# Artificial Intelligence Threat Reporting and Incident Response System

## D.5.3 IRIS lab pods for CERTs/CSIRTs

| | |
|---|---|
| **Project Title:** | Artificial Intelligence Threat Reporting and Incident Response System |
| **Project Acronym:** | IRIS |
| **Deliverable Identifier:** | Document number |
| **Deliverable Due Date:** | 31/10/2023 |
| **Deliverable Submission Date:** | 15/11/2023 |
| **Deliverable Version:** | 1.0 |
| **Main author(s) and Organisation:** | Bruno Vidalenc (THALES), Lorens Barraud (THALES) |
| **Work Package:** | WP5: Virtual Cyber Range and Training Environment |
| **Task:** | Task 5.3: IRIS lab pods for CERTs/CSIRTs |
| **Dissemination Level:** | CO: Confidential, only for members of the Consortium (including the Commission Services) |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 101021727. Content reflects only the authors' view and European Commission is not responsible for any use that may be made of the information it contains.

## Quality Control

| | Name | Organisation | Date |
|---|---|---|---|
| Editor | Bruno Vidalenc, Lorens Barraud | THALES | 14/11/2023 |
| Peer Review 1 | Marius Preda, Mihail Guranda | DNSC | 15/11/2023 |
| Peer Review 2 | Andrew Roberts | TALTECH | 15/11/2023 |
| Submitted by (Project Coordinator) | Gonçalo Cadete | INOV | 15/11/2023 |

## Contributors

| Organisation |
|---|
| THALES |
| CLS |
| CERTH |
| ICCS |
| KEMEA |
| SID |
| ATOS |
| TUD |
| INOV |
| ÎNTRA |

## Document History

| Version | Date | Modification | Partner |
|---|---|---|---|
| V0.1 | 14/11/2023 | Full draft | THALES |
| V0.2 | 15/11/2023 | Peer Review | DNSC |
| V0.3 | 15/11/2023 | Peer Review | TALTECH |
| V1.0 | 15/11/2023 | Final editing | INOV |

## Legal Disclaimer

IRIS is an EU project funded by the Horizon 2020 research and innovation programme under grant agreement No 101021727. 

# CONTENTS

# List of Figures



# List of Tables

# List of Abbreviations and Acronyms

| Abbreviation/ Acronym | Meaning |
|---|---|
| VCR | Virtual Cyber Range |
| RAN | Radio Access Network |
| VM | Virtual Machine |
| ATA | Automated Threat Analytics |
| CTI | Collaborative Threat Intelligence |
| EME | Enhanced MeliCERTes Ecosystem |
| CERT | Computer Emergency Response Team |
| CSIRT | Computer Security Incident Response Team |
| ATIO | Advanced Threat Intelligence Orchestrator |

# Executive Summary

This report reflects one part of task 5.2, to implement the training scenario of the IRIS project. The creation of the asset requires to re-create an IT infrastructure in the cyber range to train the IRIS end-users. This includes, amongst others, CERT/CSIRT team members, cybersecurity professionals, cybersecurity security practitioners, security services providers, as well as decision makers, operators of AI services and infrastructures. The other part of task 5.2 concerns the IRIS platform implementation in the cyber range and is reported in this deliverable. The scope of this deliverable is to present the technical details of the assets and the cyber-threat that has been chosen for the training scenario described in D5.1. The assets will create the virtual infrastructure used to emulate cybersecurity attacks and demonstrate how the end-user can utilize the IRIS functionalities to mitigate and/or prevent them. In this context, this report is divided in the description of the assets of three distinct training scenarios that were designed to illustrate the IRIS functionalities, the first scenario describes the assets for an attack that is conducted in a 5G mobile network infrastructure, targeted at taking control of the core network. The end-user is expected to understand the attack sequence, the actions that are performed by the IRIS solution in the background and interact with the platform when required. The second scenario involves a ransomware attack in a company that manages data from an IoT infrastructure, targeting to also align with the Pilot Use Case 1 (PUC1) of the IRIS project (IoT system in tram station in Barcelona). The attack is rolled-out and the end-user can familiarize with the network topology, understand the background processes that are conducted in the IRIS platform, as well as interact with the platform through decision making procedure to contain or mitigate the attack. Finally, the third scenario relates to PUC3 of the IRIS project, describing a collaborative exercise tailored to cross-border threat intelligence scenarios.

# 1 INTRODUCTION

## 1.1 Project Introduction

The IRIS project aims at tackling recent and upcoming cybersecurity challenges that emerge in IoT and AI-enabled networks and platforms. For this reason, the IRIS activity integrates several innovative technical solutions into an easy-to-use single platform to assist CERTs/CSIRTs for detecting, evaluating, responding, and communicating information regarding threats & vulnerabilities of IoT and AI-driven ICT systems.

The functionalities of the IRIS platform will be demonstrated in 3 pilots with the engagement of 3 smart cities (in Helsinki, Tallinn and Barcelona) along with the involvement of national CERTs/CSIRTs, and cybersecurity authorities.

The project duration extends from September 2021 to August 2024.

## 1.2 Deliverable Purpose

This deliverable aims to describe the different modules of the IRIS project and their status concerning their deployment on the INTRA platform and the cyber range. The cyber range must be able to recreate an infrastructure similar to the real infrastructure of the project end-users.

## 1.3 Structure of the deliverables

Section 2 describes the IRIS scenario landscape. Section 3 starts by describing the platform on which the training scenarios will be implemented and the assets and the technical details of the cyber threat used in the three scenarios that were developed for training purposes. Section 4 will detail the different pods of the IRIS platform. Finally, Section 5 will present the different modules of the IRIS platform and their deployment status.

# 2 IRIS TRAINING LANDSCAPE

## 2.1 IRIS Cyber Range

The Cyber Range tool is based on the Hynesim Cyber Range edited by Diateam (https://www.diateam.net/what-is-a-cyber-range/ ). It provides virtualized resources, such as network L2/L3 equipment's, and fully customizable virtual machines. These resources can be linked together in architectures called "topologies" to simulate complex network environments. Furthermore, external equipment can be connected to these topologies, allowing the use of specific hardware or systems. Memory dump and network capture features provide introspection on the environment.

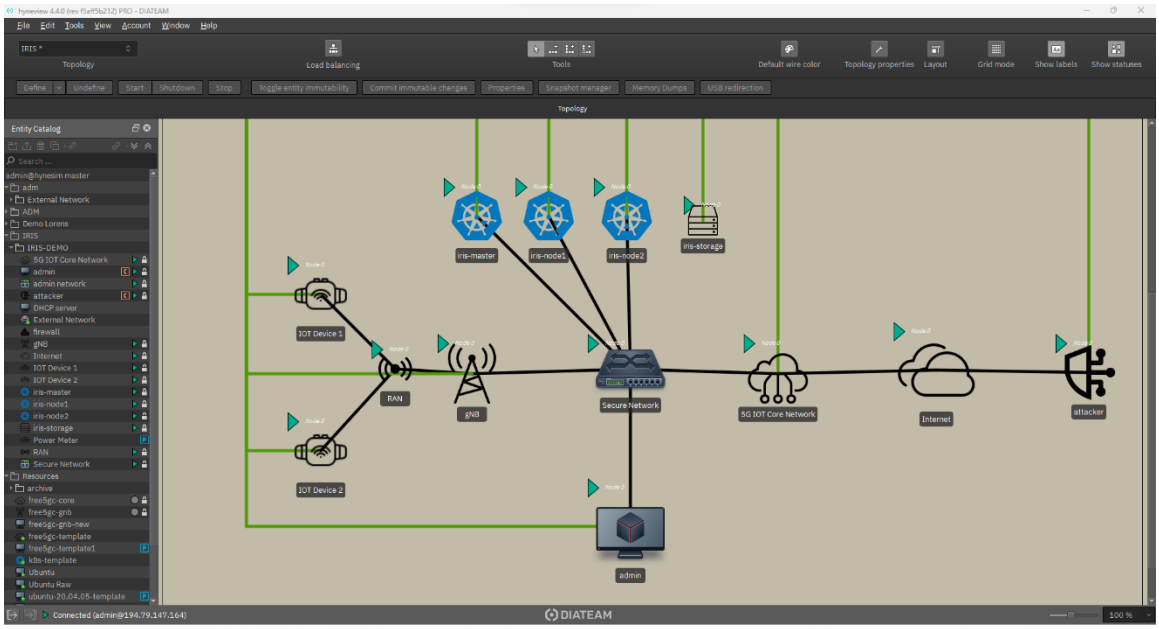


*Figure 1: Cyber Range tool*

Fine-grained access control can be configured on the different entities, allowing direct console access to the virtual machines, and asymmetric blue team/read team training scenarios.

The Hynesim Cyber Range is a distributed application. The master server hynesim-master manages the communications between the hynesim-nodes used to virtualise and emulate the entities of a topology. User access is provided by a software client called hyneview. Communication between the hyneview clients and the hynesim-master goes through a relay named hynesim-glacier (Figure 2).

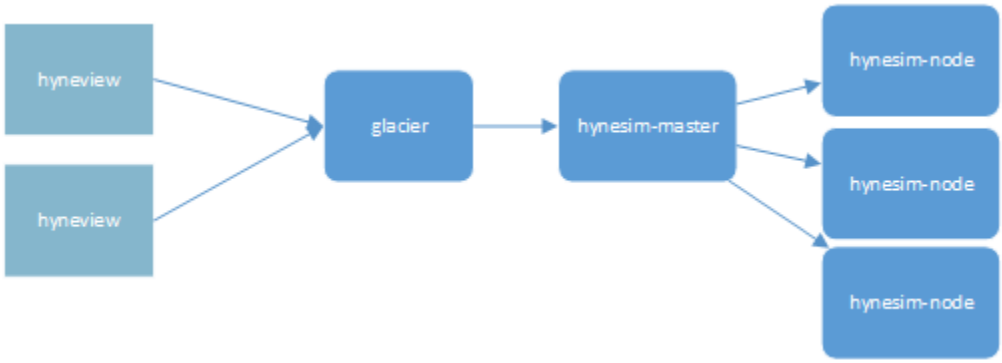


*Figure 2: Cyber Range architecture*

## 2.2 Training scenario

The first scenario in the Cyber Range emulates an IoT network interconnected with a 5G network. The attacker targets to compromise the 5G core network through a container breakout attack, gaining administrator access on the container's underlying host. The overview of this scenario is depicted in the following figure.

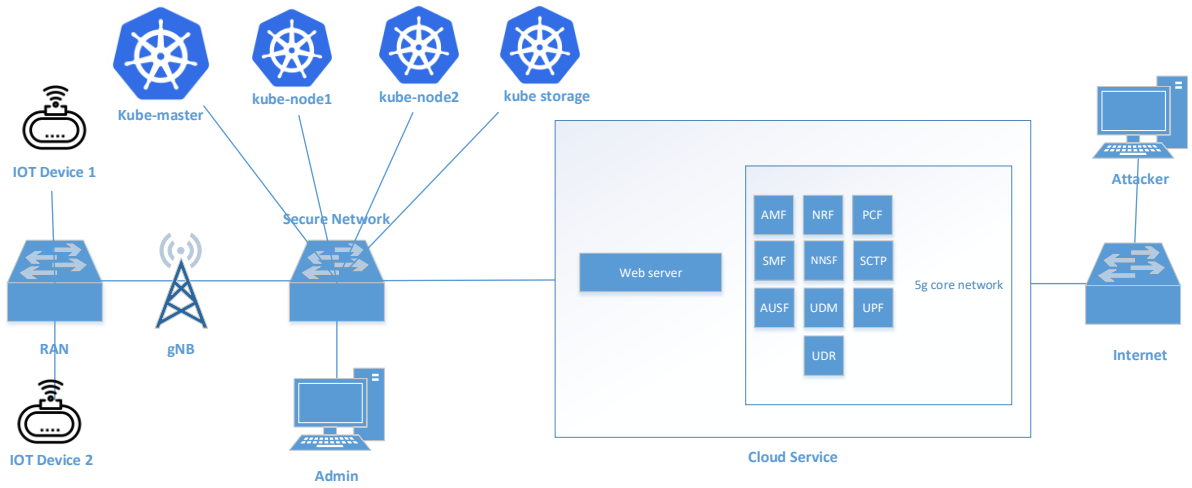


*Figure 3: IoT connected with a 5G network scenario*

In the second training scenario (Figure 4), it is assumed that hackers have infected IoT devices with malware to turn them into botnets that probe access points or search for valid credentials in device firmware that they can use to enter the network. Having network access through an IoT device, attackers can exfiltrate data to the cloud and threaten to keep, delete, or make the data public unless paid a ransom.

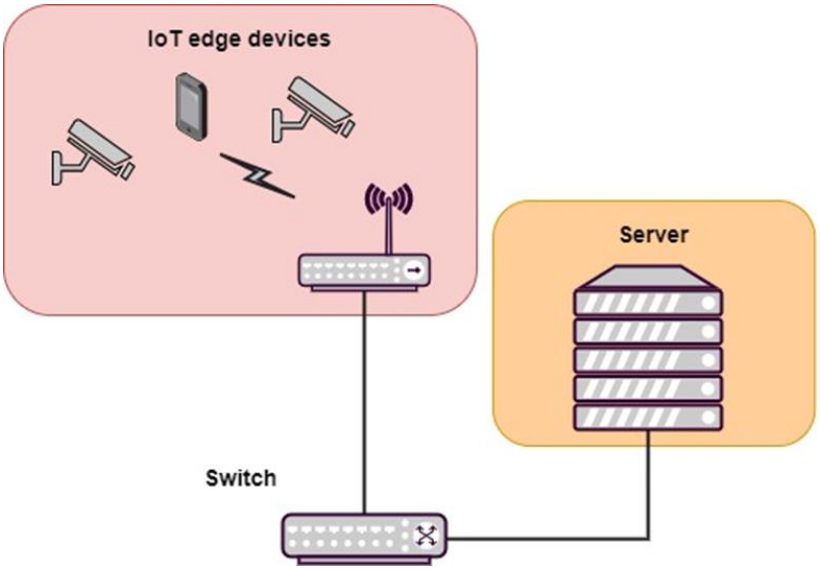


*Figure 4: Overview of the IoT Ransomware scenario*

And the last training scenario is also the PUC3 (Figure 5) where The Cyber Range emulates the data transmission and receiving of smart grid data including cross-border data exchange from the Tallinn Substation to the Helsinki Smart Grid. A threat actor with access to the smart grid system intercepts and reads the smart grid data and develops a manipulated data stream which is then implanted in the network to malform the input of the Smart Grid system. The aim of the attacker is to trigger abnormal/irregular smart grid system events, such as equipment failure/power surge/wrong energy readings.

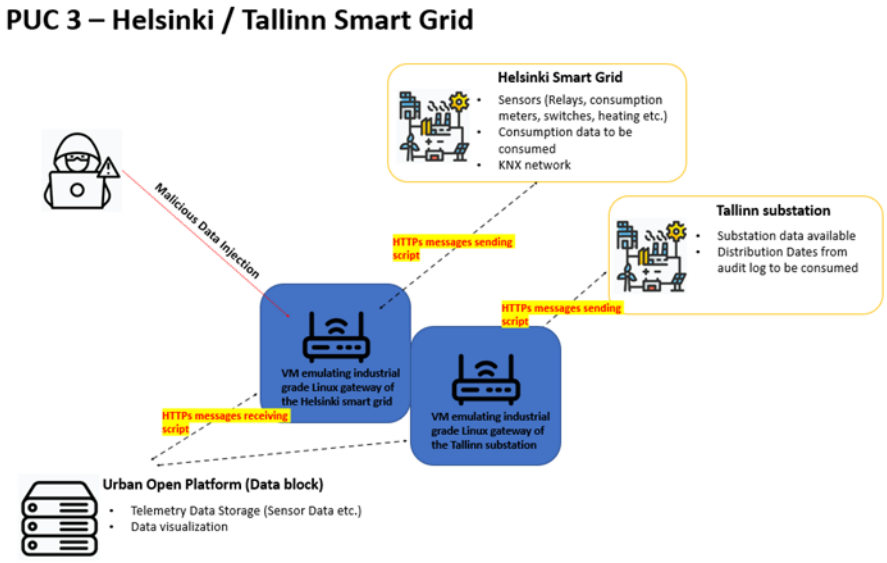


*Figure 5: Overview of PUC3 scenario*

# 3 IRIS PLATFORM COMPONENT FOR PODS

In order to provide training on the IRIS tools, these tools are deployed on the Cyber Range training environment. Several IRIS components are meant to be deployed on a Kubernetes environment, therefore a Kubernetes environment needs to be set-up on the Cyber Range.

This section discusses the Kubernetes implementation choices for the Cyber Range.

## 3.1 IRIS platform reference

The Integration work package (WP6) provides a Kubernetes environment, described in the repository https://gitlab.iris-h2020.eu/h2020-iris/iris-platform-integration-and-testing/hetzner-integration-infrastructure.

In order to standardize environments, the objective is to remain as close as possible to the above implementation.

However, the Integration environment relies on a cloud provider's (https://www.hetzner.com/cloud) infrastructure and services. As these services are not available on the bare-metal deployment of Kubernetes in the Cyber Range, adaptations and replacements must be made.

## 3.2 Components supporting the IRIS platform

### 3.2.1 Kubernetes

The Generic Kubernetes components are presented in the following figure:

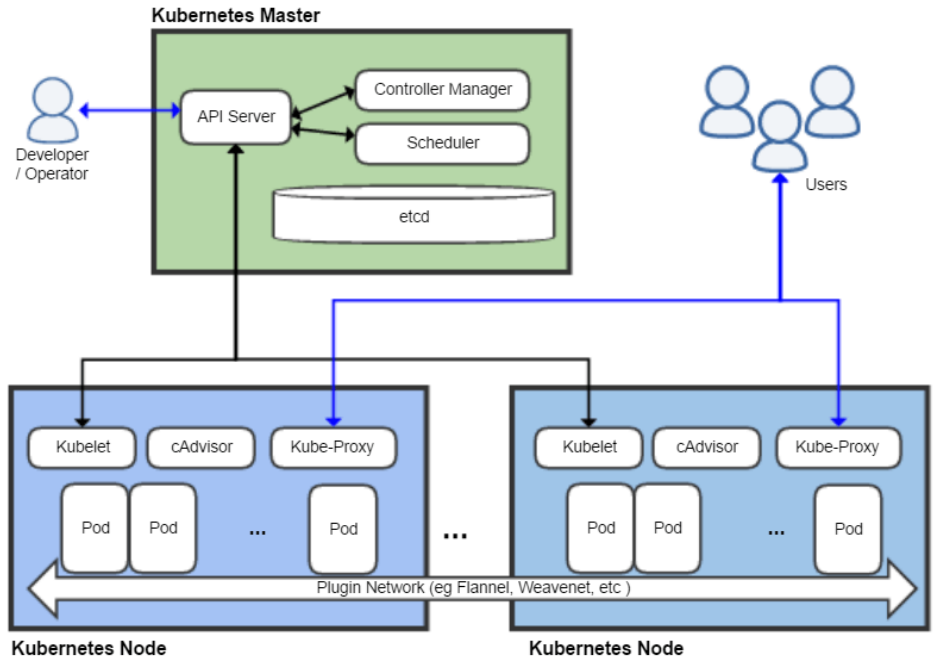


*Figure 6: Generic Kubernetes*[1]

### 3.2.2 Tools

Here is an overview of the different functions provided in the Integration work package, and their equivalent in the Cyber Range. Text is red where the Cyber Range environment differs.

| function | Integration environment | Cyber Range environment |
|---|---|---|
| **Deployment tool** | Kubeadm | kubeadm |
| **dashboard** | kubernetes-dashboard | kubernetes-dashboard |
| **CNI** | Flannel | flannel |
| **Gitalb runner** | gitlab runner | (not implemented) |
| **Ingress** | NGINX ingress controller | NGINX ingress controller |
| **Metrics** | metrics-server | metrics-server |
| **Monitoring** | Grafana | grafana |
| **Reflector** | emberstack/reflector | emberstack/reflector |
| **CSI** | csi.hetzner.cloud | nfs.csi.k8s.io |
| **Load Balancer** | hetzner.cloud | metalLB |
| **Public DNS** | hetzner.cloud | ori-edge/k8s_gateway |

*Table 1: Comparison pods between INTRA and the Cyber Range*

[1] https://en.wikipedia.org/wiki/Kubernetes

Note: for the Ingress resource, the same provider is used but with a slightly modified configuration.

## 3.3 Cyber Range specifics

In this section, we will address the functions where the Cyber Range diverges from the Integration infrastructure and discuss the reasoning for the choices.

### 3.3.1 Gitlab Runner

The Cyber Range is not part of the IRIS gitlab CI/CD system as the two systems are not interconnected. Therefore the gitlab runners are not deployed.

### 3.3.2 CSI

The Container Storage Interface (CSI) expose storage to the Kubernetes infrastructure. It is used as a backend for the "PrivateVolume" ressources.

Here the only production driver available that allows a shared access to storage and does not require the existence of a storage system is the NFS driver (https://github.com/kubernetes-csi/csi-driver-nfs).

Therefore we create a dedicated virtual machine with a NFS export and the NFS provider ("nfs.csi.k8s.io") to manage the export.

### 3.3.3 Load Balancer

The NGINX ingress controller makes use of an external network load balancer to provide the external IP. However, Kubernetes does not provide an implementation of network load balancers out of the box.

We use the MetalLB implementation (https://metallb.org/) to provide the load balancer function transparently.

### 3.3.4 External DNS

Public DNS resolution of Ingress FQDNs needs to be provided (these are the addresses from which the services are accessed by the end users).

The k8s_gateway CoreDNS plugin provides DNS resolution for Kubernetes external resources, ie Ingress and Service of type LoadBalancer. We deploy a CoreDNS pod with this plugin to provide external DNS resolution to clients using https://github.com/ori-edge/k8s_gateway .

It is configured to answer to iris-h2020.eu and iris-h2020.lan, and forward all other requests to the resolver configured on the /etc/resolv.conf of the host.

The IP of the external DNS service can be retrieved by the command (colum EXTERNAL-IP):

*kubectl –n kube-system get service external-dns*

Or to get only the IP:

*kubectl –n kube-system get service external-dns –o jsonpath='{.status.loadBalancer.ingress[0].ip}'*

That IP can then be configured as the only DNS server in the client's configuration ("blue team" machine).

## 3.4 Implementation

### 3.4.1 Servers

The following virtual machines are deployed in the Cyber Range to support the Kubernetes infrastructure.

| server | Role |
|---|---|
| **k8s-master** | Kubernetes master |
| **k8s-node1** | Kubernetes node |
| **k8s-node2** | Kubernetes node |
| **k8s-nfs** | NFS server for CSI |

*Table 2: Kube architecture*

### 3.4.2 K8 Services

The deployment of services as been configured as follow:

Forked from https://gitlab.iris-h2020.eu/h2020-iris/iris-platform-integration-and-testing/hetzner-integration-infrastructure.git and adapted for Cyber Range infrastructure.

(Hetzner cloud services not available in this environnement)

```
## Nodes

| **Node**      | **Role** | **Internal IP**      |
|:------------:|:--------:|:--------------------:|
|  master      |  master  |     192.168.20.101   |
|  node1       |   node   |     192.168.20.102   |
|  node2       |  node    |     192.168.20.103   |
|  k8-storage  |    nfs   |     192.168.20.110   |

External network:  192.168.21.100-192.168.21.250 (managed by
metal-lb)

# Kubernetes Cluster
```

```
## Versions

- Kubernetes (kubelet, kubeadm, kubectl): 1.25.5
- OS-Image: Ubuntu 20.04.5 LTS
- Kernel: 5.4.0-131-generic
- Container-Runtime: containerd<span>://1.6.9</span>
- runc: 1.1.4
- CNI Plugins: 1.1.1
- Flannel CNI Plugin: 1.1.0
- ingress-nginx-controller: 1.4.0
- hcloud-cloud-controller-manager: 1.13.2
- hcloud-csi-driver: 1.6.0

## Nodes

| **Node** | **Role** | **Internal IP** | **External IP** | **CPU
Cores** | **RAM** | **Filesystem** |
|:--------:|:--------:|:---------------:|:---------------:|:-----
--------:|:-------:|:--------------:|
|  iris01  |  master  |     10.0.0.2    |   88.99.13.107  |
4        |   8GB    |      160GB      |
|  iris02  |  worker  |     10.0.0.5    |        -        |
8        |   16GB   |      240GB      |
|  iris03  |  worker  |     10.0.0.3    |        -        |
8        |   16GB   |      240GB      |

## Links & Endpoints

Kubernetes API -- k8s-api.platform.iris-h2020.eu:6443

NGINX Ingress Controller -- platform.iris-h2020.eu

Kubernetes Dashboard -- https://k8s-dashboard.platform.iris-
h2020.eu

Grafana -- https://grafana.platform.iris-h2020.eu

## Installation Steps

### Node Preparation

1. Update the system
```
sudo apt-get update && sudo apt-get upgrade && sudo apt-get dist-
upgrade -y
```

2. Disable ufw
```
sudo systemctl stop ufw.service
sudo systemctl disable ufw.service
```
```

````
3. Forwarding IPv4 and letting iptables see bridged traffic
```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
```

### Install a container runtime
1. Install containerd
```
wget
https://github.com/containerd/containerd/releases/download/v1.6.9
/containerd-1.6.9-linux-amd64.tar.gz
sudo tar Cxzvf /usr/local containerd-1.6.9-linux-amd64.tar.gz
wget
https://raw.githubusercontent.com/containerd/containerd/main/cont
ainerd.service
sudo mv containerd.service /usr/lib/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable --now containerd
```

2. Install runc
```
wget
https://github.com/opencontainers/runc/releases/download/v1.1.4/r
unc.amd64
sudo install -m 755 runc.amd64 /usr/local/sbin/runc
```

3. Install CNI plugins
```
sudo mkdir -p /opt/cni/bin
wget
https://github.com/containernetworking/plugins/releases/download/
v1.1.1/cni-plugins-linux-amd64-v1.1.1.tgz
sudo tar Cxzvf /opt/cni/bin cni-plugins-linux-amd64-v1.1.1.tgz
````
````

```
```

4. Configure the systemd cgroup driver
```
sudo mkdir -p /etc/containerd/
containerd config default > /etc/containerd/config.toml
sudo nano /etc/containerd/config.toml
"""
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
  ...

[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.opt
ions]
    SystemdCgroup = true
"""
sudo systemctl restart containerd
```

### Deploy Cluster
1. Install kubeadm, kubelet and kubectl
```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-
keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-
keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" |
sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

2. Initialize the control-plane
```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-
cert-extra-sans=88.198.93.121,10.0.0.2,k8s-api.platform.iris-
h2020.eu
```

3. Make kubectl work for the non-root user
```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

4. Deploy flannel CNI
```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documenta
tion/kube-flannel.yml
```

````
```

5. Join the worker nodes
```
sudo kubeadm join 10.0.0.2:6443 --token wiyxzd.i7dfyrjv6na9eore \
     --discovery-token-ca-cert-hash
sha256:c4b0d277f82aa5c93014705ce018c96b255d9dd675385f8246eb646f33
3a5206
```

## Template Config Files
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: __BUILD_PIPELINE__-pvc
  namespace: __NAMESPACE__
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  volumeMode: Filesystem
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: __BUILD_PIPELINE__
  namespace: __NAMESPACE__
  labels:
    app: __BUILD_PIPELINE__
spec:
  replicas: 3
  selector:
    matchLabels:
      app: __BUILD_PIPELINE__
  template:
    metadata:
      labels:
        app: __BUILD_PIPELINE__
    spec:
      containers:
        - name: __BUILD_PIPELINE__
          image: irish2020/__BUILD_PIPELINE__
          imagePullPolicy: Always
          resources:
            limits:
              memory: "500Mi"
              cpu: "100m"
            requests:
              memory: "250Mi"
````

```
              cpu: "50m"
          env:
            - name: NODE_ENV
              value: __BUILD_PIPELINE__
          ports:
            - containerPort: __SERVICE_PORT__ #internal
          command: [ "__START_COMMAND__" ]
          args: [ "__START_COMMAND_ARGUMENTS__" ]
          volumeMounts:
            - name: __BUILD_PIPELINE__-volume-mount
              mountPath: /path/to/mount/data
      volumes:
        - name: __BUILD_PIPELINE__-volume-mount
          persistentVolumeClaim:
            claimName: __BUILD_PIPELINE__-pvc
      imagePullSecrets:
      - name: regcred
---
apiVersion: v1
kind: Service
metadata:
  name: __BUILD_PIPELINE__-service
  namespace: __NAMESPACE__
  annotations:
    load-balancer.hetzner.cloud/name: "kubelb"
    load-balancer.hetzner.cloud/use-private-ip: "true"
spec:
  selector:
    app: __BUILD_PIPELINE__
  ports:
  - protocol: TCP
    port: __SERVICE_PORT__ #external
    targetPort: __SERVICE_PORT__ #internal
  type: LoadBalancer
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: __BUILD_PIPELINE__-ingress
  namespace: __NAMESPACE__
spec:
  ingressClassName: nginx
  tls:
  - hosts:
      - __BUILD_PIPELINE__.platform.iris-h2020.eu
  rules:
  - host: __BUILD_PIPELINE__.platform.iris-h2020.eu
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
```

```
          service:
            name: __BUILD_PIPELINE__-service
          serviceName: __SERVICE_PORT__
```

### Variables Details
- __NAMESPACE__ : The network namespace the component should be deployed to. Only components in the same namespace can see each other, without any extra network configuration (iris-wpxx).
- __BUILD_PIPELINE__ : The name of the application to be deployed.
- __START_COMMAND__ : The command the container will run on boot.
- __START_COMMAND_ARGUMENTS__ : The arguments to pass to the command that will run when the container first starts up.
- __SERVICE_PORT__ #external : The port to expose the container on the "internet".
- __SERVICE_PORT__ #internal : The port to expose the container on the local network. This should match the port exposed on the Dockerfile.

# 4 IRIS LAB PODS DEPLOYEMENT IN THE CYBER RANGE

This section presents the description of the deployment of each IRIS component in the IRIS cyber range.

## 4.1 ATA components

### 4.1.1 Nightwatch

#### 4.1.1.1 Introduction

Nightwatch is an AI-based threat detection tool, which enables the identification of threats targeting IoT and AI-provisioned systems through activity readings and endpoint behavior heuristics. Nightwatch leverages CLS's patent-protected artificial intelligence technologies for accurately and rapidly determining the likelihood that an IoT or AI-provisioned infrastructure/system has been compromised.

#### 4.1.1.2 Description

The overall architecture of Nightwatch along with its main components (the Probe and Cortex) and its interaction with other IRIS components is shown in Figure 7. As it can be seen, the Nightwatch-Probe is tasked with monitoring the traffic from the devices/assets connected to the targeted infrastructure, generating logs relative to the endpoints monitored as well as collecting information relative to various network protocols e.g., DNS, ICMP, http, etc. The Nightwatch-Probe seamlessly pushes the generated logs to a Redis database connected to the Nightwatch-Cortex for analysis and inspection by the Cortex's detection mechanism which is composed by 'Nodes' (i.e., fleet of detectors). Upon detection of threat events, the Nightwatch-Cortex pushes threat alerts to the IRIS Advanced Threat Intelligence Orchestrator (ATIO) in the form of STIX2.1-formatted JSON reports. The threat reports produced by Nightwatch can then be forwarded to the IRIS Threat Intelligence Sharing component of the CTI module through MISP and to the IRIS Dashboard to become visible to the IRIS end-users.

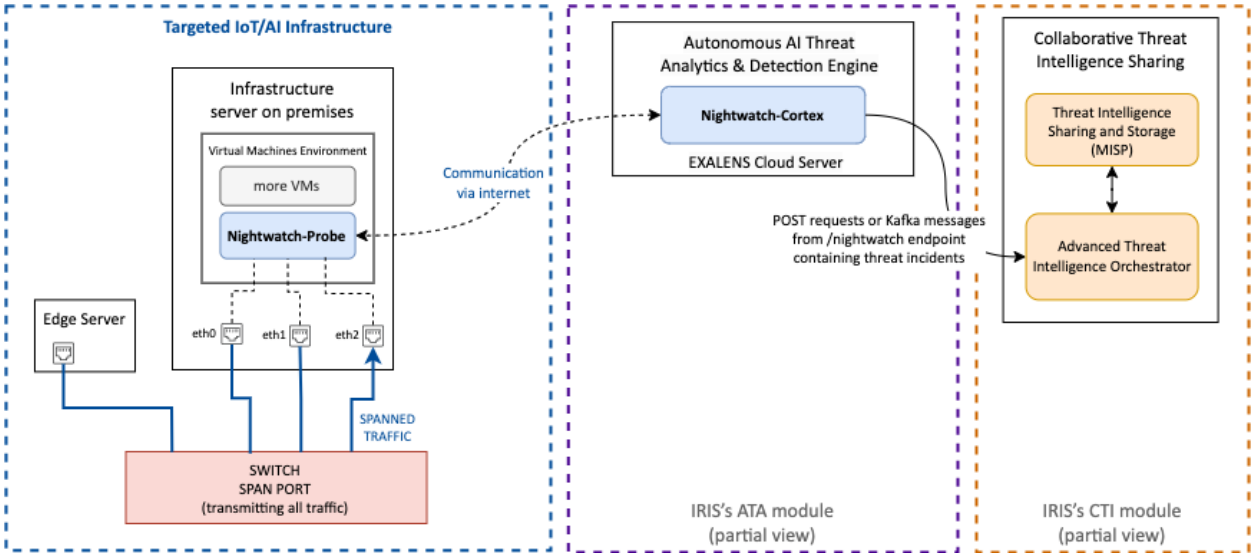


*Figure 7: ATA's Nightwatch Architecture*

### 4.1.1.3 Development status

On M26 of the project, several detection 'Nodes' of the Nightwatch tool have been developed. These detections include among other network port scanning detection, denial of service using flood attacks on ICMP, TCP, and UDP, SSH brute force detection, and ARP cache poisoning detection. The 'Nodes' currently supported by Nightwatch along with their status, is shown in the Table below.

| | Node/Detection name | Protocol | Status |
|---|---|---|---|
| **TD1** | **C2 Beacon**<br>Suspicious {service indicator} beaconing to {external/internal} host | HTTP | Working |
| | | TCP | Working |
| | | UDP | Working |
| **TD2** | **Network Scanning**<br>Unusual internal {proto} network scanning activity | TCP | Working |
| | | UDP | Working |
| | | ARP | Working |
| | | ICMP | Working |
| **TD3** | **BruteForce**<br>Suspicious number of failed {service_indicator} login attempts in small time interval | SSH | Working |
| | | FTP | Working |
| | | RDP | Working |
| | | SMBv2 | Working |
| **TD4** | **Network Flood**<br>Unusual number of consecutive {service_indicator} connections with small interval | TCP | Working |
| | | UDP | Working |
| | | ICMP | Working |
| **TD5** | **ARP Poison**<br>Suspicious unsolicited ARP messages | Suspicious ARP replies | Working |
| **TD6** | **Discovery**<br>Suspicious {deployment} discovery / enumeration activity via {service_indicator} | Network share discovery | Work In progress |
| | | System accounts discovery | Work In progress |

| | Node/Detection name | Protocol | Status |
|---|---|---|---|
| **TD7** | **Execution**<br>Remote service/ task creation<br>Suspicious {deployment} remote Service / Task creation via {service_indicator} | wmi_service | Work In progress |
| | | scheduled_task | Work In progress |
| **TD8** | **Impact**<br>Suspicious {deployment} remote Shutdown / Reboot via {service_indicator} | System Shutdown/Reboot | Work In progress |
| **TD9** | **Defence Evasion**<br>Suspicious attempt to remotely clear Windows event logs via {service_indicator} | Remote windows event log removal | Work In progress |
| **TD10** | **Lateral Movement**<br>Suspicious file transfer over {service_indicator} | OS Credential Dumping (DCSync) | Work In progress |
| | | Failed SMB connections | Work In progress |
| **TD11** | **Reconnaissance**<br>Directory Bruteforce | HTTP, HTTPS | Work In progress |
| **TD12** | **Credential Access** | HTTP, HTTPS | Work In progress |
| **TD13** | **LLMNR/NBNS session hijacking detection** | HTTP, HTTPS | Work In progress |
| **TD14** | **Path Enumeration detection** | HTTP, HTTPS | Work In progress |

*Table 3. Nightwatch's list of supported detection 'Nodes'*

#### 4.1.1.4 Deployment status

An initial version of the Docker container deployment of Nightwatch on the IRIS's GitLab has been created, and unit tests have been included. This initial version is not the final iteration of Nightwatch's container, and further enhancements and adjustments will be made in subsequent iterations.

#### 4.1.1.5 PUC deployment status

The VM of the Nightwatch Probe has been successfully deployed in Cisco's server on-premises to support the execution of PUC1, while the Nightwatch-Cortex is already deployed as an Elastic Compute Cloud (EC2) instance on CLS's AWS cloud server. No deployment effort has been put forward until M26 of the project to support the execution of PUC2. The deployment of the Nightwatch-Probe VM to Tallinn's AI-enabled infrastructure is scheduled to take place by M28 of the project.

### 4.1.2 Sivi

#### 4.1.2.1 Introduction

IRIS's visual-aided anomaly detection system, namely SiVi, is capable of monitoring and identifying a variety of security threats. Graphs that can quickly, reliably, and clearly provide a network overview are the tool's main innovation. To provide the administrator with a complete anomaly detection environment, SiVi uses a variety of data visualization

techniques, including both standard and more advanced approaches (graph lines, activity gauge, tables, etc.).

#### 4.1.2.2 Description

The "Security monitoring and analysis mechanism" is an Intrusion Detection System that integrates multiple sensors. SiVi monitors and analyzes the multiple communication protocols at the network layer using the Suricata sensor and Machine Learning (ML) sensors.

At the network level, the sensors (Suricata and ML sensors) take network packets as input and transform them to network flows. These flows are analyzed, and each sensor produces a security record to alert the tool operator to probable security breaches. All these security records from various sensors will be combined into a common security event that a correlation engine can readily interpret (not included in SiVi).

Finally, the data is shown on the SiVi dashboard, which provides both quantitative and qualitative indicators, allowing security administrators to gain a better understanding of the network.

#### 4.1.2.3 Deployment status

All configuration files (K8 manifest, etc.) have been uploaded to the IRIS GitLab repository. SiVi is ready to be deployed both in PUC2 infrastructure and in PUC3 environment.

### 4.1.3 Vulnerability Manager (VDM)

#### 4.1.3.1 Introduction

The Vulnerability Discovery Manager (VDM) is the tool included in the IoT and AI-Provision Risk and Vulnerability Assessment Module of the IRIS Platform for the dynamic identification, analysis and reporting of vulnerabilities detected on environments with IoT devices and AI-based systems.

#### 4.1.3.2 Description

As described in section 3.3 of *IRIS D3.1 – IRIS risk and vulnerability assessment module*[2], the main functionalities included in the VDM are:

- Identification of vulnerabilities on the target infrastructure, which can include IoT devices and the platforms where AI-systems are running.
- Perform an impact assessment to classify and assign priorities to the vulnerabilities detected.
- Perform a risk assessment of the infrastructure considering risk models that includes vulnerabilities that can affect environments with IoT devices and platforms running AI-systems and take into account the relevance of the assets where they were found to suggest mitigation measures.
- Provide intelligence sharing functionalities so the information about the vulnerabilities can be shared though Threat Intelligence Platform in SITX format with authorized CERTS and CSIRTs.
- A novel automated pentesting AI-based engine to orchestrate penetration tasks learning during the process. This engine implements a reinforcement learning algorithm (DQN) and, once modelled the pentesting process of the target system as a Markov Decision Process (MDP), it discovers the optimal path to compromise the system. This policy indicates which are the best attack actions (exploits of vulnerabilities identified) that should be taken in each of the potential states the target system can be, which will help to prioritize the vulnerabilities that must be fixed.

The modules included in the Vulnerability Discovery Manager are also described in section 3.3 of D3.1. In summary, it has a modular design including a Vulnerability Scanner sub-component to manage scanning requests and interact with the infrastructure, a Vulnerability Assessment sub-component to manage the classification, prioritization and treatment of the vulnerabilities identified, a Vulnerability Reporting sub-component to address the generation and sharing of the vulnerability reports, a Vulnerability Storage sub-component and a Pentesting-AI Engine sub-component. Additionally, it has been added a plugin to integrate syslog messages generated in PUC1 with the VDM.

#### 4.1.3.3 Deployment Status

The development of the VDM has been completed during Task 3.2 and the docker images are available in https://hub.docker.com/u/irish2020, although further

[2] https://partners.inov.pt/iris-h2020/index.php/f/35206

enhancements and adjustments can be done during the deployment testing and the execution of the Pilot Use Cases. A first version of Kubernetes manifest files for the deployment of the VDM subcomponents in IRIS Cloud infrastructure are also available at https://gitlab.iris-h2020.eu/h2020-iris/iris-autonomous-threat-analytics/vdm. All these containers have been deployed in the namespace *iris-wp03*. Currently, we are working on including unit tests for the different components and on testing that the deployment and configuration of the different VDM sub-modules in the IRIS Cloud infrastructure is correct and they can interact among them.

#### 4.1.3.4 PUC deployment status

VDM has been successfully deployed in Cisco's server on-premises and a first round of tests have been performed for vulnerability scanning of the devices in PUC1 and to test integration with the Advance Threat Intelligence Orchestrator (ATIO) component. By the moment no deployment has been done of VDM in PUC2 and PUC3 yet.

### 4.1.4 Risk Based Response And Self Recovery

#### 4.1.4.1 Introduction

The risk-based response and self-recovery module (RRR) is a critical component of the IRIS platform for improving the security and resilience of IoT and AI-provisioned platforms. The module operates as a dynamic platform for incident response. It utilizes threat and vulnerability detection telemetry to generate incident response procedures by employing statistical analysis, optimization techniques, and game theory principles, all within the context of selected response and self-recovery actions. The framework integrates an optimization model designed to evaluate the least risky course of action in incident response strategies. Additionally, it includes a self-recovery mechanism that adapts a programmable API to IoT and AI-enabled platforms, enabling the execution of the remediation actions.

#### 4.1.4.2 Description

Figure 8 illustrates the refence architecture of the RRR module.

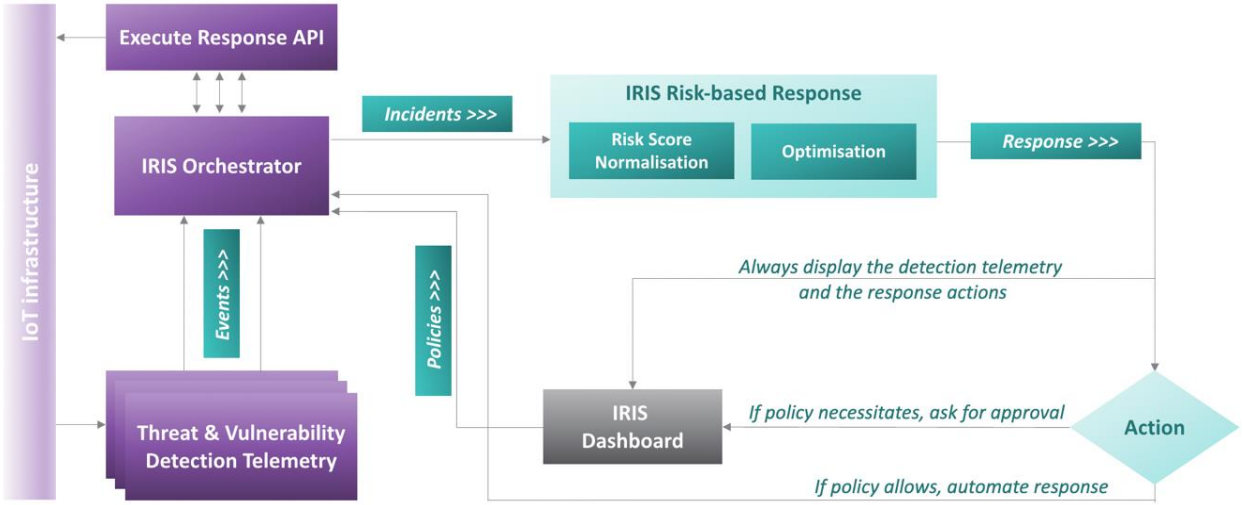


*Figure 8: Risk-based response and self-recovery module's related architecture.*

As it can be seen, the RRR module ingests detection telemetry from the ATA's threat analytics and vulnerability detection components, as well as security policies and asset criticalities from the IRIS end-users (CERTs/CSIRTs). Recommended response actions are generated to cover a range of response categories, including containment, hardening, and recovery. Where applicable, the module provides a list of clear and specific execution steps and commands to the IRIS Advanced Threat Intelligence Orchestrator (ATIO) aimed at restoring the monitored IoT/AI-enabled system in the face of potential threats. Responses generated by the module are communicated to the IRIS orchestrator via a RESTful interface and an external API.

#### 4.1.4.3 Deployment status

An initial version of the Docker container deployment of RRR on the IRIS's GitLab has been created, and preliminary unit tests have been included. This initial version is not the final iteration of RRR's container, and further enhancements and adjustments will be made in subsequent iterations.

#### 4.1.4.4 PUC deployment status

The module is currently deployed on CLS's cloud server to facilitate the interconnectivity with the Advanced Threat Intelligence Orchestrator (ATIO) and support the execution of the IRIS pilots. RRR is planned to be deployed in the respective pilot infrastructures as an "integrated package" along with EME and ATIO by M28 of the project.

### 4.1.5 SiHoneyPot

#### 4.1.5.1 Introduction

SiHoneyPot is Sidroco's platform to support the automated threat intelligence orchestration for implementing proactive defense measures. SiHoneyPot platform supports a variety of honeypots working as decoy systems designed to attract, detect, and distract cybercriminals from the real targets. Digital twin honeypots simulate the real system or network and its vulnerabilities, and monitor the activities of the attackers in a

controlled environment. These virtual representations of real assets are also enhanced to predict and prevent future cyberattacks, by using the data collected to improve the security posture of the real system or network.

#### 4.1.5.2 Description

SiHoneyPot platform provides two different solutions. Honeypots are very use-case specific and thus different pilot infrastructures require a specific solution, applicable and suitable to its needs.

*PUC2 solution – LiDAR Honeypot*

LiDAR Honeypot is designed to emulate a real-life LiDAR sensor, similar to the ones utilized in Autonomous Vehicles. LiDARs are pivotal sensors to an AV's autonomous navigation system and its malfunction can lead to various hazards. The LiDAR honeypot creates a virtual representation of a real LiDAR by replicating its main functionalities: a) handles properly formatted LiDAR-like data (point clouds), b) streaming of LiDAR-data towards a TCP socket, c) achieving real-LiDAR file transmission rate.

*PUC3 solution Modbus Honeypot*

Modbus honeypot aims to imitate both server and client devices using Modbus/TCP, thus misleading potential cyber attackers and hiding the real assets. Modbus supports a variety of operations interpreted into particular function codes. Although many critical infrastructures adopt Modbus, it is characterized by severe cybersecurity issues since it does not comprise sufficient authentication and authorization mechanisms. Consequently, potential cyberattacks can execute a plethora of cyberattacks.

Both LiDAR and Modbus honeypots share the SiHoneyPot dashboard where telemetry data, analytics and alerts are depicted in various visualization forms and tables. The dashboard offers additional functionalities, specific to each honeypot solution.

#### 4.1.5.3 Deployment status

All configuration files, for both LiDAR and Modbus, have been uploaded to the IRIS GitLab repository. SiHoneyPot solutions are ready to be deployed both in PUC2 infrastructure and in PUC3 environments.

## 4.2 DPA components

### 4.2.1 DPA crypto and DLT tools

#### 4.2.1.1 Introduction

The DPA (Data Protection and Accountability) module was designed to support auditing functions for incident response workflows, ensuring accountability and traceability.

#### 4.2.1.2 Description

It has three main components:

1) CryptoTools (Task 4.4 of IRIS, more info on D4.5) – provides self-encryption and Shamir Secret Sharing mechanisms to the DPA
2) HLF distributed network (blockchain) (Task 4.5 of IRIS, more info on the upcoming D4.6) – provides safe and immutably storage of audit data metadata
3) Off-chain database (Task 4.5 of IRIS, more info on the upcoming D4.6) – provides off-chain storage of encrypted audit data

The simplest use-case scenario where the DPA is used in the context of IRIS is the following:

1) The CTI orchestrator (an authorized system) POSTs audit logs to the DPA, when needed;
2) An authorized auditor queries the DPA to gain access to said audit logs.As for the development/deployment status of the DPA - currently, the DPA is implemented and is deployed in the IRIS integration infrastructure (as prepared by INTRA), ready for integration.

#### 4.2.1.3 Development status

As for the development/deployment status of the DPA - currently, the DPA is implemented and is deployed in the IRIS integration infrastructure (as prepared by INTRA), ready for integration.

## 4.3 CTI components

### 4.3.1 Advanced Threat Intelligence Orchestrator

#### 4.3.1.1 Introduction

*Advanced Threat Intelligence Orchestrator* (ATIO) is a full stack solution that acts like a middleware system, due to, its central location in the architecture, all data is transferred via it. Therefore, ATIO links ATA, EME (which includes CTI and DPA), and Infrastructure. Four backend services and two frontend services compose ATIO.

#### 4.3.1.2 Description

User interfaces (UIs), include the Workflow Designer (OWM), Sharing and Response Task Management and Tracking, and are shown in the figure below.

The former UI enables end-users to create or use pre-made, **cyber incident detection of threats response/recovery and reporting/sharing workflows**. Also, they can monitor the targeted workflows and be aware for the status of each workflow step throughout the end-to end process execution. The workflows can be executed either automatically or semi automatically. Additionally, the end-user will be able to choose from among workflows depending on each category of attack type and modify or gain experience from them.

ATIO is a restful solution, providing OpenAPIs interfaces, communicating with STIX.2.1 formats and its extensions. The latter UI lists the potential response actions received by RRR. Both UI solutions would be integrated on EME Unified Dashboard.

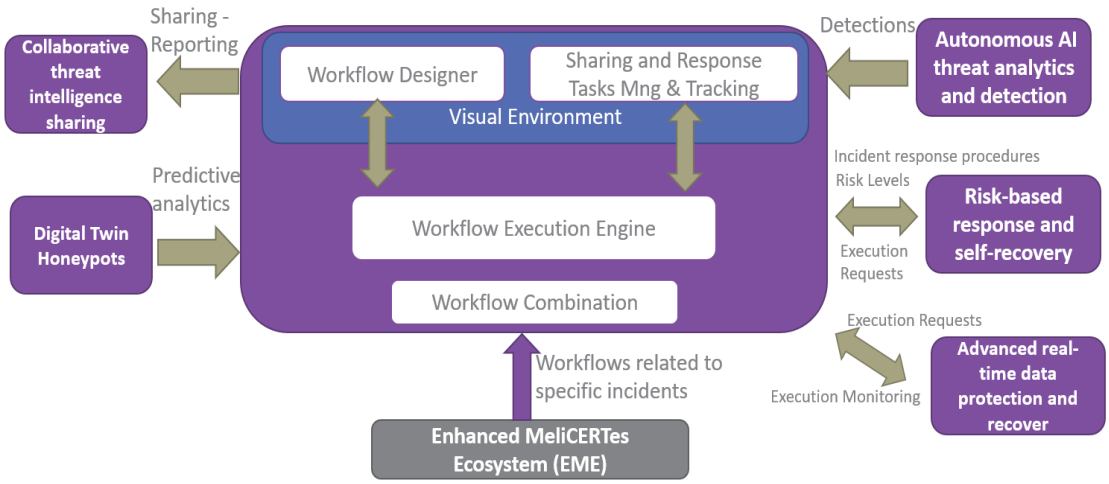


*Figure 9:* The internal structure of the Advanced Threat Intelligence Orchestrator and its relationship to input and output information.

Backend services, include the Workflow Execution Engine, Workflow Combination Engine, Data Exchange Framework, Command Execution Requests Framework, and altogether with the front-end services provide to the system the below capabilities.

1. ATIO aggregates multiple flows to one direction (e.g. event detection originating from the infrastructure into a single point).
2. Modifies the events by transforming from one format to another format.
3. Filters the information.
4. Translates from STIX to MISP format.
5. Forwards these events to the appropriate modules that are required for event storage, analysis, enrichment, risk and response calculation, as well as user-friendly presentation and auditor backlog.
6. Efficient routing

ATIO is a restful solution, providing OpenAPIs interfaces, communicating with STIX.2.1 formats and STIX extensions.

#### 4.3.1.3 Development status

The tool is already deployed in INTRA premises.

### 4.3.2 Threat Intelligence Sharing and Storage

#### 4.3.2.1 Introduction

The Cyber Threat Intelligence Sharing and Storage tool is based on the MISP Open-Source Threat Intelligence Platform. MISP is a repository that can be used for the collection and storing of threats and vulnerabilities targeted to IoT and AI-driven ICT systems. This tool aims to create dynamic taxonomies and ontologies of threats, attacks and vulnerabilities in order to assist researchers and practitioners in developing a common lexicon about threats, attacks and vulnerabilities with the end goal of setting standards and best practices for managing the cybersecurity of ICT systems against attackers. Additionally, the added value of the Cyber Threat Intelligence (CTI) tool is two-fold. It supports extensive analysis through a secure and trusted environment (e.g., MISP) as well as leads to the overall situational awareness.

#### 4.3.2.2 Description

The CTI Sharing and Storage tool is able to collect, store, correlate, and share information about threats, attacks and vulnerabilities from both internal and external sources. Internal sources comprise honeypot instances, firewalls, SIEM, IDSs etc. and they are selected from the IoT and AI-based infrastructures from WP3 ATA tools (Vulnerability Manager, NIGHTWATCH, BINSEC, SiVi, SiHoneypot). External sources include among others vulnerability databases, CERT feeds, databases with Proof-of-Concept (PoC) exploits, social media platforms, as well as various sources from both Surface and Dark Web that are already stored in MISP. The CTI Sharing and Storage tool gathers cybersecurity information. Following, the next step includes the CTI extraction from the gathered information. Then simple and advanced correlation techniques have been used in order to enrich the extracted CTI. The enriched CTI is stored and shared through MISP. MISP provides a user-friendly dashboard that the user can use to interact with the stored CTI.

The CTI Sharing and Storage tool creates taxonomies and ontologies following the steps:

- The extracted CTI from ATA tools is received from the ATIO and pushed as input to MISP. The gathered information is correlated to find associations between data and intelligence collected.
- The data received is used to extract the most valuable information (threats, attacks and vulnerabilities) for the taxonomy generation. Named Entity Recognition (NER), BERTopic modelling and Pattern matching are used. More specifically, NER facilitates the identification and extraction of various named entities including malware names, hashes, the purpose of attacks and other relevant information. BERTopic modelling utilizes embedding to convert input sentences into a numerical representation. Then through a clustering procedure, it creates topics. Last, through a technique called c-TF-IDF representative names are inserted into the topics. Pattern matching matches terms in specific fields.
- After running these algorithms, the taxonomies are created.

- The generated taxonomies are used to update existing threat taxonomies (e.g. MISP Taxonomies[3]) using the taxonomies' terms identified by NER, BERTopic and Pattern matching.
- The REBEL[4] model is used for Relation Extraction of relationships between the different taxonomies. Based on the extracted information, a merged ontology is generated by the different taxonomies.
- Then, a merged ontology is created by the different generated ontologies.
- The ontologies are developed using OWLready2 package. Two existing ontologies are used namely MALOnt[5] and IoTsec[6] in the whole procedure. The terms of the merged ontology are used to update the existing ontologies using a semantic search procedure.

#### 4.3.2.3 Development Status

The CTI Sharing and Storage tool has been already deployed. We continue working on any request arises from the WP6 leader Integrator.

### 4.3.3 EME

#### 4.3.3.1 Introduction

Within IRIS, the MeliCERTes platform will form the basis of developments and will be extended to facilitate Collaborative Cyber-Threat Intelligence Sharing, among CI Operators (e.g., smart city infrastructure operators, IoT infrastructure operators, etc.) and CERTs/CSIRTs with focus on AI and IoT relevant threats and attacks. The IRIS-Enhanced MeliCERTes ecosystem (EME) will incorporate the majority of the technical developments that concern the CTI sharing in IRIS and act as a CTI sharing and collaboration interface towards the envisaged users of the IRIS platform. EME will act as a distributed and customized solution, and provide for secure and trusted online communication, collaboration and information sharing among CI operators and CERTs/CSIRTs allowing them to interact with the IRIS platform through a unified customizable dashboard.

#### 4.3.3.2 Description

EME, consists of a selection of MeliCERTes CSP v2.0 developed components which will be extended and configured to support IRIS project's objectives and provided functionality. In addition, EME is built in a modular way, allowing for seamless integration with the IRIS developments in the context of CTI. More specifically, within EME the following components are included.

- Cerebrate (MeliCERTes 2): Cerebrate will support the IRIS users and organizations definitions offering a visual environment for managing IRIS roles and entities that results to the description of Trusted Circles (TC). TCs aim to drive the CTI communication and sharing of the CTI data that are generated by the IRIS platform.

[3] https://www.misp-project.org/taxonomies.html
[4] https://github.com/Babelscape/rebel
[5] https://github.com/aiforsec/MALOnt
[6] https://github.com/brunomozza/IoTSecurityOntology/tree/master

- MISP: EME will host a MISP instance that will support the CTI communication towards the IRIS users. The particular MISP instance will reflect the work that was been described in the context of "Threat Intelligence Sharing & Storage" module.
- KEYCLOAK IMS: EME will incorporate an identity and access management solution that will support the secure authentication and authorization of IRIS users and services.
- REST API and DB to facilitate Dashboard communication and storage requirements.
- Unified UI: EME will include a unified visual environment (dashboard). The UI will facilitate the CTI information visual representation to the IRIS users. The unified dashboard will loosely integrate all the IRIS developed visual environments, safeguarding the coherence of the IRIS platform towards its users. More specifically will provide:
    - CTI information sharing home page: Presenting the CTI threats detected by the platform.
    - CTI orchestration information: Presenting CTI mitigation actions' workflows to the IRIS users.
    - Audit log query private/security view: Presenting audit logs that are stored in the DPA module's BC. Stringent security mechanisms will be applied in order to allow only authenticated and authorized security personnel to have access to this view.

#### 4.3.3.3 Development status

The development of the IRIS-Enhanced MeliCERTes Ecosystem (EME) is considered to proceed smoothly according to plan. The core EME components, namely backend RESP API, EME unified UI, MISP, CEREBRATE and KEYCLOAK development is well advanced and at the same time ongoing. The latest version of the EME core components' source code has been uploaded to the IRIS GitLab private instance and the EME modules have been deployed to the IRIS integration and testing cloud infrastructure.

Currently, the focus in EME development is put on one hand on the final development of the EME unified UI and on the other on the REST API (EME backend) that realizes the communication of the CTI events from the IRIS-enabled infrastructure to the CERTs/CSIRTs EME instances. The last stage of the EME development activities concern the integration tasks that will also realize the incorporation of the IRIS tool-specific dashboards (ATIO, SiHoneypots) into the EME unified dashboard.

# 5 CONCLUSION

In this deliverable, the technical details of the emulated IRIS infrastructure for training scenario has been described. This emulated platform is provided by the cyber range. It is composed of different assets to re-create a topology close to IRIS integration platform. The description of these components, in term of resources, software and main configuration has been provided. Along with this platform, the description of the different modules that will be integrated in the IRIS platform with their status.