Data Communications, Computer Networks and Protocols – Project

Brno University of Technology

Petr Stehlík <xstehl14@stud.fit.vutbr.cz>

April 20, 2017

1 Assignment

The task in hand was to create whitelisting of detected alerts for the NEMEA system with connection to the vizualization part. Whitelisting is only a partial feature which was needed. Because of this the NEMEA Alert Filtering 2.0 was created.

This document describes the analysis and implementation of this system while defining the format for configuration which is the cornerstone of the whole system. The configuration should be central for every reporting module available.

2 Analysis

2.1 System Architecture

The current reporters architecture is modular and easily expandable. Each reporter provides a callback conversion function (`conv_func`) which converts given alert to IDEA[1] message format to the main module `report2idea`.

`report2idea` connects to the TRAP interface, receives alerts and performs desired actions – stores alert to MongoDB, resends alert to TRAP interface or reports the alert to Warden system. These options depends on arguments passed to the reporter on startup.

The new architecture expands the `report2idea` module where on startup a YAML configuration file is loaded, checked and parsed by `report_config` module. When an alert is received it is matched against rules specified in the configuration and actions or elseactions are performed (more about this in section 2.2).

`report_config` uses MENTAT Filtering[2] module to parse conditions for rules and match alerts against them. Mentat internally uses its own module `jpath` to traverse paths in python dictionaries which is used in conditions to retrieve values from an alert.

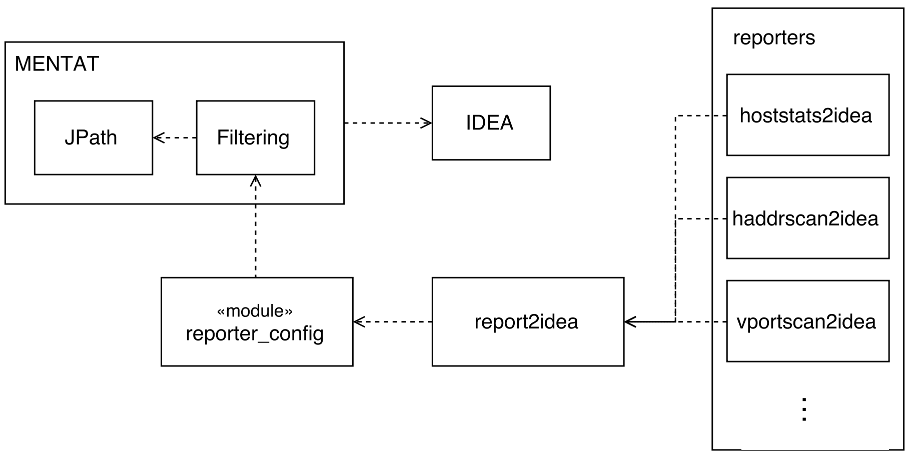


Figure 1: NEMEA reporters configuration architecture

[1]https://idea.cesnet.cz

[2]The lead creators are Jan Mach, Pavel Kácha and Andrea Kropáčová

2.2 Configuration Design

The configuration is divided into several parts: rules, address groups, custom actions and actions. Each part will be described in detail. The example configuration can be seen at 5.1.

2.2.1 Rules

The list of rules is ordered by ID. The list is sorted by ID and evaluated by all reporters from the first rule to the last one. Each rule is composed of ID, condition, actions and elseactions which is an optional item.

The unique ID is an integer value by which the rules are sorted and evaluated.

Filtering condition consists of unary and binary operations, constants and JSON paths that are parsed by the Mentat filtering module created by the Mentat team. When an alert meets the condition, list of actions is performed in specified order. If list of elseactions is defined and the filtering condition is not met this list of actions is performed.

2.2.2 Actions

Each rule specifies a list of actions that are of two types: implicit and custom actions. Currently the only implicit action is to `drop` the alert which will imidiatelly stop processing the alert. This action mustn't be specified in custom actions.

The `custom_actions` is a list of actions identified by ID string key – this identifier is used to reference an else/action in a rule. Each custom action must specify its type. Currently available types of custom actions are as follows:

- file – store message into file
  - dir[boolean] – if set to `False` each message is stored in separate file, otherwise it is appended to specified file
  - path[string] – path to file/folder respectively to dir option
- email – send alert via email
  - to[string] – recipient email address
  - subject[string] – subject field
- mongo – store alert to MongoDB
  - host (optional)[string] – MongoDB instance host (defaults to *localhost*
  - port (optional)[number] – MongoDB instance port (defaults to *27017*
  - db[string] – database name
  - collection[string] – collection name
  - username (optional)[string] – MongoDB username (needed in case of enabled authentication)
  - password (optional)[string] – MongoDB user's password (needed in case of enabled authentication)
- mark – add value into alert/modify IDEA message
  - path[JSONPath] – JSONPath in alert, if non-existent it is created
  - value[string] – value to add
- trap – send the message via TRAP IFC[3] if it is opened for the reporter. Otherwise, the action takes no effect.

[3]http://nemea.liberouter.org/trap-ifcspec/

2.2.3 Address Groups

Condition might refer to a named list of addresses. This feature extends the Mentat Filtering. Each address group is identified by its string ID. Each address group must specify a file or a list of IP addresses/subnetworks. An address group file is a list of IP addresses/subnetworks separated by newline.

3 Implementation

The whole system of NEMEA reporters and Mentat Filtering is implemented in Python and therefore the `report_config` is a Python module as well. The report2idea takes `config` argument which specifies path to configuration file.

Instance of Config class takes path to configuration file. The file is loaded and parsed by PyYAML module to dictionary. Afterwards Mentat Filtering is initialized in order to parse rules conditions. Next address groups are parsed. Depending on given configuration it can load and parse file address group which translates to a list.

With address groups parsed it continues to parse custom actions. Each action is recognized by its type and an Action class is instanciated accordingly. This enables to create unlimited number of types and its actions in the future. Each action takes different arguments as specified in 2.2.2 and implements `run` method which performs desired action. Right after custom actions parsing the implicit drop action is added as well.

The last part on inicialization procedure is rules parsing. Each rule takes its dictionary representation, list of actions, list of address groups and optionally a Mentat Filtering instance (if not specified each rule creates its own instance). Rule then parses its condition – finds all occurences of address groups IDs and replaces them with a list of IP addresses/subnetworks and then parses the condition via Mentat Filtering. Finally the rule links all its specified actions and elseactions with actions available in custom actions and in implicit actions.

Afterwards the report2idea `Run` function starts to receive alerts from TRAP interface. Each received alert is converted to IDEA message, matched against available rules and the list of actions or elseactions is performed accordingly. This continues until TRAP interface or the reporter is terminated.

The visualization part of whitelisting is done via a mark action (as shown in 5.1 in *markwhitelisted* custom action). It adds *Whitelisted* flag to the alert with boolean value. Afterwards Liberouter GUI NEMEA Events module which displays alerts saved to MongoDB looks for such flag and if hiding of whitelisted alerts is enabled, the alert is not displayed. Otherwise such action is visually different from non-whitelisted alerts as seen in 2.

4 Conclusion

NEMEA Alert Filtering 2.0 was designed and implemented successfully. Only minimal changes were needed in reporters (the only change is in arguments passed to the reporter). During the development of the module a bug in Mentat Filtering was found and fixed.

Further work will focus on autonomous recognition of configuration and addressgroups changes which will trigger reload and parsing of the updated configuration, deeper integration with NEMEA system and Liberouter GUI which will enable to edit the configuration comfortably via web interface with minimal margin for error.

The whole project is available on Github in Nemea-Framework repository[4].

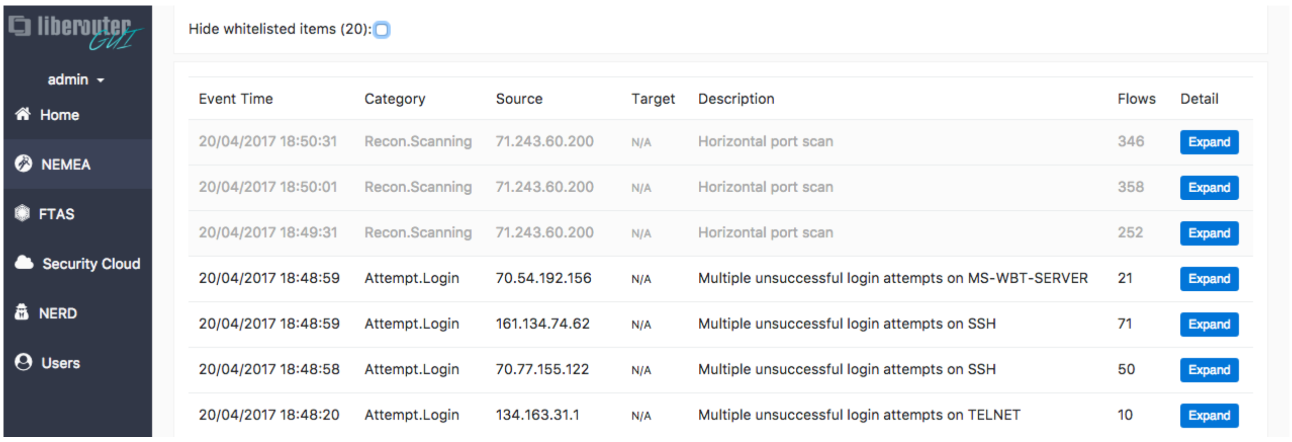
[4]https://github.com/CESNET/Nemea-Framework/tree/reporter-config/pycommon

5 Annexes

5.1 Example Configuration File

```
custom_actions:
- id: mongo1
  type: mongo
  host: localhost
  db: nemeadb
  collection: alerts
- id: marktest
  type: mark
  mark:
    path: Test
    value: 'FooBar'
- id: markwhitelisted
  type: mark
  mark:
    path: _CESNET.Whitelisted
    value: 'True'
- id: savefile
  type: file
  dir: true
  path: "/var/log/nemea/events"
addressgroups:
- id: main_whitelist
  file: "/etc/nemea/reporters/whitelists/whitelist1"
- id: whitelist2
  list:
  - 1.1.0.0/24
  - 1.2.3.4
rules:
- id: 1
  condition: Source.IP4 in main_whitelist or Target.IP4 in main_whitelist
  actions:
  - markwhitelisted
  - mongo1
  - drop
  elseactions:
  - marktest
  - savefile
  - mongo1
- id: 2
  condition: >
   Node.SW == 'detector1' and
   Category == 'Recon.Scanning' and
   Target.IP4 in whitelist2
  actions:
  - marktest
- id: 3
  condition: Source.IP4 == whitelist2
  actions:
  - drop
- id: 4
  condition: Node.SW in ['detector1', 'detector2']
  actions:
  - savefile
  - drop
```

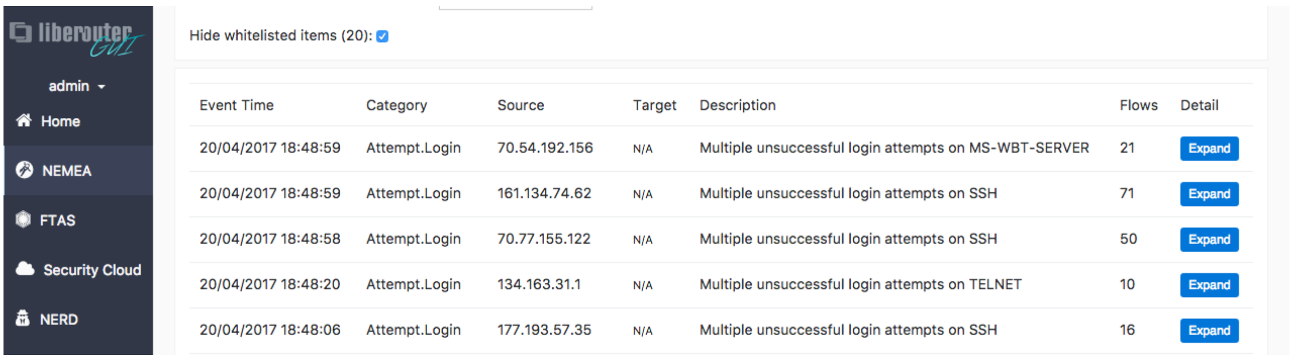
5.2 Whitelisting visualization



Hide whitelisted items (20): ☐

Event Time	Category	Source	Target	Description	Flows	Detail
20/04/2017 18:50:31	Recon.Scanning	71.243.60.200	N/A	Horizontal port scan	346	Expand
20/04/2017 18:50:01	Recon.Scanning	71.243.60.200	N/A	Horizontal port scan	358	Expand
20/04/2017 18:49:31	Recon.Scanning	71.243.60.200	N/A	Horizontal port scan	252	Expand
20/04/2017 18:48:59	Attempt.Login	70.54.192.156	N/A	Multiple unsuccessful login attempts on MS-WBT-SERVER	21	Expand
20/04/2017 18:48:59	Attempt.Login	161.134.74.62	N/A	Multiple unsuccessful login attempts on SSH	71	Expand
20/04/2017 18:48:58	Attempt.Login	70.77.155.122	N/A	Multiple unsuccessful login attempts on SSH	50	Expand
20/04/2017 18:48:20	Attempt.Login	134.163.31.1	N/A	Multiple unsuccessful login attempts on TELNET	10	Expand

Figure 2: Visually separated whitelisted alerts in Liberouter GUI NEMEA Events



Hide whitelisted items (20): ✓

Event Time	Category	Source	Target	Description	Flows	Detail
20/04/2017 18:48:59	Attempt.Login	70.54.192.156	N/A	Multiple unsuccessful login attempts on MS-WBT-SERVER	21	Expand
20/04/2017 18:48:59	Attempt.Login	161.134.74.62	N/A	Multiple unsuccessful login attempts on SSH	71	Expand
20/04/2017 18:48:58	Attempt.Login	70.77.155.122	N/A	Multiple unsuccessful login attempts on SSH	50	Expand
20/04/2017 18:48:20	Attempt.Login	134.163.31.1	N/A	Multiple unsuccessful login attempts on TELNET	10	Expand
20/04/2017 18:48:06	Attempt.Login	177.193.57.35	N/A	Multiple unsuccessful login attempts on SSH	16	Expand

Figure 3: Hidden whitelisted alerts in Liberouter GUI NEMEA Events