MISP Dashboard

Real-time overview of threat intelligence from MISP instances

Team CIRCL

info@circl.lu

December 5, 2019
MISP includes a flexible publish-subscribe model to allow real-time integration of the MISP activities:

- Event publication
- Attribute creation or removal
- Sighting
- User login

→ Operates at global level in MISP
MISP ZeroMQ functionality can be used for various models of integration or to extend MISP functionalities:

- Real-time search of indicators into a SIEM\(^1\)
- Dashboard activities
- Logging mechanisms
- Continuous indexing
- Custom software or scripting

\(^1\)Security Information & Event Management
MISP-Dashboard: An introduction
MISP-DASHBOARD - REALTIME ACTIVITIES AND THREAT INTELLIGENCE
MISP-DASHBOARD - FEATURES

- Subscribe to multiple **ZMQ** MISP instances
- Provides historical geolocalised information
- Present an experimental **Gamification of the platform**
- Shows when and how MISP is used
- Provides real time information showing current threats and activity
MISP-Dashboard: Architecture and development
1. Be sure to have a running redis server: e.g.
   ▶ redis-server -p 6250
2. Update your configuration in config.cfg
3. Activate your virtualenv:
   ▶ . ./DASHENV/bin/activate
4. Listen to the MISP feed by starting the zmq_subscriber:
   ▶ ./zmq_subscriber.py
5. Start the dispatcher to process received messages:
   ▶ ./zmq_dispatcher.py
6. Start the Flask server:
   ▶ ./server.py
7. Access the interface at http://localhost:8001/
MISP-Dashboard architecture
Writing your handler

```python
# Register your handler
dico_action = {
    "misp_json": handler_dispatcher,
    "misp_json_event": handler_event,
    "misp_json_self": handler_keepalive,
    "misp_json_attribute": handler_attribute,
    "misp_json_object": handler_object,
    "misp_json_sighting": YOUR_CUSTOM_SIGHTINGS_HANDLER,
    "misp_json_organisation": handler_log,
    "misp_json_user": handler_user,
    "misp_json_conversation": handler_conversation,
    "misp_json_object_reference": handler_log,
}
```
# Implement your handler

def handler_user(zmq_name, jsondata):
    # json action performed by the user
    action = jsondata["action"]
    # user json data
    json_user = jsondata["User’]
    # organisation json data
    json_org = jsondata["Organisation’]
    # organisation name
    org = json_org["name’]
    # only consider user login
    if action == ‘login’:
        timestamp = time.time()
        # users_helper is a class to interact with the DB
        users_helper.add_user_login(timestamp, org)
Recent changes in the MISP-dashBoard

- MISP authentication can now be used in the misp-dashboard
- Improved TLS/SSL support in the default misp-dashboard
- Self-test tool to debug and test ZMQ connectivity
Future development

Optimizing contribution scoring and model to encourage sharing and contributions enrichment

Increasing geolocation coverage

Global filtering capabilities
- Geolocation: Showing wanted attribute or only on specific region
- Trendings: Showing only specified taxonomies

Tighter integration with MISP
- Present in MISP by default
- ACL enabled version
Conclusion

MISP-Dashboard can provide realtime information to support security teams, CSIRTs or SOC showing current threats and activity by providing:

- Historical geolocalised information
- Geospatial information from specific regions
- The most active events, categories, tags, attributes, ...

It also proposes a prototype of gamification of the platform providing incentive to share and contribute to the community.