Security Information and Event Management - User Manual

SPHINX

A Universal Cyber Security Toolkit for Health-Care Industry

Table of contents

Table of figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

1 Introduction

The SIEM component is responsible for triggering alerts and to match information using log files that are collected from multiple resources such as data collected from Data traffic monitoring, system log files, auditing checks, vulnerability assessments. The data can be either constructed data (json, csv files) or raw text files that can be converted to structured data by using regular expressions. The SIEM continuously monitors the events created from the log files and trigger the alerts. The alerts could include executions of specific bash scripts, sending of emails, API Calls or publishing to Kafka topics. Consequently, every time a specific predefined condition is met to the scheduled query the above actions can be triggered.

2 Installation/Deployment

The installation is based on docker images for deploying the SIEM or using NPM and NodeJS to execute locally. The ports expose a WebUI and REST API for getting data using POST methods. The SIEM is also possible to parse log files using agents that use a specific TCP port. Having the docker images it is possible to deploy the SIEM.

2.1 Prerequisites and hardware

a. CPU: medium CPU like Intel I7
b. GPU: no needed
c. RAM: medium Ram like 16 GB RAM
d. HDD: for WEB backend server binary files needs 18 MB

2.2 Deployment with Docker

The Document Manager is the core component of the SIEM and a docker-compose file is included in this repository. This docker-compose deploy all the required micro-services. Git clone all the repositories of the project and then inside the document-manager execute the following command:

> docker-compose up -d. In case debugging is required skip the tag -d.

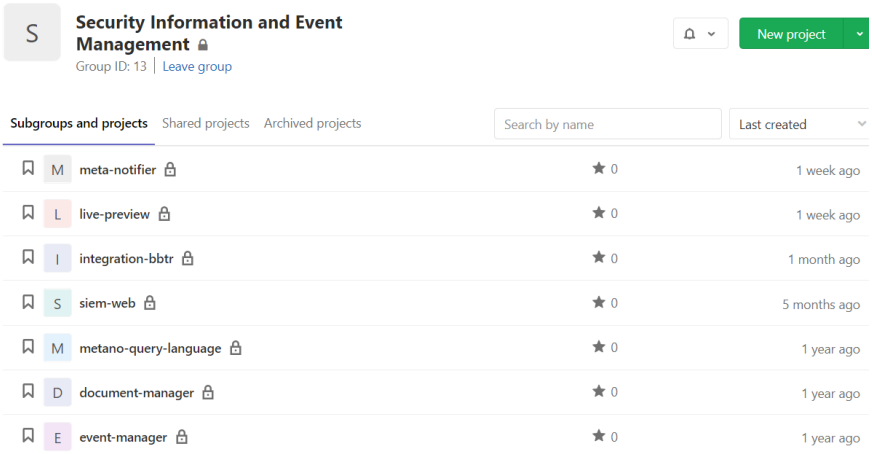

Figure 2.1 SIEM Dashboard

2.3 Deployment with K8S

A K8s folder is included for the required deployment options for Kubernetes cluster deployment.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

3 Overall functions

The menu includes the Input (Upload, Paths, Listeners), Query (Scheduled, Executions) and Database. The Playground function is currently disabled, and the Dashboard is still under development (**Figure 2.1**). The Search function is the overall query for retrieving the events. The following steps are important for engaging into the SIEM:

1. Upload a log file, setup a listener or monitor an internal path for log files. The log files are text-based files and can be unconstructed or constructed files (JSON, CSV, XML). The same approach applies to every of the above options.
2. The user must set up the parser regular expressions and setup where the logfiles will be stored.
3. After the log-files are parsed the user must define the queries which will schedule and will extract information from the parsed log-files according to the specified rules. The user is then possible to see the overall executions.
4. The user can go to the database and check the database-list, while it is possible to create cron jobs which will execute when the defined rules are met. The SIEM can respond executing a shell script, sending Emails or publish data to Kafka Topics when the defined conditions/rules are met.

Every option is previewed in the SIEM main menu (**Figure 2.1**).

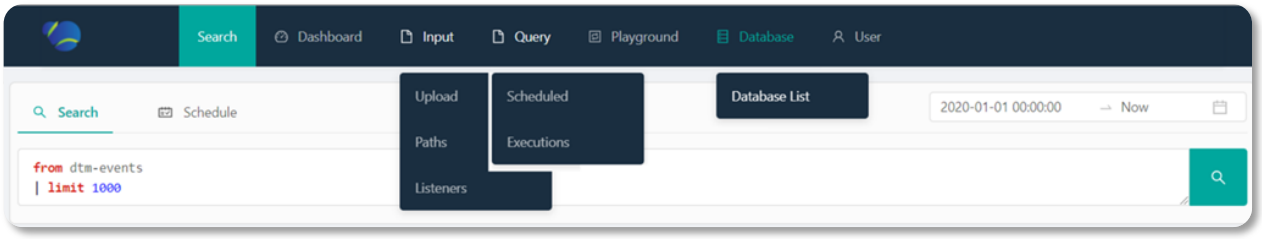

Figure 3.1 **Overall Functions of the SIEM**

Function 01: Parsing and Extraction Rules (Upload): The first step is to upload/retrieve log files. After setting up the listeners or uploading the file it is important to define the structure using regular expressions (**Figure 3.1**). Then the user defines the Index (Index of the Database for log-file data to be stored) and the time-zone. The Source Type (that includes the defined regular expressions) it is possible to be stored for future usage as well. The extraction rules are set, and a preview window is presented (**Figure 3.1**).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

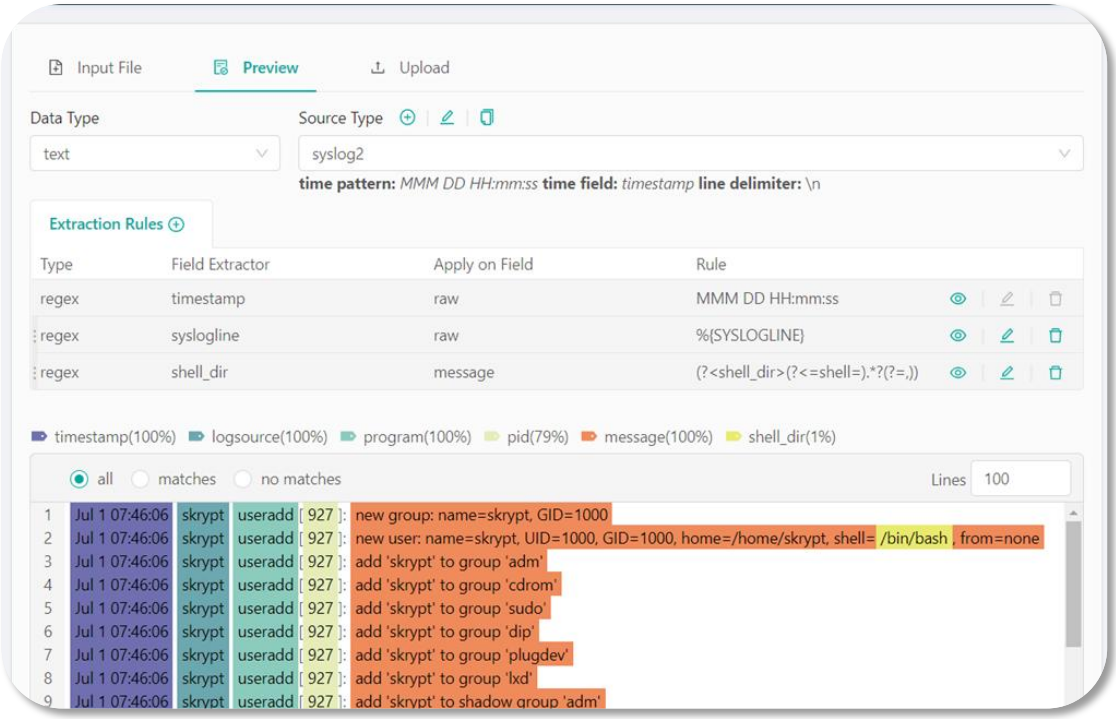

***Figure 3.2* Parsing and usage of regular expressions to parse the log-files**

Monitor Path/Folders for files: Similarly, it is possible to set a monitored path and the SIEM will parse each log file that is inside the specified folder (**Figure 3.3**). The extraction rules must be defined as well and on the advanced tab the index must be defined.

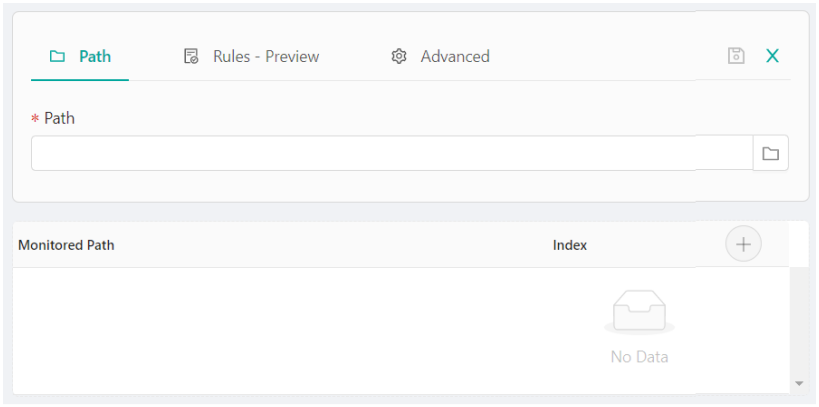

Figure 3.3 Defining a folder to monitor for logfiles

Listeners and Agents: Finally, it is possible to set up a listener to parse log-files from various sources. The listeners can include HTTP requests (agents will be used), TCP and it is possible also to subscribe to Kafka topics to retrieve data directly from Kafka topics (**Figure 3.4**).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

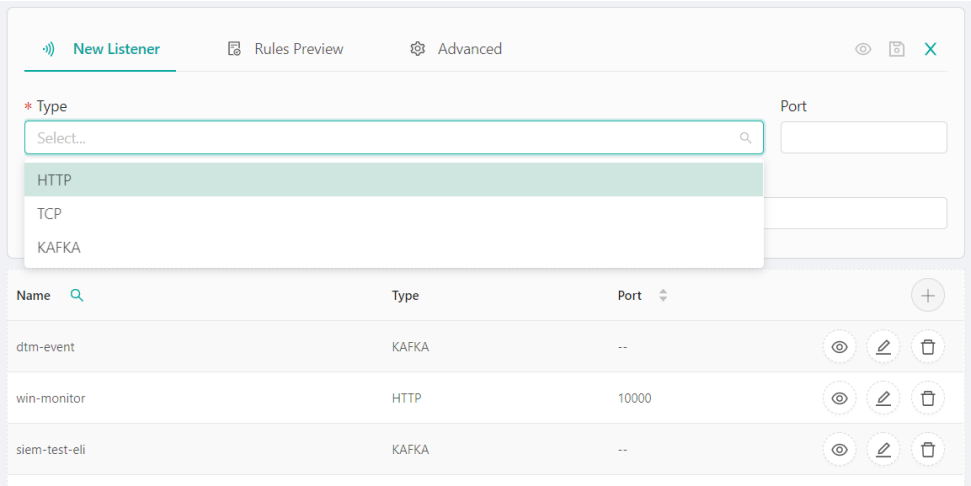

Figure 3.4 Setting up a listener (HTTP, TCP, Kafka subscription)

When setting up the listeners it is possible to have a live-view for debugging purposes and then the extraction rules have to be defined according to the structured. There are three predefined source types, and it is possible to add/save new source types as well.

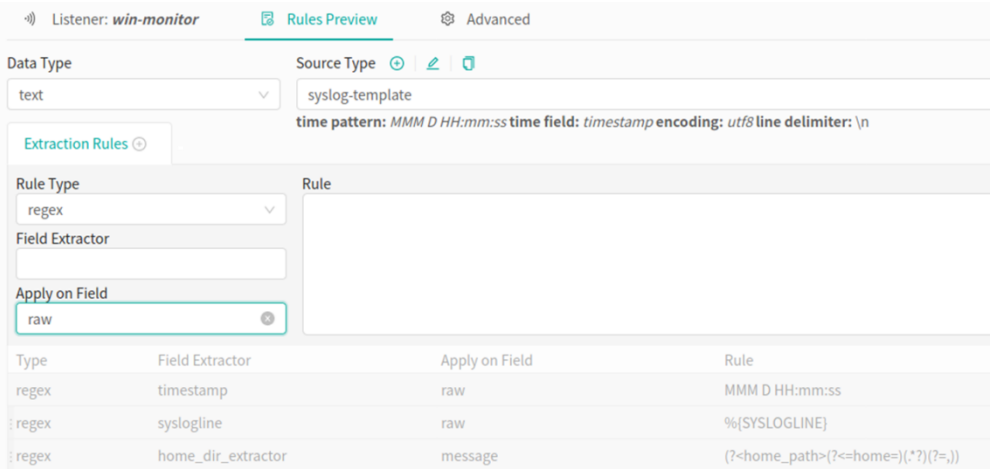

Figure 3.5 Defining the source types and define the extraction rules

For setting up the agents which will contact to the listeners it is important to execute a Go language script which is also compiled as windows binary file. Before executing the agent, it is important to configure which files or it is possible to monitor other interfaces directly (e.g., ethernet interfaces, databases etc.). To configure the agents the config.toml file is required to be edited (**Figure 3.6**).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
#TOML Config File
[outputs.metago]
Urls = [ "metago://localhost:10003" ]

[file.nmap-scan]
Name = "nmap-scan"
Path = "/home/ubutnu/Downloads/nmap.log"
#Encoding = "utf-8"
#SourceType = "syslog"
Index = "pt_syslog"
Disabled = false
Rules = [ "outputs.metago" ]

[file.win-monitor]
Name = "win-monitor"
Path = "/home/siem/Documents/files/win_monitor.csv"
#Encoding = "utf-8"
#SourceType = "syslog"
Index = "win_monitor"
Disabled = true
Rules = [ "outputs.metago" ]

[if.enp0s3]
Name = "enp0s3"
SourceType = "netflow"
Index = "netflow_index"
Rules = [ "outputs.metago" ]
Disabled = true
Fields = [
"time",
"bytes",
"src_ip",
"dst_ip",
"src_port",
"dst_port",
]
```

Figure 3.6 Configuration of the SIEM agents (config.toml)

This file contains information about the IP that the SIEM listeners are set and the files that will be monitored are defined as well (**Figure 3.6**). In the first lines the URL of the SIEM is defined and the port of the matching listener that was defined before (see **Figure 3.4**)**.** Then the tag [file.nmap-scan] defines which file the agent will monitor defining the name, the path where the file is, the index where the events will be stored (this is the index of the Elastic Search) and possible rules that might apply. It is possible to monitor multiple files, and these are defined similarly as presented in (**Figure 3.6**).

In summary, to proceed on the monitoring using the listeners a listener must be created (defining the port number and type of connection), the data type of the monitored data and the extraction rules for converting the raw text to constructed data using tags and the configuration of the agents and execution of them.

Function 02: Queries/Scheduled Executions: The data which are monitored are pushed in the elastic search (database) and now it is important to set scheduled queries which will extract events that apply to specific conditions. For defining the scheduled queries, you have to select the database list and then by selecting a specific index to define the rules that will apply. This way the scheduled queries will match to specific indexes and will extract events according to the predefined conditions. By using the search tab it is possible to test your query (**Figure 3.7**).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

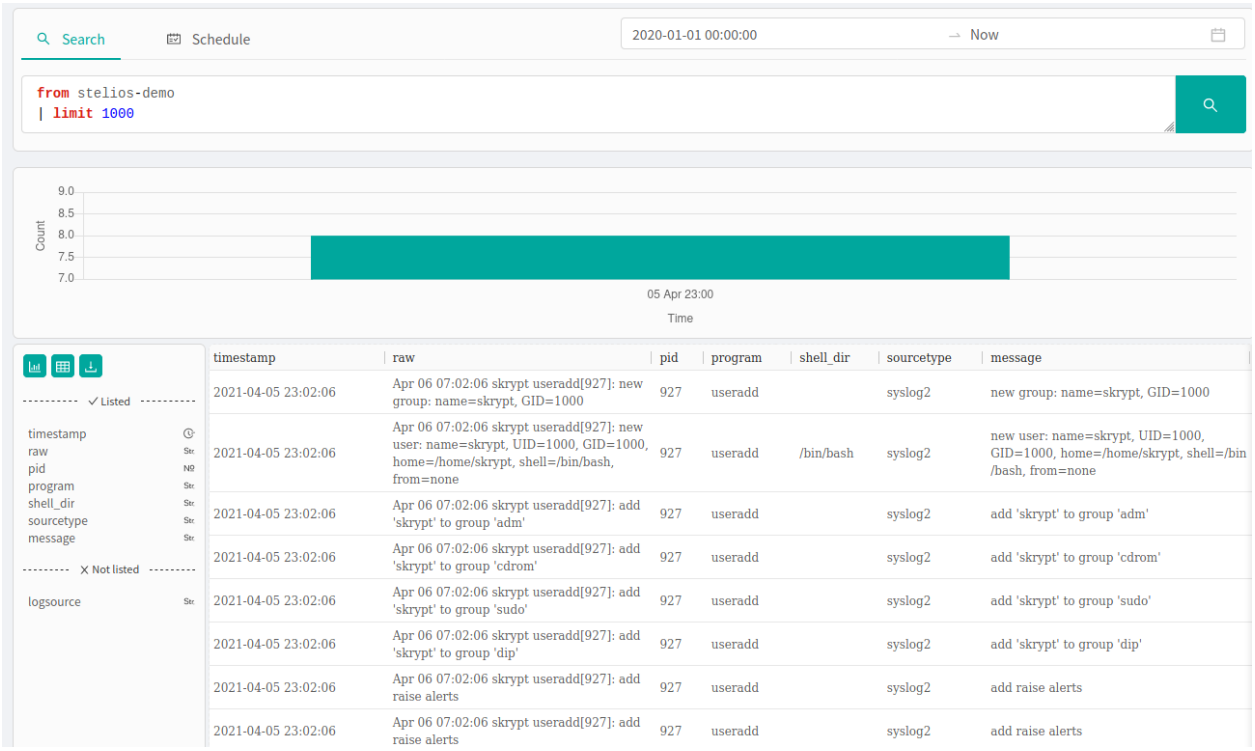

Figure 3.7 Scheduling Queries

The scheduled queries overall can be seen in the main tab Query > Scheduled (**Figure 3.8**). There it is possible to check which executions of the scheduled queries were also done.

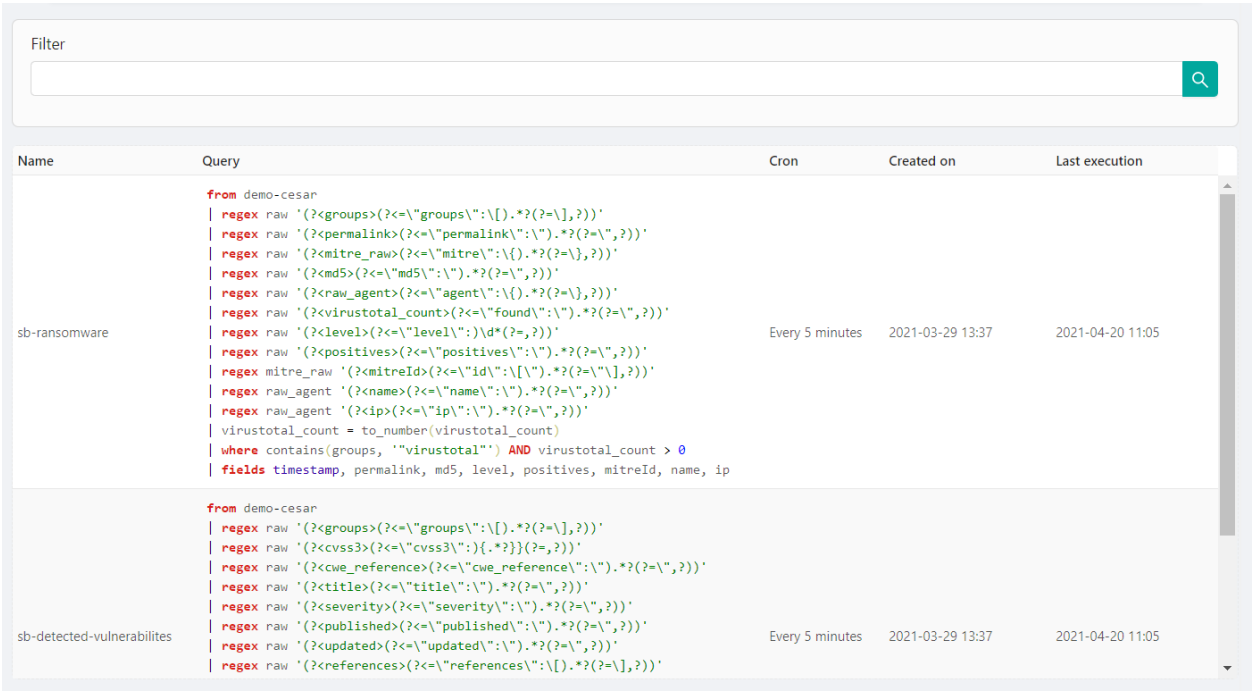

Figure 3.8 Overall scheduled queries

For setting up the queries complex regular expressions might be required. For being easier to handle this task the taglines and values must be checked regarding their consistency and we suggest to go on executing simpler queries in the first place and then to continue to setting up more complex queries. For example in the first query (sb-ransomware) we select the index demo-cesar and we define that if the virustotal_count is higher than 0 to extract an event.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Function 03: Event Investigation/Scheduling/Response: After setting up the query by using the regular expressions it is important to define which actions will be triggered. The possible options include the execution of an Email, API, Shell script and publishing to Kafka topics (**Figure 3.9**).

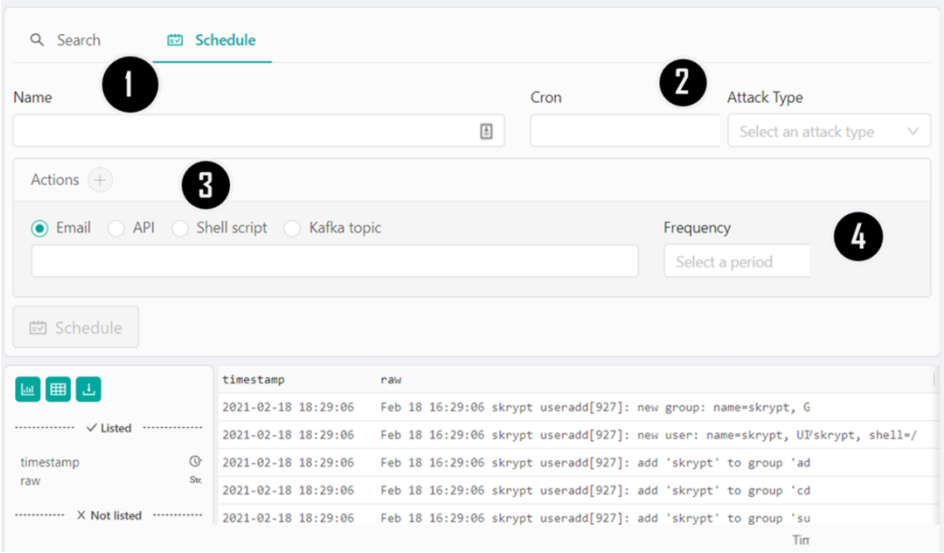

Figure 3.9 Triggered actions/tasks for each query

In 1 the name of the query is defined, 2. The attack type can be optionally defined and 3. the actions can be set. Finally, it is important to set the frequency of whether this query will be scheduled to be executed. Overall an example in **Figure 3.10** describes the DTM. In 3. It is possible to preview the extracted values for each tagline and then in 2 to test an execution of the query. Then it is possible to proceed on the schedule and set the query to execute frequently. In 4. we see the columns for each tagline (e.g., timestamp, destination port, source port etc.).

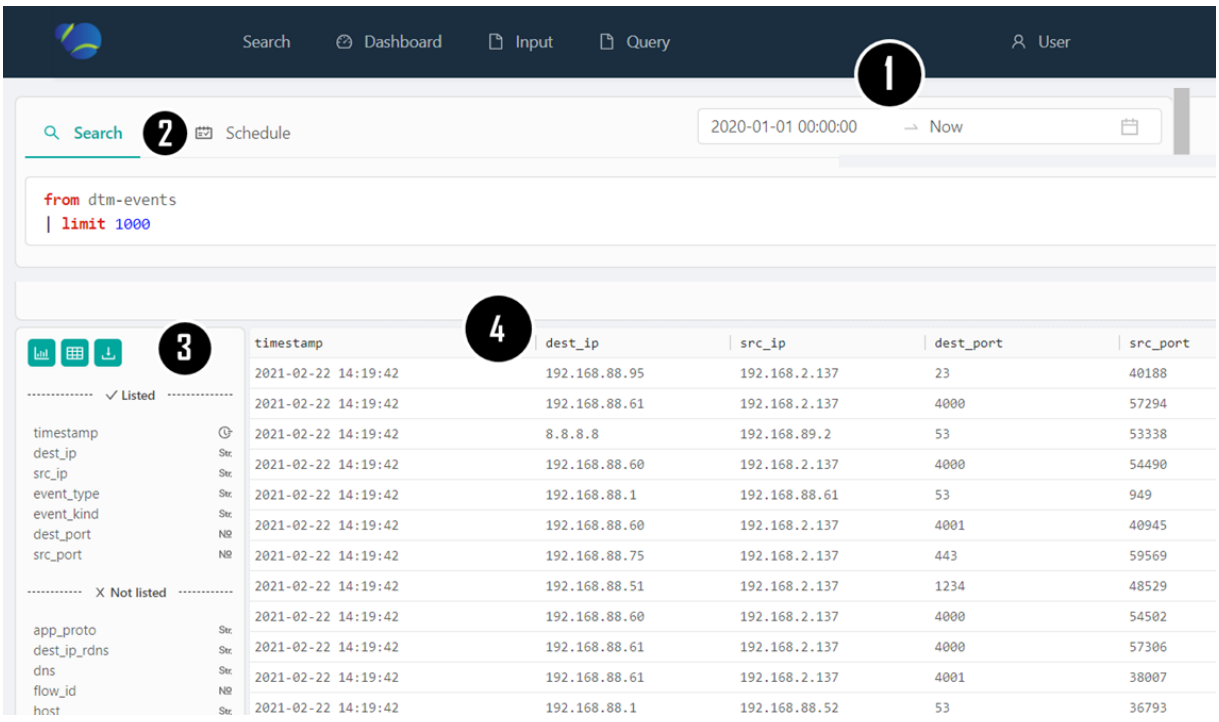

Figure 3.10 DTM overview events

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Possible KPIs: True/False Positives, Insignificant Number Alerts, Mean Response Time that Alert triggers when Incident happens, Deployment options, Engagement/Training.

3.1 Basic Case Examples

We have 2 case-examples for the SIEM usage. **Case-Example 1:** The log files are uploaded or parsed from the SIEM. The tags are generated, and the rules are set. The log files include insights from a ransomware attack and the SIEM published the corresponding threat intelligence to the Kafka topics and sending also to the BBTR. **Case-Example 2:** A vulnerability reports is presented to the SIEM and this is published to the Kafka topics for the RCRA and other components to get information regarding the vulnerabilities from systems.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.