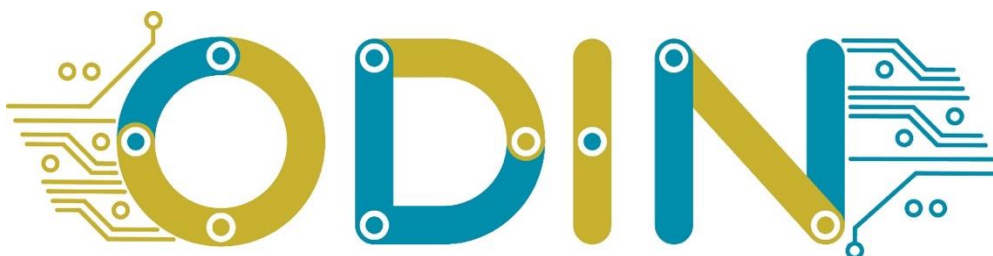


Open-Digital-Industrial and Networking pilot lines using modular components for scalable production

Grant Agreement No : 101017141
Project Acronym : ODIN
Project Start Date : 1st January, 2021
Consortium : UNIVERSITY OF PATRAS – LABORATORY FOR MANUFACTURING SYSTEMS AND AUTOMATION
 FUNDACION TECNALIA RESEARCH & INNOVATION
 KUNGSLIGA TEKNISKA HOEGSKOLAN
 TAMPEREEN KORKEAKOULUSAATIO SR
 COMAU SPA
 PILZ INDUSTRIELEKTRONIK S. L.
 ROBOCEPTION GMBH
 VISUAL COMPONENTS OY
 INTRASOFT INTERNATIONAL SA
 GRUPO S21SEC GESTIÓN, S.A.
 FUNDACION AIC AUTOMOTIVE INTELLIGENCE CENTER FUNDAZIOA
 DGH ROBOTICA, AUTOMATIZACION Y MANTENIMIENTO INDUSTRIAL SA
 PSA AUTOMOBILES S.A.
 AEROTECNIC COMPOSITES SL. U.
 WHIRLPOOL EMEA SPA
 WHIRLPOOL MANAGEMENT EMEA SRL



Title : ODIN Networked Component validation report – final version
Reference : D4.4
Availability : Public
Date : 30/12/2023
Author/s : INTRA, S21SEC
Circulation : EU, Consortium

Summary:

ODIN Networked Component validation report – final version: Final report of ODIN Networked Component validation at pre-industrial scale.

Table of Contents

| | |
|---|----|
| LIST OF FIGURES | 3 |
| LIST OF TABLES | 4 |
| EXECUTIVE SUMMARY | 5 |
| 1. INTRODUCTION | 6 |
| 1.1. Approach..... | 6 |
| 1.2. OpenFlow Deployment and Validation | 7 |
| 1.3. Cyber Security Deployment and Validation | 8 |
| 2. OPENFLOW DEPLOYMENT AND VALIDATION | 9 |
| 2.1. Introduction..... | 9 |
| 2.2. Pre-Industrial Deployment and Validation | 9 |
| 2.2.1. Automotive Pilot Case | 9 |
| 2.2.2. White Goods Pilot Case | 16 |
| 2.2.3. Aeronautics Pilot Case | 21 |
| 3. CYBERSECURITY DEPLOYMENT AND VALIDATION | 25 |
| 3.1. Introduction..... | 25 |
| 3.2. ODIN cybersecurity threat modeling..... | 25 |
| 3.2.1. Attack implementation included in the final version of the cybersecurity solution | 27 |
| 3.3. ODIN cybersecurity incident detection and response final version | 28 |
| 3.3.1. Monitored endpoint..... | 28 |
| 3.3.2. SIEM | 30 |
| 3.3.3. SOAR..... | 33 |
| 4. CONCLUSIONS | 36 |
| 5. GLOSSARY | 37 |
| 6. REFERENCES | 38 |

LIST OF FIGURES

| | |
|--|----|
| Figure 1: Pre-industrial deployment and testing goals | 6 |
| Figure 2: Planning for deployment and testing at pre-industrial scale | 8 |
| Figure 3: ODIN cybersecurity incident and response solution final architecture..... | 8 |
| Figure 4: OpenFlow reference architecture | 9 |
| Figure 5: Automotive Pre-Industrial Setup (Operation 1) | 10 |
| Figure 6: Automotive Pilot Case Validation Product Plans..... | 10 |
| Figure 7: Automotive Pilot Case Validation Resources | 11 |
| Figure 8: Automotive Operation 1 Schedule Task Graph | 13 |
| Figure 9: Automotive Operation 1 – OpenFlow UI - Action Level | 14 |
| Figure 10: Automotive - Tracking and logging COMAU mobile resource location..... | 15 |
| Figure 11: Screwing in motion task diagram | 15 |
| Figure 12: Automotive Pilot Case - Operation 2-Events Logging | 16 |
| Figure 13: White Goods Pre-Industrial Setup picture..... | 16 |
| Figure 14: White Goods Pilot Case Validation Product Plans | 17 |
| Figure 15: White Goods Pilot Case Validation Resources | 17 |
| Figure 16: White Goods Pilot Case M36 Validation Scenario - Actions | 20 |
| Figure 17: White Goods M36 schedule execution UI | 21 |
| Figure 18: Aeronautic Pilot Case Validation Product Plans – OpenFlow UI..... | 22 |
| Figure 19: Aeronautics Pilot Case OpenFlow Validation Schedules | 23 |
| Figure 20: Aeronautics Pilot Case - ERP Connection UI..... | 23 |
| Figure 21: Resource Location Monitoring – Tecnalia Mobile Dual Arm robot..... | 24 |
| Figure 22: ODIN cybersecurity incident and response solution final architecture..... | 25 |
| Figure 23: ODIN Cyber Kill Chain | 26 |
| Figure 24: RosPento execution | 27 |
| Figure 25: Cybersecurity incident and response solution final flow diagram | 28 |
| Figure 26: Evidences of Alert Check and writing in ROS..... | 29 |
| Figure 27: Cybersecurity alert publish in ROS..... | 29 |
| Figure 28: ROSPenTO detection in SIEM: Dashboard | 30 |
| Figure 29: ROSPenTO detection in SIEM: Events..... | 30 |
| Figure 30: ROSPenTO detection in SIEM: Events detail..... | 30 |
| Figure 31: Active response for ROSPenTo attack with SIEM | 31 |
| Figure 32: Evidence of attack in the Iptables of the Monitored Endpoint..... | 31 |
| Figure 33: Evidence of the unsuccessful connection of ROSPenTo | 32 |
| Figure 34: SSH Brute Force detection in SIEM: Dashboard..... | 32 |
| Figure 35: SSH Brute Force detection in SIEM: Events | 32 |
| Figure 36: SSH Brute Force detection in SIEM: Events detail | 33 |
| Figure 37: SSH Brute Force attack alerts in SOAR..... | 33 |
| Figure 38: SSH Brute Force attack alert detail in SOAR | 34 |
| Figure 39: SSH Brute Force attack alert observables in SOAR | 34 |
| Figure 40: SSH Brute Force attack alert responders in SOAR..... | 35 |
| Figure 41: Monitored Endpoint Iptables..... | 35 |

LIST OF TABLES

| | |
|---|----|
| Table 1: Automotive Pilot Case Validated Module Interfaces | 12 |
| Table 2: Automotive Pilot Case – Operation 3 - Validated Module Events..... | 14 |
| Table 3: White Goods Pilot Case Validated Module Interfaces | 18 |
| Table 4: White Goods Pilot Case Validated Module Events | 19 |
| Table 5: Aeronautics Pilot Case Validated Interfaces | 22 |
| Table 6: Aeronautics Pilot Case Validated Module Events..... | 22 |

EXECUTIVE SUMMARY

This is the final report for the ODIN Networked Component validation process that took place in the ODIN Project Task 4.3 “Deployment and testing at pre-industrial scale”.

The validation has been performed in two different stages and in two different versions that are described hereafter.

The two stages of the validation are the following:

1. The **individual prototype validation stage** that assessed the successful connectivity of the ODIN software modules, including mobile manipulators and stationary robots.
2. The **pilot case execution validation stage** where the ODIN architecture has been validated for its ability to dispatch the correct tasks to the correct resources and to undertake the transfer of the resources the OpenFlow orchestrator.

The two versions of the validation are the following:

- a) The **preliminary deployment and testing version** that has been completed and reported in Deliverable D4.2 “ODIN Networked Component report – initial version”.
- b) The **final validation version** that is reported in this document.

This report provides information regarding the final validation version. The final validation has been focused on the second validation stage, in particular the execution of the pilot cases. This was mainly performed in the context of the pre-industrial pilot case setups of ODIN.

The Networked component and the integration with other modules have been validated in pre-industrial settings. For each one pilot cases, namely the Automotive, White Goods and Aeronautics, product plans and production scenarios have been created based on the requirements of the pilot cases developed in WP1 and in collaboration with the responsible partners. Therefore, a preliminary deployment and testing step has been completed in both stages, the individual prototype validation stage and the pilot case execution validation stage.

Finally, the ODIN Networked component has performed initial deployment and validation tests in terms of cyber security, which is an important topic of the modern digital environment. In particular, during this preliminary deployment and testing period validation process, the Cyber-Security module prototype and the OpenFlow module prototype security related functionalities and integration were validated.

In addition, specific integration points between the two modules were developed and tested that hardened the security of the OpenFlow module, by allowing the Cyber-Security module to detect potential security issues that affected the OpenFlow operation and also allowed the OpenFlow to orchestrate shopfloor related cybersecurity responses.

1. INTRODUCTION

1.1. Approach

The goal of Task 4.3 “Pre-industrial deployment and testing” is the validation of the Networked Component and the OpenFlow Architecture in terms of robustness, security and performance by the deployment in pre-industrial scale. These goals are depicted graphically in Figure 1. The pre-industrial deployment and testing is executed in two steps and each step covers two stages that will be executed in two different versions. Each of the two different versions include both stages.

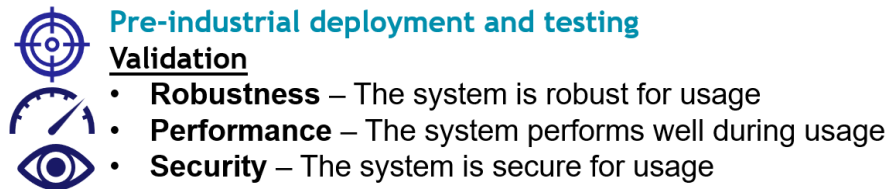


Figure 1: Pre-industrial deployment and testing goals

The two stages of the validation are the following:

1. The **individual prototype validation stage** that assessed the successful connectivity of the ODIN software modules, including mobile manipulators and stationary robots.
2. The **pilot case execution validation stage** where the ODIN architecture has been validated for its ability to dispatch the correct tasks to the correct resources and to undertake the transfer of the OpenFlow orchestrator’s resources.

The two versions of the validation are:

- a) The **preliminary deployment and testing version** that has been completed and reported in Deliverable D4.2 “ODIN Networked Component report – initial version”.
- b) The **final validation version** that is reported in this document.

The initial validation approach that was followed for ODIN, which was first reported in D4.2, is based on the ODIN software architecture and the following principles:

1. Scenario Definition:

- Pre-industrial HRC scenarios are defined for each pilot case, Automotive, White Goods, and Aeronautics. The definition of the pre-industrial and industrial scenarios is an important step for the exploitation of the innovative ODIN architecture. The defined scenarios and their processes take advantage of the innovations in the ODIN architecture and are not bounded by the limitations of conventional approaches. There is a close relationship between machines and processes, since the capabilities and limitations of a process often depend on the design and operation of the machine being used [1].

2. Emulation Process:

- OpenFlow's emulation engine simulates network resources needed for production schedules, enabling task and action execution without physical hardware.
- Emulated interfaces, compatible with ROS middleware, are managed by OpenFlow through the ActionLib protocol, allowing for faster validation and interface development.

3. Pre-industrial Pilots Validation Process:

- Execution of the operations that will be carried out in the industrial environment but executed in a controlled environment.
- The correctness of operations and its robustness is verified in close collaboration with the pilot case leaders and the end users. In addition, it is supported by the OpenFlow Collection Engine and Error Reports Engine for collecting and logging metrics and errors.
- Metrics validation involves capturing data during schedule execution, displaying it in the OpenFlow UI, and providing aggregate statistics for optimization decisions.

- Error Report validation captures and stores errors in the OpenFlow Knowledge Repository, aiding in tracking and addressing issues for a successful schedule completion.
- 4. Cybersecurity Integration:**
- OpenFlow responds to cybersecurity events by stopping robot resources, notifying the operator through the AR application of ODIN presented in D2.5, and projecting information for resolution.
 - Event recovery involves the operator addressing the safety event, requesting a reset, and triggering the Safety module to resume production execution.

Scaling the validation process to a larger industrial context and deploying it in industrially generated pilot cases run in pre - industrial conditions significantly enhances its efficacy. In a larger scale, the comprehensive validation steps provide a robust framework for ensuring seamless cohesion between hardware/software modules and OpenFlow resources in highly complex industrial environments. This validation considered human robot collaboration combined with robot's mobility as the major enablers for flexibility in current assembly systems. Mobile Robot Platforms (MRPs) that can navigate in different workstations for performing various assembly operations, may highly contribute in the reconfigurability of the production system both in structural as well as process level. A key requirement for enabling the efficiency of these hybrid systems in the effective interaction between the different kind of resources [2].

The results validation process, with its metrics collection and error reporting functionalities, becomes crucial for managing the intricacies of numerous tasks and actions concurrently. Additionally, in the advanced pilot cases, the real-life execution of scenarios in specifically designed environments ensures a more accurate reflection of potential challenges and successes in a production setting. This comprehensive approach not only optimizes the efficiency and performance of the systems but also bolsters the adaptability of the validation process to the diverse demands of large-scale industrial applications.

1.2. OpenFlow Deployment and Validation

The Automotive, White goods and Aeronautics use cases have been performed in a pre-industrial scale in this final validation phase and the OpenFlow framework has collected validation data with its functionalities, namely the Metrics Collection Engine and the Error Reports Engine. The data that were collected are actual data from the real resources and actors that performed the pilot cases, providing much more valid data than emulations. The validation process during pre-industrial execution now involves the utilization of more advanced modules for both software and hardware, representing a significant progression from the initial stages. The entire system is physically present and observable during this phase. This enhanced prototype showcases a more refined and sophisticated version of the modules intended for use in the pilot cases. The validation setup of the real scenarios is explained in D4.2, while the validation procedure during the performance of those scenarios is explained in detail in section 2. The results from the final validation are positive and satisfactory. As is shown in Figure 2, the output of the validation will be used to define conditions and improve the final industrial application of the ODIN solution.

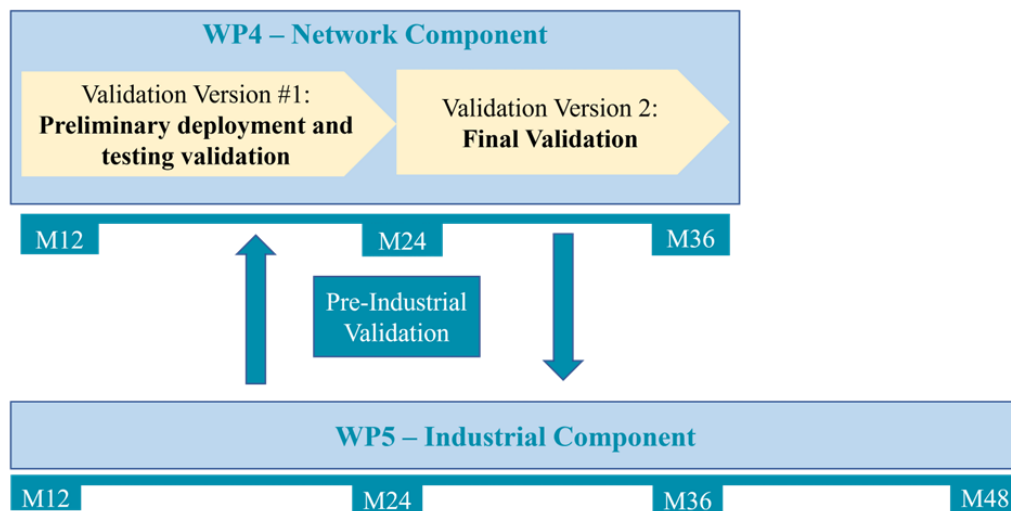


Figure 2: Planning for deployment and testing at pre-industrial scale

1.3. Cyber Security Deployment and Validation

The Cyber Security Deployment was also performed by using docker containers. The validation of the Cyber Security module was performed in both the LMS and TECNALIA premises in the context of the pre-industrial pilot cases.

The development of the ODIN Cyber Security solution and its validation in pre-industrial demonstrators followed two key phases. The first phase was the solution's design. In this phase the cybersecurity threat modelling was conducted specifically for the OpenFlow component. This modelling proved instrumental in pinpointing potential attacks on this component, facilitating the development of a cyber kill chain.

Once the cyber kill chain had been established, it was leveraged in the second phase in order to devise and implement the module for cybersecurity incident detection and response. The primary goal of this module is to fortify the OpenFlow component against identified attacks.

In particular, the components that make up this module have been described in deliverable D4.3 and can also be seen in the Figure 3.

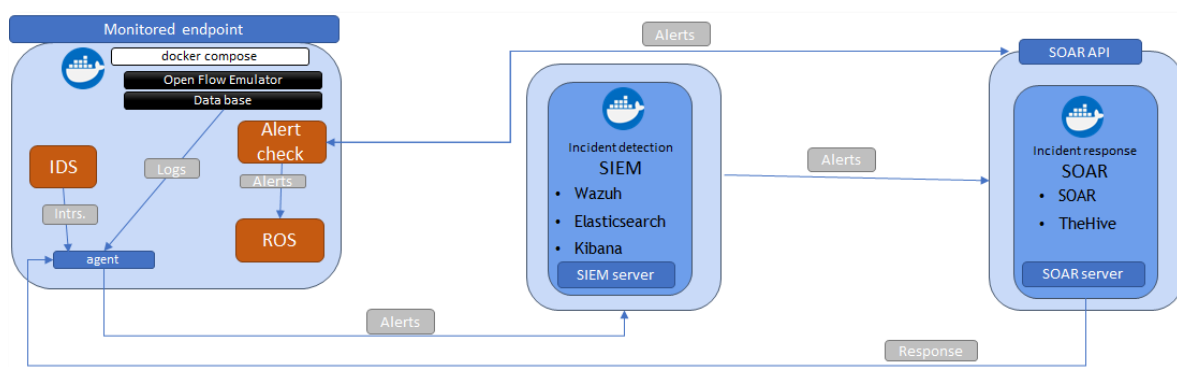


Figure 3: ODIN cybersecurity incident and response solution final architecture

2. OPENFLOW DEPLOYMENT AND VALIDATION

2.1. Introduction

The validation of the OpenFlow final prototype aims to validate the robustness and performance of the OpenFlow in specific pre-industrial HRC scenarios that are defined for each pilot case of ODIN.

The defined scenarios and their processes take advantage of the innovations and utilize the functionalities of the OpenFlow and the ODIN architecture. The scenarios are closely defined based on the ODIN final demonstrators' setup that will be implemented in WP5 "ODIN Industrial Component for robust Large scale Pilot Lines" as presented in deliverable D5.4. However, validated the software and hardware modules, functionalities and software interfaces will be in the largest part the same with the ones used in the industrial pilot lines. Therefore, the validation process has also validated at large the industrial setups of the pilot cases. The integrated system of each ODIN pilot case follow the OpenFlow reference architecture shown in Figure 4.

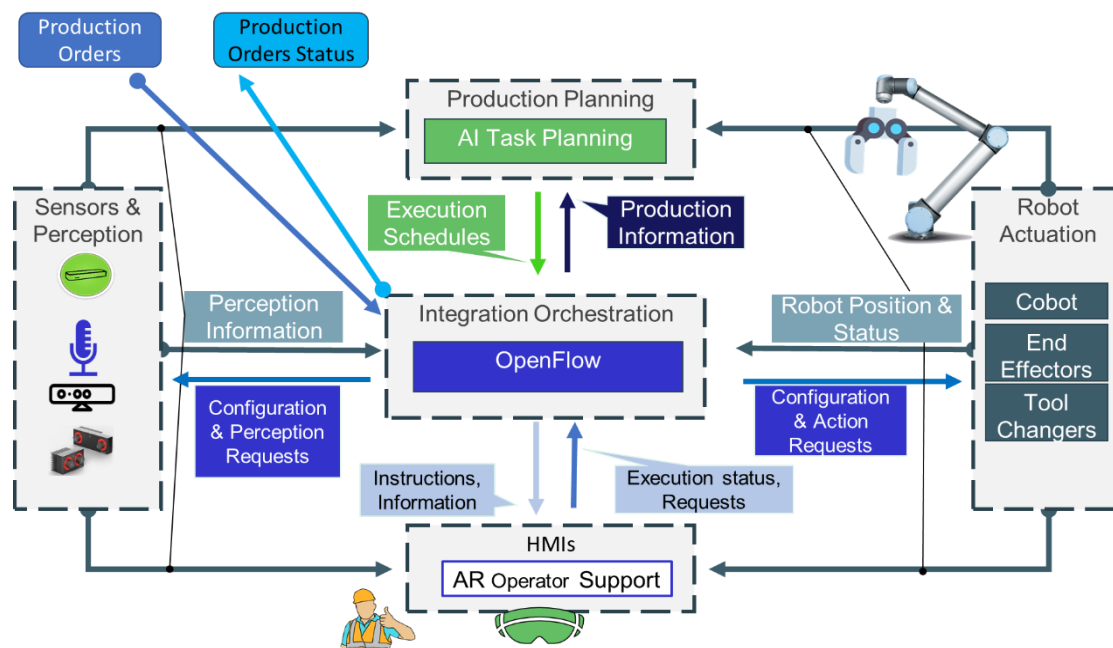


Figure 4: OpenFlow reference architecture

2.2. Pre-Industrial Deployment and Validation

This section presents the pre-industrial deployment and validation efforts on the perspective of the OpenFlow module. These validation efforts also included the validation of other ODIN software and hardware modules.

2.2.1. Automotive Pilot Case

The Automotive pilot case industrial scenario can be divided into three sections:

- Operation 1: Motor & gearbox assembly.
- Operation 2: Additional parts assembly on the motor and gearbox assembly.
- Operation 3: Quality check of installed parts on the motor and gearbox assembly.

A picture of the pre-industrial setup that was also used for the validation can be shown in Figure 5. The pre-industrial setup for the automotive pilot case was located in LMS premises in Patras, Greece and TECNALIA premises in San Sebastian, Spain.



Figure 5: Automotive Pre-Industrial Setup (Operation 1)

2.2.1.1. Validation Scenarios

The validation of the OpenFlow module in the framework of the Automotive Pilot Case was performed on the four dedicated scenarios that have been created for the purposes of the pre-industrial setups. The scenarios are modelled as product plans. There is a product plan for each of the three operations and a product plan for the execution of all three operations in the appropriate sequence. A screenshot of these validation product plans from the OpenFlow User Interface is shown in the following figure.

| # | Name | Plan New Schedule |
|---|----------------------------|-------------------|
| 1 | Automotive M36 | Plan New Schedule |
| 2 | Automotive M36 Operation 1 | Plan New Schedule |
| 3 | Automotive M36 Operation 2 | Plan New Schedule |
| 4 | Automotive M36 Operation 3 | Plan New Schedule |

Figure 6: Automotive Pilot Case Validation Product Plans

2.2.1.2. Manufacturing Resources and Software Modules

The automotive validation scenarios include four different resource types. In particular human operators and three types of robotic resources, namely the AURA cobot, the COMAU mobile robot and the TECNALIA Mobile robot. The manufacturing resources, that have been used to validate the OpenFlow module are shown in Figure 7.

Execution Status Schedules Product Plans **Resources** Statistics

Detected Issues ERP Connection Settings

stellantisuser Sign out

Network Description Digital Description - Resources Digital Description - Standards

Show 25 entries Search:

| ID | Name | Modules | Events |
|----|-----------------|----------------------|----------------------|
| 1 | Operator 2 | View | View |
| 2 | AIC mobile | View | View |
| 3 | TECNALIA mobile | View | View |
| 4 | AURA | View | View |
| 5 | COMAU Mobile | View | View |
| 6 | Operator 1 | View | View |

Showing 1 to 6 of 6 entries Previous 1 Next

Figure 7: Automotive Pilot Case Validation Resources

In addition to the Networked Component's modules, the following modules interfaces have been used and validated in the Automotive Pilot Case pre-industrial setups.

- Cobot API and Mobile Cobot API Module.
 - This module was used to control the COMAU Mobile and the COMAU AURA Cobot. The validation included cartesian and joint based movements, mobile base movements and configuration of the TCP.
- Robot End Effectors API Module.
 - The controlling of the gripper has been validated.
- Easy Robot Programming Tool.
 - In the context of the OpenFlow validation, this module was used to execute skills programmed using the Easy Robot Programming Tool by the TECNALIA Mobile Robot.
- AR Operator Support Module.
 - The request to execute a task and the confirmation of completeness as well as the notification showing to the operator have been validated.
- AI Task Planning Module.
 - The interfaces for process planning functionality of the AI Task Planner has been validated. In particular, the OpenFlow modules provides to the AI Task Planner the lists of pending tasks and available resources and receives an assignment of these resources to tasks.
- Environment and Process Perception Modules
 - These modules have been used and successfully validated for object detection, assembly detection and quality inspection.

The manufacturing resources, software modules and interfaces that have been used to validate the OpenFlow module are listed in Table 1.

Table 1: Automotive Pilot Case Validated Module Interfaces

| Manufacturing Resource | Software Module | Interface | Description | Type |
|------------------------|-----------------------------|--------------------|--|--------|
| COMAU Mobile | Cobot API | Move Arm Joint | Move Arm Joint of cobot mobile | Action |
| | | Move Arm Cartesian | Move Arm Cartesian of cobot mobile | Action |
| | | Mobile Control | Mobile Control | Action |
| Operator | AR Operator Support | Notify Operator | Sends notifications to operators | Action |
| | AR Operator Support | Execute Task | Support Human Task | Action |
| AURA | Cobot API | Move Arm Joint | Move Arm Joint | Action |
| | Robot End Effectors API | Control Gripper | Control Gripper | Action |
| | Cobot API | Move Arm Cartesian | Move Arm Cartesian | Action |
| | Cobot API | Configure TCP | Configure TCP | Action |
| TECNALIA mobile | Easy Robot Programming Tool | Skill Execution | Executes composite skills | Action |
| - | AI Task Planning | Task Planning | Responsible for Assigning tasks to resources | Action |
| - | Environment Perception | Detect Object | Detect Object | Action |
| | Process Perception | Quality Inspection | Quality Inspection | Action |
| | Process Perception | Assembly Detection | Assembly Detection | Action |

2.2.1.3. Validation Schedules

This section provides an overview of key validation results and activities that took place in the validation of each schedule. All OpenFlow schedules mentioned in this section have been composed by the OpenFlow module after getting Task Planning information from the AI Task Planner using the information defined in the Product Plans.

The OpenFlow UI can visualize all schedules in both task level and action level. Figure 8 shows the validation schedule of operation 1 in task level. In particular, it shows the different tasks of picking and placing and connecting correctly the motor and the gearbox.

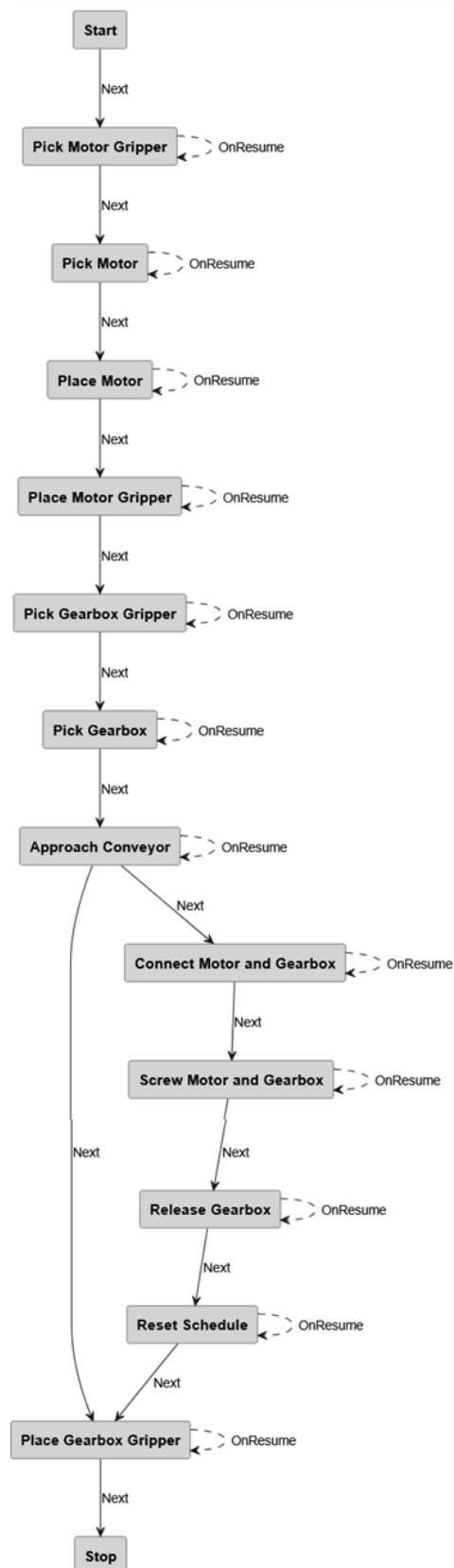


Figure 8: Automotive Operation 1 Schedule Task Graph

The ability of the OpenFlow to execute, control and monitor automotive operation 1 has been validated in the pre-industrial demonstrator at LMS premises. Figure 9 shows a screenshot of the schedule's execution screen of the OpenFlow UI during the execution of Automotive Operation 1.

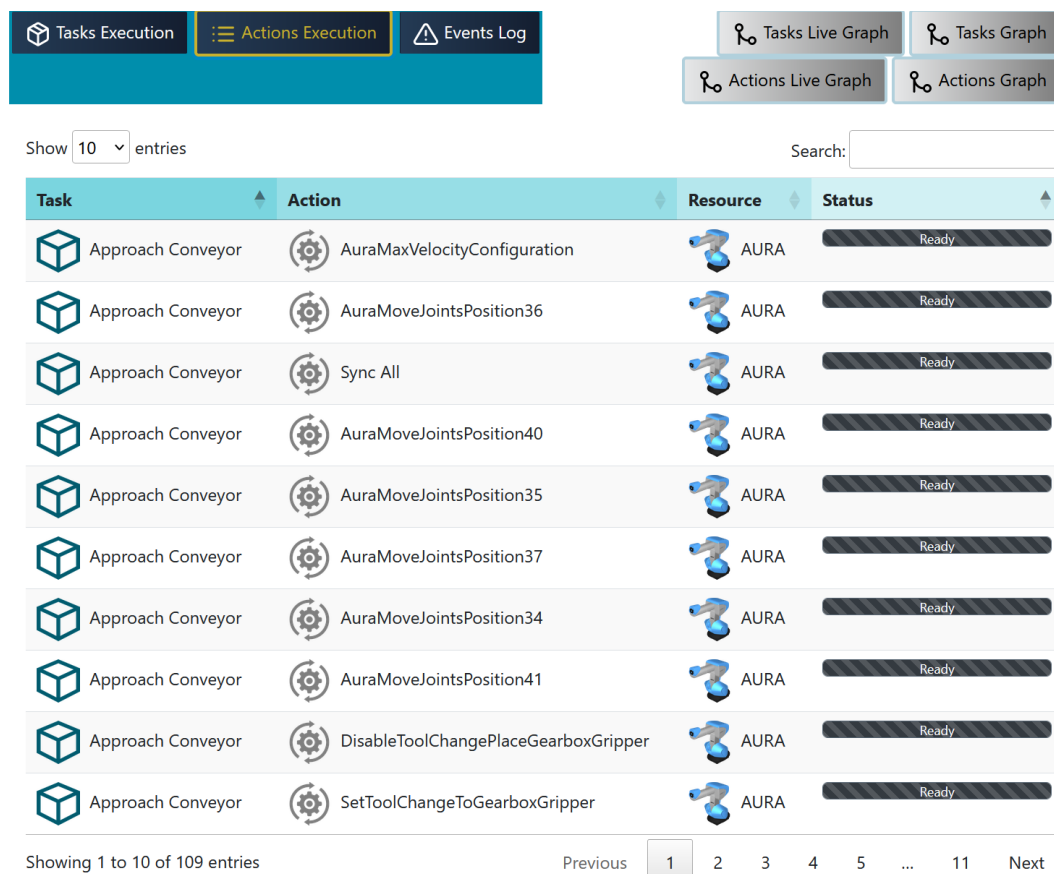


Figure 9: Automotive Operation 1 – OpenFlow UI - Action Level

The OpenFlow event-based functionalities were also validated, especially in operation 3. In particular the OpenFlow receives and processes results from inspection as inspection result events of different types. The OpenFlow processes the results from each event and determines the required corrective actions. At the end of the complete inspection process, the OpenFlow decides on the appropriate correction actions, requests the corrective actions execution and monitors the correction progress. The corrective actions are executed by the operator who can choose the sequence of corrective actions and perform them in any order he/she may choose. Using the appropriate interface, the OpenFlow is notified whenever the operator marks a corrective action as completed. Table 2 summarizes the event types handled in Operation 3 of the automotive pilot case.

Table 2: Automotive Pilot Case – Operation 3 - Validated Module Events

| Schedule | Events | Emitting Module | Description | Handling Module |
|-------------------------|-------------------------------|--------------------|--|-----------------|
| Automotive M36 Schedule | Connector Inspection Results | Quality Inspection | Event tracking the inspection of connectors | OpenFlow |
| | Screw Inspection Results | Quality Inspection | Event tracking the inspection of screws | OpenFlow |
| | Valve Pipe Inspection Results | Quality Inspection | Event tracking the inspection of Valve Pipes | OpenFlow |

In addition to the aforementioned events, OpenFlow events are used to track the position of the COMAU mobile robot. In particular, the OpenFlow can understand and log when a mobile resource departs from or arrives to predefined positions. This functionality comes in addition to the actual position monitoring that is captured from the position reporting of each module. This is shown graphically in an example in Figure 10. In particular, the COMAU Mobile resource constantly publishes its location. In addition to

that, the OpenFlow logs when the COMAU Mobile Robot departs from one station, e.g. Station 1 on another station e.g. Station 2.

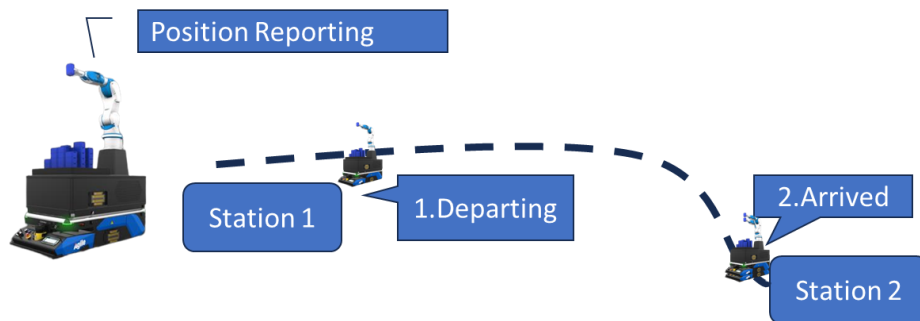


Figure 10: Automotive - Tracking and logging COMAU mobile resource location

In addition to the COMAU Mobile Robot, this functionality has also been validated using the schedule shown in Figure 11 with the TECNALIA Mobile Robot. Each of the tasks in the schedule of Figure 11 corresponds to a skill execution in the easy programming tool module. Furthermore, within each of the aforementioned tasks OpenFlow actions are used to log the departure and arrival of the TECNALIA Mobile Robot, in the “Screwing” and “Home” stations.

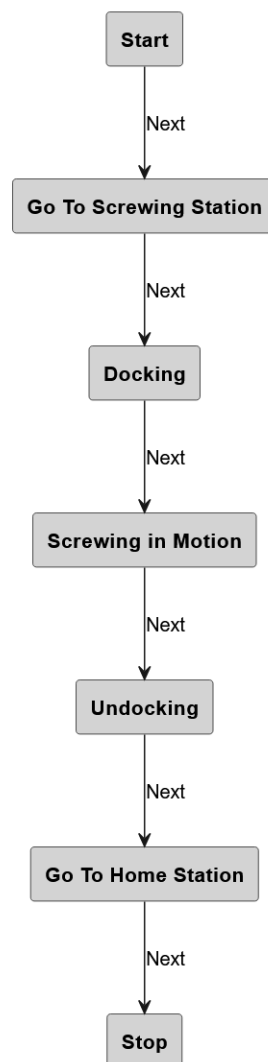


Figure 11: Screwing in motion task diagram

The OpenFlow UI captures the events received during the execution of the schedule and can visualize them in the schedule event logging menu as shown in Figure 12.

| Tasks Execution | Actions Execution | Events Log | Tasks Live Graph | Tasks Graph |
|--------------------------------------|-----------------------------|------------|-----------------------------|----------------------------|
| | | | Actions Live Graph | Actions Graph |
| Search: <input type="text"/> | | | | |
| ID | Name | Type | Handling Action | Time |
| db08c5f8-de24-4863-b25e-b8f45ead2362 | Arrived at Home Station | Arrival | Arrived at Home Station | 2023-12-13, 16:53:02 - 836 |
| 2fc19ebd-2db9-48a0-83f5-2164ab6f78d7 | Moving to Home Station | Departure | Moving to Home Station | 2023-12-13, 16:53:02 - 719 |
| e90639db-cffd-4e4f-8b7b-d92b025b3b40 | Arrived at Screwing Station | Arrival | Arrived at Screwing Station | 2023-12-13, 16:53:02 - 704 |
| ec485d1c-abc9-45be-be9e-62ae8060ea27 | Moving to Screwing Station | Departure | Moving to Screwing Station | 2023-12-13, 16:53:02 - 355 |
| Showing 1 to 4 of 4 entries | | | | |
| | | | Previous | 1 Next |
| Filters Active - 0 | | | Collapse All | Show All Clear All |

Figure 12: Automotive Pilot Case - Operation 2-Events Logging

2.2.2. White Goods Pilot Case

The validation scenarios created for the White Goods pilot case not only aim to validate the functionality of the White Goods pilot case but also include dedicated scenarios for the targeted validation of specific functionalities and integration between modules.

The pre-industrial setup for the White Goods pilot case was located in LMS premises in Patras, Greece. Additional physical pre-industrial validation was performed in TAU premises in Finland, which was focused on the OpenFlow and projector-based modules' integration validation.

Furthermore, additional, targeted integrated validation tests with the Cybersecurity module, the Digital Simulation module and the AI Task Planner were performed using dedicated scenarios.

A picture of the pre-industrial setup at LMS used for modules' validation can be shown in Figure 13.



Figure 13: White Goods Pre-Industrial Setup picture

2.2.2.1. Validation Scenarios

In order to validate the OpenFlow module in the framework of the White Goods pilot case five dedicated scenarios have been created for the purposes of the final validation. The scenarios are modelled as product plans. The White Goods M36 Product plan captures the pre-industrial White Goods operations. White Goods Simulation Demo is product plan created for the validation of the usage of the OpenFlow module in simulation. Finally, the White Goods Static Borders and “Hello World-Projector” product plans model dedicated validation tests that take place in TAU premises. Figure 14 shows a screenshot of the OpenFlow UI, showing the aforementioned product plans.

| # | Name |
|---|-----------------------------|
| 1 | WhiteGoods Simulation Demo |
| 2 | Hello World - Projector |
| 3 | White Goods - Security Demo |
| 4 | WhiteGoods M18 |
| 5 | WhiteGoods M36 |
| 6 | WhiteGoods Static Borders |

Showing 1 to 6 of 6 entries

Figure 14: White Goods Pilot Case Validation Product Plans

2.2.2.2. Manufacturing Resources and Software Modules

The White Goods OpenFlow module validation scenarios include three different resource types. In particular, human operator, the projector interface and a robotic resource, namely the UR10 Cobot. The manufacturing resources, that have been used to validate the OpenFlow module are shown in Figure 15.

| ID | Name | Modules |
|----|---------------------|----------------------|
| 1 | ur10-Cobot | View |
| 2 | Operator | View |
| 3 | Projector Interface | View |

Showing 1 to 3 of 3 entries

Figure 15: White Goods Pilot Case Validation Resources

The White Goods validation involves the interfaces of modules shown in Table 3. As presented in the aforementioned table, the validation covers both integration via the ROS framework as well as through a RESTful API.

Table 3: White Goods Pilot Case Validated Module Interfaces

| Resource | Module | Interface | Description | Type |
|----------------------------|--------------------------|----------------------------------|--|---------|
| Operator | AR Operator Support | Show Notification to Operator | A customizable notification is shown to the operator | Action |
| | | Gesture Control | Operator controls any gesture to interact | Action |
| | | Execute Human Task | The Operator supports human task through execution | Action |
| ur10-Cobot | Robot End Effectors API | Control Gripper | Control the gripper | Action |
| | Cobot API | Move Arm Joint | Move Cobot's Arm Joint | Action |
| | Robot End Effectors API | Changer Control Tool | Changer Control Tool | Action |
| | Cobot API | Configure Tcp | Configure Tcp | Action |
| | Cobot API | Move Arm Cartesian | Move Cobot's arm at cartesian system | Action |
| | OpenFlow | Detect Object | Detect object's position | Action |
| | OpenFlow | Control Schedule Execution | OpenFlow Api controls schedule's execution | Action |
| Projector Interface | Cobot API | Configure Payload | Configure Payload | Action |
| | HMI: Projector Interface | Operator Static Border | Operator Static Border | Service |
| | HMI: Projector Interface | Set Preset UI Projection | Set Preset UI Projection | Action |
| | HMI: Projector Interface | Set Layout Static Borders | Set Layout Static Borders | Service |
| | HMI: Projector Interface | Unset Projection | Clears specified projection | Action |
| | HMI: Projector Interface | Set Indication Projection | Show an indication | Action |
| | HMI: Projector Interface | Set Virtual Button Chance Color | Change color of Web UI | Action |
| | HMI: Projector Interface | Release Robot Static Border | Release Robot Static Border | Service |
| | HMI: Projector Interface | Set Safety Border Projection | Set Safety Border Projection | Action |
| | HMI: Projector Interface | Release Operator Static Border | Release Operator Static Border | Service |
| | HMI: Projector Interface | Set HTML Instructions Projection | Set HTML Instructions Projection | Action |
| | HMI: Projector Interface | Set Instructions Projection | Set Instructions Projection | Action |
| | HMI: Projector Interface | Book Robot Static Border | Book Robot Static Border | Service |
| | HMI: Projector Interface | Set Virtual Buttons Projection | Set Virtual Buttons Projection | Action |
| | HMI: Projector Interface | Set Switch Interface Projection | Set Switch Interface Projection | Action |

| Resource | Module | Interface | Description | Type |
|----------|------------------------------|----------------|--|---------|
| - | AI Task Planning | Task Planning | Responsible for Assigning tasks to resources | Action |
| - | Digital Simulation | Move Joints | Move joints of robot | Topic |
| - | Digital Simulation | Move Cartesian | Move joints Cartesian | Topic |
| - | Digital Resource Description | Get standards | Gets all standards | RESTful |
| - | Digital Resource Description | Get Resources | Gets all resources | RESTful |
| - | Digital Resource Description | Authenticate | Authenticates | RESTful |

The White Goods pilot case also features event driven, reactive logic in order to model reactions to real or digital world events whose occurrence cannot always be foreseen, such as safety or security events, as well as to trigger actions when something expected takes place. The list of events that have been used to validate the OpenFlow event driven, reactive features are shown in Table 4.

Table 4: White Goods Pilot Case Validated Module Events

| Schedule | Events | Emitting Module | Description | Handling Module |
|---|-------------------------------|--------------------------|---|-----------------|
| WhiteGoods M36 | Release Gripper Request Event | Robot End Effectors API | Release Gripper Request Event | OpenFlow |
| | Gesture control request event | Gesture Control | Gesture Control Request Event | OpenFlow |
| | Safety Event | Safety Event | Handle Side Safety Violation | OpenFlow |
| | Safety Status Event | Safety Status Event | Handle Safe and Unsafe Status Events | OpenFlow |
| | Cyber Security Event | Security Event | Handle Security Events | OpenFlow |
| | Show Notification | AR Operator Support | Show Notification to Operator Safety Response Event | OpenFlow |
| | Safety Mode Event | Safety System | Handle Safety Mode to Normal Events | OpenFlow |
| | Async Task Completed | AR Operator Support | Execute Human Task Result Published Event | OpenFlow |
| White Goods Simulation Validation Schedule | Move Joints Completed | Cobot API | Move Joints Result Event Action | OpenFlow |
| | Move Joints Started | Cobot API | Move Joints Goal Publish Action | OpenFlow |
| White Goods - Security Validation Schedule | Cyber Security Event | Cybersecurity | Handle Security Events | OpenFlow |
| Hello World - Projector Schedule | Virtual Button Event | HMI: Projector Interface | Handle Virtual Button Events | OpenFlow |

2.2.2.3. Validation Schedules

This section presents an overview of key validation results and activities that took place for the validation of each schedule of the White Goods pilot case. All OpenFlow schedules mentioned in this section have been composed by the OpenFlow module after getting Task Planning information from the AI Task Planner using the information defined in the Product Plans mentioned in section 2.2.2.1.

The pre-industrial validation was performed in the White Goods M36 validation scenario. This scenario was developed in an agile way, adding more functionality as it became available. Progressively the latest versions of software and hardware modules were installed in the pre-industrial setup and controlled by the OpenFlow module. The complexity of the scenario also gradually increased, adding more complex functionalities after the first integration tests.

The resulting OpenFlow schedule is shown in Figure 16 and it is responsible for the orchestration in the White Goods pilot case. The schedule creates a lot of different functionalities and action. In this case an enlarged detailed view of the schedule is shown in the same figure.

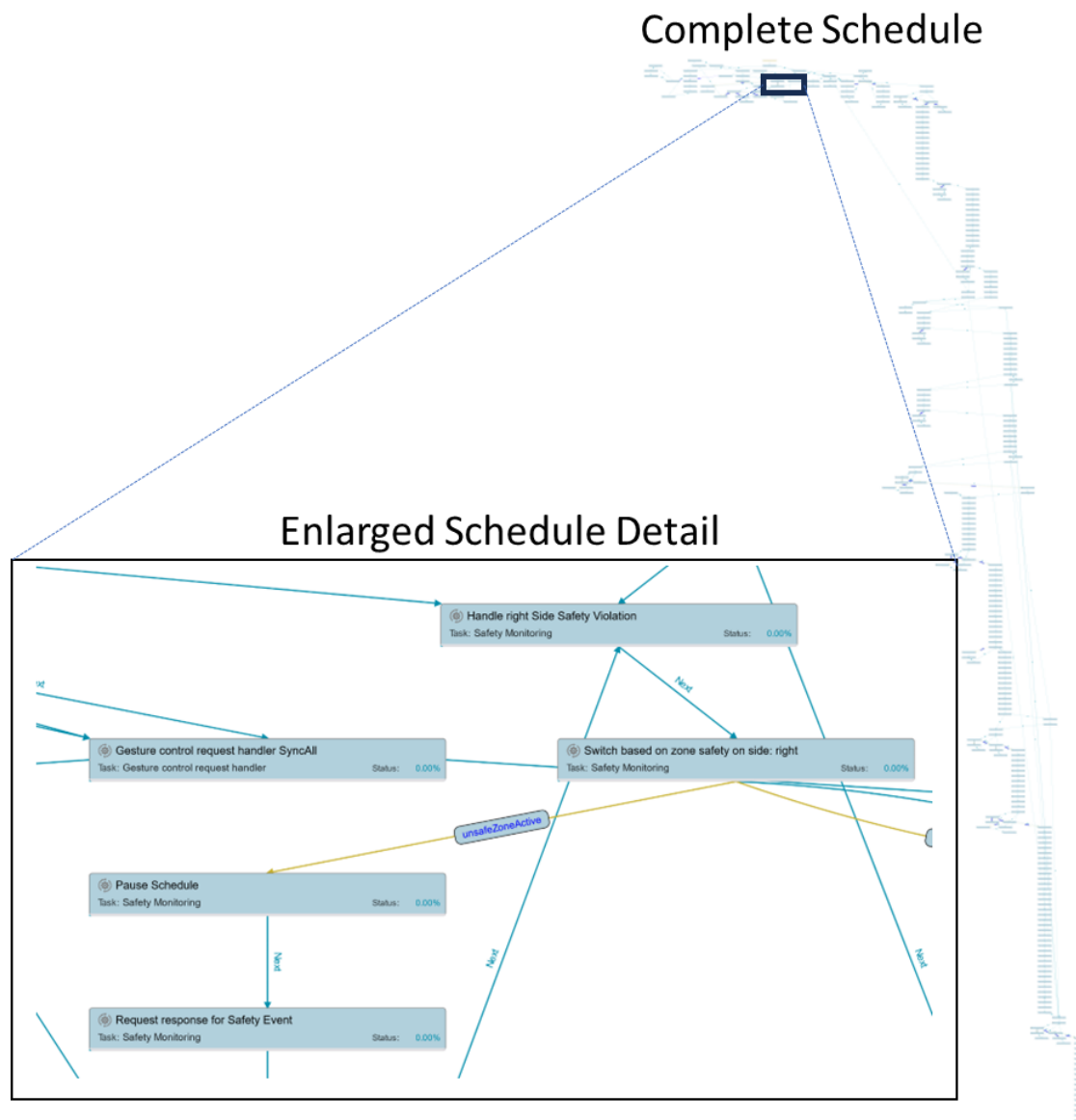


Figure 16: White Goods Pilot Case M36 Validation Scenario - Actions

The schedule uses and therefore validates several system level functionalities that are implemented by OpenFlow and the OpenFlow connection's with other modules. In particular the schedule implements the orchestration and information exchange for the following task types.

- Collaborative Mode Monitoring
- Gesture Control
- Cooktops installation on the cooktop burner
- Knobs installation on the cooktop burner
- Transformer installation in the oven
- Pick parts from kitting table
- Provide parts to the operator
- Remove parts from the kitting table
- Tool changing

All the aforementioned tasks have been implemented through the use of the software interfaces and events mentioned in Table 3 and Table 4. Through the validation process, the OpenFlow orchestrated successfully and multiple times the schedule demonstrating excellent performance and robustness. Figure 17 shows the schedule monitoring and control user interface of the OpenFlow module as shown during the execution of this schedule.

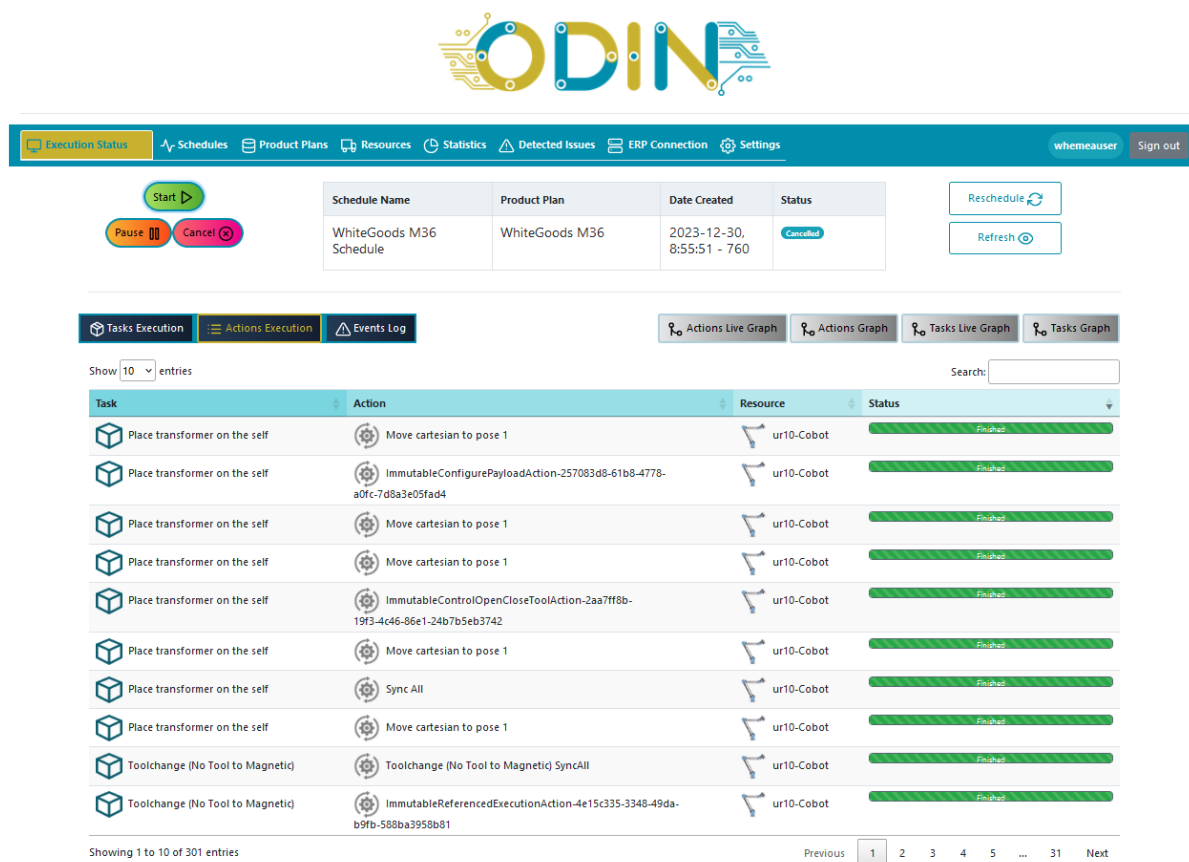


Figure 17: White Goods M36 schedule execution UI

2.2.3. Aeronautics Pilot Case

2.2.3.1. Validation Scenarios

In the Aeronautics pilot case three different scenarios have been created to validate the OpenFlow module final prototype and the OpenFlow architecture. In the Aeronautics pilot case, the OpenFlow module receives incoming orders from ERP systems and determines. The OpenFlow also reports back the execution progress of the orders as well as information about resources, such as their location. In order to validate the functionality required for the Aeronautics pilot case, three different scenarios were created, capturing key functionalities required for the completion of the investigated operations. These scenarios have been modelled as OpenFlow product plans. Figure 18 shows the three validation product plans, as viewed from the OpenFlow UI.

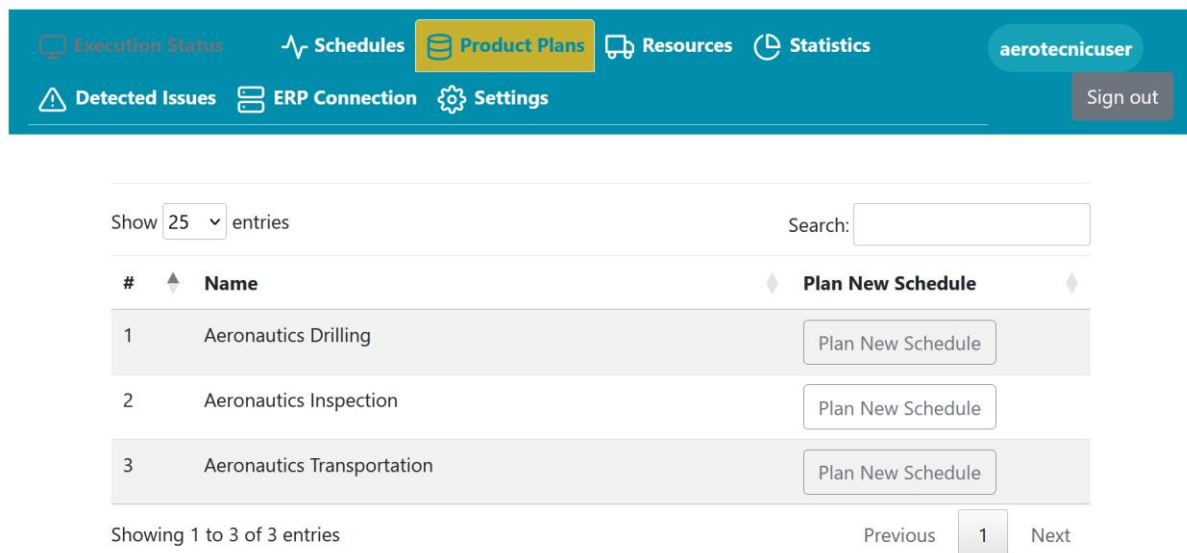


Figure 18: Aeronautic Pilot Case Validation Product Plans – OpenFlow UI

2.2.3.2. Manufacturing Resources and Software Modules

In the Aeronautics pilot case, the OpenFlow communicates and orchestrates the TECNALIA Mobile robot through the versatile Skill Execute interface. The use of the Skill Execution interface to orchestrate the TECNALIA Mobile robot is shown in Table 5.

Table 5: Aeronautics Pilot Case Validated Interfaces

| Resource | Module | Interface | Description | Type |
|------------------------|-----------------------------|-----------------|--|--------|
| TECNALIA mobile | Easy Robot Programming Tool | Skill Execution | Executes a skill serialized in the appropriate skill format. | Action |

In order to track the location of each Mobile Robot in the shopfloor in relation to predefined location the OpenFlow module is capable of tracking Arrival and Departure events as shown in Table 6.

Table 6: Aeronautics Pilot Case Validated Module Events

| Events | Emitting Module | Description | Handling Module |
|-----------|-----------------------------|---|-----------------|
| Departure | Easy Robot Programming Tool | The TECNALIA Mobile Robot starts moving to a specific station | Open Flow |
| Arrival | Easy Robot Programming Tool | The TECNALIA Mobile Robot has arrived at a specific station | Open Flow |

The actions and events of Table 5 and Table 6 are combined in the three OpenFlow schedules. These schedules' overview is shown in Figure 19. In the aeronautics pilot case, the OpenFlow is required to operate in a higher-level orchestration while the Easy Robot Programming Tool is responsible for the programming and execution of each skill. Despite the fact that this synergy is validated with a single robot, it offers a lot benefits when multiple mobile robots are present, allowing for more autonomy on the individual resources.

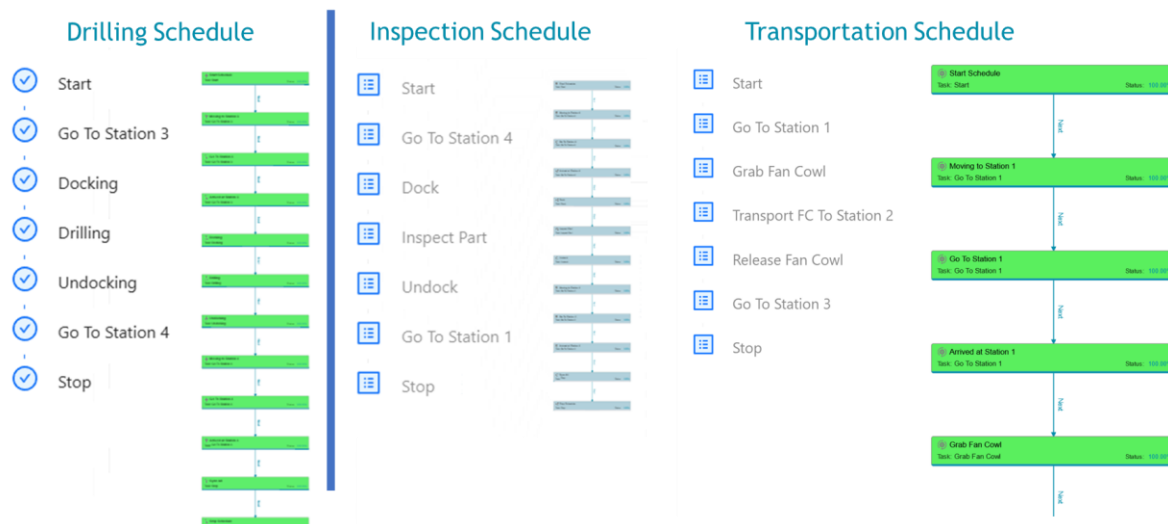


Figure 19: Aeronautics Pilot Case OpenFlow Validation Schedules

2.2.3.3. Validation Schedules

For the Aeronautics pilot case validation, the execution of a schedule is performed as a response to an incoming order. During validation, the orders that were used corresponded to the existing product plans. Figure 20 shows the OpenFlow ERP Connection user interface that allows the user to see the incoming orders and select an order to fulfil. For each order the related product plans can be shown and the user can request to generate an OpenFlow schedule to fulfil the order.



Incoming Orders

| | Product Name | Quantity | Progress | Status |
|-------------------------------------|----------------------------|----------|----------|---------|
| <input checked="" type="checkbox"/> | Aeronautics Transportation | 2 | 0% | PENDING |
| <input type="checkbox"/> | Aeronautics Drilling | 1 | 0% | PENDING |
| <input type="checkbox"/> | Aeronautics Inspection | 1 | 0% | PENDING |

Select product from Incoming Orders

Available Product Plans

| ID | Product Plan | Description | Plan New Schedule |
|--------------------------------------|----------------------------|---------------------------------------|------------------------------------|
| 24cbf1d6-1bda-459a-98a4-60f926213899 | Aeronautics Transportation | Aeronautics Transportation Validation | <button>Generate Schedule</button> |

Figure 20: Aeronautics Pilot Case - ERP Connection UI

During the execution of the OpenFlow manufacturing schedule, that aims to fulfil a particular order, the OpenFlow module tracks not only the progress of all actions and tasks but also the location of the mobile resources. The OpenFlow module can also forward this information to other systems if needed. This has been validated in all current aeronautic scenarios and is shown graphically in Figure 21. The OpenFlow can not only capture the position reported from mobile resources, such as the TECNALIA mobile dual arm robot but can also track and log the arrival and departure of the TECNALIA mobile dual arm robot in specific, predefined locations.

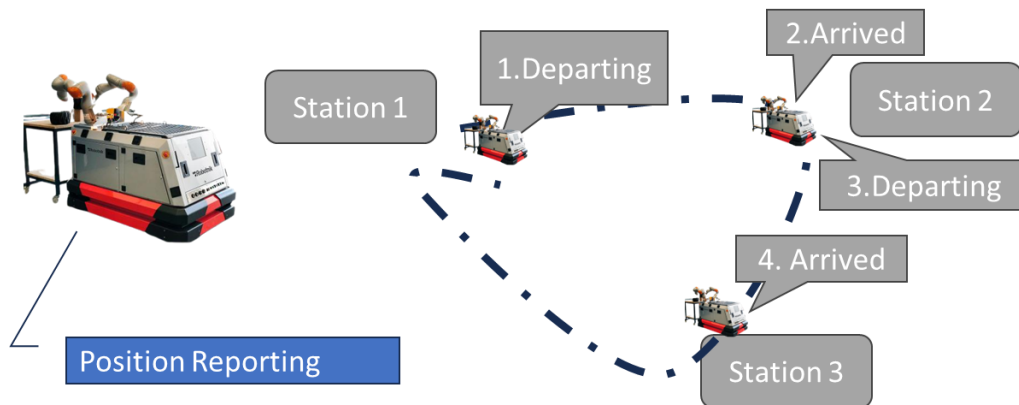


Figure 21: Resource Location Monitoring – Tecnalia Mobile Dual Arm robot

For instance, Figure 21 shows how moving from Station 1 to Station 2 and then Station 3 is understood by the OpenFlow. From the one hand there can be the constant position monitoring of the mobile resource, that can track each reported location of the mobile resource as a stream of location events. In parallel, the OpenFlow through the schedule understands and logs when the resource has arrived or is departing from a specific station.

3. CYBERSECURITY DEPLOYMENT AND VALIDATION

3.1. Introduction

This section describes the final version of the development of the cybersecurity module that has been developed in the ODIN project.

Specifically, this deliverable presents the functioning of this module once it has been installed in the LMS and TECNALIA premises in the pre-industrial scenarios.

In the development of the final cyber security solution, the project can be divided into two phases. On the one hand, in the first phase of the solution design, cybersecurity threat modelling was performed for the OpenFlow component presented in the previous sections. This modelling helped us to identify some attacks that can be made on this component and thus develop a cyber kill chain.

Once the cyber kill chain was defined, the cyber kill chain was used to design and implement the cyber security incident detection and response module. The objective of this is to be able to defend the OpenFlow component from detected attacks. In particular, the components that make up this module have been described in deliverable D4.3 and can also be seen in the Figure 22.

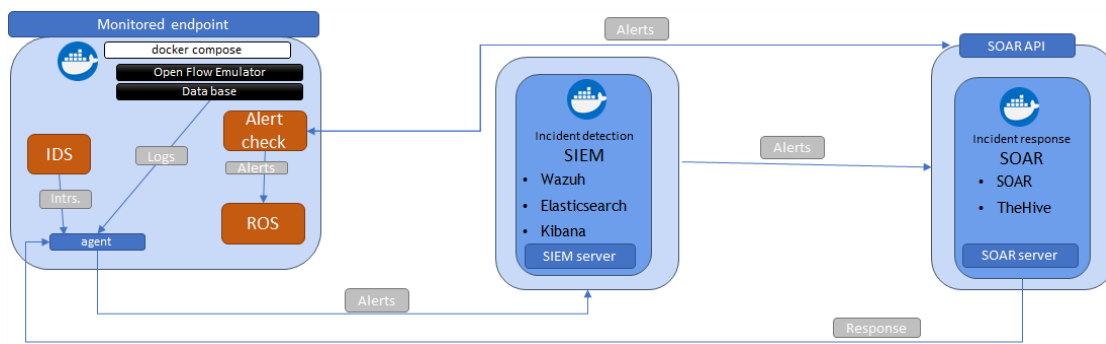


Figure 22: ODIN cybersecurity incident and response solution final architecture

3.2. ODIN cybersecurity threat modeling

The cyber kill chain is a series of steps that trace stages of a cyberattack from the early reconnaissance stages to the exfiltration of data. The kill chain helps us understand and combat ransomware, security breaches, and advanced persistent attacks (APTs).

There are several core stages in the cyber kill chain. They range from reconnaissance (often the first stage in a malware attack) to lateral movement (moving laterally throughout the network to get access to more data) and data exfiltration (getting the data out). All of the common attack vectors – whether phishing or brute force or the latest strain of malware – trigger activity on the cyber kill chain.

Each stage is related to a certain type of activity in a cyber-attack, regardless of whether it's an internal or external attack:

- Reconnaissance

The observation stage: attackers typically assess the situation from the outside-in, in order to identify both targets and tactics for the attack.

- Intrusion

Based on what the attackers discovered in the reconnaissance phase, they're able to get into your systems: often leveraging malware or security vulnerabilities.

- Exploitation

The act of exploiting vulnerabilities, and delivering malicious code onto the system, in order to get a better foothold.

- Privilege Escalation

Attackers often need more privileges on a system to get access to more data and permissions: for this, they need to escalate their privileges often to an Admin.

- Lateral Movement

Once they're in the system, attackers can move laterally to other systems and accounts in order to gain more leverage: whether that's higher permissions, more data, or greater access to systems.

- Obfuscation / Anti-forensics

In order to successfully pull off a cyberattack, attackers need to cover their tracks, and in this stage, they often lay false trails, compromise data, and clear logs to confuse and/or slow down any forensics team.

- Denial of Service

Disruption of normal access for users and systems, in order to stop the attack from being monitored, tracked, or blocked

- Exfiltration

The extraction stage: getting data out of the compromised system.

In ODIN project the following Cyber kill chain has been developed:

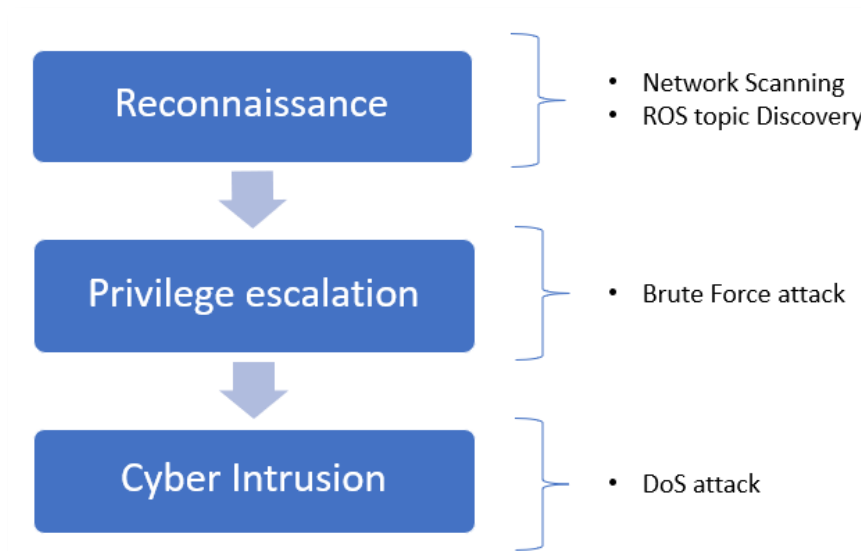


Figure 23: ODIN Cyber Kill Chain

Beginning with this Cyber kill chain, a set of attack scenarios has been developed with the following steps:

1. Attacks:
 - a) Network Discovery: Port Scanning (Nmap).
 - b) Environment Discovery: ROS Topic Discovery (RosPenTo).
 - c) Inhibition of services/resources: DoS (hPing3).
 - d) SSH authentication brute force attack.
2. SIEM: Security Incident detection and match with MITRE attack technique.
3. SOAR: Automated link from the SIEM to the Incident Response for further incident investigation and enrichment.
4. Response action:
 - a) Automated action: temporary blocking of malicious IP in the network.
 - b) Manual action: blocking of malicious IP in the network.
 - c) Notification: Publication in ROS Security Topic.

The first step is the attack to the network. It can be 3 different types of attacks:

1. Network discovery: One of the most typical network attacks is a port scanning, to discover what ports are opened. This can be performed with Nmap tool.
2. Environment discovery: In ODIN this applies to ROS Topic discovery, where the attacker discovers the topics that are used for communication. In this last version, the SSH authentication brute force attack has been included.
3. Inhibition of services/resources: this attack is also known as Denial of Service (DoS) and tries to get down the services that are attacked. One of the most typical tools to perform this attack is hping3 that allows the attacker to send multiple ping packets in a small slot of time.

The second step of the scenario is related with the detection of the incidents (the attack that has been performed). This step is made through the agent located in the monitored endpoint that has been attacked and the SIEM server, where all the events detected by the agents are sent. The SIEM after processing the information shows it in the dashboard and each event is matched to a MITRE attack technique.

The third step is the incident response tool. This tool is used to investigate the alerts received from the SIEM and enrich each case with further information to determine the most suitable response for that alert.

And finally, the response action that there are 3 kinds of responses:

1. Automated action: for example, a temporary blocking of malicious IP that is performed without the action from the operators.
2. Manual action: for example, a temporary blocking of malicious IP that requires an action from the operators.
3. Notification: in ODIN, the publication of a message in ROS security Topic.

3.2.1. Attack implementation included in the final version of the cybersecurity solution

In this latest version of the cybersecurity incident and response solution, two new attacks have been validated. The detection and response to the port scanning attack (with NMap) and the DoS attack (with hping3) were validated in the already presented deliverable D4.2. In this last deliverable we will focus on the validation of two new attacks.

The first one is done with a penetration testing tool for ROS using RosPenTo. This attack is specific to the ROS environment used in the project. This specific software is able to connect to the ROS environment and analyse the topics where the information of the industrial processes is being published. Through this attack, the attacker would be able to read all the information that is being shared through ROS. In addition, the attacker could include new malicious ROS messages in the topics producing unwanted actions in the industrial process.

```
root@36af9d45f3cc:/# rospento
RosPenTo - Penetration testing tool for the Robot Operating System(ROS)
Copyright(C) 2018 JOANNEUM RESEARCH Forschungsgesellschaft mbH
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under certain conditions.
For more details see the GNU General Public License at <http://www.gnu.org/licenses/>.

What do you want to do?
0: Exit
1: Analyse system...
2: Print all analyzed systems
1

Please input URI of ROS Master: (e.g. http://localhost:11311/)
http://192.168.15.112:11311/
```

Figure 24: RosPento execution

Finally, in order to validate what would happen if we made the Monitored Endpoint accessible on the internet, an installation of this component has been created in a private cloud. When this installation was carried out, SIEM monitoring showed that there were ssh authentication attempts on the published machine. This attack is critical because in the event that the attacker is able to enter the Monitored

Endpoint via ssh, he/she would be able to carry out any type of modification or attack on the monitored environment, with repercussions in the industrial environment. Due to this test, it has been decided to include this attack in the ODIN cyber kill chain and provide also the necessary response to mitigate, along with the others, this type of attack.

3.3. ODIN cybersecurity incident detection and response final version

In Figure 25, the final version of the cybersecurity incident detection and response solution flow diagram is presented.

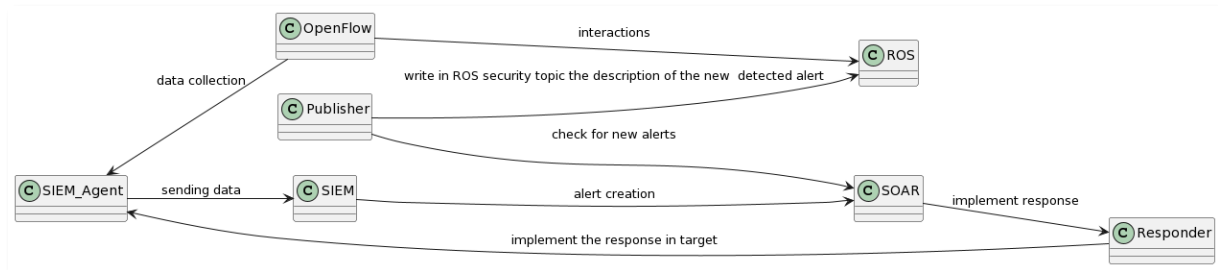


Figure 25: Cybersecurity incident and response solution final flow diagram

This diagram shows how the different modules developed and deployed in ODIN are interconnected.

The flow starts with the detection of incidents from the OpenFlow component. This is done through the IDS (to monitor the network) and a specific component that is able to read the OpenFlow logs. The logs and intrusions detected by the IDS are passed to the SIEM Agent which, in turn, normalizes the data and applies rules and decoders to analyze whether the information that has arrived is a cybersecurity alert or not (it can be related to an intrusion attempt detected by the IDS or an attack attempt identified in the OpenFlow logs).

In case the Agent identifies the entry as a cybersecurity alert, it is sent to the SIEM. This component is responsible for normalizing, enriching, and displaying this information on the dashboard. In turn, specific decoders have been developed to parse the data of the specific attacks that are analysed in ODIN and specific rules that allow to detect the mentioned attacks and pass them to the SOAR. In addition, for some specific cases, responses to the detected incidents have been created in the SIEM to minimize the damage they may cause.

Once the alerts have arrived at the SOAR, they are presented in the SOAR user interface. The cybersecurity operator will be in charge of analysing each one of them to decide whether to discard them (false positives) or escalate them to a case, perform a more exhaustive analysis and respond to them with a reply.

The responder is in charge of launching the response to mitigate the detected cybersecurity alert. To do so, it makes use of the observables that have been generated in the alert. These responses can be launched automatically, in case the alert is previously known, or manually.

Finally, and with the intention of closing the cycle with ROS, a specific module called Publisher has been created, which is in charge of analysing if there are new alerts in the SOAR, accessing the API and writing them in a ROS topic. This allows us to have the alerts in ROS so that they can be visualized in OpenFlow (or the augmented reality glasses installed in the LMS) and take the necessary measures in the production environment (such as stopping the industrial process in the event that a specific type of attack is detected).

3.3.1. Monitored endpoint

This section describes the new module that has been integrated into the Monitored Endpoint. The module has been created due to the need to write cybersecurity alerts in a ROS topic. In the first version of this module, the alerts were sent from the SOAR component to the Monitored Endpoint so that it could write them.

Due to confidentiality aspects, it is preferable for the end users to avoid VPN connections in order to minimize any risks for accessing their internal network systems. Thus, a VPN connection to the SOAR component would be quite challenging.

To solve this problem, it has been decided to modify the alert and script capture module in ROS. In this second version, an AlertChecker module has been created to perform queries to the SOAR API to bring the new cybersecurity alerts detected (it analyzes if they are repeated or not). These alerts are stored in a folder created in the Monitored Endpoint.

On the other hand, another component SecurityAlertPublisher has been created and is launched in the ROS environment. This component is in charge of analyzing the directory where the alerts are saved, and for new alerts, it formats them and writes them in the ROS security_events topic.

The evidence of operation is presented in Figure 26 where in step 1 it is presented how the alerts are brought from the SOAR API, in step 2 how the new alerts are written in the ROS topic and in step 3 how (by echoing the ROS security_event topic) the cybersecurity alerts are displayed in ROS.

Figure 26: Evidences of Alert Check and writing in ROS



Figure 27: Cybersecurity alert publish in ROS

The following sequence is required to perform the scenario (Figure 27).

- Run AlertChecker as a continuous daemon.
- ROS Subscription to ROS security topic - to see if cybersecurity alerts are being ingested in a ROS topic.
- ROS execution securityalert Publisher – write in ROS topic /execution/cyber_security/integration/topics/security_event.

This demonstrator has been implemented using the implementation provided in GitHub repository for OpenFlow on M28 ODIN-Project-Architecture/Docker/OpenFlowSecurityM28 which includes a

security topic ready to process the messages in the right format and delivered by the securityalertPublisher which is compiled in ROS environment.

3.3.2. SIEM

3.3.2.1. ROSPenTo attack

This section analyses the evidence of detection of the ROSPenTo attack with the SIEM.

In the Figure 28 the detection of the ROSPenTo attack in the dashboard is presented. In the same dashboard, the events related to this attack are visualized in Figure 29. The detailed information of these events can be seen in Figure 30.

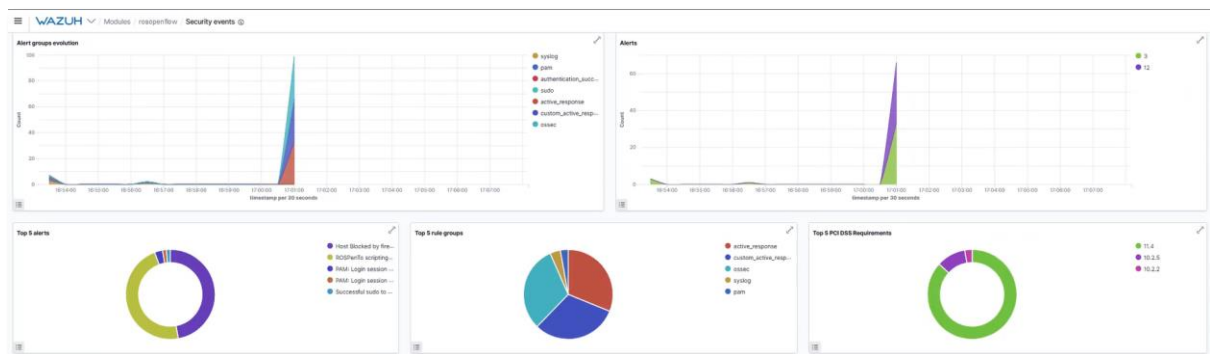


Figure 28: ROSPenTO detection in SIEM: Dashboard

| | |
|-------|--|
| T1595 | ROSPenTo scripting engine detected, Scan Analyse system. |
| T1595 | ROSPenTo scripting engine detected, Scan Analyse system. |
| T1595 | ROSPenTo scripting engine detected, Scan Analyse system. |

Figure 29: ROSPenTO detection in SIEM: Events

| SON | Rule |
|------------------|--|
| agent.ip | 192.168.15.112 |
| agent.name | rosopenflow |
| agent.id | 005 |
| manager.name | wazuh-manager |
| rule.firedtimes | 33 |
| rule.mail | true |
| rule.level | 12 |
| rule.description | ROSPenTo scripting engine detected, Scan Analyse system. |
| rule.groups | custom_active_response_rules |
| rule.mitre.id | T1595 |
| rule.id | 100202 |
| decoder.name | json |
| full_log | <pre>{ "timestamp": "2023-01-03T16:01:03.726878+0000", "flow_id": "1904482203979206", "iface": "ens0s3", "event_type": "alert", "src_ip": "192.168.15.112", "src_port": 11311, "dest_ip": "192.168.15.112", "dest_port": 4444, "protocol": "TCP", "service": "HTTP", "category": "Misc activity", "severity": 3, "metadata": { "affected_product": "Windows_XP_Vista_7_and_10_Server_32_64_BIT", "attack_target": "Client_Endpoint", "informational": true, "updated_at": "2022-04-18T00:00:00Z", "http": { "hostname": "192.168.15.112", "http_port": 11311, "url": "/", "http_user_agent": "XML-RPC.NET", "http_content_type": "text/xml", "http_method": "POST", "http_status": 200, "http_body": "[]", "gaps": false, "state": "CLOSED", "stored": false, "size": 268, "tx_id": 303 }, "app_proto": "http", "flow": { "pkts_toserver": 68, "pkts_toclient": 68, "bytes_toserver": 18514, "bytes_toclient": 17297, "start": 1672780863.726878, "end": 1672780863.726878 } } }</pre> |
| location | /var/log/suricata/eve.json |

Figure 30: ROSPenTO detection in SIEM: Events detail

In the case of the ROSPenTo attack, it has been analysed to launch the attack from the SOAR, in the same way as it has been done with the other attacks above. Due to the time needed for the alert to reach the SOAR is sufficient for ROSPenTo not to give a result. It has been tested launching the response automatically from the SIEM component every time one of these events is detected.

We know that evidencing the ROS topics that are running in an environment, in our case in the Monitored Endpoint, can generate an important cybersecurity problem. In the event that someone


```

root@36af9d45f3cc:/# rospento
RosPenTo - Penetration testing tool for the Robot Operating System(ROS)
Copyright(C) 2018 JOANNEUM RESEARCH Forschungsgesellschaft mbH
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under certain conditions.
For more details see the GNU General Public License at <http://www.gnu.org/licenses/>.

What do you want to do?
0: Exit
1: Analyse system...
2: Print all analyzed systems
1

Please input URI of ROS Master: (e.g. http://localhost:11311/)
http://192.168.15.112:11311/
System (http://192.168.15.112:11311/) does not respond!

```

Figure 33: Evidence of the unsuccessful connection of ROSPenTo

3.3.2.2. SSH Brute Force attack

In this section, evidence of the detection of the SSD Brute Force attack is presented. Figure 34 shows how the events are represented in the SIEM dashboard. In the Figure 35 the resume of detected events can be seen and in Figure 36 the details of each of these events are presented.

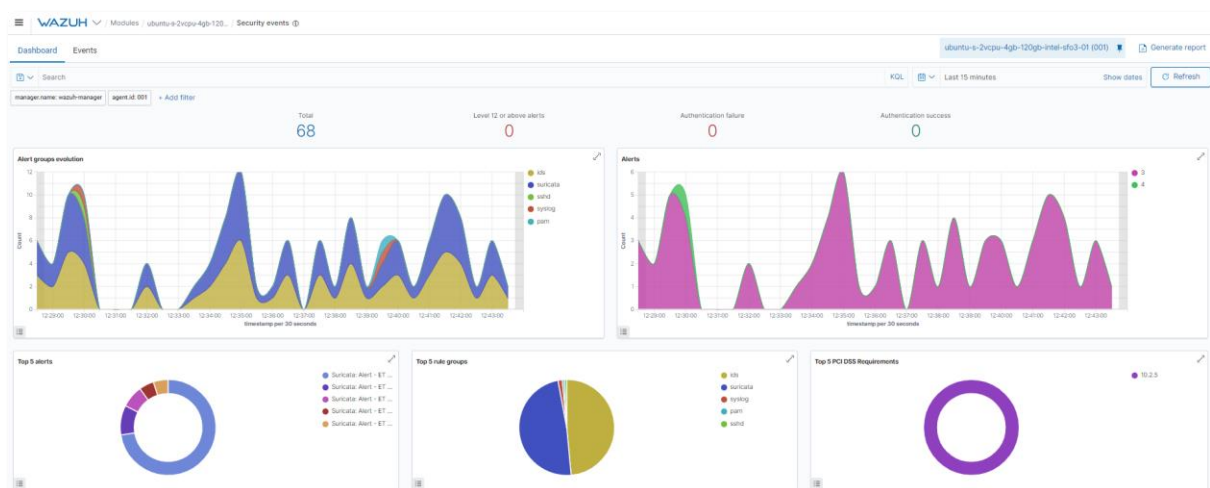


Figure 34: SSH Brute Force detection in SIEM: Dashboard

| | | | |
|-------------------------------|-------|-------------------|--|
| > Dec 22, 2023 @ 11:53:37.009 | T1110 | Credential Access | sshd: Attempt to login using a non-existent user |
| > Dec 22, 2023 @ 11:53:34.993 | T1110 | Credential Access | sshd: Attempt to login using a non-existent user |
| > Dec 22, 2023 @ 11:53:32.991 | T1110 | Credential Access | sshd: Attempt to login using a non-existent user |
| > Dec 22, 2023 @ 11:53:32.991 | T1110 | Credential Access | sshd: Attempt to login using a non-existent user |
| > Dec 22, 2023 @ 11:53:30.990 | T1110 | Credential Access | sshd: Attempt to login using a non-existent user |
| > Dec 22, 2023 @ 11:53:28.990 | T1110 | Credential Access | sshd: Attempt to login using a non-existent user |
| > Dec 22, 2023 @ 11:53:28.988 | T1110 | Credential Access | sshd: Attempt to login using a non-existent user |
| > Dec 22, 2023 @ 11:53:28.988 | T1110 | Credential Access | sshd: Attempt to login using a non-existent user |

Figure 35: SSH Brute Force detection in SIEM: Events

| 2023 @ 11:53:37.009 T1110 Credential Access sshd: Attempt to login using a non-existent user | |
|--|---|
| JSON | Rule |
| agentIp | 143.198.48.210 |
| agent.name | ubuntu-s-2vcpu-4gb-120gb-intel-sfo3-01 |
| agent.id | 001 |
| manager.name | wazuh-manager |
| rule.mail | true |
| rule.level | 12 |
| rule.pci_dss | 10.2.4, 10.2.5, 10.6.1 |
| rule.hipaa | 164.312.b |
| rule.tsc | CC6.1, CC6.8, CC7.2, CC7.3 |
| rule.description | sshd: Attempt to login using a non-existent user |
| rule.groups | syslog, sshd, invalid_login, authentication_failed |
| rule.nist_800_53 | AU.14, AC.7, AU.6 |
| rule.gdpr | IV.35.7.d, IV.32.2 |
| rule.firedtimes | 8 |
| rule.mitre.technique | Brute Force |
| rule.mitre.id | T1110 |
| rule.mitre.tactic | Credential Access |
| rule.id | 100206 |
| rule.gpg13 | 7.1 |
| decoder.name | sshd |
| full_log | Dec 22 10:53:35 ubuntu-s-2vcpu-4gb-120gb-intel-sfo3-01 sshd[3989576]: Disconnected from invalid user admin 122.187.178.195 port 56811 [preauth] |
| location | /var/log/auth.log |

Figure 36: SSH Brute Force detection in SIEM: Events detail

As these events are more complex (attacks are carried out from different source machines at different times), it has been decided to pass them to the SOAR to analyse them there and implement the response to mitigate the attack.

3.3.3. SOAR

This section presents evidence of the alerts that have reached SOAR and how they have been responded to.

3.3.3.1. SSH Brute Force attack

In Figure 37, acquired evidence of SSH Brute Force attack alerts in the SOAR are presented. Figure 38 visualizes the detail of these alerts.

| TheHive + New Case My tasks 0 Waiting tasks 0 Alerts 356 Dashboards Search Caseld Organisation ODIN/odin_project | | | | | | | | | |
|---|------|--|--------|-------------|----------------------|-----------|-------------|--|--|
| Severity | Read | Title | # Case | Type | Source | Reference | Observables | Dates | |
| 4 | Read | sshd: Attempt to login using a non-existent user | #3083 | wazuh-alert | Wazuh: wazuh-manager | 8cf516 | 1 | O. 12/21/23 16:44 C. 12/21/23 16:44 U. 12/21/23 16:44 a few seconds | |
| <div> <div>SIEM:Wazuh alert</div> <div> <div>Source Api: SIEM:Wazuh</div> <div>Automated: false</div> <div>Source Alert Id: 1703173459.9028812</div> <div>Source Rule Level: 12</div> <div>Rule description: sshd: Attempt to login using a non-...</div> </div> </div> | | | | | | | | | |
| <div> <div>Source Rule Id: 100206</div> <div>Mitre ID: T1110</div> <div>Mitre Tactic: Credential Access</div> <div>Mitre Technique: Brute Force</div> <div>Source Agent Id: 001</div> </div> | | | | | | | | | |
| <div> <div>Agent Name: ubuntu-s-2vcpu-4gb-120gb-intel-s...</div> <div>Agent IP: 143.198.48.210</div> <div>Event Type: alert</div> <div>Alert Action:</div> <div>Alert Signature:</div> <div>Alert Category:</div> </div> | | | | | | | | | |
| Source Alert Severity: 2 | | | | | | | | | |
| 4 | Read | sshd: Attempt to login using a non-existent user | #3082 | wazuh-alert | Wazuh: wazuh-manager | 4fb4c7 | 1 | O. 12/21/23 16:44 C. 12/21/23 16:44 U. 12/21/23 16:44 a few seconds | |
| <div> <div>SIEM:Wazuh alert</div> <div> <div>Source Api: SIEM:Wazuh</div> <div>Automated: false</div> <div>Source Alert Id: 1703173453.9028233</div> <div>Source Rule Level: 12</div> <div>Rule description: sshd: Attempt to login using a non-...</div> </div> </div> | | | | | | | | | |
| <div> <div>Source Rule Id: 100206</div> <div>Mitre ID: T1110</div> <div>Mitre Tactic: Credential Access</div> <div>Mitre Technique: Brute Force</div> <div>Source Agent Id: 001</div> </div> | | | | | | | | | |
| <div> <div>Agent Name: ubuntu-s-2vcpu-4gb-120gb-intel-s...</div> <div>Agent IP: 143.198.48.210</div> <div>Event Type: alert</div> <div>Alert Action:</div> <div>Alert Signature:</div> <div>Alert Category:</div> </div> | | | | | | | | | |
| Source Alert Severity: 2 | | | | | | | | | |

Figure 37: SSH Brute Force attack alerts in SOAR

Case # 3083 - sshd: Attempt to login using a non-existent user

Wazuh Odin integration 12/21/23 16:44 20 hours 21 cases 1 alert

Sharing (0) | Close

Details Tasks Observables TTPs

Basic Information

Title sshd: Attempt to login using a non-existent user

Severity High

TLP TLPAED

PAP PROBANDER

Assignee Wazuh Odin integration

Date 12/21/23 16:44

Tags SIEM:Wazuh Alert

Additional information

[Add](#) [Layout](#)

| | | |
|-------------------------------------|--|---|
| Source Api SIEM:Wazuh | Automated False | Source Alert Id 1703175459.9028812 |
| Source Rule Level 12 | Rule description sshd: Attempt to login using a non-existent user | Source Rule Id 100206 |
| Mitre ID T1110 | Mitre Tactic Credential Access | Mitre Technique Brute Force |
| Source Agent Id 001 | Agent Name ubuntu-s-2vcpu-4gb-120gb-intel-sfo3-01 | Agent IP 143.198.48.210 |
| Event Type alert | Alert Action Not Specified | Alert Signature Not Specified |
| Alert Category Not Specified | Source Alert Severity 2 | |

Related cases

Newest (Case # 2916 - sshd: Attempt to login using a non-existent user)
Created on 12/21/23 16:36
Shares 1 observable
Tagged as SIEM:Wazuh Alert

Oldest (Case # 3107 - sshd: Attempt to login using a non-existent user)
Created on 12/21/23 16:45
Shares 1 observable
Tagged as SIEM:Wazuh Alert

[See all \(21 related cases\)](#)

Figure 38: SSH Brute Force attack alert detail in SOAR

Due to the configuration that has been programmed into the SOAR, it is able to detect and extract two observables from each of these alerts. Figure 39 shows that in the observables, the source IP of the machine from which the attack was launched, and the general log of the alert are extracted.

These observables are used to be able to launch the response to the attacks by means of the responses that can be seen in Figure 40. Specifically, the Firewall Drop responder adds a rule to block the attacker's source IP (extracted from the observable) in the Iptables firewall of the Monitored Endpoint. With this, all incoming packets to the Monitored Endpoint from this IP are cut off. Figure 41 shows how the block entries have been added in the IPTables of the Monitored Endpoint.

Details Tasks Observables TTPs

No observable selected [+ Add observable\(s\)](#) [Export](#)

Filters

[+ Add a filter](#)

List of observables (2 of 2)

| Flags | Type | Value/Filename |
|---|-------|--|
| <input type="checkbox"/> 🌟 👁 🔍 | ip | 110[.]137[.]195[.]210 source No reports available |
| <input type="checkbox"/> 🌟 👁 🔍 | other | "Dec 21 15:44:17 ubuntu-s-2vcpu-4gb-120gb-intel-sfo3-01 sshd[3419390]: Invalid user machao from 110[.]137[.]195[.]210 port 9367" raw-json-alert No reports available |

Figure 39: SSH Brute Force attack alert observables in SOAR

Run responders

Please select the responder you want to run

 Filter responders

Microsoft Teams Bot_1_0

Block an IP on a host via Wazuh agent (API v4)

Wazuh Firewall Drop APIv4_1_0

Block an IP on a host via Wazuh agent (API v4)

SSH Case Sender_1_0

Sends json case info file to a remote host/folder

Figure 40: SSH Brute Force attack alert responders in SOAR

```
root@ubuntu-s-2vcpu-4gb-120gb-intel-sfo3-01:~/alertChecker# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
DROP      all  --  hosted-by.alsycon.net  anywhere
DROP      all  --  212.70.149.150         anywhere
DROP      all  --  111.198.221.98         anywhere
DROP      all  --  211.118.215.8          anywhere
DROP      all  --  211.118.215.8          anywhere
DROP      all  --  211.118.215.8          anywhere
DROP      all  --  211.118.215.8          anywhere
DROP      all  --  211.118.215.8          anywhere
DROP      all  --  user178.151-252-135.netatonce.net anywhere
DROP      all  --  user178.151-252-135.netatonce.net anywhere
DROP      all  --  user178.151-252-135.netatonce.net anywhere
DROP      all  --  user178.151-252-135.netatonce.net anywhere
DROP      all  --  ip.bkhost.vn           anywhere
DROP      all  --  static.58.102.12.190.cps.com.ar anywhere
DROP      all  --  ubuntu-s-2vcpu-4gb-120gb-intel-sfo3-01 anywhere

Chain FORWARD (policy DROP)
target     prot opt source                destination
DROP      all  --  hosted-by.alsycon.net  anywhere
DROP      all  --  212.70.149.150         anywhere
DROP      all  --  111.198.221.98         anywhere
DROP      all  --  211.118.215.8          anywhere
DROP      all  --  211.118.215.8          anywhere
DROP      all  --  211.118.215.8          anywhere
DROP      all  --  211.118.215.8          anywhere
DROP      all  --  211.118.215.8          anywhere
DROP      all  --  user178.151-252-135.netatonce.net anywhere
DROP      all  --  user178.151-252-135.netatonce.net anywhere
DROP      all  --  user178.151-252-135.netatonce.net anywhere
DROP      all  --  user178.151-252-135.netatonce.net anywhere
DROP      all  --  ip.bkhost.vn           anywhere
DROP      all  --  static.58.102.12.190.cps.com.ar anywhere
DROP      all  --  ubuntu-s-2vcpu-4gb-120gb-intel-sfo3-01 anywhere
```

Figure 41: Monitored Endpoint Iptables

4. CONCLUSIONS

This report has presented the final validation of the Networked Component for both the OpenFlow and Cybersecurity modules. In particular, the final validation of the Networked Component has been performed in the context of the pre-industrial pilot case setups of ODIN, validating that the Networked Component is ready to be applied in the industrial pilot cases of the ODIN project.

More specifically, the OpenFlow module was tested in specially designed pre-industrial settings for each pilot case, namely the Automotive, White Goods and Aeronautics pilot cases. The pre-industrial product plans and production scenarios that have been created have also been used for the validation of the OpenFlow functionalities in terms of suitability, robustness and performance. These product plans and scenarios are based on the requirements of the pilot cases developed in WP1 and in collaboration with responsible partners. Therefore, a preliminary deployment and testing step has been completed in two stages, the individual prototype validation stage and the pilot case execution validation stage.

Finally, the ODIN Networked component has performed initial deployment and validation tests in terms of cyber security, which is an important topic of the modern digital environment. In particular, during this preliminary deployment and testing period validation process the Cyber-Security module prototype and the OpenFlow module prototype security related functionalities and integration were validated.

Specific integration points between the two modules were developed and tested that hardened the security of the OpenFlow module, by allowing the Cyber-Security module to detect potential security issues that affected the OpenFlow operation and also allowed the OpenFlow to orchestrate shopfloor related cybersecurity responses.

The validation process as a whole was a thorough, diligent work that has validated both the OpenFlow and Cybersecurity modules but also execution of the pre-industrial setups, ensuring and streamlining the integration in the industrial environment.

The next step for the validated Networked Component is to be integrated into the industrial pilot case setups as part of the ODIN OpenFlow software architecture. In the framework of WP5 “ODIN Industrial Component for robust Large scale Pilot Lines” all ODIN Components, namely the ODIN Digital Component, the ODIN Open Component, the ODIN Networked Component and the ODIN Industrial Component will be integrated in the Automotive, Aeronautics and White Goods pilot lines where the performance of the ODIN system will be applied and assessed.

5. GLOSSARY

| | |
|-------|---|
| AI | Artificial Intelligence |
| AGV | Automated Guided Vehicle |
| API | Application Programming Interface |
| AR | Augmented Reality |
| CRUD | Create, Reade, Update, Delete |
| C&C | Command and Control |
| DB | Database |
| DDD | Domain Driven Design |
| ERP | Enterprise Resource Planning |
| IDS | Intrusion Detection System |
| IEC | International Electrotechnical Commission |
| IP | Internet Protocol |
| IPS | Intrusion Prevention System |
| ISA | Industry Standard Architecture |
| IT | Information Technology |
| HMI | Human Machine Interface |
| HRC | Human Robot Collaboration |
| KR | Knowledge Repository |
| MES | Manufacturing Execution Systems |
| OSINT | Open-Source Intelligence |
| OT | Operational Technology |
| PLM | Product Lifecycle Management |
| ROS | Robot Operating System |
| SCADA | Supervisory Control and Data Acquisition |
| SOA | Service Oriented Architecture |
| SOAR | Security Orchestration, Automation and Response |
| SIEM | Security Information and Event Management |
| SOC | Security Operation Centre |
| UI | User Interface |
| URL | Uniform Resource Locator |

6. REFERENCES

1. Chryssolouris, G., Manufacturing Systems: Theory and Practice, 2nd Edition, Springer-Verlag, New York, New York, (2006)
2. S. Koukas, N. Kousi, S. Aivaliotis, G. Michalos, R. Bröchler, S. Makris, ‘ODIN architecture enabling reconfigurable human – robot based production lines’, *Procedia CIRP*, Volume 107, pp 1403-1408 (2022)
3. S, Aivaliotis, K. Lotsaris, C. Gkournelos, N. Fourtakas, S. Koukas, N. Kousi, S. Makris ‘An augmented reality software suite enabling seamless human robot interaction’, *International Journal of Computer Integrated Manufacturing*, pp1-27(2022)
4. publisher = {Taylor & Francis}, N. Kousi, S. Koukas, G. Michalos, S. Makris, G. Chryssolouris, "Service oriented architecture for dynamic scheduling of mobile robots for material supply", *CIRPe2016*, *Procedia CIRP*, 5th CIRP Global Web Conference-Research and Innovation for Future Production [Volume 55, pp. 18-22](#) (2016)
5. S. Papanastasiou, N. Kousi, P. Karagiannis, C. Gkournelos, A. Papavasileiou, K. Dimoulas, K. Baris, S. Koukas, G. Michalos, S. Makris, "Towards seamless human robot collaboration: integrating multimodal interaction", *The International Journal of Advanced Manufacturing Technology*, [Volume 105, pg. 3881-3897](#), (2019)
6. G. Michalos, N. Kousi, P. Karagiannis, C. Gkournelos, K. Dimoulas, S. Koukas, P. Mparis, A. Papavasiliou, S. Makris, "Seamless human robot collaborative assembly – An automotive case study", *Mechatronics*, [Volume 55, pg 194-211](#), (2018)
7. S. Makris, P. Karagiannis, S. Koukas, A. S. Matthaiakis, "Augmented reality system for operator support in human–robot collaborative assembly", *CIRP Annals - Manufacturing Technology*, [Volume 65, Issue 1, pp. 61-64](#), (2016)
8. Open-source UR10 robots’ ROS drivers, https://github.com/ros-industrial/ur_modern_driver
9. [GitHub - jr-robotics/ROSPenTo: Penetration testing tool for ROS](#)