European Commission

Horizon 2020
European Union funding
for Research & Innovation

Cyber Security PPP: Addressing Advanced Cyber Security Threats and Threat Actors

Cyber Security Threats and Threat Actors Training - Assurance Driven Multi- Layer, end-to-end Simulation and Training

D2.2: Emulated components monitoring module†

Abstract: This deliverable defines tools and mechanisms, able to monitor the status of the THREAT-ARREST emulated components in real-time and provide information to the rest of the platform's components. The main component that is analysed in this document is the *'Emulated Components Monitor'*, which captures the status of each running emulated component (i.e. VM) and reports the details (e.g. CPU and RAM usage) to the rest tools, via the platform's message broker.

Contractual Date of Delivery	30/11/2019
Actual Date of Delivery	30/11/2019
Deliverable Security Class	Public
Editor	*Michael Vinov (IBM)*
Contributors	*George Hatzivasilis (FORTH)*, *Hristo Koshutanski (ATOS)*, *Marinos Tsantekidis (TUBS)*
Quality Assurance	*Fulvio Frati (UMIL),* *Dirk Wortmann (SIMPLAN)*

† *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 786890.*

The *THREAT-ARREST* Consortium

Foundation for Research and Technology – Hellas (FORTH)	Greece
SIMPLAN AG (SIMPLAN)	Germany
Sphynx Technology Solutions (STS)	Switzerland
Universita Degli Studi di Milano (UMIL)	Italy
ATOS Spain S.A. (ATOS)	Spain
IBM Israel – Science and Technology LTD (IBM)	Israel
Social Engineering Academy GMBH (SEA)	Germany
Information Technology for Market Leadership (ITML)	Greece
Bird & Bird LLP (B&B)	United Kingdom
Technische Universitaet Braunschweig (TUBS)	Germany
CZ.NIC, ZSPO (CZNIC)	Czech Republic
DANAOS Shipping Company LTD (DANAOS)	Cyprus
TUV HELLAS TUV NORD (TUV)	Greece
LIGHTSOURCE LAB LTD (LSE)	Ireland
Agenzia Regionale Strategica per la Salute ed il Sociale (ARESS)	Italy

Document Revisions & Quality Assurance

Internal Reviewers

1. *Fulvio Frati (UMIL)*
2. *Dirk Wortmann (SIMPLAN)*

Revisions

Version	Date	By	Overview
0.6	28/11/2019	Editor	Minor fixes according to UMIL final feedback
0.5	27/11/2019	Editor	UMIL, FORTH, SIMPLAN feedbacks addressed
0.4	24/11/2019	FORTH	FORTH's update
0.3	21/11/2019	Editor	UMIL feedback addressed: added use-cases
0.2	19/11/2019	Editor	UMIL review feedback addressed
0.1	07/10/2019	Editor	First Draft

Executive Summary

This deliverable provides a technical description of the design and development of the THREAT-ARREST's *Emulated Components Monitor (ECM)* module. The overall component is a collection of collaborative services aimed to provide accurate readings of the Platform's hardware and virtual resources and make them available for both automatic resource management tools and resource visualisation tools. The *OpenStack's Nova service* is mainly utilized for this purpose. The Monitor will report the state of the emulated components (i.e. RAM, CPU, disk, and network usage) and the effects of the training actions on the system (e.g. in case of a Denial of Service (DoS) attack or an infected machine). This deliverable is the result of the task T2.2 activities and follows the first-year achievements for the Emulation Tool in the deliverables "D2.1 – Emulated components' generator module v1" (THREAT-ARREST, 2019a), "D2.3 – Interlinking of emulated components module v1" (THREAT-ARREST, 2019b), and "D2.4 – Emulation tool interoperability module v1" (THREAT-ARREST, 2019c), respectively.

Table of Contents

List of Abbreviations

CTTP Cyber Threat and Training Preparation

DoS Denial of Service

DDoS Distributed DoS

ECM Emulated Components Monitor

HPC High-Performance Computing

IaaS Infrastructure-as-a-Service

JSON JavaScript Object Notation

NaaS Networking-as-a-Service

REST Representational State Transfer

VM Virtual Machine

List of Figures

1 Introduction

The objective of this document is to provide a technical description of the design and development of the Emulated Components Monitor (ECM), which records the status of the deployed emulated components (virtual machines (VMs)) and reports the details to the rest THREAT-ARREST platform, mainly the Training/Dashboard and Visualization Tools.

The goal of the task T2.2 is to create services and mechanisms, able to monitor the status of the emulated components in real-time and provide information to the rest of the platform's tools. These modules compose the ECM, which reports information about the state of the emulated nodes as the training process takes place. Moreover, the monitor reports the state of the emulated components and the effects of the training actions on the system, like a Denial of Service (DoS) attack performed by a VM controlled by the trainer to a VM controlled by the trainee.

The project platform is based on a model-driven approach where Cyber Threat and Training Preparation (CTTP) models are built to specify attack scenarios on one side, and the security controls that can be used against them on the other side. A model includes the possibility to define the structures needed to drive the training process and the controls to monitor the current state of the cyber system together with security assurance mechanisms to assess the relevance of training.

OpenStack has been chosen as a reference architecture for the THREAT-ARREST Emulation Tool due to its widespread acceptance and its recognized leadership in the market of open source cloud management infrastructures. For the monitoring of the deployed VMs status we utilized one of the OpenStack underlying services – *Nova Compute* – as it is described in this document.

The rest deliverable is organized as follows. Section 2 provides two use-cases, where Emulated Components Monitoring module serves as an essential THREAT-ARREST Platform capability. Then, Section 3 provides an overview of the technologies that are utilized by the Emulation Tool in order to deploy the environment. Finally, Section 4 describes the high-level architecture, major components, the algorithm and APIs used to fetch and publish information regarding the resource configurations and utilization.

2 Monitoring Use-Cases

The THREAT-ARREST training platform is aimed to accommodate a wide range of users with different levels of expertise – from novice computer system users to experienced system administrators and security officers (e.g. (Manifavas et al., 2014; Ferrera et al., 2018; Soultatos et al. 2019)). The ECM information for each emulated component is meant to be depicted like a task manager tab by the Visualization Tools. Each tab will show usage data, like speed and utilization, for the main features of the related VM, such as the CPU, Memory, Disk, and Networking elements. This Section briefly describes the three main scenarios that can be observed via the collected ECM data. For the trainee's point of view, these includes the normal functional state of a VM or the operation under an ongoing attack (e.g. a machine infected with a disruptive malware). For the THREAT-ARREST platform operator (e.g. the trainer), this may involve potential misconfigurations by the OpenStack environment.

2.1 Normal Operation

The types of data that are recorded by the ECM are detailed in the next section. Here, we present the high-level information that is available to the user in the form of CPU, Memory, Disk, and Network Interfaces graphs. As an example, we consider the VM for the captain's PC in the smart shipping scenario that has been deployed by the related CTTP model, as described in the deliverables D2.1 and D2.3. This is a PC with a CPU at 2.4 GHz, a 20GB Hard disk, a 16GB RAM, and an Ethernet and WiFi network interfaces. The next figure illustrates the relevant graphs for the normal operation of this VM.

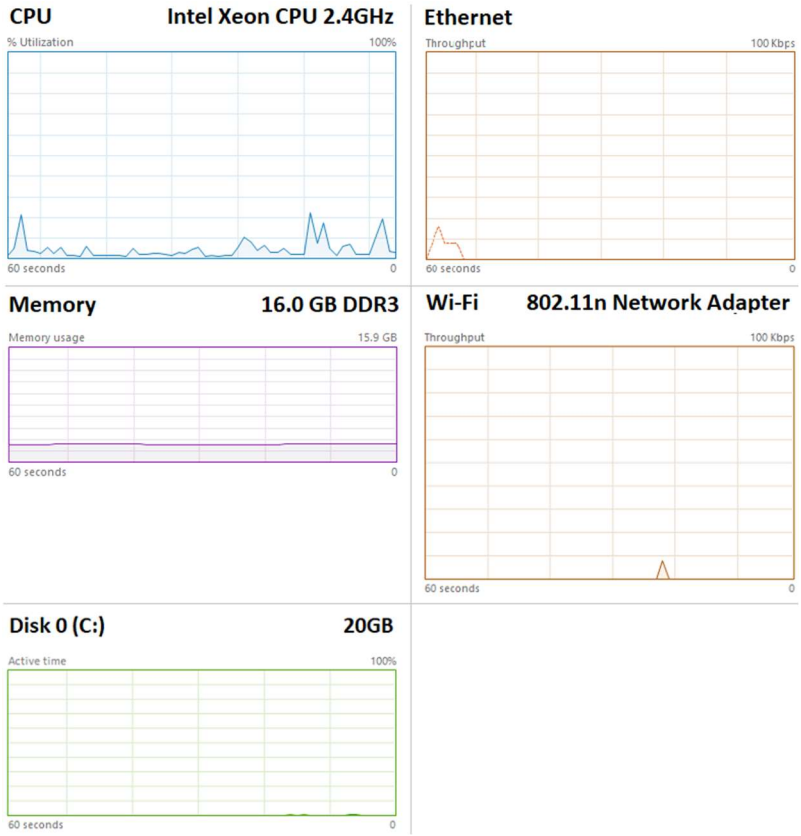

Figure 1 The emulated captain's PC under normal operation

The PC has just started, and the resource consumption is low.

2.2 Resource-targeting Cyber-Attacks

However, except from the normal operation, there is a wide range of known cyber-attacks which can result in dramatic system performance degradation. Denial-of-Service (DoS) attack is one of them; or its more sophisticated form – Distributed DoS (DDoS) attack (Wikipedia, 2019b; Zargar et al., 2013; Hatzivasilis et al., 2019) – is another one. Recently introduced and being widely adopted Cryptocurrency (Wikipedia, 2019a; Antonopoulos, 2018; Alexandris et al., 2018) ignited a brand new type of cyber-attacks – the Crypto-mining malware (Webopedia, 2019; Zimba et al., 2018). In such attacks, an attacker hijacks a victim's computer system, and then controls and severely overloads it.

The following figure depicts the CPU usage of the examined VM under such an attack. The CPU utilization of infected machine reaches the 100%. The instantiation is done via a CTTP model that deploys a VM where the malware is already installed in the image. The trainee has to understand that something is wrong, detect the problem, and perform the mitigation actions that have been taught during the main training courses (i.e. perform an anti-virus scan to block and delete the malware (Hatzivasilis et al., 2018; Papaefstathiou et al., 2014)).

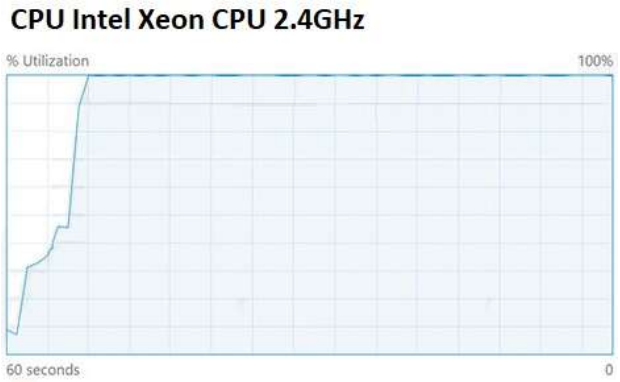

Figure 2 The emulated captain's PC CPU utilization under a crypto-mining attack

The offered resource monitor capabilities can be further leveraged by trainees with high security knowledge, like security officers and administrators, using THREAT-ARREST platform as their training tool. An example provided above is applicable here as well. High average and long-lasting pick CPU utilization figures, provided by the Monitor, may serve as an indicator of resource-hungry attack being performed as a part of the trainee scenario. Observing such a fact the trainee learns to detect such attacks and can be taught to take corrective and preventive measures.

Moreover, ECM as a system tool can help the THREAT-ARREST's system security officer to detect such attacks and take corrective actions. For example, high average and long-lasting pick CPU utilization figures, provided by the Monitor, may be caused by a resource monitor-hungry attack on the THREAT-ARREST Platform. Observing such a fact, a system security officer can take corrective measures to disinfect and reconfigure the system and prevent future attacks of similar kind.

2.3 System Misconfiguration

Misconfigured OpenStack environment is possible as well. It could result in poor overall system performance, observed as slow computation, bad responsiveness or even hang-ups. ECM can also help in such cases, by enabling the THREAT-ARREST platform administrators to detect these problems and properly fixing them. For example, high average and long-lasting pick memory utilization figures, provided by the Monitor, may be caused by sub-optimal memory

configuration. In some occasions, the requested hard disk may not be mounted properly with no utilization being recorded in the related tab. Another example is where long-lasting saturation of CPU utilization is observed. The picture below (Figure 3) depicts the case where the examined VM was overloaded for a long time. Such overload can be caused by misconfiguration as well. Upon observing such a behaviour, an administrator can reconfigure the system to gain better performance.

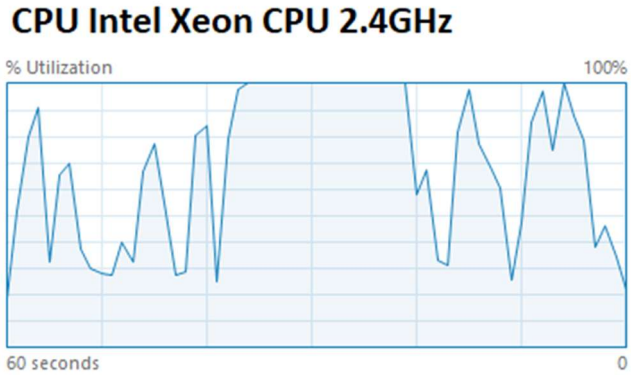

Figure 3 The emulated captain's PC CPU utilization for a misconfigured OpenStack instance

3 Emulation Tool Infrastructure

This Section briefly re-introduces the various sub-components of the Emulation Tool and their main operation.

3.1 Emulation Platform

One of the goals of the THREAT-ARREST project is the development of a model-driven advanced training platform including emulation and simulation capabilities, together with the provision of a simple interface managing visualization of the state of the VMs and the possibility to set and offer interactive games. The aim is to prepare stakeholders to defend high-risk cyber systems and organizations and contrast advanced, known and novel cyber-attacks.

A CTTP model includes the possibility to define the structures needed to drive the training process and the controls to monitor the current state of the cyber system together with security assurance mechanisms to assess the relevance of training. The full CTTP model and the language for its description has been initially addressed in "D3.1 – CTTP Models and Programmes Specification Language" (THREAT-ARREST, 2019d), where the requirements for the definition of the language and the specification of the basic constructs have been presented.

The platform and the initial technical description of the design and the development of the functionalities of the Emulation Tool that support the generation of the emulated components has been described in D2.1 (THREAT-ARREST, 2019a) and D2.3 (THREAT-ARREST, 2019b), that mainly deal with the definition of the virtual networks.

In the project, a cyber-system in a typical scenario is described as a network of information systems involving nodes and routers where the basic applications and services are deployed. Then, the platform will instantiate a set of VMs that are automatically created and configured starting from the CTTP model file defined for the specific training scenario. In the following subsection, we will report a synthetic description of the Emulation Tool infrastructure presented in D2.1 (THREAT-ARREST, 2019a).

3.2 Emulation Tool Infrastructure

The Emulation Tool infrastructure is depicted in Figure 4, where the main modules are also reported.

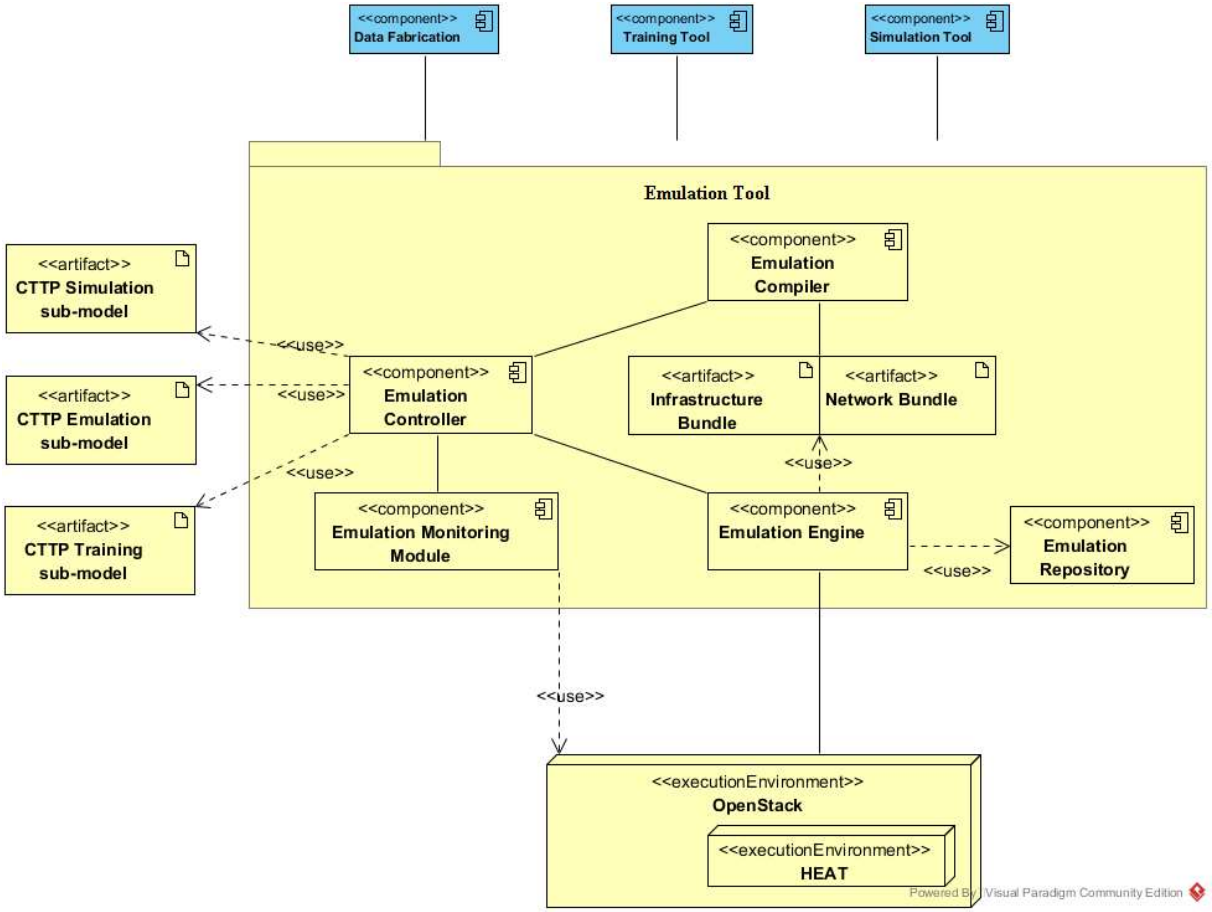

Figure 4 Emulation Tool Infrastructure

The infrastructure relies on:

- the **Emulation Controller** that provides the interface with the other tools of the THREAT-ARREST framework in the form of Representational State Transfer (REST) APIs,
- the **Emulation Compiler** that receives in input the CTTP Emulation Sub-model, describing the training scenario to deploy, and create the YAML file to be used for deploying the virtual machines and networks inside OpenStack,
- the **Emulation Engine** that executes the YAML file, provided by the emulation Compiler, in OpenStack through its orchestrator HEAT (OpenStack, 2019c),
- and the **Emulation Monitoring Module** that provides the monitoring interface described in this deliverable.

The emulation platform has been implemented relying on the OpenStack (OpenStack, 2019a) cloud computing platform, that is an open source collection of software tools providing the capability of controlling processing, storage, and networking resources, accessible through RESTful APIs and using different programming languages, such as Java, Python and many others.

3.3 OpenStack

OpenStack is an open source software for creating private and public clouds. The OpenStack software controls large pools of computational, storage, and networking resources throughout a datacenter, managed through a dashboard or via the OpenStack API. OpenStack works with

popular enterprise and open source technologies making it ideal for heterogeneous infrastructure.

Beyond the standard Infrastructure-as-a-Service (IaaS) functionality, additional components provide, among other, orchestration, service and fault management to ensure high availability of user applications.

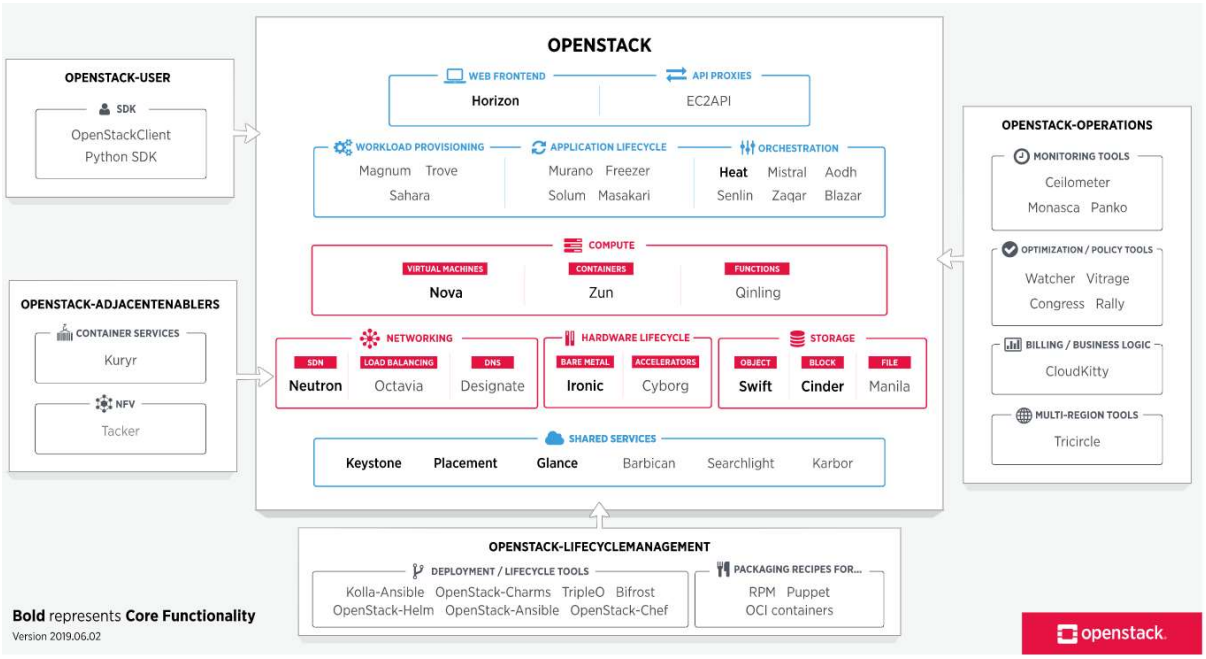

Figure 5 OpenStack Core Functionalities

The modular composition of OpenStack allows its deployment in different infrastructures and technical requirements. Different projects are developed independently to manage different aspects of Cloud management, and they integrate together into a single product. Figure 5 depicts the projects stack that composes the overall infrastructure.

In particular, the main projects that compose a basic OpenStack infrastructure are the following (OpenStack, 2019b):

- **Nova**: it implements services and associated libraries to provide massively scalable, on demand, self-service access to compute resources, including bare metal, virtual machines, and containers (OpenStack, 2019d).
- **Neutron**: it is a Software-Defined Networking project focused on delivering Networking-as-a-Service (NaaS) in virtual computing environments. The exploitation of Neutron as provider of the scenario's virtual network will be provided in deliverable D2.3 (THREAT-ARREST, 2019b).
- **Horizon**: it is the canonical implementation of OpenStack's dashboard, extensible and providing a web-based user interface to most OpenStack services.
- **Keystone**: it is the OpenStack service that provides API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStack's Identity API. It supports LDAP, OAuth, OpenID Connect, SAML and SQL.
- **Swift**: it is a highly available, distributed, eventually consistent object/blob store. Organizations can use Swift to store lots of data efficiently, safely, and cheaply. It is built for scaling and optimized for durability, availability, and concurrency across the entire data set. Swift is ideal for storing unstructured data that can grow without bound.
- **HEAT**: this module orchestrates the infrastructure resources for a cloud application based on templates in the form of text files that can be treated like code. HEAT provides

both an OpenStack-native REST API and a Cloud Formation-compatible Query API. It also provides an auto-scaling service that integrates with the OpenStack Telemetry services, so it is possible to include a scaling group as a resource in a template (OpenStack, 2019c).

OpenStack has been chosen as reference architecture for the THREAT-ARREST emulation environment due to its widespread acceptance and its recognized leadership in the market of open source cloud management infrastructures. Furthermore, the use of HEAT as orchestrator has made possible the straightforward adoption of the model-driven deployment for the virtual environment basing on the CTTP Model described in D3.1 (THREAT-ARREST, 2019d).

4 Emulated Components Monitor

In order to effectively control and utilize THREAT-ARREST platform resources and to support advanced use-cases, it is required first and foremost to get accurate readings of the platform's hardware and virtual resources and make them available for both Dashboard and the Training Tool modules, presenting the readings to the platform users for monitoring and taking configuration decisions and corrective actions.

4.1 High-Level Architecture

Emulation Components Monitor is a collection of collaborative services aimed to satisfy such a requirement. Figure 6 presents a high-level architecture of the Monitoring sub-system along with its major components.

The Monitor will report the state of the emulated components and the effects of the training actions on the system.

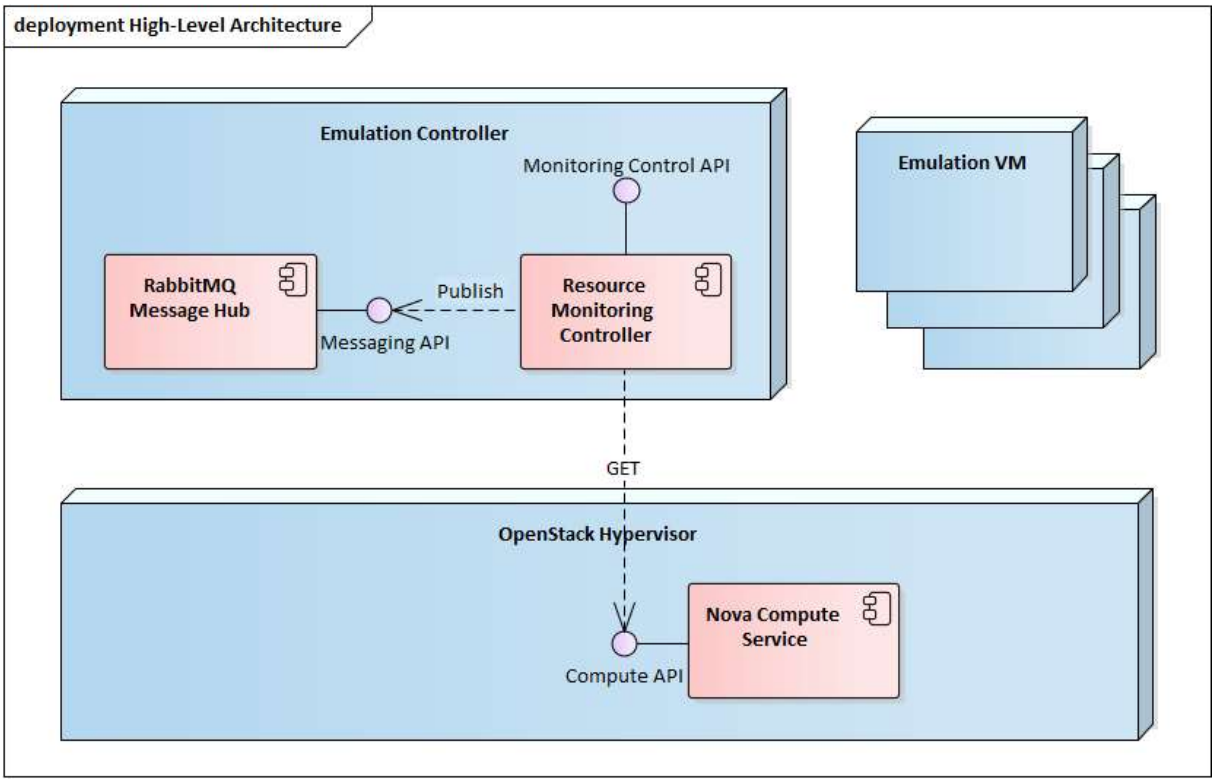

Figure 6 Emulated Components Monitor Architecture

- **Nova Compute Service** is a cloud computing fabric controller, which is the main part of an IaaS system. It is designed to manage and automate pools of computer resources and can work with widely available virtualization technologies, as well as bare metal and high-performance computing (HPC) configurations. One of the Nova Compute Service capabilities is the ability to provide information about hardware and virtual resource configuration and utilization, hence it is used as a backbone technology for resource monitoring in the THREAT-ARREST Emulation Environment (OpenStack, 2019d).
- **RabbitMQ Message Hub** is lightweight and easy to deploy on premises and in the cloud. It supports multiple messaging protocols. RabbitMQ can be deployed in distributed and federated configurations to meet high-scale, high-availability

requirements. THREAT-ARREST Platform leverages RabbitMQ Messaging Hub for asynchronous communication between Platform's components. In particular, it is used as a communication channel to provide THREAT-ARREST Emulation Platform resource configuration and utilization information to consumers (RabbitMQ, 2019).

- **Resource Monitoring Controller** is an active component responsible for fetching information about the THREAT-ARREST Emulation Platform resource configuration and utilization and delivering this information to consumers.

4.2 Operational Flow

As shown on Figure 7, the Resource Monitoring Controller periodically communicates to the Nova Compute Service, collects resource utilization info through a series of GET-calls to the Service's RESTful API, packs the information into a JavaScript Object Notation (JSON) formatted message payload, and finally publishes the message to the RabbitMQ Message Hub to be consumed by the rest of the THREAT-ARREST system. The exact format of the published messages will be finalized later, as a part in the deliverable "D2.6 – Interlinking of emulated components module v2" of the task "T2.3 – Interlinking of emulated components".

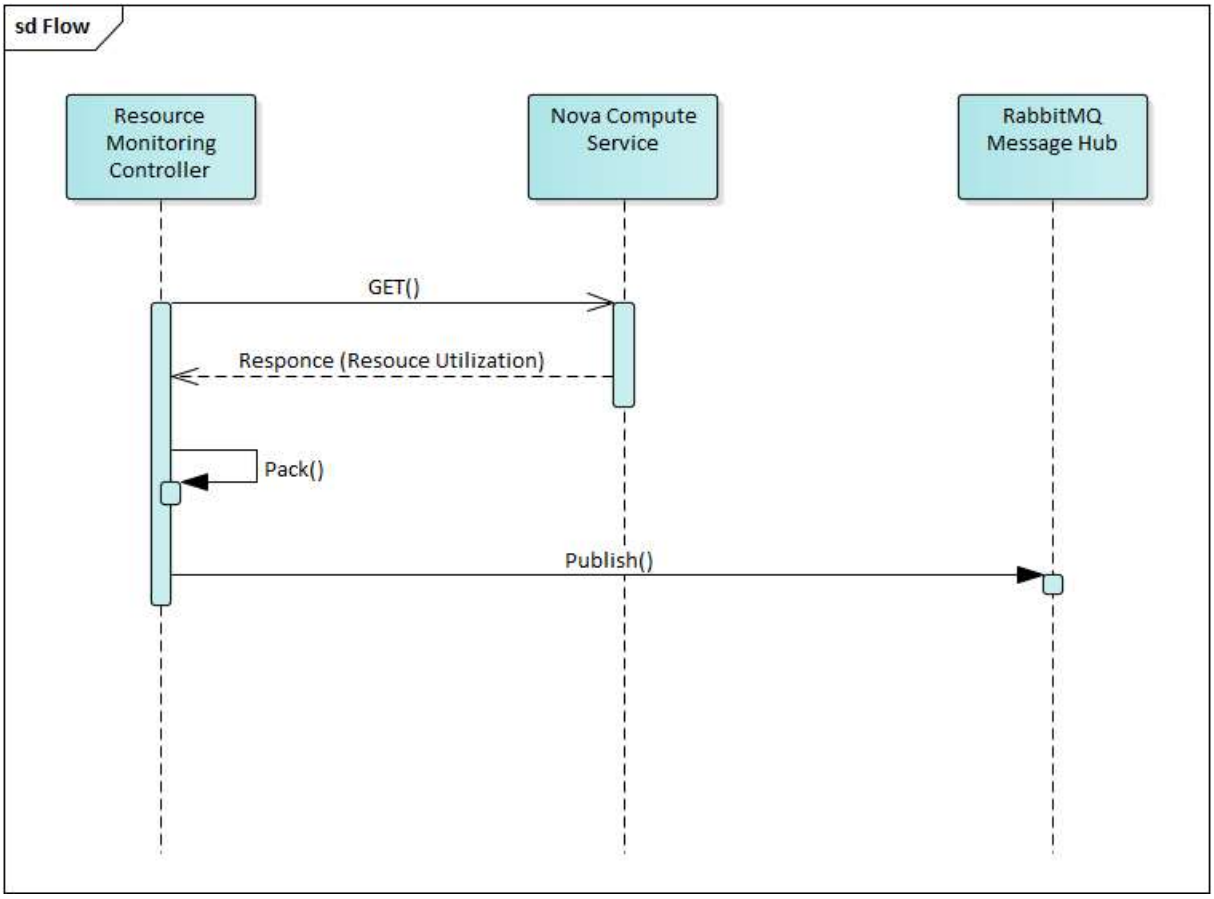

Figure 7 Operational Flow

4.3 Implementation

The Resource Monitoring Controller is implemented as a Java-coded Web-Service, operated under Apache Tomcat Web Server. It exposes RESTful APIs to fetch resource configuration and utilization information (shown on Figure 8), and to control the monitoring operations (shown on Figure 11).

```
1  @Path("/monitor")
2  public class Monitor {
3
4      // Returns a list of VMs along with their configuration
5      @GET
6      @Path("/vms")
7      @Produces(MediaType.APPLICATION_JSON)
8      public Response getVMs() {
9          ...
10     }
11
12     // Returns resource utilization info of a specific VM
13     @GET
14     @Path("/vms/{vm}")
15     @Produces(MediaType.APPLICATION_JSON)
16     public Response getVM(@PathParam("vm") String vm) {
17         ...
18     }
19 }
20
```

Figure 8 Resource Monitor API

- **/monitor/vms** – this GET API call returns a list of Virtual Machines along with their configuration. An example snippet returned by Nova Compute Service in response to such a call is presented on Figure 9. A payload of such a response contains important properties like VM ID, name, IP address and much more.
- **/monitor/vms{vm}** – this GET API call returns resource utilization information of a specific Virtual Machine. An example snippet returned by Nova Compute Service in response to such a call is presented on Figure 10. A payload of such a response contains important information of CPU, memory and storage utilization and failures.

```
{
  "servers": [
    {
      ...
      "addresses": {
        "provider": [
          {
            ...
            "addr": "10.126.111.3",
            ...
          }
        ]
      },
      ...
      "id": "d3931b13-ef2a-4182-99b8-3fc50bc1d25e",
      "name": "server2",
      ...
      "os-extended-volumes:volumes_attached": [
        {
          "id": "abe33fb5-144f-47e5-8fb4-c9420cf5e9e8"
        }
      ],
    },
    {
      ...
      "addresses": {
        "provider": [
          {
            ...
            "addr": "10.126.111.10",
            ...
          }
        ]
      },
      ...
      "id": "d87cad2b-a1c2-4656-ae7a-053ce8d9dabd",
      "name": "server1",
      ...
      "os-extended-volumes:volumes_attached": [
        {
          "id": "56772998-279c-4102-a680-7829a2acd04e"
        }
      ],
    }
  ]
}
```


Figure 9 VMs Configuration info snippet

```
{
  "cpu1_time": 636420000000,
  "tap60a50c13-84_rx_drop": 0,
  "tap60a50c13-84_rx": 389099195,
  "vda_write": 3724227584,
  "tap60a50c13-84_rx_errors": 0,
  "cpu0_time": 218840000000,
  "memory-major_fault": 1508,
  "memory-swap_out": 0,
  "memory-available": 2047576,
  "memory-last_update": 1573736870,
  "memory-usable": 1318616,
  "tap60a50c13-84_tx_errors": 0,
  "vda_read": 396185600,
  "cpu3_time": 665230000000,
  "vda_write_req": 78175,
  "memory-actual": 2097152,
  "tap60a50c13-84_tx_drop": 0,
  "memory-swap_in": 0,
  "tap60a50c13-84_rx_packets": 935155,
  "tap60a50c13-84_tx": 237472729,
  "memory-minor_fault": 6398727,
  "tap60a50c13-84_tx_packets": 90624,
  "cpu2_time": 252400000000,
  "memory": 2097152,
  "memory-rss": 2151140,
  "memory-unused": 140936,
  "vda_read_req": 23601,
  "vda_errors": -1
}
```


Figure 10 VM resource utilization and fault info

```
@Path("/monitor/actions")
public class MonitorController {

    // Starts monitoring session
    @POST
    @Path("/start")
    public Response start() {
        ...
    }

    // Stops monitoring session
    @POST
    @Path("/stop")
    public Response stop() {
        ...
    }

    // Pauses monitoring session
    @POST
    @Path("/pause")
    public Response pause() {
        ...
    }

    // Resumes previously paused monitoring session
    @POST
    @Path("/resume")
    public Response resume() {
        ...
    }
}
```

Figure 11 Resource Monitor Control API

- **/monitor/actions/start** – this API call starts a resource monitoring session. As a result, the Monitor Controller starts a periodic worker thread, which performs actions, described in Section 3.2 above.
- **/monitor/actions/stop** – this API call gracefully terminates a previously started monitoring session.
- **/monitor/actions/pause** – this API call pauses a previously started monitoring session.
- **/monitor/actions/resume** – this API call resumes a previously started though paused monitoring session.

5 Conclusions

In this deliverable, we have provided a technical description of the design and development of the THREAT-ARREST *Emulated Components Monitoring (ECM)* module. The deliverable is the outcome of the activities under the task "T2.2 – Emulated components' monitor".

ECM is a collection of collaborative services aimed to provide accurate readings of the platform's hardware and virtual resources and make them available for both automatic resource management and resource visualisation tools. The deployed monitoring module reports the state of the emulated components and the effects of the training actions on the system.

The work has led to the development of a working prototype of the Emulated Components Monitoring Controller that fetches information about the resource configuration and utilization of the emulated counterparts and delivers this information to the platform users. The Monitoring Controller utilizes the OpenStack Nova Compute Service as its backbone technology. The RabbitMQ Message Hub is used for sharing the resource monitoring information to the rest THREAT-ARREST tools.

The exact format of the published messages will be finalized later, as a part in the deliverable "D2.6 – Interlinking of emulated components module v2" of the task "T2.3 – Interlinking of emulated components". The final implementation of the Emulated Components Monitor will be available along with the final version of the Emulation Tool.

References

[1] Alexandris, G., et al., 2018. Blockchains as enablers for auditing cooperative circular economy networks. 23rd IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD 2018), IEEE, Barcelona, Spain, 17-19 September 2018, pp. 1-7.

[2] Antonopoulos A. M. (2018). *Mastering Bitcoin: unlocking digital cryptocurrencies.* O'Reilly, 3rd Edition, pp. 1-410.

[3] Ferrera, E., et al. 2018. IoT European Security and Privacy Projects: Integration, Architectures and Interoperability. CRIStin – SINTEF, Next Generation Internet of Things. Distributed Intelligence at the Edge and Human Machine-to-Machine Cooperation. Book Chapter 7, pp. 207-292.

[4] Hatzivasilis, G., et al., 2018. CloudNet Anti-Malware Engine: GPU-Accelerated Network Monitoring for Cloud Services. International Workshop on Information & Operational Technology (IT & OT) Security Systems (IOSec), Heraklion, Crete, Greece, 13 September 2018, Springer, LNCS, 11398, pp. 122-133.

[5] Hatzivasilis, G., et al., 2019. WARDOG: Awareness detection watchdog for Botnet infection on the host device. IEEE Transactions on Sustainable Computing – Special Issue on Sustainable Information and Forensic Computing, IEEE, vol. 4, pp. 1-15.

[6] Manifavas, C., et al., 2014. DSAPE – Dynamic Security Awareness Program Evaluation. Human Aspects of Information Security, Privacy and Trust (HCI International 2014), 22-27 June, 2014, Creta Maris, Heraklion, Crete, Greece, Springer, LNCS, vol. 8533, pp. 258-269.

[7] OpenStack. (2019a). *OpenStack*. From https://www.openstack.org

[8] OpenStack. (2019b). *OpenStack Services*. From https://www.openstack.org/software/project-navigator/openstack-components

[9] OpenStack. (2019c). *Heat - OpenStack*. Retrieved 2019, from https://wiki.openstack.org/wiki/Heat

[10] OpenStack. (2019d). *Nova - OpenStack*. From https://wiki.openstack.org/wiki/Nova

[11] Papaefstathiou, I., et al., 2014. International Conference on Advanced Technology & Sciences (ICAT 2014), 12-15 August 2014, Antalya, Turkey, pp. 1-9.

[12] RabbitMQ. (2019). *RabbitMQ*. From https://www.rabbitmq.com/

[13] Soultatos, O., et al., 2019. Pattern-Driven Security, Privacy, Dependability and Interoperability Management of IoT Environments. 24th IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD 2019), IEEE, Limassol, Cyprus, 11-13 September 2019, pp. 1-6.

[14] THREAT-ARREST. (2019a). *D2.1 Emulated components generator module v1.* From https://www.threat-arrest.eu/

[15] THREAT-ARREST. (2019b). *D2.3 Interlinking of emulated components module v1.* From https://www.threat-arrest.eu/

[16] THREAT-ARREST. (2019c). *D2.4 – Emulation tool interoperability module v1.* From https://www.threat-arrest.eu/

[17] THREAT-ARREST. (2019d). *D3.1 CTTP Models and Programmes Specification Language.* From https://www.threat-arrest.eu/

[18] Webopedia. (2019). *Cryptomining Malware*. From https://www.webopedia.com/TERM/C/cryptomining-malware.html

[19] Wikipedia. (2019a). *Cryptocurrency*. From https://en.wikipedia.org/wiki/Cryptocurrency

[20] Wikipedia. (2019b). *Denial-of-service attack*. From https://en.wikipedia.org/wiki/Denial-of-service_attack

[21] Zargar S. T., Joshi J., Tipper D. (2013). *A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks.* IEEE Communications Surveys & Tutorials, vol. 15, issue 4, pp. 2046-2069.

[22] Zimba A., et al. (2018). *Crypto mining attacks in information systems: an emerging threat to cyber security.* Journal of Computer Information Systems, Taylor & Francis, pp 1-12.