RE4DY

MANUFACTURING DATA NETWORKS

RE4DY TOOLKIT

Name of the Tool	Incident Detection
Tool Owner	Industry Commons Foundation
Version	1.0
Date	Nov 2025
Version	V1.0

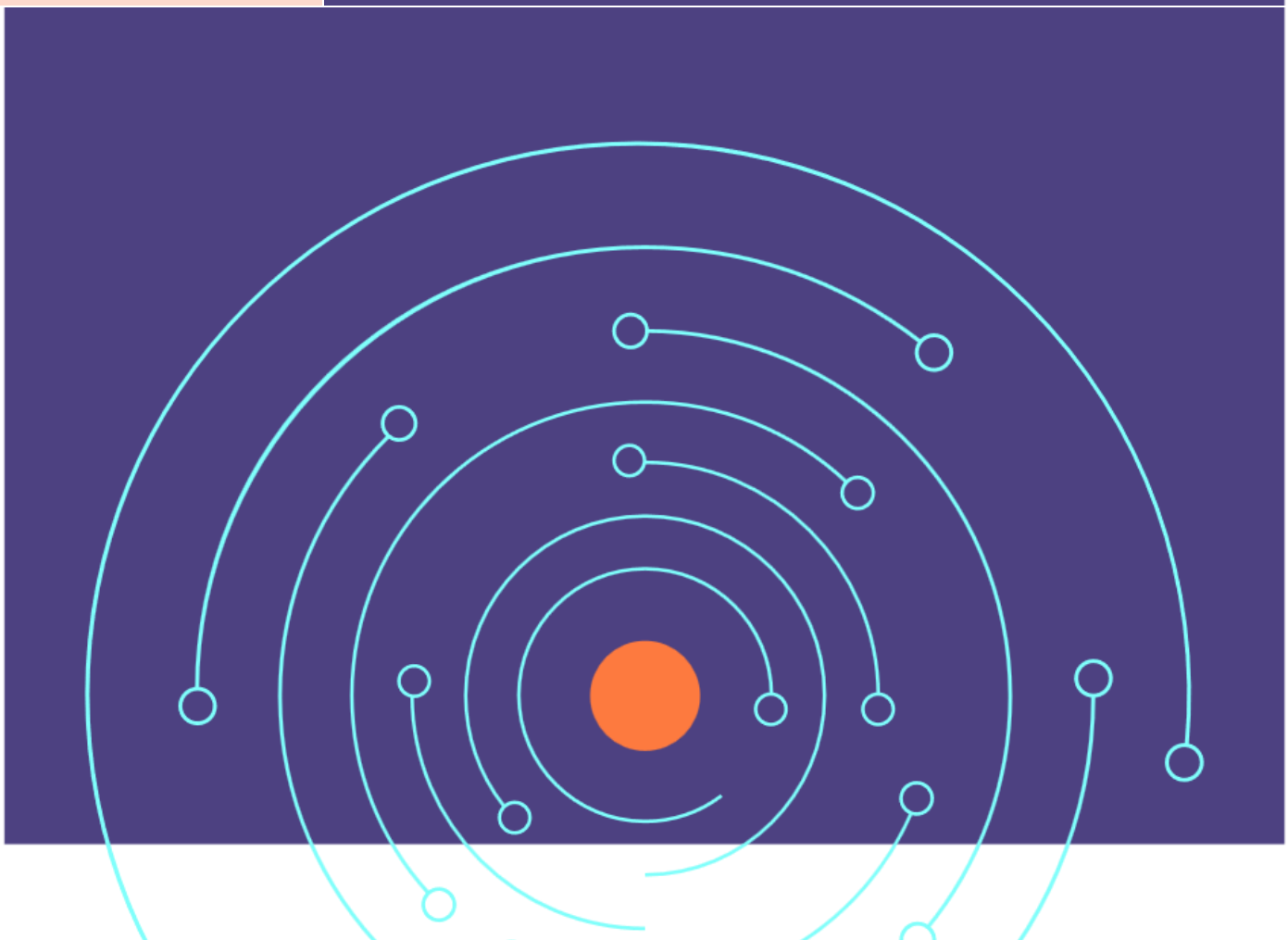

Table of contents

1. Component Description

The Incident Detection component acquires monitoring awareness from all the events received from the monitored endpoints: authentication errors, software/malware installation, real time file monitoring (looking for unauthorized modifications), network attacks, privilege escalation, or others that can be configured. Incident detection monitoring also normalizes all the info received to prepare it for the next phase.

This component provides cybersecurity functionality to RE4DY endpoints and enables the detection of the events described above, both endpoints as component running.

2. Input

The Incident Detection communicates with the agents in the monitored endpoints to get the information about all the events registered, such as authentication errors, software/malware installation, real time file monitoring (looking for unauthorized modifications), network attacks, privilege escalation, or others that can be configured.

The incident detection component is integrated by the agent and the server:

1) Agent: Running in the endpoint, is in charge of collecting all the information from the different logs in the endpoints and other components and sends them to the server.
2) Incident detection: Collects all logs received from the agents in the network.

The agent running in each endpoint must be enrolled in the server and for that it is needed to change the configuration in the ossec.conf file, as showed in figure 1, and restart the service.

.

```
<ossec_config>
  <client>
    <server>
      <address>10.10.10.3</address>
      <port>18504</port>
      <protocol>tcp</protocol>
    </server>
    <enrollment>
      <manager_address>10.10.10.3</manager_address>
      <port>2626</port>
      <agent_name>test-idi</agent_name>
    </enrollment>
    <config-profile>ubuntu, ubuntu22, ubuntu22.04</config-profile>
    <notify_time>10</notify_time>
    <time-reconnect>60</time-reconnect>
    <auto_restart>yes</auto_restart>
    <crypto_method>aes</crypto_method>
  </client>
```

Figure 1 ossec.conf file configuration example

3. Output

The incident detection component communicates with the incident response component and sends all the normalized alerts detected to be treated. For each alert or incident a json file will be generated as the following one:

{

"_index": "wazuh-alerts-4.x-2023.02.27",

"_type": "_doc",

"_id": "9q8DlIYBeKdNQkVPVHDn",

"_version": 1,

"_score": None,

"_source": {

"input": {

"type": "log"

},

"agent": {

"ip": "192.168.15.112",

"name": "RE4DY agent 1",

"id": "005"

},

"manager": {

"name": "wazuh-manager"

},

"data": {

"srcip": "192.168.15.177",

"in_iface": "enp0s3",

"src_ip": "192.168.15.186",

"src_port": "6443",

"event_type": "alert",

"alert": {

"severity": "3",

"signature_id": "2210046",

"rev": "2",

"gid": "1",

"signature": "SURICATA STREAM SHUTDOWN RST invalid ack",

"action": "allowed",

"category": "Generic Protocol Command Decode"

},

"tls":{

"version":"TLS 1.3",

"ja3":{

"hash":"89be98bbd4f065fe510fca4893cf8d9b",

"string":"771,49199-49200-49195-49196-52392-52393-49171-49161-49172-49162-156-157-47-53-49170-10-4865-4867-4866,5-10-11-13-65281-18-43-51,29-23-24-25,0"

},

"ja3s":{

"hash":"f4febc55ea12b31ae17cfb7e614afda8",

"string":"771,4865,43-51"

}

},

"app_proto":"tls",

"flow_id": "1331957630929130.000000",

"dest_ip": "192.168.15.177",
"proto": "TCP",
"dest_port": "48580",
"flow": {
"pkts_toserver": "18",
"start": "2023-02-27T01:54:58.624874+0000",
"bytes_toclient": "5318",
"bytes_toserver": "2212",
"pkts_toclient": "17"
},
"timestamp": "2023-02-27T01:59:53.111013+0000"
},
"rule": {
"firedtimes": 1,
"mail": True,
"level": 12,
"description": "DoS attack has been detected.",
"groups": [
"custom_active_response_rules"
],
"mitre": {
"id": [
"T1498"
],
"tactic": [
"Impact"

```
      ],
      "technique": [
        "Network Denial of Service"
      ]
    },
    "id": "100200"
  },
  "location": "/var/log/suricata/eve.json",
  "decoder": {
    "name": "json"
  },
  "id": "1677463193.114738",
  "full_log": "{\"timestamp\":\"2023-02-27T01:59:53.111013+0000\",\"flow_id\":1.33195763092913e+15,\"in_iface\":\"enp0s3\",\"event_type\":\"alert\",\"src_ip\":\"192.168.15.186\",\"src_port\":6443,\"dest_ip\":\"192.168.15.177\",\"dest_port\":48580,\"proto\":\"TCP\",\"alert\":{\"action\":\"allowed\",\"gid\":1,\"signature_id\":2210046,\"rev\":2,\"signature\":\"SURICATA STREAM SHUTDOWN RST invalid ack\",\"category\":\"Generic Protocol Command Decode\",\"severity\":3},\"tls\":{\"version\":\"TLS 1.3\",\"ja3\":{\"hash\":\"89be98bbd4f065fe510fca4893cf8d9b\",\"string\":\"771,49199-49200-49195-49196-52392-52393-49171-49161-49172-49162-156-157-47-53-49170-10-4865-4867-4866,5-10-11-13-65281-18-43-51,29-23-24-25,0\"},\"ja3s\":{\"hash\":\"f4febc55ea12b31ae17cfb7e614afda8\",\"string\":\"771,4865,43-51\"}},\"app_proto\":\"tls\",\"flow\":{\"pkts_toserver\":18,\"pkts_toclient\":17,\"bytes_toserver\":2212,\"bytes_toclient\":5318,\"start\":\"2023-02-27T01:54:58.624874+0000\"}}",
  "timestamp": "2023-02-27T01:59:53.809+0000"
},
"fields": {
  "data.timestamp": [
    "2023-02-27T17:54:49.468Z"
```

```
      ],
      "timestamp": [
        "2023-02-27T17:54:49.909Z"
      ]
    },
    "highlight": {
      "rule.id": [
        "@kibana-highlighted-field@100201@/kibana-highlighted-field@"
      ]
    },
    "sort": [
      1677520489908
    ]
  }
```


4. Information Flow

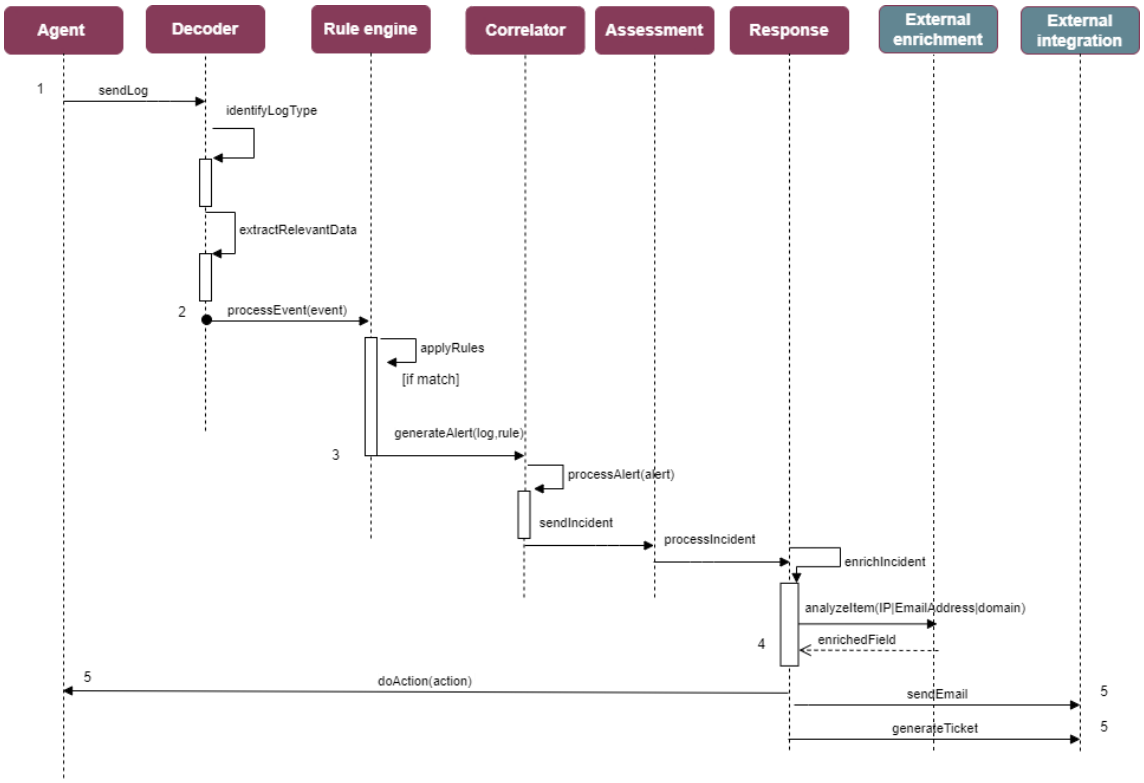

Figure 2 Incident detection and response information flow diagram

- Incident Detection will receive logs and information from the agents deployed.
- Incident Detection will decode the log, identify the type of log and extract some useful fields.
- Incident Detection will have a ruleset to be applied to the received logs.
- Incident Detection will apply the active rules to the received log, and if there is a match, it will generate an alert.
- Incident Response will normalize the alert event and correlate until determine if it is only a simple alert or a real incident.
- Incident Response will enrich the incident with useful information, to facilitate the assignment of the risk level of the incident and the response actions to be done.
- Incident Response can do predefined actions for incident mitigation depending on the incident, such as communicate with the agent so that it performs an action, send an email or send the incident to a ticketing system.

Incident Detection will update information on a GUI so that the admin user can see the status of the agents and the alert/incident information.

5. Internal Architecture

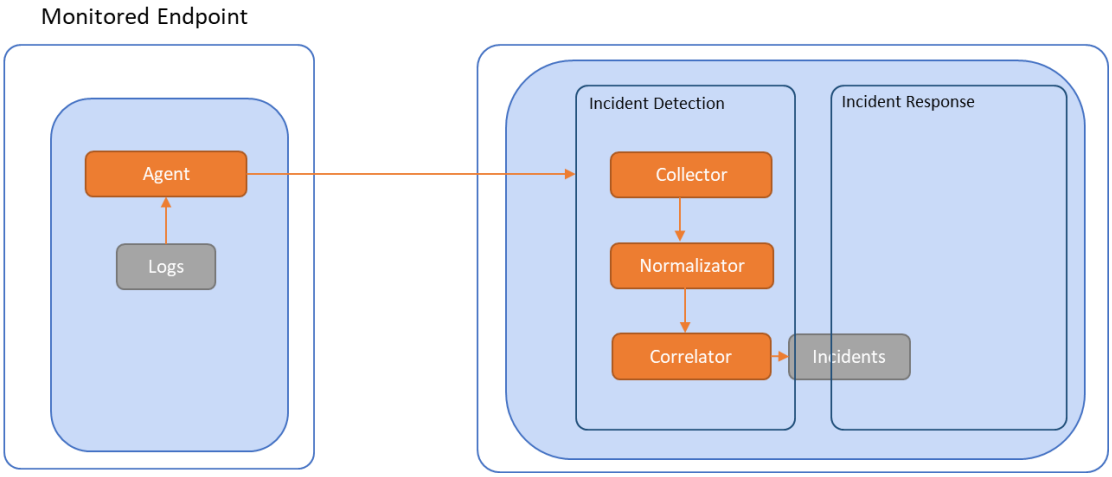

Figure 3 Incident detection component architecture

In the Figure *3* it is shown how the different blocks are connected in the component. In the left side it is shown the monitored endpoint with an agent running, that collects all the information from the different logs.

In the right side, it is shown the server side, where the incident detection and incident response are running. The Agent sends the info to the incident detection component and after passing through all the steps send the incidents that must have a response to the incident response component

6. API

METHOD	DESCRIPTION	ENDPOINT
PUT	Run an Active Response command on all agents or a list of them	{SIEM}/active-response
PUT	Restart all agents or a list of them	{SIEM}/agents/restart
PUT	Restart the specified agent	{SIEM}/agents/{agent_id}/restart
POST	Add an agent specifying its name, ID and IP. If an agent with the same ID already exists, replace it using 'force' parameter	{SIEM}/agents/insert

POST	Add a new agent with basic info	{SIEM}/agents
DELETE	Delete all agents or a list of them based on optional criteria	{SIEM}/agents
GET	Obtain a list with information of the available agents	{SIEM}/agents
PUT	Restart the manager	{SIEM}/manager/restart
GET	Return statistical information for the current or specified date	{SIEM}/manager/stats
PUT	Replace configuration with the data contained in the API request	{SIEM}/manager/configuration
GET	Return enabler configuration used	{SIEM}/manager/configuration
GET	Basic information such as version, compilation date, installation path	{SIEM}/manager/info
GET	Return the status of the monitoring server	{SIEM}/manager/status

7. Implementation Technology

Wazuh is open-source software which can be installed and executed on bare metal, on a virtual machine, as well as in containerized manner using Docker or Kubernetes. We have chosen the container-based approach for all services related to incident detection of RE4DY platform. Therefore, Docker is installed on a virtual machine running Ubuntu 22.04 Linux OS and the compose plugin is used to orchestrate the deployment of the necessary services. We use the official wazuh Docker image as base for our Incident Detection instance.

8. Comments

The component will be integrated with incident response component.