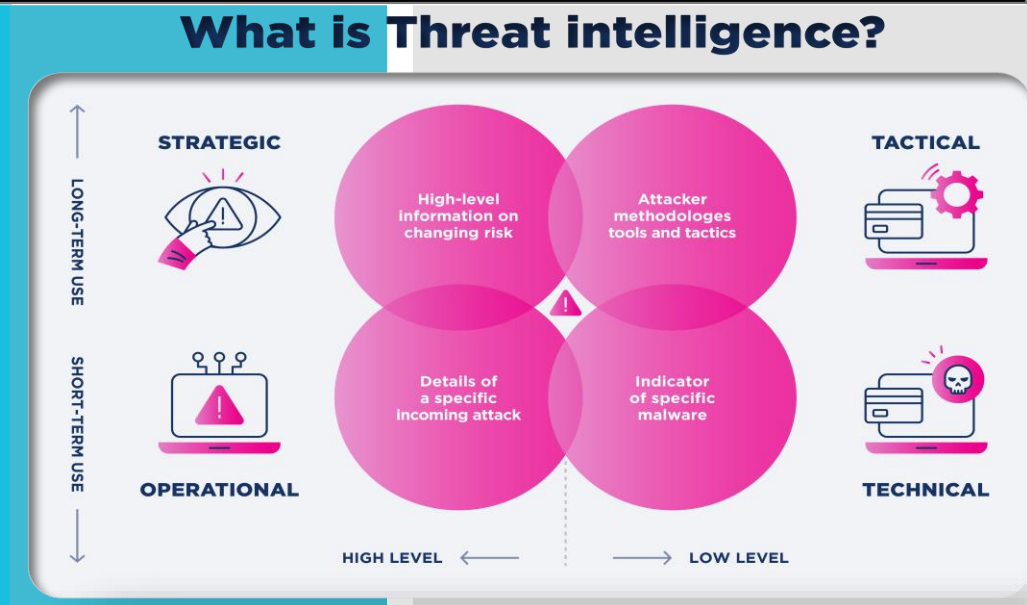
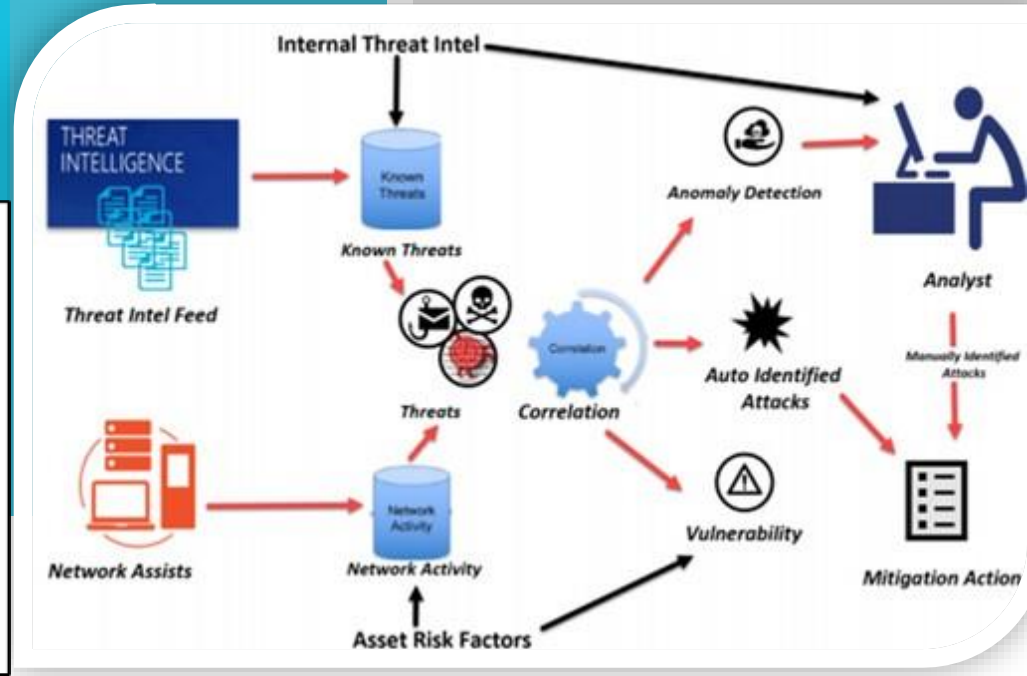
# Project Title

# IMPLEMENTING SECURITY MEASURES WITHIN A THREAT INTELLIGENCE SHARING PLATFORM

# Security Assessment of Existing Platform Architecture

Implementation Steps:

1. Conduct a Threat Modeling Exercise

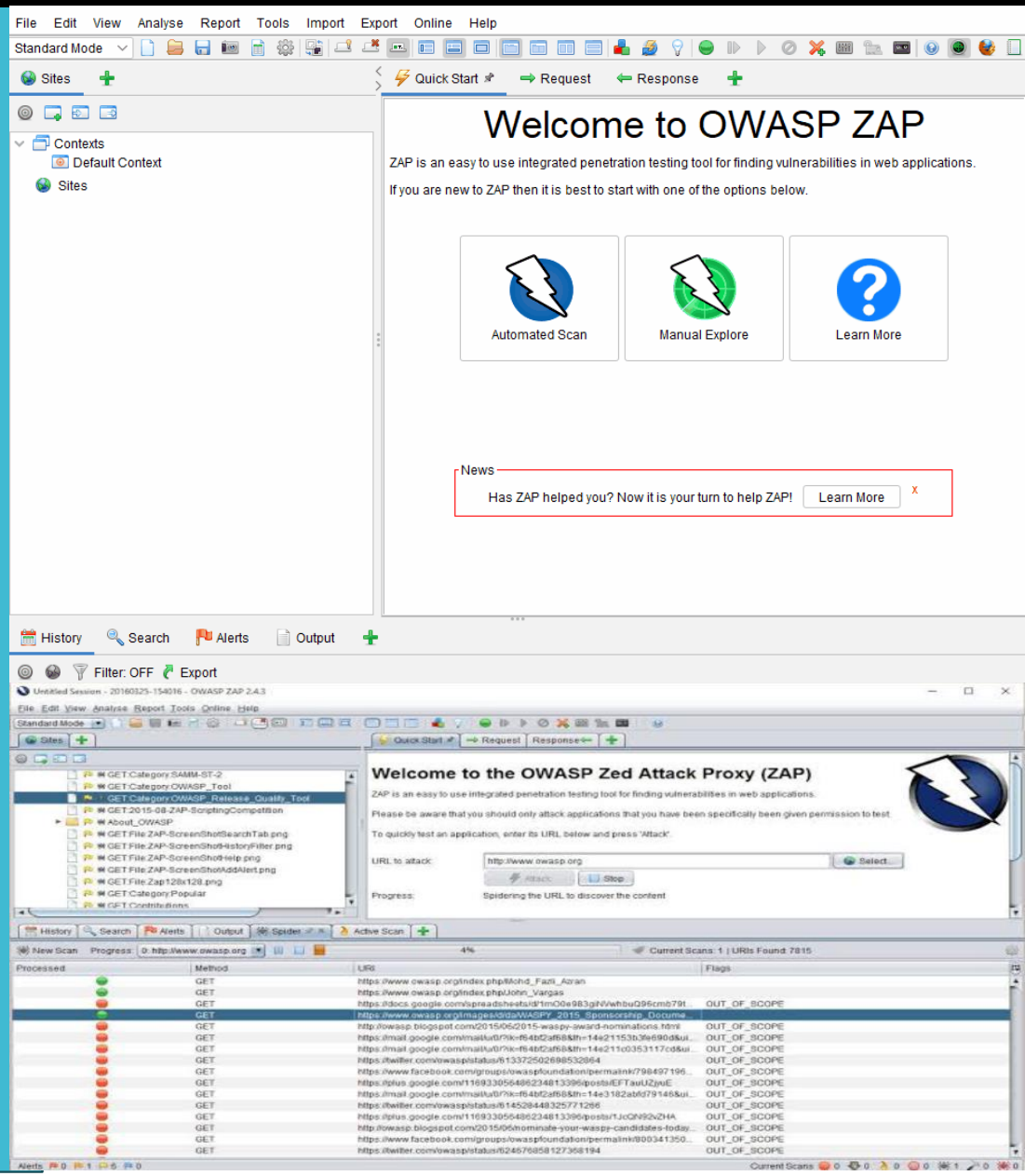
   Use **STRIDE** or **MITRE ATT&CK** to classify risks (Spoofing, Tampering, Denial of Service, etc.).
   Identify vulnerabilities in MISP's **data ingestion, processing, and sharing mechanisms**.

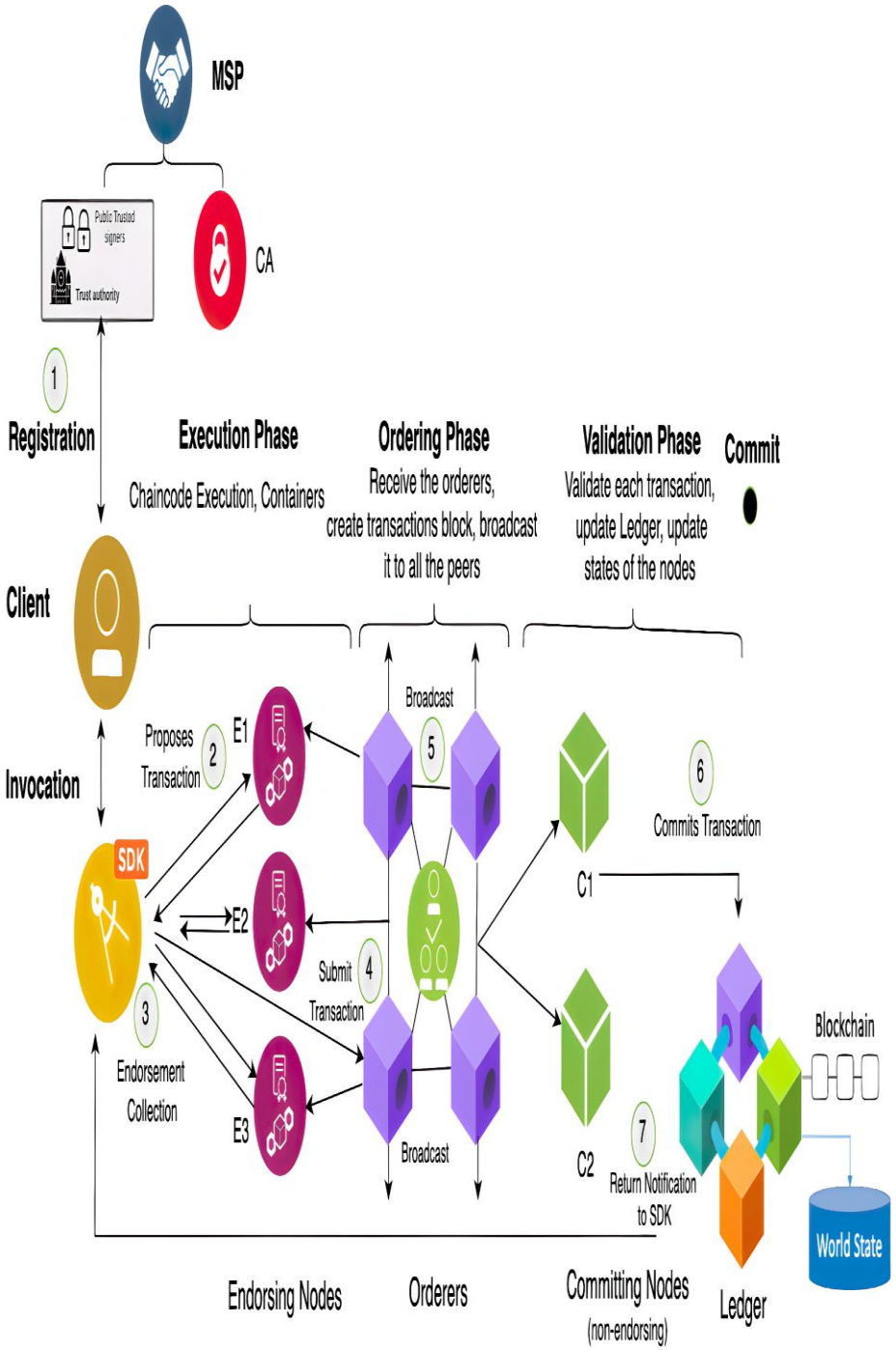
2. Perform Vulnerability Assessments

   Run penetration testing using **OWASP ZAP, OpenVAS, Nessus**
   Test for **API security flaws, SQL injection, privilege escalation**

3. Review Encryption & Authentication Mechanisms

   Audit **TLS configurations**, ensuring strong transport encryption (TLS 1.3)
   Validate authentication methods (password policies, **OAuth 2.0, API keys**).






## Key Tasks and Implementation Plan

### 1. Conduct a Threat Modeling Exercise

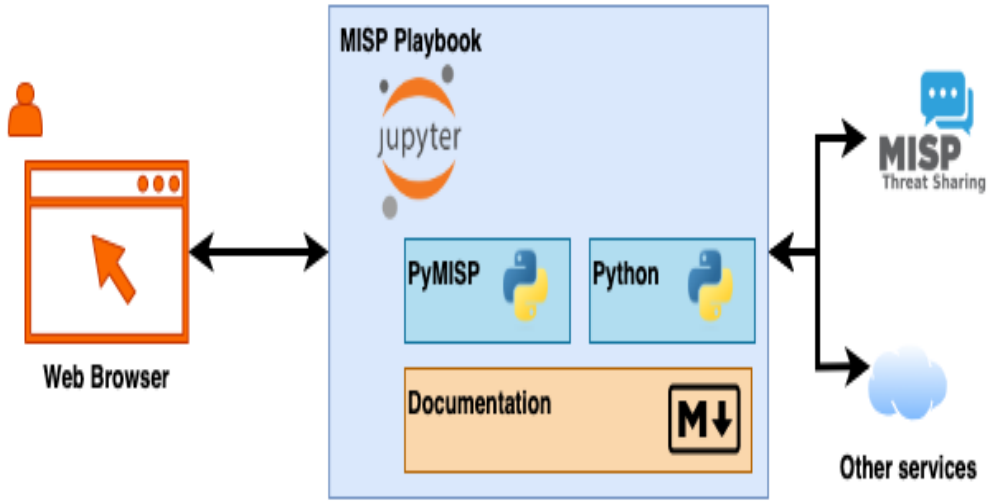
Use the **STRIDE framework** to model threats:

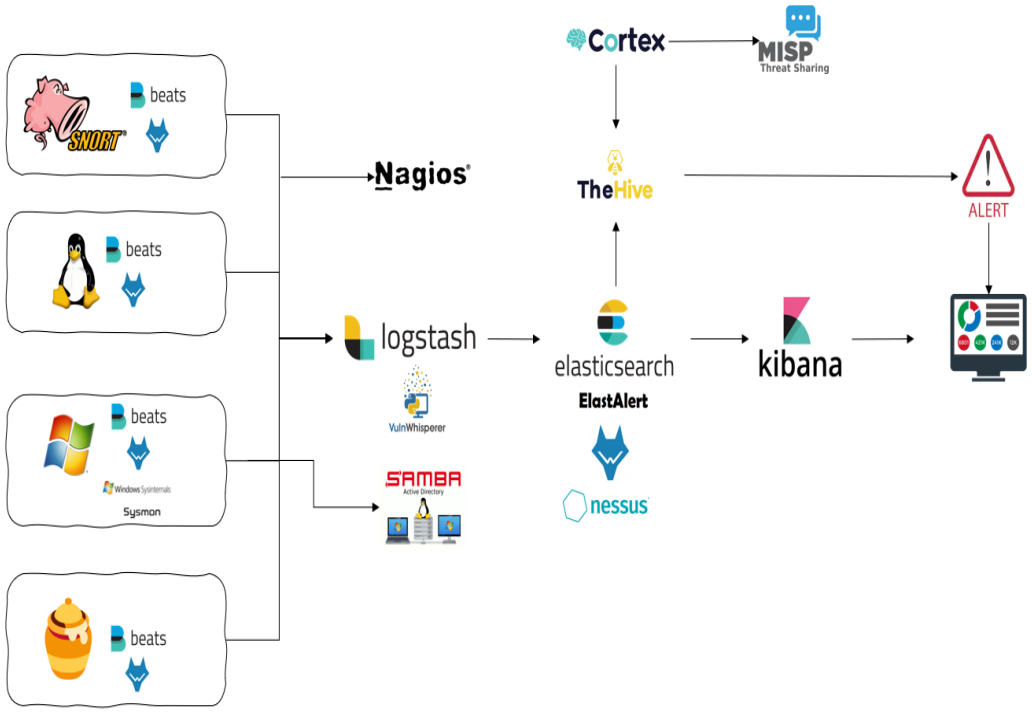
- **Spoofing** (fake identities)
- **Tampering** (data manipulation)
- **Repudiation** (lack of logs)
- **Information Disclosure** (sensitive data exposure)
- **Denial of Service**
- **Elevation of Privilege**

Focus areas in MISP:

- User authentication and role-based access
- API endpoints and integrations
- Event sharing (data disclosure)
- Background workers (possible DoS target)

Tool: Use **Microsoft Threat Modeling Tool** or **OWASP Threat Dragon**




## 2. Perform Vulnerability Assessments

- Run **OpenVAS**, **Nikto**, or **Nmap** scans on the server hosting MISP.
- Conduct **authenticated vulnerability scans** on:
  - Web UI (Apache/PHP)
  - API endpoints (access control)
  - Database (MySQL/MariaDB)

Use **OWASP ZAP** to test MISP's web interface for:

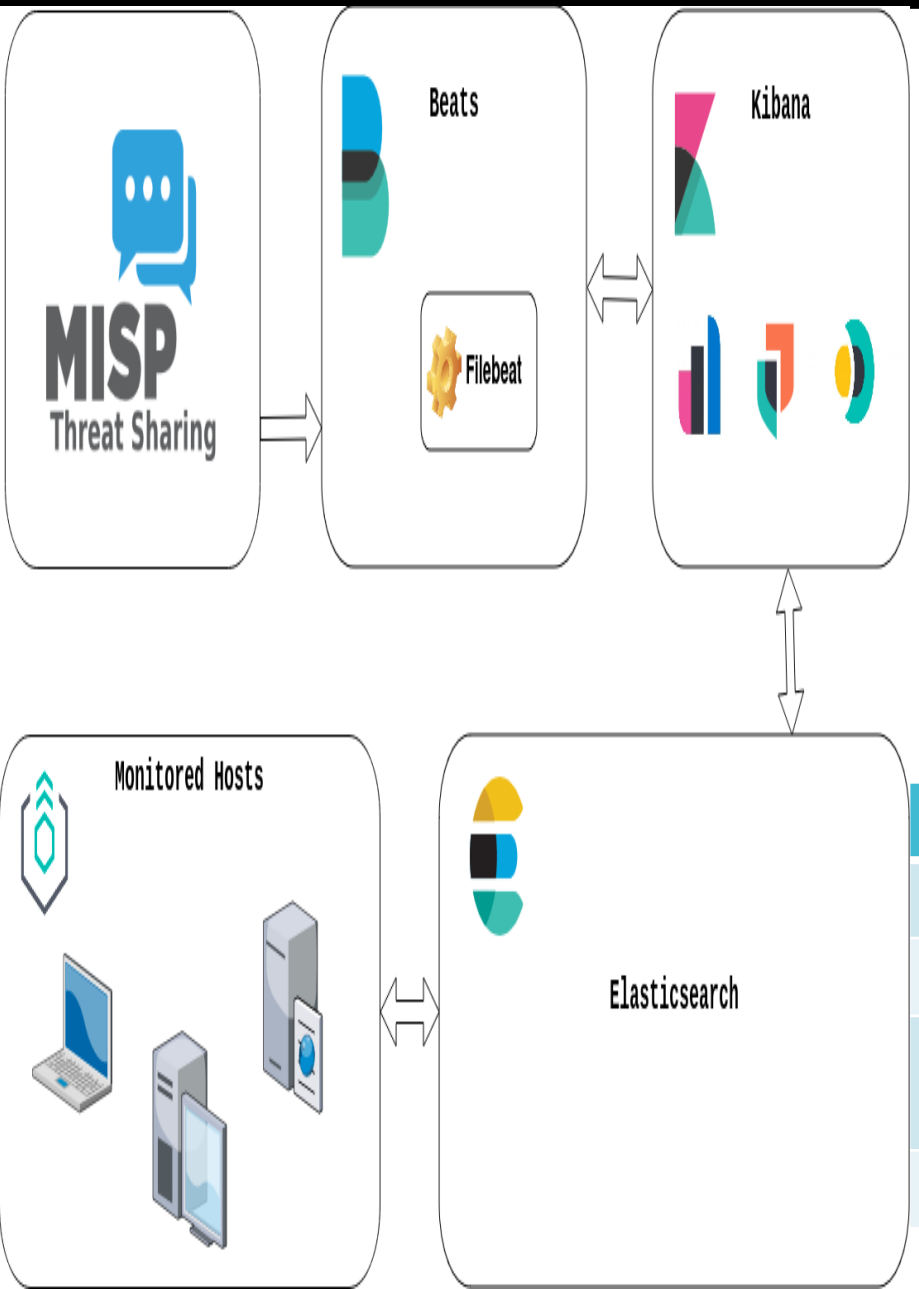
- XSS
- CSRF
- Insecure headers
- Auth bypass attempts

## 3. Review Current Security Controls

Check:

- **Encryption**
  - Is HTTPS/TLS enforced for all web/API traffic?
  - Are PGP/GPG keys used to sign/share events?
- **Authentication**
  - Are default users disabled?
  - Is MFA enforced (via SSO/SAML)?
- **Data Privacy**
  - Are event distribution settings correctly scoped (Org Only, Sharing Groups)?
  - Are audit logs active?

# Practical Step: Perform a Basic Vulnerability Scan with Nmap on MISP Server




🛠 **Example**

bash

```
# Scan open ports and service versions
    nmap -sV -p- <misp-server-ip>
```

Then check for vulnerabilities:

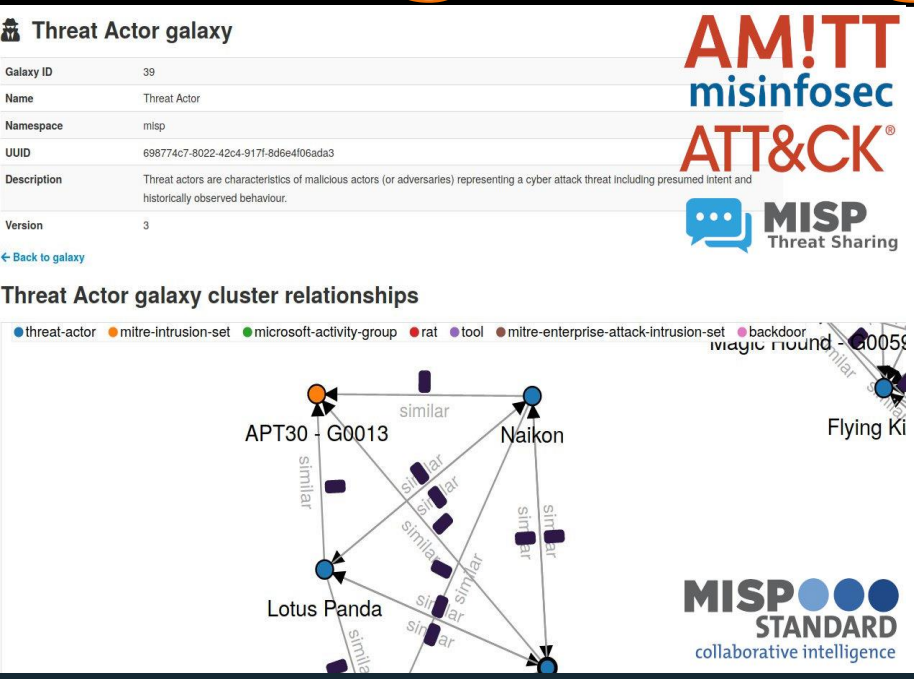
```
nmap --script vuln -p 80,443 <misp-server-ip>
```

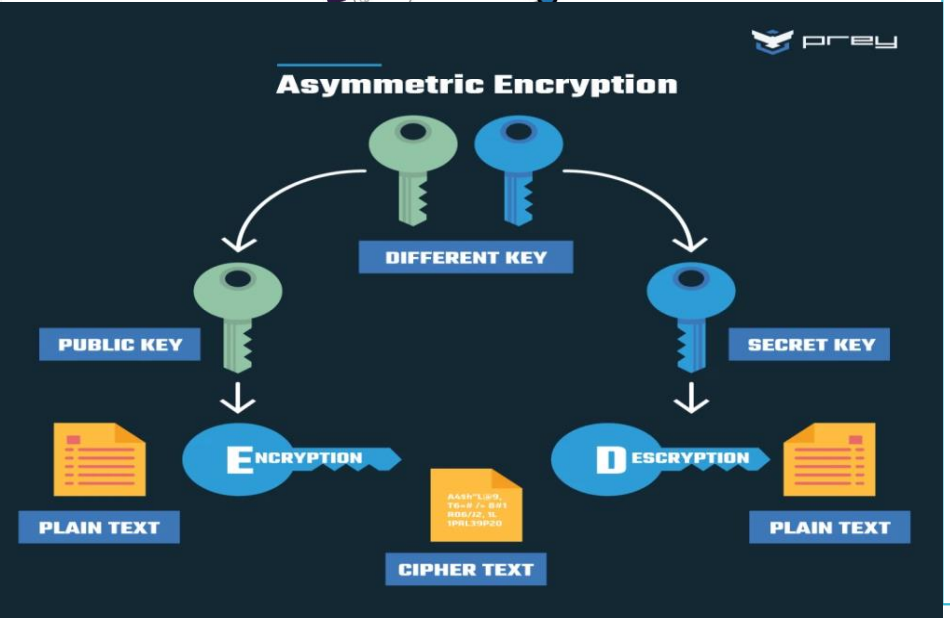
This identifies potential issues like outdated Apache modules, SSL weaknesses, or misconfigurations.

Output might show:

| PORT | State Service | Version |
|---|---|---|
| 22/TCP | Open SSH | Open SSH 8.2 |
| 80/TCP | Open HTTP | Apache HTTPD 2.4.41 |
| 443/TCP | Open SSL/HTTP | Apache HTTPD 2.4.41 (OPEN SSL) |
| 3306/TCP | Open MYSQL | MYSQL 5.7.31 |

# Strengthening Data Encryption in MISP




## Step 1: Secure Data in Transit (TLS/HTTPS)

**Goal:** Prevent unauthorized access or interception of data exchanged between users, MISP, and integrated systems.

**Actions:**

### 1. Enable HTTPS (TLS 1.2 or 1.3)

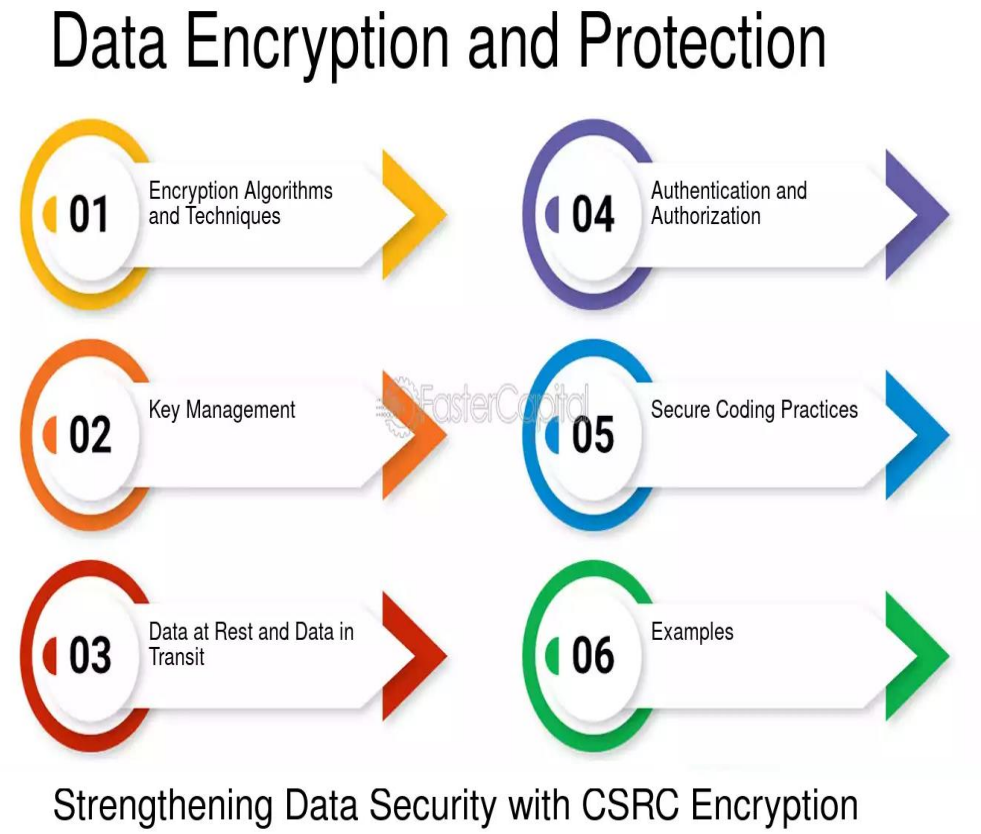
- Use a valid SSL/TLS certificate (e.g., Let's Encrypt or enterprise CA).
- Configure Apache/Nginx (depending on your MISP setup) for HTTPS:

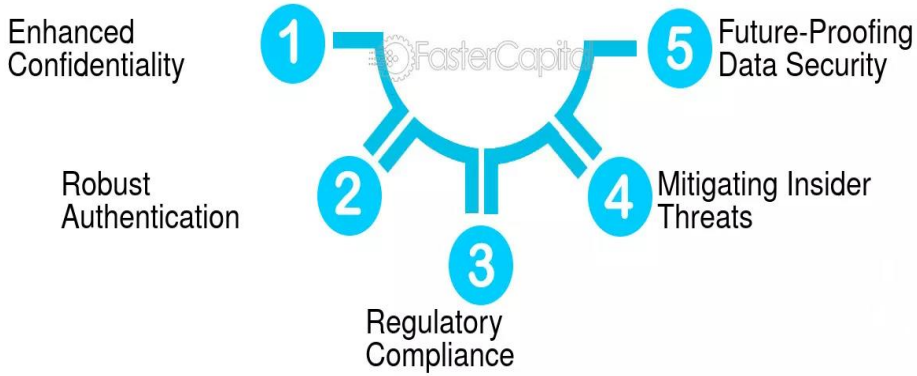
```
sudo a2enmod ssl
sudo a2ensite default-ssl
sudo systemctl restart apache2
```

### 2. Configure TLS securely

- Disable weak ciphers and SSL versions:

```
SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite HIGH:!aNULL:!MD5
```




### 3. Force HTTPS

- ❖ Enforce HTTPS redirect in web server config or via HTTP headers (HSTS).

### 4. Secure API Connections

- ❖ MISP's REST API should only be accessed via HTTPS.
- ❖ Validate certificates in all integrations (e.g., PyMISP, other MISP peers).

**Step 2: Encrypt Data at Rest**

**Goal:** Prevent access to stored data if the server is compromised.

**Actions:**

1. **Use Full Disk Encryption (FDE):**

   Enable LUKS (Linux Unified Key Setup) on MISP server volumes
   This encrypts the entire disk and requires passphrase/key at boot.

**Encrypt Database Storage:**

- Use **MySQL with InnoDB tablespace encryption**:

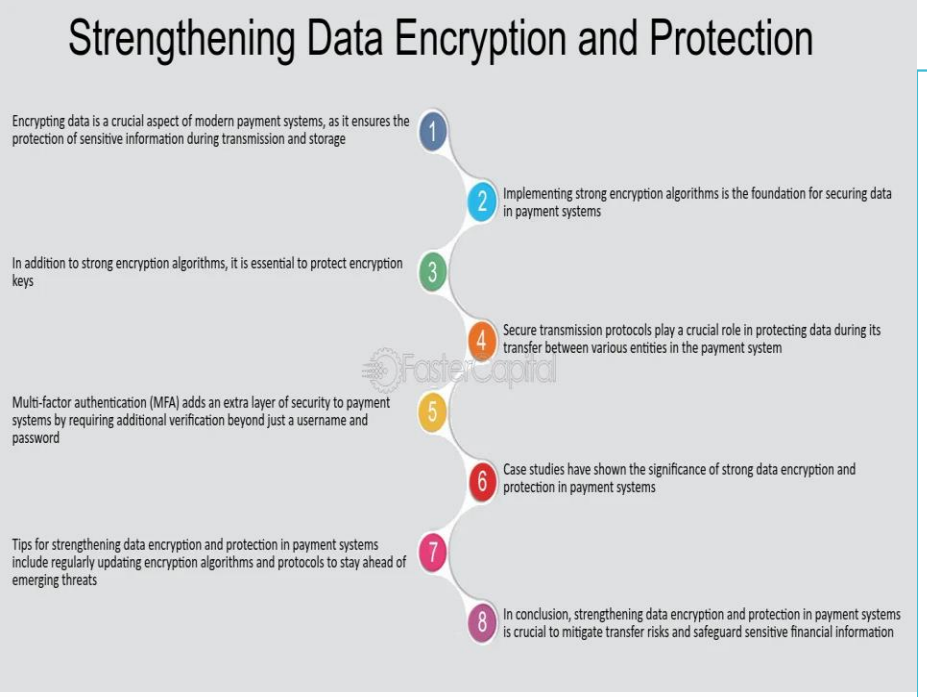
  ALTER INSTANCE ROTATE INNODB MASTER KEY;

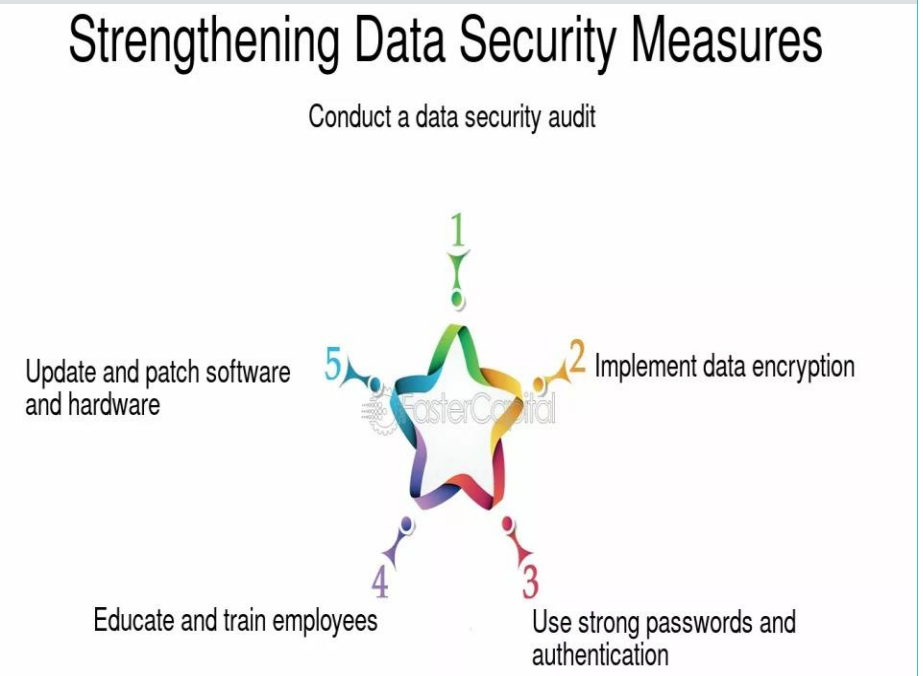
  Enable "innodb_encrypt-tables and innodb_encrypt_log".

Encrypt MISP Attachments (Optional):

Store attachments (malware samples, documents) using gpg or AES-256.

Protect file directories using file-level encryption like enCryptfs




## Backup Encryption:

- Use gpgor openssl to encrypt backups before storage or transfer..
- Automate with cron jobs:

```
mysqldump ... | openssl enc -aes-256-cbc -e -out backup.sql.enc
```

## Step 3: Implement Secure Key Management

**Goal:** Protect encryption keys from unauthorized access.

### Actions:

### 1. Use a Key Management System (KMS):

Options:

a) **HashiCorp Vault**

b) **AWS KMS / Azure Key Vault / GCP KMS** (if using cloud)

Store MySQL master keys, TLS certs, and backup keys securely.

### 2. Best Practices:

a) Rotate keys periodically (automate if possible).

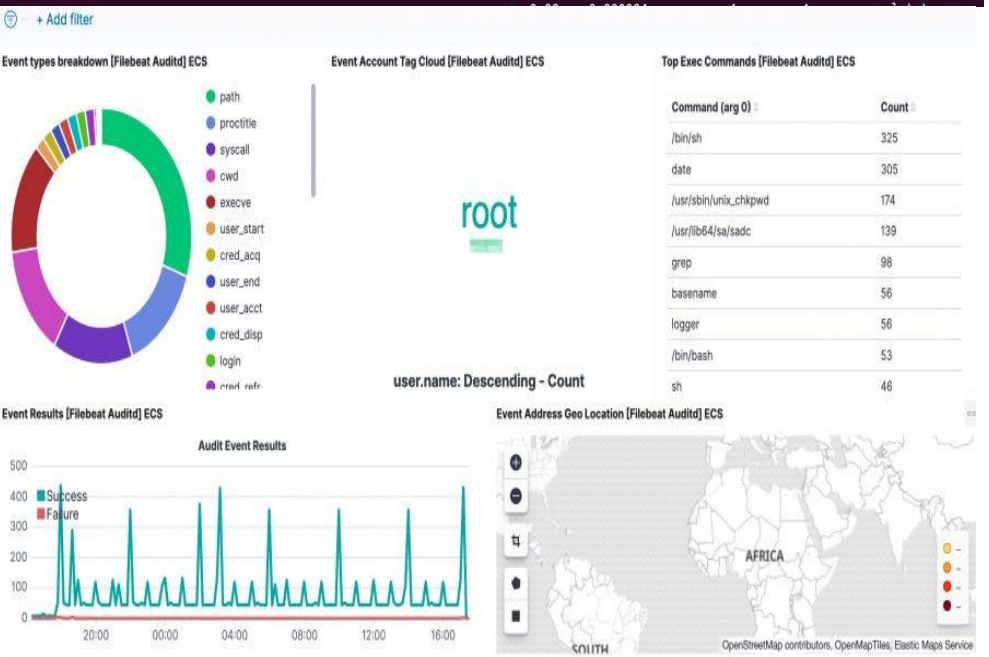
b) Use separate keys for different functions (e.g., database vs. file encryption

c) Restrict key access to only required services/users (principle of least privilege).

### 3. Integrate Vault with MISP (optional advanced):

a) Use Vault CLI/API to load secrets dynamically at runtime.

b) Example: Pull DB passphrase on system boot and mount encrypted volum


## Step 4: Monitor and Audit

**Goal:** Detect misconfigurations and unauthorized access attempts.

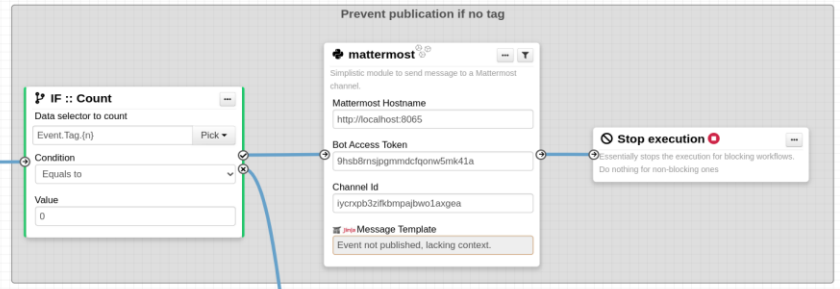
**Actions:**

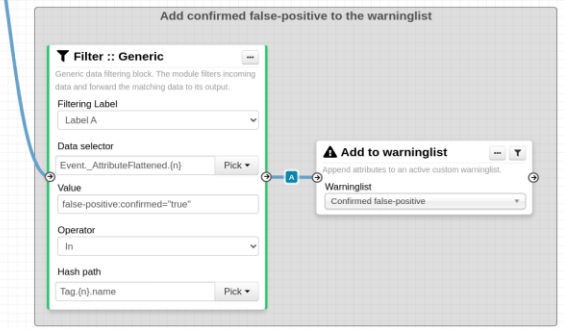
- Enable **logging** for:
  1. TLS handshakes
  2. Access to encrypted volumes
  3. Key usage and rotation
- Use tools like **Auditd**, **Filebeat**, or **OSSEC** for file and key access auditing.
- Regularly **review MISP logs** in/var/www/MISP/app/tmp/logs

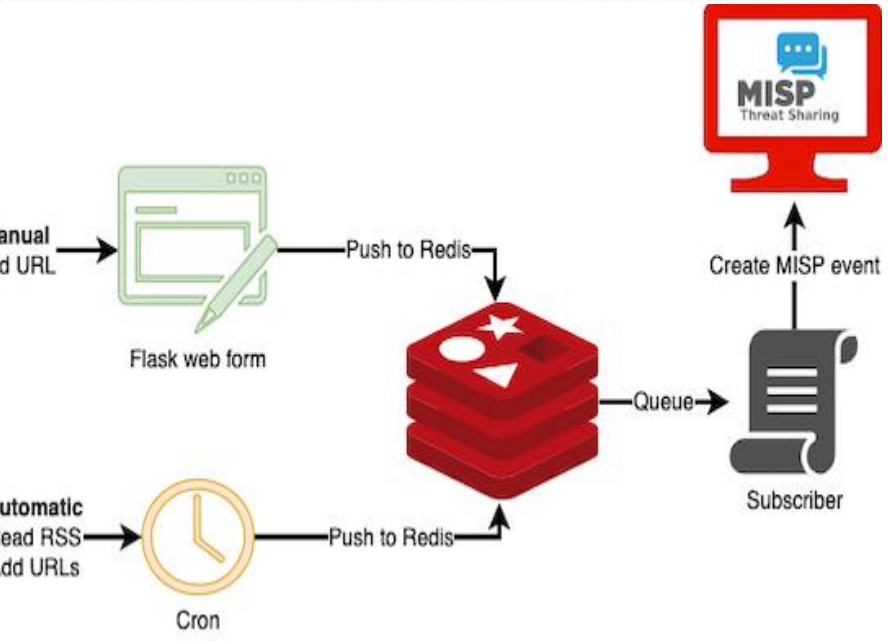
## Step 5: Documentation & Maintenance

1. Maintain a **data encryption policy** with:
   - Encryption algorithms used
   - Key lifecycle and access control rules
   - TLS certificate renewal schedule
2. Train admins on secure handling of encrypted assets and key material.

# Enhancing Authentication and Authorization in MISP




## Step 1: Implement Multi-Factor Authentication (MFA)

- **Objective:** Add a second layer of security beyond username/password

## Actions:

### 1. Enable TOTP-based MFA in MISP (Time-based One-Time Password)

MISP supports **Google Authenticator** or **any TOTP-compatible app**.

- Go to **Administration → List Users**
- Click on the target user → Enable **Two-Factor Authentication**
- Scan QR code in authenticator app (Google Authenticator, Authy, etc.)

### 2. Enforce MFA Platform-wide

- In config.php (/var/www/MTSP/app/config/config.php) set:

```
Configure::write('Security.require_mfa', true);
```



### 3. Log & Monitor MFA Events

- All login attempts (success/failure) are logged in:

/var/www/MISP/app/tmp/logs/*

### Step 2: Configure Role-Based Access Control (RBAC)

- **Objective:** Assign permissions based on user responsibilities.

### Actions:

1. **Define Roles in MISP**
2. Go to **Administration → Roles**
3. Predefined roles:
   I. **Org Admin**
   II. **Site Admin**
   III. **User**
4. You can also create custom roles (e.g., Read-Only Analyst, Data Contributor, Threat Analyst)

### 1. Assign Specific Permissions to Roles

For each role, set:

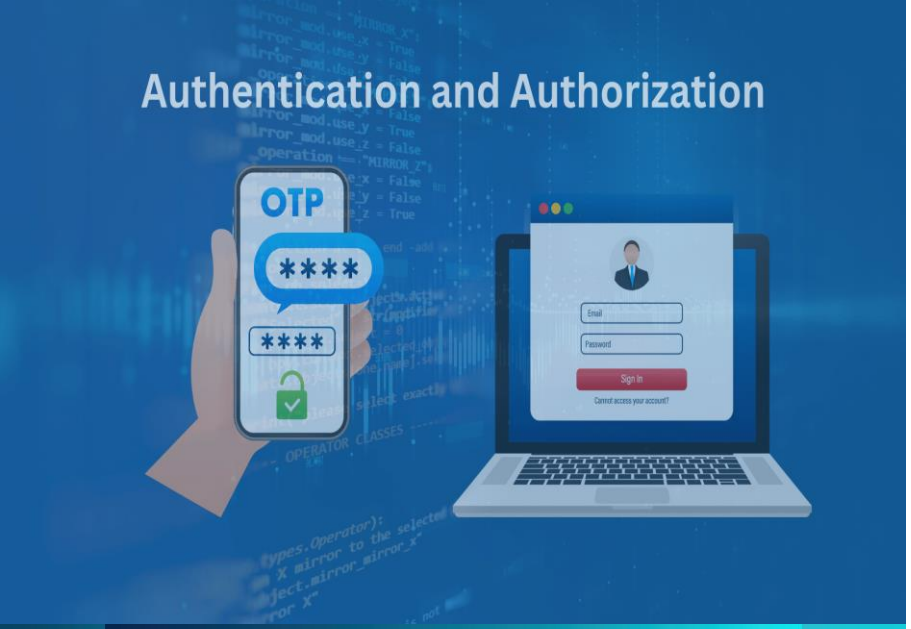
- ❑ Event publishing rights
- ❑ Tagging rights
- ❑ Access to specific object types
- ❑ Sync permissions with other MISP instances

### 2. Assign Users to Roles

- ❑ Go to **Administration → Users**
- ❑ Edit the user → Choose appropriate role from dropdown

### 3. Use Orgs for Isolation

Group users into **organizations** to segregate data and access across teams or partners.




## Step 3: Integrate with Single Sign-On (SSO)

**Objective:** Streamline secure user access through identity federation.

### MISP Supports SSO via:

- SAML 2.0
- LDAP/AD
- HTTP Header-based Auth (e.g., behind a proxy like Keycloak or Shibboleth)

### Actions (SAML / Keycloak Example):

1. **Use a Reverse Proxy (Apache/Nginx)**

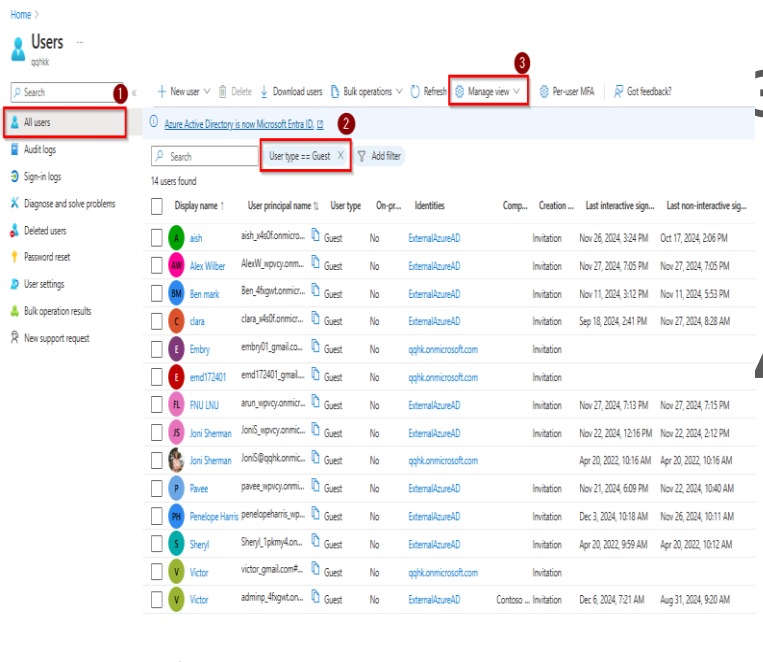
   Place MISP behind Keycloak, which handles SSO.

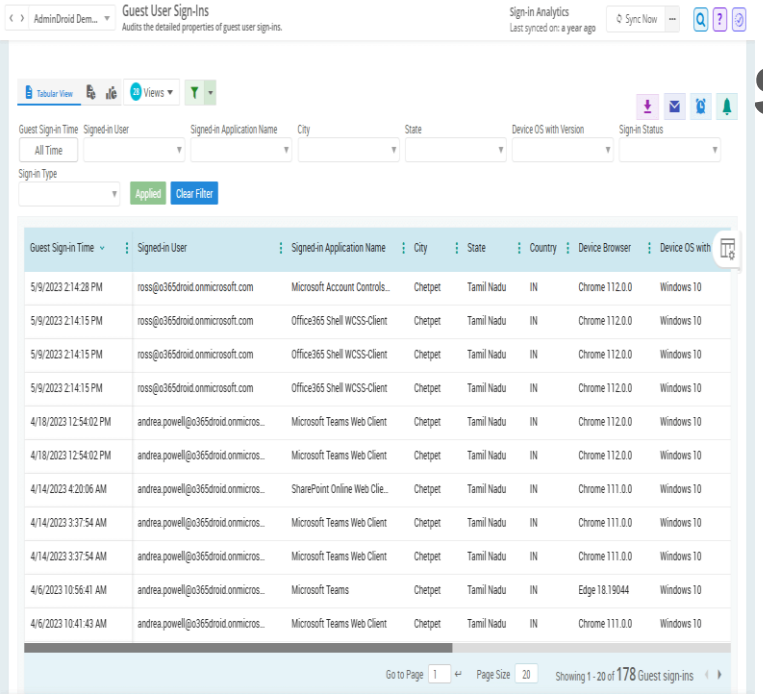
   Use mod_auth_opendc for Apache.

2. **Configure HTTP Header-based Login in MISP**

   In config.php, enable header-based login:

```
Configure::write('Security.auth_enforced', true);
Configure::write('Security.auth_user_header', 'HTTP_REMOTE_USER');
```




# 3. Set User Creation via Header

- You can auto-create users if they don't exist in MISP:

```
Configure::write('Security.advanced_authkeys', true);
```

# 4. SSO Providers Supported:

Keycloak (OpenID Connect)

Okta / Azure AD / ADFS (SAML 2.0)

LDAP via Apache's mod_ldap.

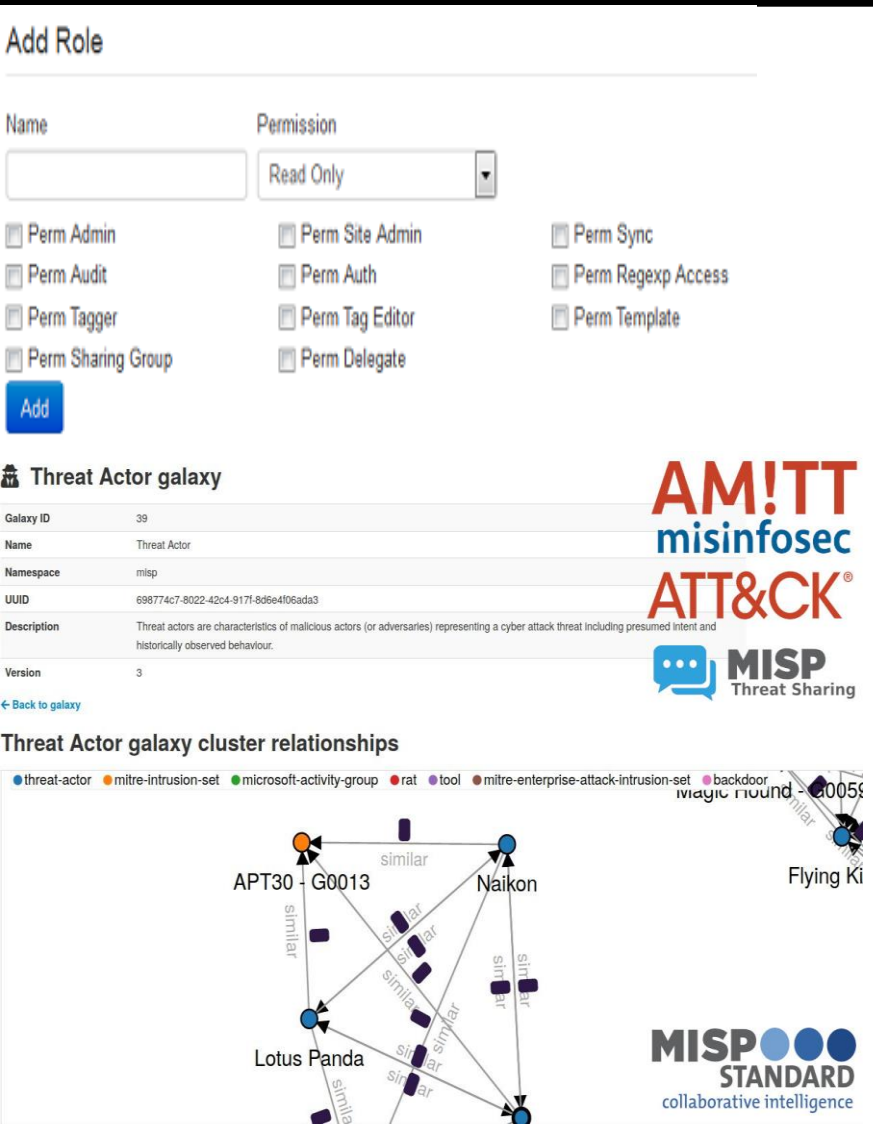
# Step 4: Review and Monitor Access

- Periodically review
- Role assignments
- MFA adoption logs
- SSO login audits
- Remove inactive users or expired sessions

Enable session expiration and timeouts:

```
Configure::write('Session.timeout', 30); // in minutes
```

# Improving Access Control for Threat Intelligence Data in MISP




## Step 1: Review & Update Access Control Lists (ACLs)

MISP uses **roles and organizations** to enforce access control.

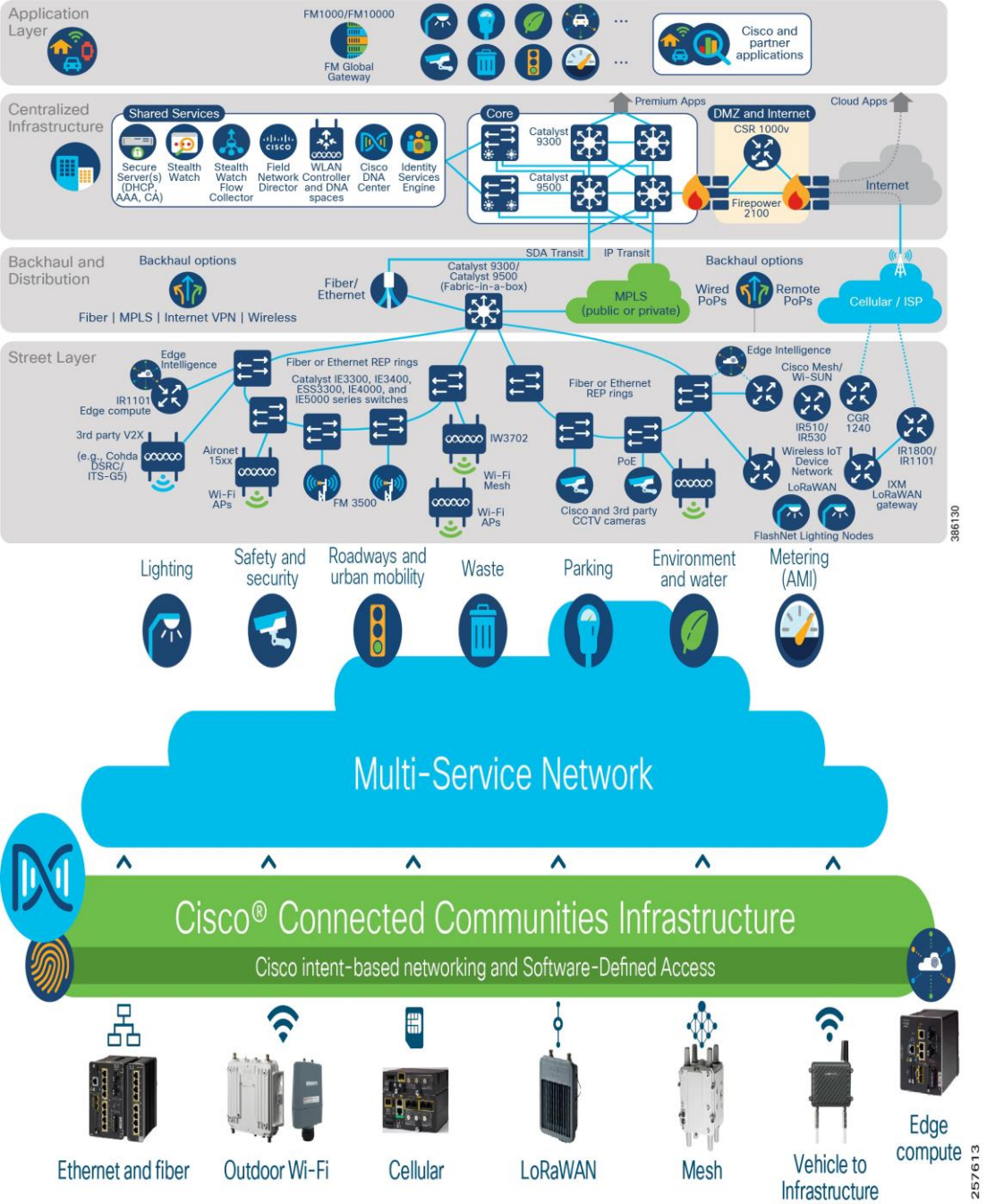
Users are grouped under organizations, and their access depends on:

- **Role permissions**
- **Event distribution settings**
- **Sharing groups**

## Actions:

### 1. Review User Roles

- Go to: **Administration → Roles**
- Check role settings for:
  - Perm_add
  - Perm_modify
  - Perm_publish
  - Perm_sync
  - Perm_site_admin




2. **Audit** Go to: **Administration → Users**
   - Review user role and org alignment
   - Disable or delete unused accounts
   - Existing Users
3. **Limit Event Access via Distribution Settings**

MISP event Distribution levels:

0 – Your Organization Only
1 – This Community Only
2 – Connected Communities
3 – All Communities
4 – Sharing Group

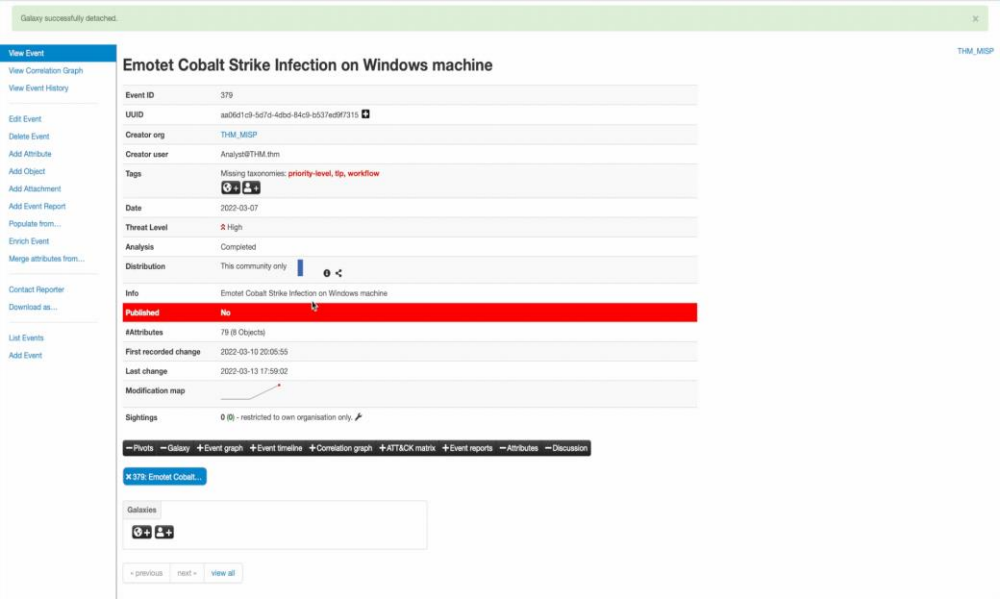
Use distribution = 0 or 4 (with custom Sharing groups) for restricted intelligence.

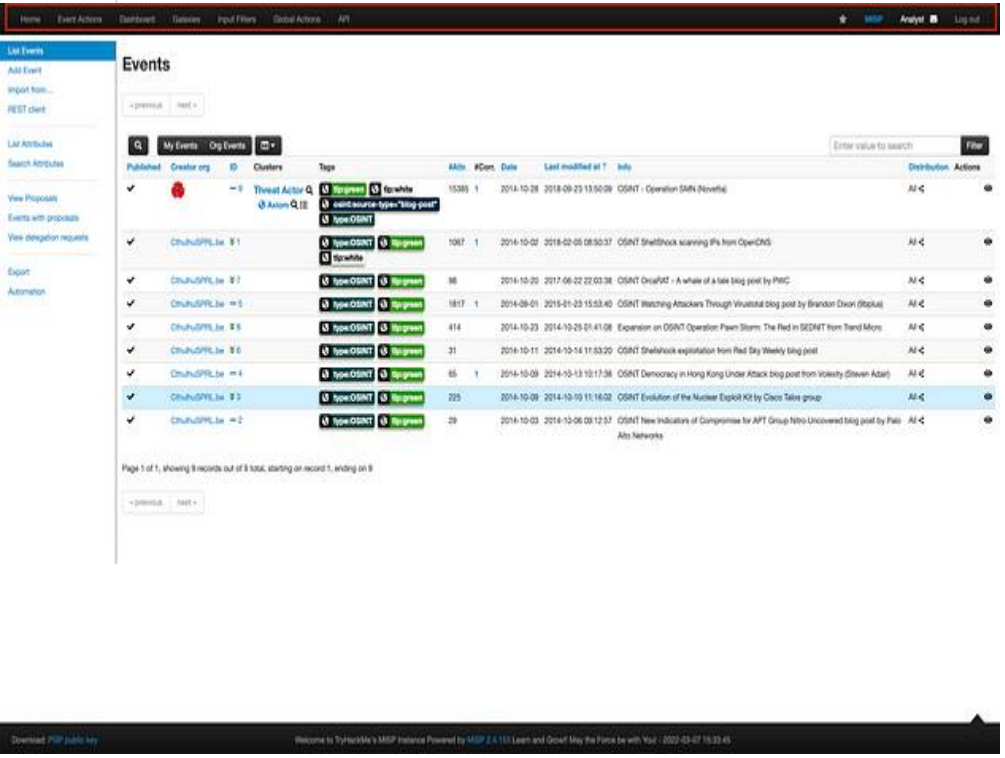
## Step 2: Implement Granular Read/Write Controls via Roles and Tags

MISP supports **fine-grained control** using **roles, tag filters,** and **object-level distribution**.

### Create Fine-Grained Roles

- Go to: **Administration → Roles → Add Role**
- Customize permissions:
  - Read-only (disable perm_add , perm_modify)
  - Analysts with limited object creation rights.
  - Admin with full event visibility.




- Use Tag-Based Access Filtering
- Tag events (e.g., tlp:red , classified , internal )
- Create roles or shaping groups that exclude/limit certain tags

Example:

Prevent users from expoting tlp:red data

## Leverage Object-Level Distribution

You can restrict individual attributes/objects in an event:

I. Set **object distribution** different from the parent event
II. Great for cases where one part of an event is more sensitive

## Step 3: Enforce Trusted Sharing Policies

MISP allows **verified sharing** through **organizations and Sharing Groups**.

## Actions:

### 1. Use Sharing Groups for Trust Control

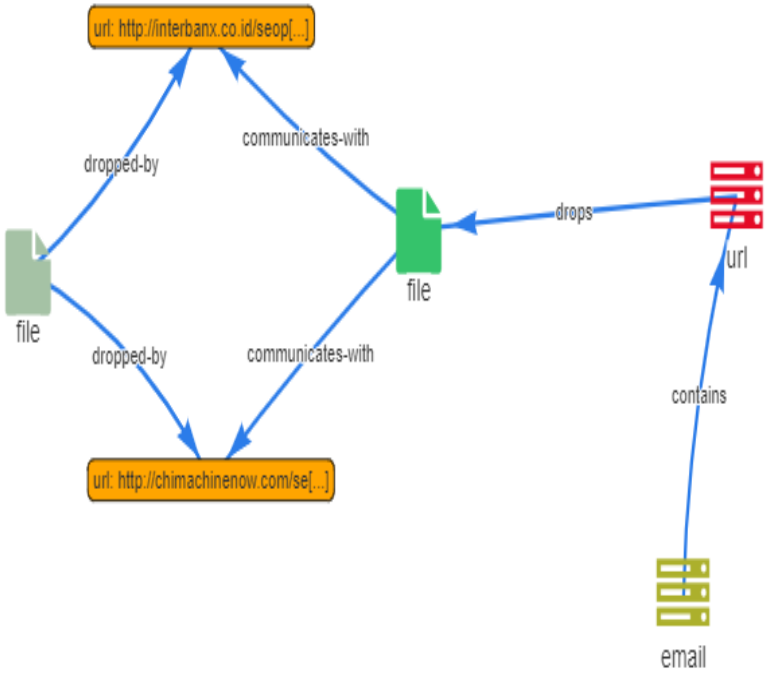
I. Go to: **Sync Actions → List Sharing Groups → Add Sharing Group**
II. Add only trusted orgs
III. Set required conditions:

Must be in the same trust level
Must accept your terms (e.g., ISAC agreements)

### 2. Restrict Attribute Distribution

Even within shared events, sensitive attributes can be scoped using:

- Disitribution – 0 (Org only)
- Distribution = 4 ith a selective Sharing Group




## 3. Configure Feed Access Control

If you're exposing MISP feeds externally:

- Use authentication tokens or IP whitelisting.
- Ensure feeds only show distribution <=2 unless controlled.

## Step 4: Monitoring & Audit

### Log Access Events

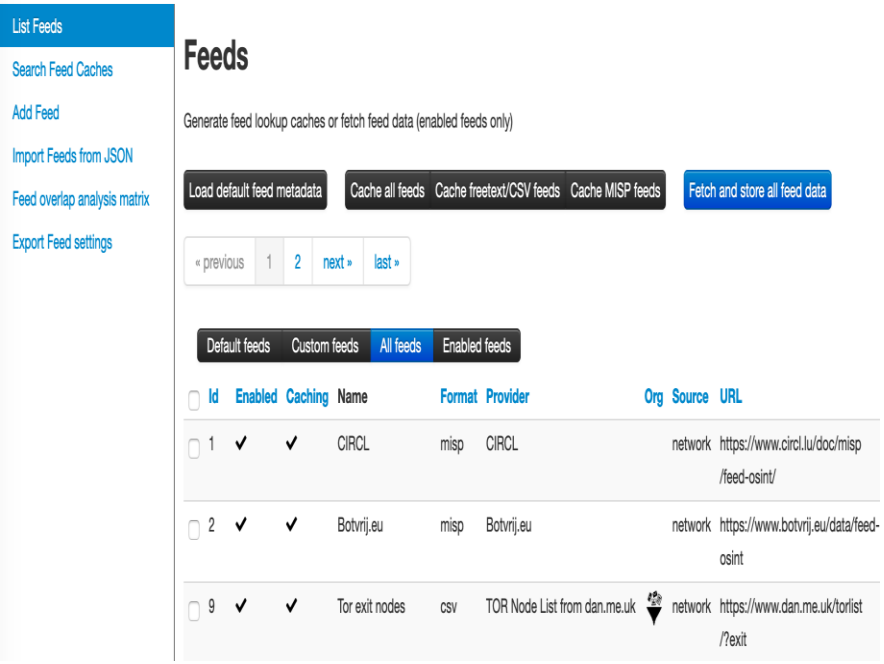
Check MISP logs: /var/ww/MISP/app/tmp/logs/.

Log types:

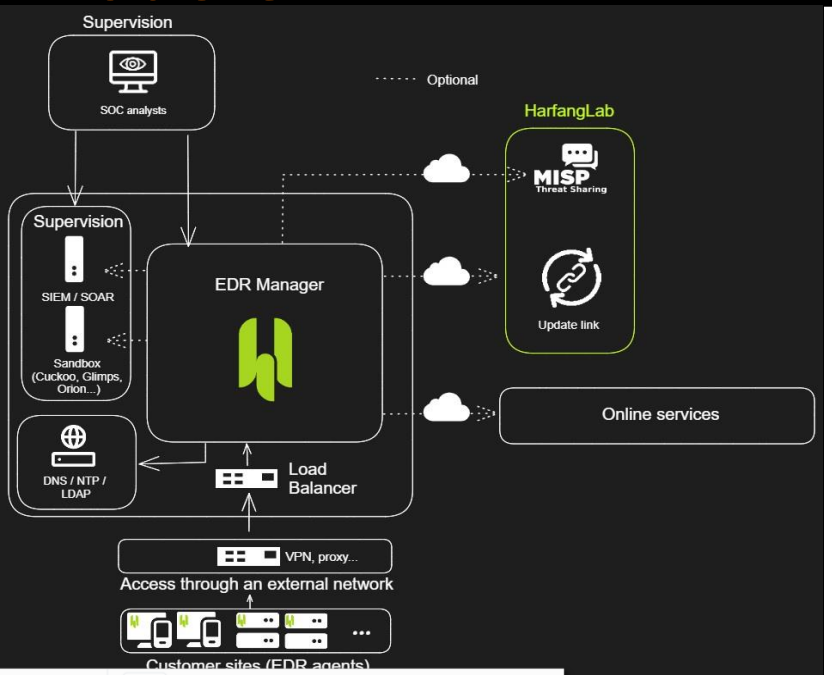
1. User access
2. Exploit requests
3. Failed logins
4. Distribution violation (if audit enabled)

Enable Auditing

- Periodically export user role and sharing group audits
- Review access to tlp:red or org-only tagges items.



# Practical Implementation Step: Create a MISP Export Filter Module

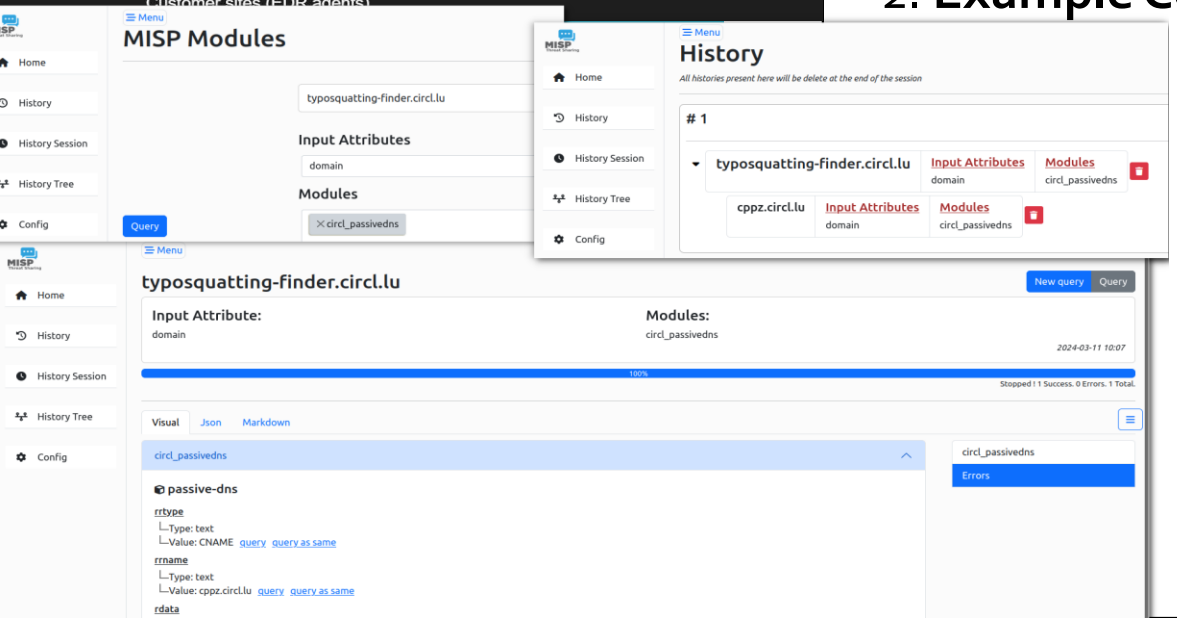


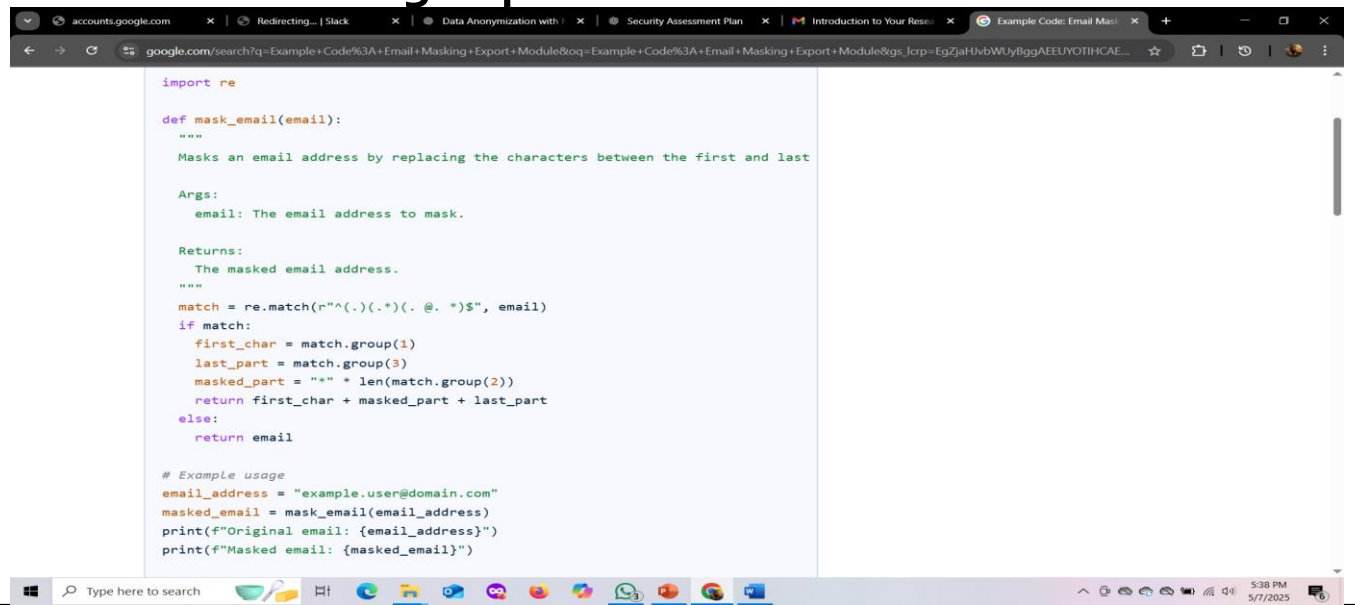

This example shows how to **mask email addresses in MISP attributes** before sharing.

**Step-by-Step: Create a Custom MISP Export Module**

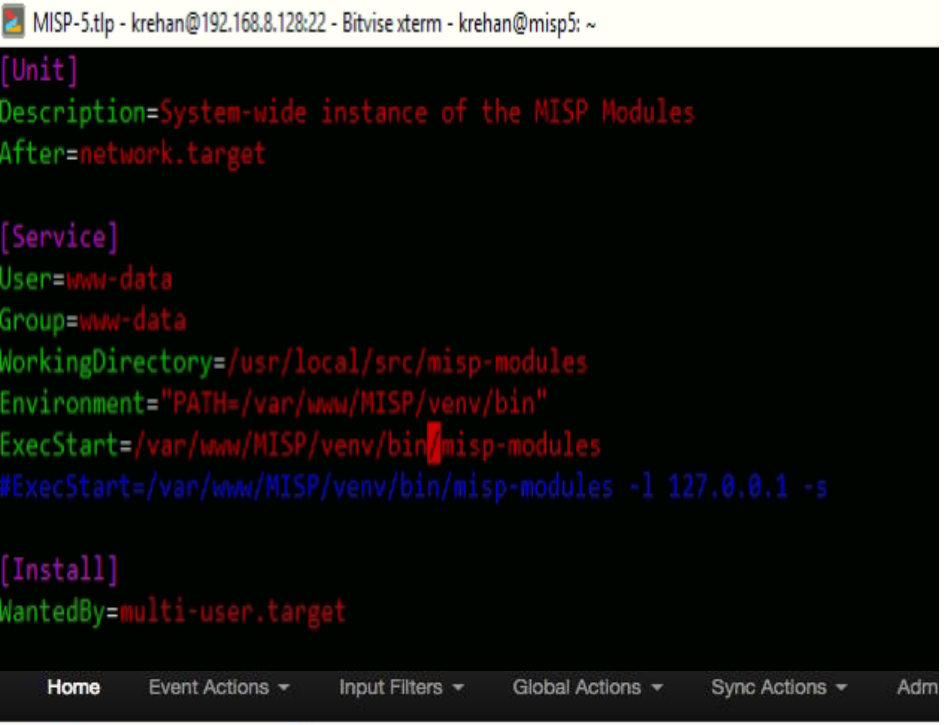
**1. Create a new Python module**

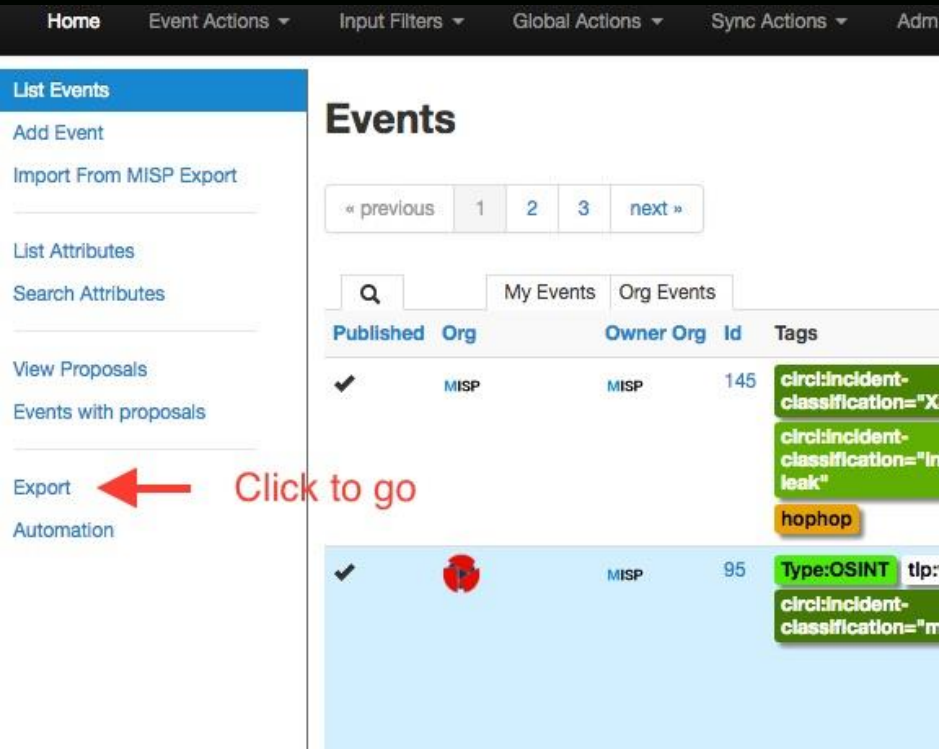
- Location: /var/www/MISP/mispmodules/misp_modules/export_mod/anon_export.py
- 2. **Example Code: Email Masking Export Module**



```python
import re

def mask_email(email):
  """
  Masks an email address by replacing the characters between the first and last

  Args:
    email: The email address to mask.

  Returns:
    The masked email address.
  """
  match = re.match(r"^(.)(.*)(. @. *)$", email)
  if match:
    first_char = match.group(1)
    last_part = match.group(3)
    masked_part = "*" * len(match.group(2))
    return first_char + masked_part + last_part
  else:
    return email

# Example usage
email_address = "example.user@domain.com"
masked_email = mask_email(email_address)
print(f"Original email: {email_address}")
print(f"Masked email: {masked_email}")
```




## 3. Activate the Module

```
cd /var/www/MISP/misp-modules
sudo systemctl restart misp-modules
```

## 4. Use the Export Module in MISP

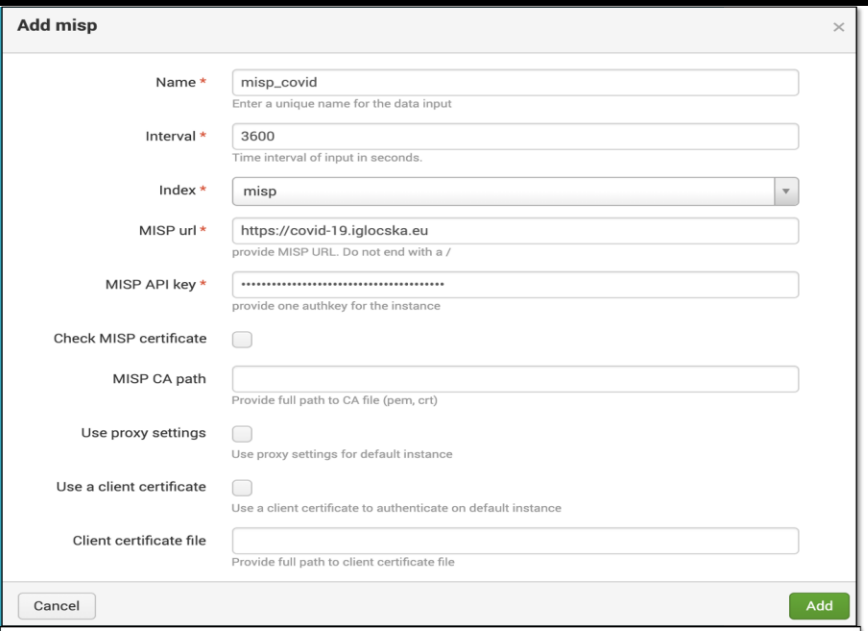
- Go to an Event
- Choose **Export → Custom Format**
- Select anon_export module.

## Final Notes

This module:

- Applies masking to emails before export.
- Can be extended to handle IPs, phone numbers, etc.
- Helps maintain GDPR/CCPA compliance by reducing risk of sharing PII.

# Practical Implementation Step: Connect MISP to Splunk for Real-Time Threat Detection




```
| mispgetioc misp_instance=misp_covid add_description=true category="External
  analysis,Financial fraud,Internal reference,Network activity,Other,Payload
  delivery,Payload installation,Payload type,Persistence mechanism,Person
  ,Social network,Support Tool,Targeting data" type="ip-dst" to_ids=true
  geteventtag=true warning_list=true limit=0 last=1d
| eval ip=misp_value
| eval description = tostring(misp_event_info)."|".tostring(misp_category)."|"
  .tostring(misp_comment)
| eval weight = 1
| table description,ip,weight
| outputlookup append=true misp_es_ip_intel
```

Here's how to **integrate MISP with Splunk** for live threat feed ingestion:

## 🛠️ Step-by-Step Integration: MISP to Splunk

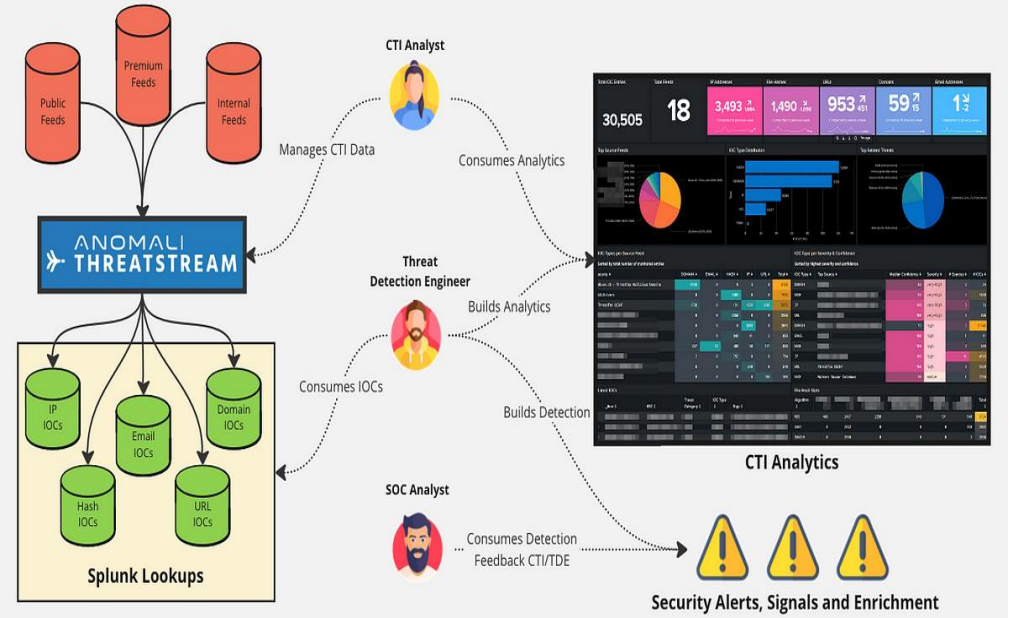
### 1. Install MISP42Splunk Add-on

- From: https://github.com/MISP/misp42splunk

```
git clone https://github.com/MISP/misp42splunk.git
cp -r misp42splunk /opt/splunk/etc/apps/
```

### 2. Configure the MISP Add-on in Splunk

- Go to the Splunk web UI
- Navigate to **Settings > Data Inputs > MISP Events**
- Add your MISP API URL, API key, and fetch schedule
- Choose data filters (e.g., by tag, org, time window)





## 3. Enable Scheduled Fetching of MISP Events

- Configure the fetch interval (e.g., every 15 mins)
- Choose output index (e.g., threatintel )

## 4. Search MISP Data in Splunk

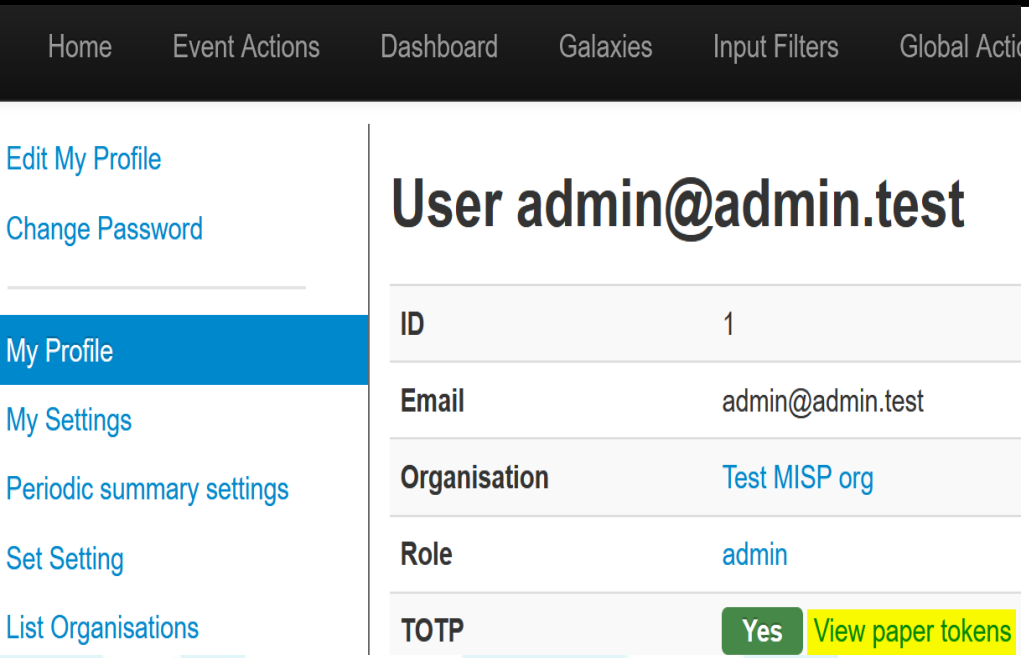
```
index=threatintel sourcetype=misp:event
```

You can now correlate these with logs for **real-time alerting and detection.**

## Outcome

- Real-time threat data from MISP available in SIEM.
- Uses **industry protocol (API)** securely.
- Can be extended to TAXII/STIX consumption later.

# Practical Implementation Step: Enable and Export MISP Logs for Auditing





## 🛠 Step-by-Step: Export MISP Logs to Syslog or ELK

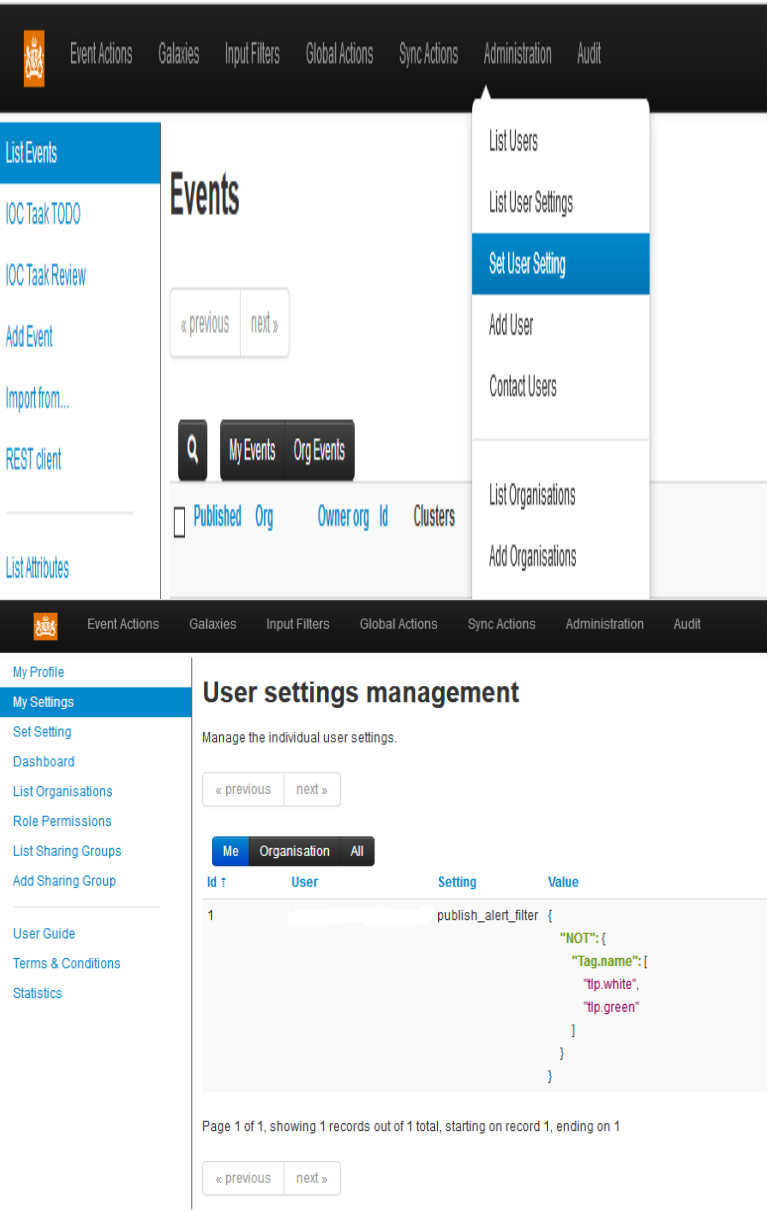
### 1. Enable Syslog Forwarding of MISP Logs

- Edit MISP's configuration file:

/var/www/MISP/app/Config/config.php

```
'Syslog' => [
        'enabled' => true,
        'facility' => LOG_LOCAL0,
        'level' => LOG_INFO
    ],
```

This enables MISP to log actions to the system's syslog.




## 2. Configure Rsyslog to Forward Logs to a SIEM

- Edit /etc/rsyslog.conf or create a custom file in /etc/rsyslog.d/

```
# Forward MISP logs to a remote server
if $programname == 'misp' then @your.siem.ip.address:514
```

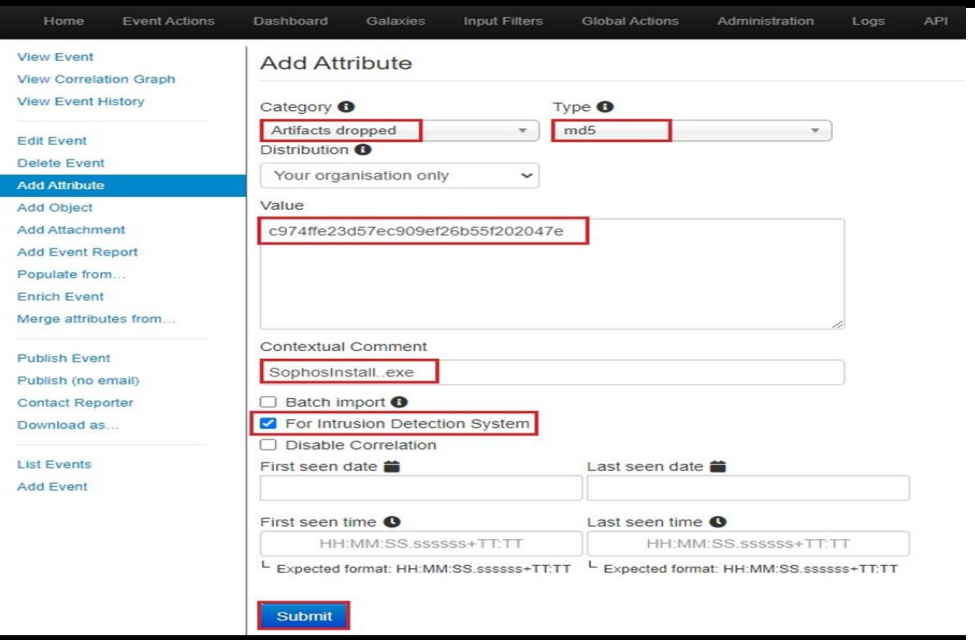
- Restart rsyslog:

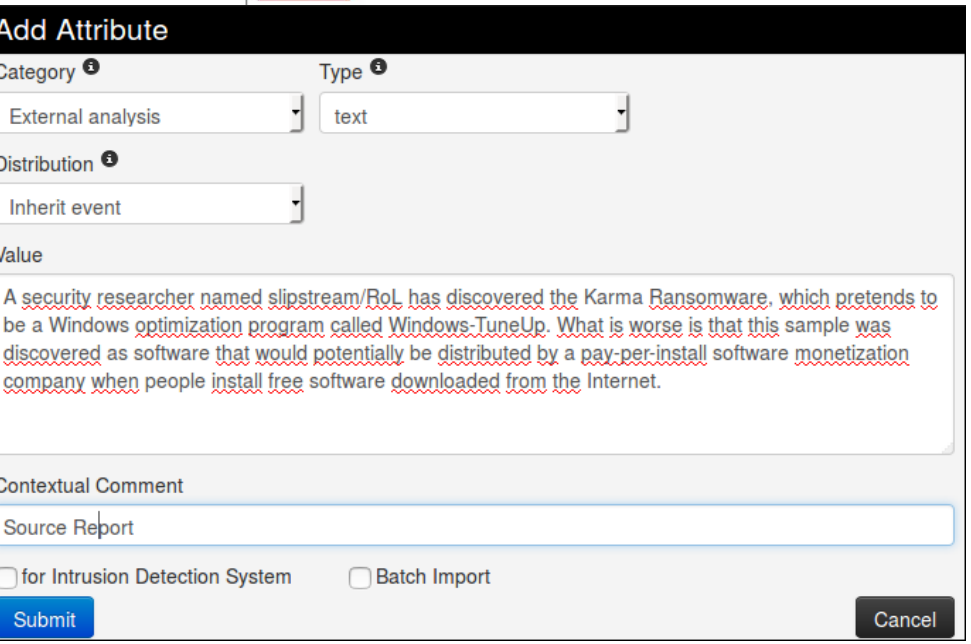
```
sudo systemctl restart rsyslog
```

## 3. Set Up a Dashboard in ELK or Splunk

- Parse logs by action , User , and IP.
- Create alerts for:
  - ❖ High number of edits/deletes in a short time
  - ❖ API key misuse
  - ❖ Failed login spikes

# Practical Implementation Step: Create an Incident Report Template in MISP




## 🛠️ Step-by-Step: Use MISP to Log and Track a Security Incident

### 1. Create a New Incident Event in MISP

Go to MISP Web UI ⟶ Event Actions ⟶ Add Event.

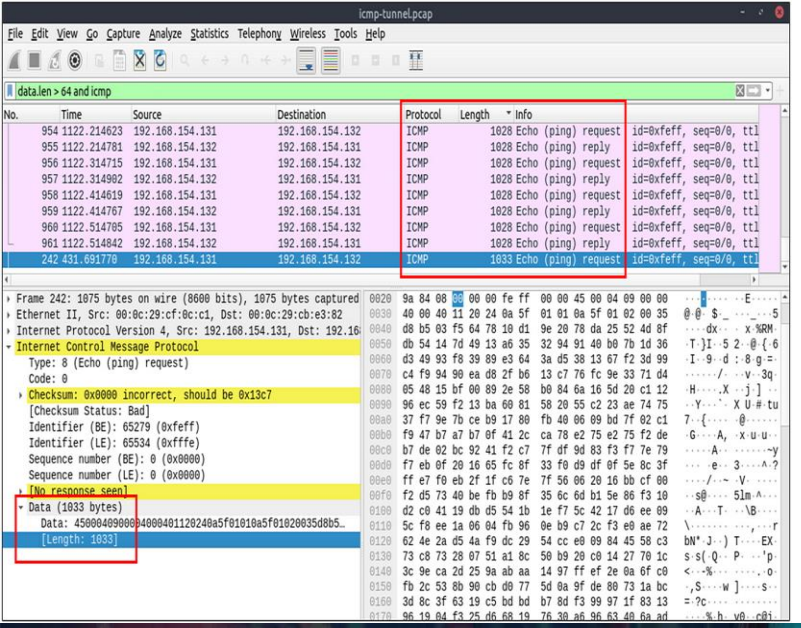
Fill in:

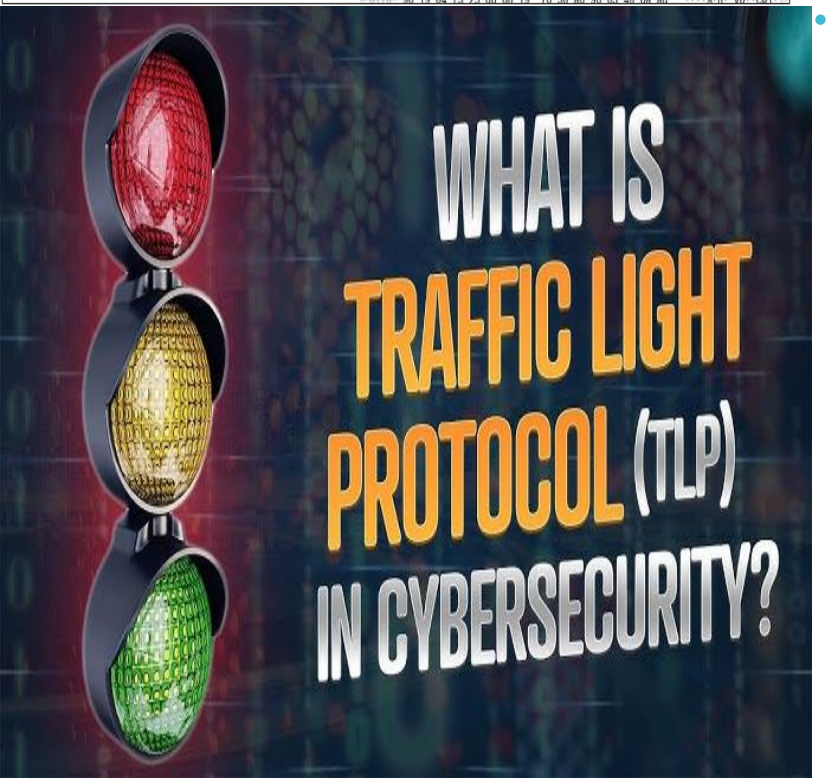
- **Info**: Security Incident – Unauthorized Access.
- **Threat Level**: High.
- **Analysis**: Initial.
- **Distribution**: Your Organization Only.

### 2. Add Attributes to the Incident

Examples:

- ip-src : 192.168.1.101.
- Comment : Suspicious login attempt from internal subnet.
- Timestamp : (use current time).
- user-agent : if known from logs.




## 3. Tag and Correlate

Apply relevant tags:

tlp:red , incident-type:unauthorized-access , actor:unknown , response:under-investigat

## 4. Export or Report the Incident

Use **Export → PDF or STIX** to generate a report for compliance or stakeholders.

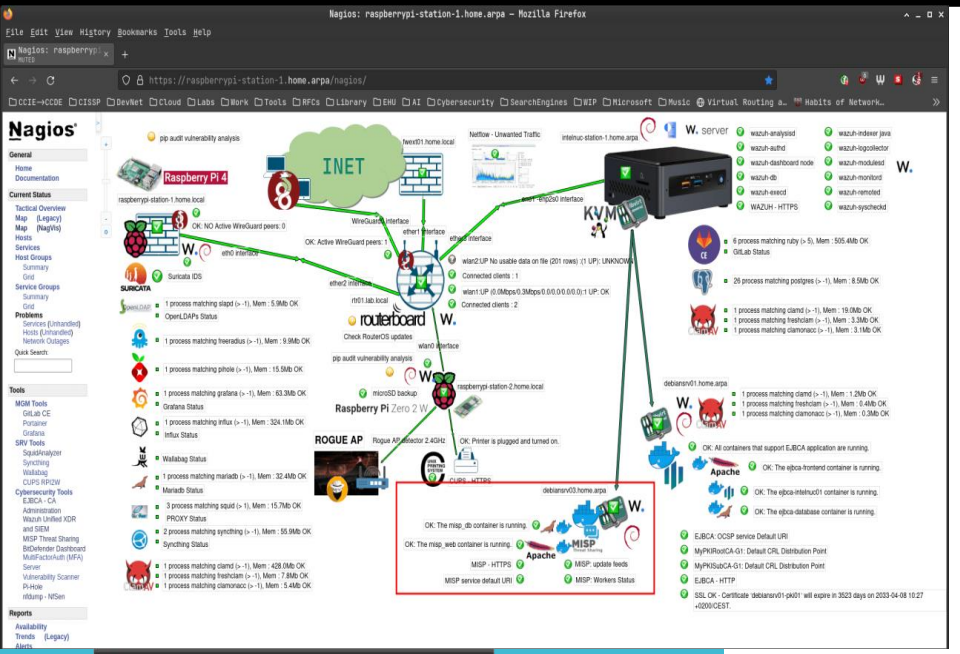
- You can also automate this via MISP's API:

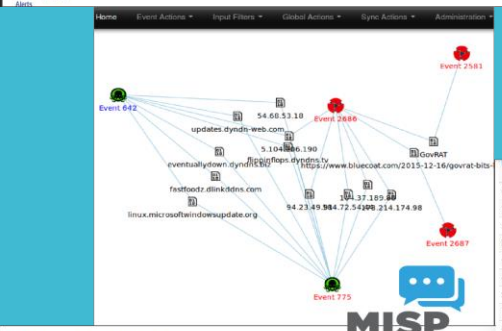
```
curl -H "Authorization: <API_KEY>" \
  -H "Accept: application/json" \
  -X GET https://<your_misp_url>/events/view/<event_id>
```

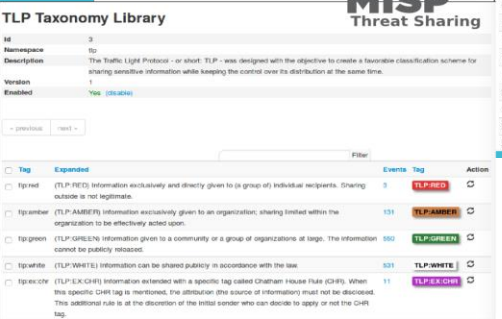
Result

- Log incidents.
- Classify and track response actions.
- Generate and export reports for audits or external communication.

# Practical Implementation Step: Create a Security Configuration Guide for MISP Deployment




- Example: misp-security-configuration.md

# MISP Security Configuration Guide

## 1. Web Server Configuration (Apache)

- Enforce HTTPS:

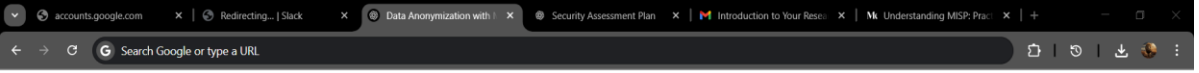
❖ SSLEngine on

❖ SSLCertificateFile /etc/ssl/certs/misp.crt

❖ SSLCertificateKeyFile /etc/ssl/private/misp.key.

- Add HTTP security headers:

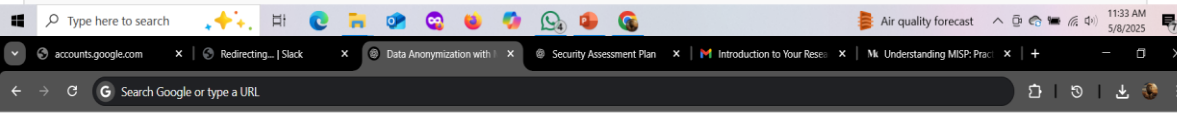
- Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains".
- Header always set Content-Security-Policy "default-src 'self'".



markdown Copy Edit

```
# MISP Secure Deployment Guide

## 1. Authentication and Access Control
- Disable default admin credentials after setup.
- Enforce strong password policies (min. 12 chars).
- Integrate with LDAP or SAML for SSO and MFA.
- Use MISP role-based permissions to assign least privilege.

## 2. TLS/HTTPS
- Install a valid TLS certificate (e.g., Let's Encrypt).
- Enforce HTTPS in Apache configuration:
```

Redirect permanent / https://your-misp-domain/

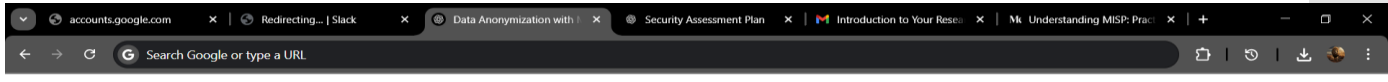


pgsql Copy Edit

```
## 3. Database Security
- Use `mysql_secure_installation` to harden MySQL.
- Limit access to MySQL from localhost only.
- Enable InnoDB encryption if required (for data at rest).

## 4. Logging and Monitoring
- Enable syslog integration to forward logs to a SIEM.
- Review logs weekly for unauthorized access attempts.

## 5. File and Permission Hardening
- Restrict file permissions on `/var/www/MISP`:
```

chown -R www-data:www-data /var/www/MISP

chmod -R 750 /var/www/MISP



yaml Copy Edit

```
## 6. Backup and Disaster Recovery
- Implement encrypted backups.
- Store backups securely off-site or in a secure cloud.

---

## Appendix: Reference Checklists
- [x] GDPR Article 32 (Security of processing)
- [x] ISO 27001: A.9, A.12, A.13 compliance areas
```