An Innovative Self-Healing Approach with STIX Data Utilisation

Arnolnt Spyros[1][a], Konstantinos Rantos[2][b], Alexandros Papanikolaou[1][c] and Christos Ilioudis[3][d]

[1]*Innovative Secure Technologies P.C., Thessaloniki, Greece*

[2]*Department of Computer Science, International Hellenic University, Kavala, Greece*

[3]*Department of Information and Electronic Engineering, International Hellenic University, Thessaloniki, Greece*

Keywords: Self-Healing Mechanisms, Cyberdefense, Cyberthreats, STIX, Mitigation.

Abstract: Organisations nowadays devote many resources in maintaining a robust security posture against emerging cyber-threats. This typically requires rapid response against newly identified or shared threat information so that appropriate countermeasures are immediately deployed to eliminate these threats or reduce the associated risks. For many shared indicators, like malicious IPs or URLs, such a response might only require minor modifications to the configuration of security appliances. Self-Healing systems are the mechanism that allows a system to discover any misconfigurations and apply the necessary corrections in an automated or semi-automated manner. This paper proposes such a mechanism that can be deployed within large organisations that either do not have the resources to devote in security and therefore automation is one of their main priorities, or they outsource their infrastructure's protection. The use of such a mechanism can relax the increased need for human resources and can also reduce response times in confronting emerging threats. The architecture and the details of a reference implementation for local public administrations is also provided.

1 INTRODUCTION

The complexity of modern computing environments continuously increases and poses significant challenges to organisations with regards to the efficiency and reliability of systems. As modern computing environments require more effort to properly control and manage, human administrators are sometimes not able to cope effectively with the aforementioned challenges. This makes systems more expensive to manage and maintain, as well as vulnerable to faults or external threats. Moreover, the increased system complexity makes them more error-prone to human errors.

Self-Healing systems are introduced as a useful approach in addressing the rising complexity requirements of systems management (Schneider et al., 2015; Keromytis, 2007). Self-Healing is described as the ability of systems to autonomously diagnose and recover from faults with transparency and within certain criteria. Self-Healing offers advantages such as reducing the required human interaction, system maintenance cost and the required workload of human administration. Furthermore, Self-Healing systems enhance durability and improve existing mitigation techniques. One of the main advantages of Self-Healing is that fault mitigation is accomplished either autonomously, i.e. without human interaction or require partial human interaction.

This paper proposes a Self-Healing mechanism that is specifically designed to facilitate rapid response to emerging threats, based on the information that the organisation receives in the context of cyber-threat information sharing (Schaberreiter et al., 2019a; Rantos et al., 2020). The aim is to provide administrators with the means to (semi-)automatically and remotely adjust their security appliances to confront cybersecurity threats. As such, it relaxes the needs for many human resources devoted to constant monitoring and adjustment of their configurations. This is particularly important for organisations, such as local public administrations, that have multiple devices to monitor but not necessarily have the required resources. The mechanism proposed in this paper was designed and developed in the context of CS-AWARE (A cybersecurity situational awareness and information sharing solution for local public administrations based on advanced big data analysis), a H2020 re-

[a] https://orcid.org/0000-0002-4681-104X
[b] https://orcid.org/0000-0003-2453-3904
[c] https://orcid.org/0000-0002-0251-0990
[d] https://orcid.org/0000-0002-8084-4339

Spyros, A., Rantos, K., Papanikolaou, A. and Ilioudis, C.
An Innovative Self-Healing Approach with STIX Data Utilisation.
DOI: 10.5220/0009893306450651
In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications (ICETE 2020) - SECRYPT*, pages 645-651
ISBN: 978-989-758-446-6

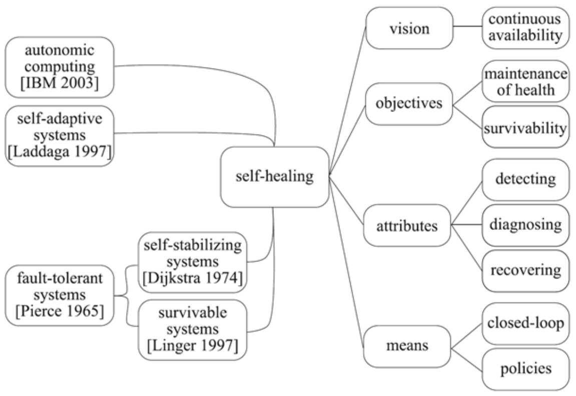
search project, which aims to provide a cybersecurity situational awareness solution for small-to medium-sized IT infrastructures.

The rest of this paper is structured as follows: Section 2 provides background information about the introduction of Self-Healing mechanisms in organisations' infrastructures. Section 3 introduces the Self-Healing mechanism in the context of the CS-AWARE project, while Section 4 describes the architecture of the proposed solution and Section 5 provides the details of a reference implementation. Section 6 concludes the paper.

2 BACKGROUND

The concept of Self-Healing is to classify and analyse sensory data in order to autonomously detect and mitigate potential system faults. The main properties of Self-Healing implementation are fault detection, failure root cause diagnosis and deriving a remedy, and recovering with a sound strategy (Psaier and Dustdar, 2011). One categorisation of Self-Healing implementation is the level of supervision. Supervision is referred to as the degree of required human interaction concerning the feedback mechanism and the expansion of Self-Healing mechanisms. Self-Healing systems are categorised into fully supervised, semi-supervised or unsupervised (Psaier and Dustdar, 2011; Dean et al., 2012).

One of the main advantages of Self-Healing is that fault mitigation can be accomplished either autonomously, i.e. without human interaction, or require partial human interaction, and therefore reduce the required human interaction, system maintenance cost and the required workload of human administration. Figure 1 provides an overview of the research work accomplished on Self-Healing properties as published in (Psaier and Dustdar, 2011).



autonomic computing [IBM 2003]
self-adaptive systems [Laddaga 1997]
self-healing
fault-tolerant systems [Pierce 1965]
self-stabilizing systems [Dijkstra 1974]
survivable systems [Linger 1997]
vision
continuous availability
objectives
maintenance of health
survivability
attributes
detecting
diagnosing
recovering
means
closed-loop
policies

Figure 1: Self-Healing Systems (Psaier and Dustdar, 2011).

There are many Self-Healing models which can be categorised based on specific criteria. Some of the most important criteria are:

- Management style.
- Application environment.
- Learning methodology.

Another categorisation of Self-Healing implementation is the level of required supervision, i.e. the degree of required human interaction concerning the feedback mechanism and the expansion of Self-Healing mechanisms. Self-Healing systems are categorised as fully supervised, semi-supervised or unsupervised. Fully supervised Self-Healing implementations require frequent human interaction, semi-supervised implementations require periodic human interaction and unsupervised implementations operate autonomously without requiring human interaction.

Although unsupervised implementations have more advantages than semi-supervised and unsupervised approaches, the most common approach of Self-Healing implementations are top-down fully supervised Self-Healing systems. This is due to the fact that semi-supervised and unsupervised approaches are more complex. Unsupervised implementations are usually implemented through the use of machine learning, thus require more resources, Furthermore, Self-Healing systems that exhibit self-elected behaviours (system behaviour that derives from autonomous configuration changes) that have not been previously tested and validated, are more difficult to maintain and meet the operational goals. Supervised Self-Healing implementation provides a controllable environment in which the produced mitigation solutions have been validated by the administrator who is also aware of the system's behaviour.

Self-Healing mechanisms have also been designed for specific domains, addressing the requirements thereof, like the ones proposed for smart grids (Elgenedy et al., 2015; Zidan and El-Saadany, 2012).

3 CS-AWARE AND SELF-HEALING

CS-AWARE enables detection, classification and visualisation of cybersecurity incidents in real-time, supporting the prevention or mitigation of cyber attacks. The solution CS-AWARE provides is a big step towards automation of cyber incident detection, classification and visualisation, and is based on mature big data analysis tools and methodologies. An overview of the CS-AWARE architecture is provided in (Schaberreiter et al., 2019b).

In the context of the CS-AWARE project, an innovative Self-Healing approach has been developed to provide a (semi-) automated external Self-Healing system which utilises Cyber Threat Intelligence. The rest of this paper refers to the Self-Healing system as Self-Healing component.

Other CS-AWARE components associated with the Self-Healing one are the Data Analysis & Pattern Recognition component and the Visualisation component. The Data Analysis & Pattern Recognition component provides input data to the Self-Healing component which the latter analyses in order to compose a mitigation rule. After the successful composition of the mitigation rule the Self-Healing component enriches the data received from the Data Analysis component and provides them to the Visualisation component.

The Cyber Threat Intelligence data that are being exchanged are structured in JSON format in compliance with the STIX 2.0 format (STIX, 2017). STIX 2.0 was chosen instead of STIX 1.x due to the advantages offered by the use of JSON schemas.

4 SELF-HEALING ARCHITECTURE

The Self-Healing component consists of three main subcomponents and three auxiliary subcomponents. The main subcomponents are defined in the deliverable D2.4 of the CS-AWARE project (CS-AWARE, 2018). Auxiliary subcomponents were designed during the development process in order to facilitate the composition of mitigation rules.

4.1 Main Subcomponents

4.1.1 Self-Healing Policies

Self-Healing Policies is a database which contains records of potential threats that might be detected in an LPA (Local Public Administration) system and the corresponding mitigation rules. The mitigation rules are stored in a human-readable format as well as in machine-readable format. Self-Healing Policies are implemented on a MySQL database whose tables describe threats, mitigation rules and policies.

Records in these tables contain fields which determine the *threat type* as well as other information related to the detected incident. This information is utilised by the Decision Engine subcomponent to determine which policies apply to the detected threat.

Moreover, the Self-Healing Policies subcomponent includes entries which contain the CLI (command-line interface) syntax of LPAs central nodes.

4.1.2 Decision Engine

The Decision Engine initiates the composition of a rule when a match is found. The matching process is done by scanning the Threat table in the Self-Healing policies subcomponent database based on the Threat Type field in order to identify the threat. Subsequently, Decision Engine performs a scan on the Policies table for a matching rule.

In case of a successful match, Decision initiates its composition. The output of Decision Engine subcomponent is a rule in a human-readable format.

4.1.3 Security Rules Composer

Security Rules Composer accepts input from the Decision Engine subcomponent. The Decision Engine provides a rule in a human-readable format as input data and Security Rules Composer subcomponent converts the mitigation rule in a machine-readable format based on the CLI syntax of the affected node. In case that the vendor of the affected node does not provide CLI, then the mitigation rule cannot be converted to a machine-readable format and the Self-Healing provides the rule in a human-readable format as a recommendation.

4.2 Auxiliary Subcomponents

4.2.1 Parser

The Parser parses the STIX package and extracts useful data for the process of mitigation rules composition. The extracted data is used for the initialisation of Java objects which contain information about detected threats, the affected LPA and the affected node or nodes.

4.2.2 Rule Applicator

Rule Applicator is responsible for enriching the STIX package with the mitigation rule, sending data to the Visualisation component and for applying the rule on the remote machine. Rule Applicator enriches the STIX package with a Course of Action SDO (STIX Domain Object) which contains information about the mitigation rule composed by the Self-Healing. Subsequently, Rule Applicator sends the enriched STIX package to the Visualisation component via a REST

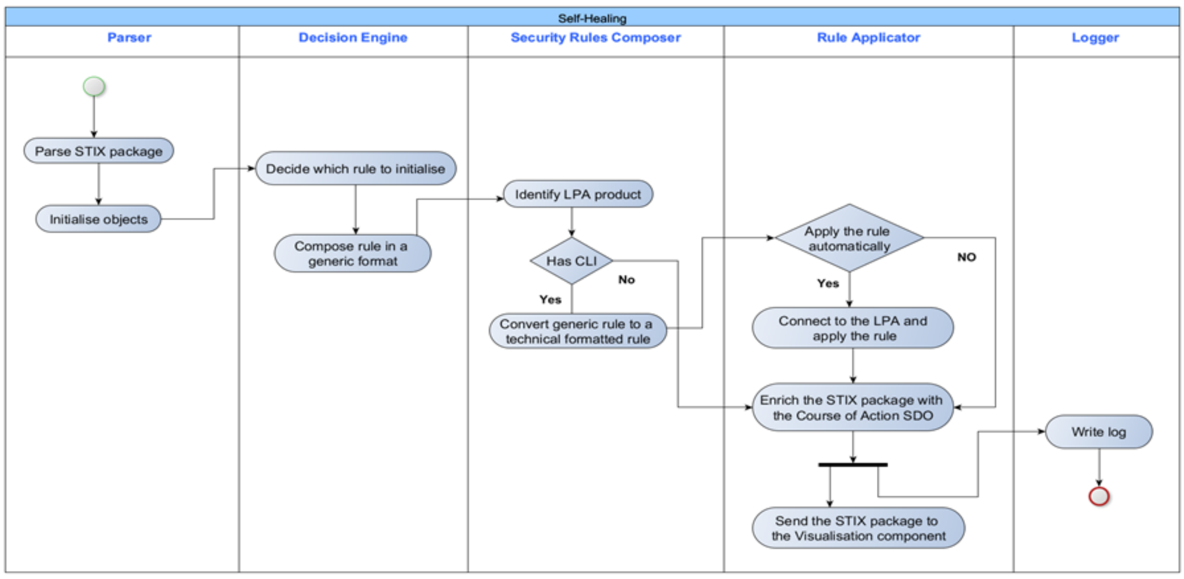

Self-Healing
Parser
Decision Engine
Security Rules Composer
Rule Applicator
Logger
Parse STIX package
Initialise objects
Decide which rule to initialise
Compose rule in a generic format
Identify LPA product
Has CLI
No
Yes
Convert generic rule to a technical formatted rule
Apply the rule automatically
NO
Yes
Connect to the LPA and apply the rule
Enrich the STIX package with the Course of Action SDO
Send the STIX package to the Visualisation component
Write log

Figure 2: Self-Healing activity diagram.

API. In case the mitigation rule must be applied remotely then the Self-Healing connects to the remote node via SSH using credentials that are defined exclusively for the Self-Healing component. Self-Healing has the required privileges on the affected node in order to apply mitigation rules without encountering any restrictions. The connection is established only if the LPA admin has provided such permission to the Self-Healing component. In case the STIX package does not include sufficient information about the threat source, then the Self-Healing provides a generic mitigation rule as a recommendation.

4.2.3 Logger

Logger writes a log entry in the log file which contains information about how the mitigation rule was implemented. Mitigation rules composed by the Self-Healing component incorporate three alternatives: Inform LPA admin upon which acts to perform in order to avoid the threat or reduce the impact, ask for admin's permission in order to apply the mitigation rule, automatically apply the rule.

4.2.4 Input-output

Self-Healing component accepts input data from the Data Analysis & Pattern Recognition component via a REST API. They contain various information about detected threats and the affected LPA which is used to determine the proper mitigation rule (Table 1).

4.3 Relevant Dynamic Behaviour

The Self-Healing component operations are organised in four levels:

- Data parsing (performed by the Parse subcomponent),
- Human-readable mitigation composition (performed by the Decision Engine subcomponent),
- Machine-readable mitigation composition (performed by the Security Rules Composer subcomponent), and
- Mitigation application (performed by the Rule Applicator subcomponent).

Figure 2 shows the interaction among the Self-Healing sub-components while Figure 3 demonstrates the interaction between the Self-Healing and Visualisation component as a result of the input provided by the Data Analysis & Pattern Recognition component.

Received data is first parsed to retrieve useful data which is then used as input data for the Decision Engine to diagnose the threat type and initiate the corresponding mitigation rule which is provided in a human-readable format. The mitigation rule is then forwarded to the Security Rules Composer subcomponent to query the Self-Healing Policies database if the LPA security mechanism has a CLI and determine whether the human-readable rule can be converted to a machine-readable set of actions.

- Security mechanism does not support CLI: The mitigation rule cannot be converted to a machine-readable format and therefore is directly provided

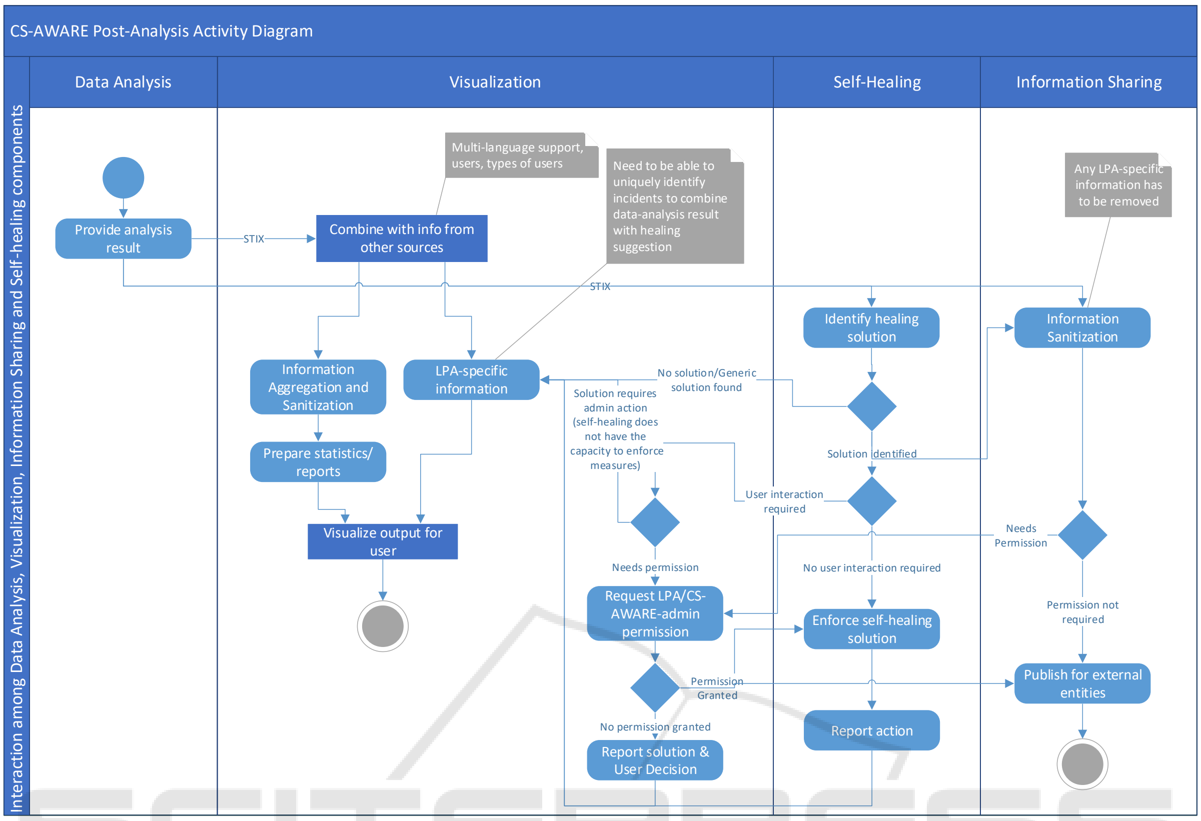

CS-AWARE Post-Analysis Activity Diagram
Interaction among Data Analysis, Visualization, Information Sharing and Self-healing components
Data Analysis
Visualization
Self-Healing
Information Sharing
Provide analysis result
STIX
Combine with info from other sources
Multi-language support, users, types of users
Need to be able to uniquely identify incidents to combine data-analysis result with healing suggestion
Any LPA-specific information has to be removed
STIX
Identify healing solution
Information Sanitization
Information Aggregation and Sanitization
LPA-specific information
No solution/Generic solution found
Solution requires admin action (self-healing does not have the capacity to enforce measures)
Solution identified
Prepare statistics/ reports
User interaction required
Visualize output for user
Needs Permission
Needs permission
No user interaction required
Request LPA/CS-AWARE-admin permission
Enforce self-healing solution
Permission not required
Permission Granted
Publish for external entities
No permission granted
Report action
Report solution & User Decision

Figure 3: Post-Analysis activity diagram.

as a recommendation. The received STIX package is enriched with a Course of Action SDO and a Relationship SDO which links the mitigation with the given threat and then forwarded to the Visualisation component. Self-Healing writes a log entry in the log file.

- Security mechanism supports CLI: The mitigation rule is converted in a machine-readable format based on product's CLI syntax. The machine-readable rule is forwarded to the Rule Applicator that forwards the rule to the Visualisation component in order to ask for the LPA administrator's permission to apply the rule automatically.
  – Administrator accepts the automatic application: The mitigation rule is applied on the affected machine. The received STIX package is enriched with a Course of Action SDO and a Relationship SDO which links the mitigation with the given threat, and is then forwarded to the Visualisation component. Self-Healing writes a log entry in the log file.
  – Administrator declines the automatic application: The mitigation rule is provided as a recommendation. The received STIX package is enriched with a Course of Action SDO and a Relationship SDO which links the mitigation with the given threat, and is then forwarded to the Visualisation component. Self-Healing writes a log entry in the log file.

Each Self-Healing subcomponent supports a specific set of operations which diagnose the threat type and compose the proper mitigation rule. At first the mitigation rule is composed in a human-readable format. Subsequently the human-readable mitigation is converted to a machine-readable rule which can then be applied on the affected node remotely and without requiring human interaction.

5 USE CASE

A STIX package is provided to the Self-Healing component as input. The STIX package contains information about a Brute Force Attack that was detected in an LPA system. Apart from threat information, the package contains information concerning the affected LPA. After finishing the parsing process the Self-Healing extracts the following information:

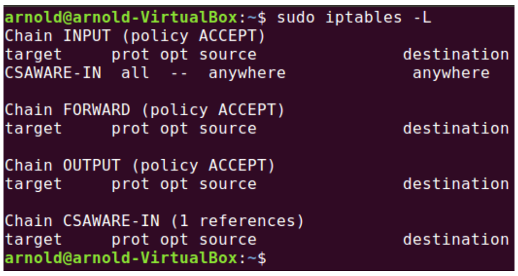
1. The Brute Force Attack is performed from the 123.183.209.131 IPv4 address.

Table 1: Fields of the STIX2.0 package received from the Data Analysis & Pattern Recognition component.

NAME	**DESCRIPTION**
Threat id	Threat's STIX package id.
Threat type *(optional)*	DoS, DDoS, Ransomware, etc.
Threat group	Server threats, Network threats, System threats, or Database Threats
LPA	Name of the LPA at which the threat was detected
Severity (LPA-specific)	Level of threat's severity with regards to the LPA.
Tool used (optional)	Information about the tool that was used to develop or spread the threat.
Affected Products	Malicious or vulnerable products info.
LPA Affected Products	LPA malicious or vulnerable products information.
OS information	OS distribution, OS version, OS system type.
Firewall information	Firewall software name.
Risk Mitigation Strategy	Actions to mitigate the detected threat.

2. The affected node runs Linux Ubuntu 16.04.
3. The affected node uses the `iptables` as firewall.

For the needs of the Use Case test the affected LPA system is virtualised using a VM (Virtual Machine) created with the Oracle VM VirtualBox. Initially, the VM contains only the default chains as well as the CSAWARE-IN custom chain (see Figure 4).



```
arnold@arnold-VirtualBox:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
CSAWARE-IN  all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain CSAWARE-IN (1 references)
target     prot opt source               destination
arnold@arnold-VirtualBox:~$
```

Figure 4: VM iptables chains listing before applying mitigation rule.

The steps followed by the Self-Healing systems as a response to receiving the aforementioned malicious IP are the following:

1. The Self-Healing component receives the following STIX package as input data.

```
{
  "type":"bundle",
  "id":"bundle--ab68225c-91cf-47fc-b27b
      -78949b703437",
  "objects":[
    {
      "type":"indicator",
      "id":"indicator--6742956c-f31a-465f-
          bde8-1453822d9948",
      "created":"2018-05-06T06:34:55.000Z",
      "modified":"2018-05-08T13:49:12.000Z
          ",
      "name":"Brute Force Attack",
      "pattern":"[ipv4-addr:value =
          '123.183.209.131']",
      "valid_from":"2018-05-06T06:34:55Z",
      "labels":[
        "malicious-activity"
      ]
    },
    {
      "type":"identity",
      "id":"identity--023d105b-752e-4e3c
          -941c-7d3f3cb15e9e",
      "created":"2016-08-23T18:05:49.307Z",
      "modified":"2016-08-23T18:05:49.307Z
          ",
      "name":"affected_LPA_x",
      "identity_class":"organization"
    },
    {
      "type":"observed-data",
      "id":"observed-data--32dab7d0-1522-4
          a08-826b-015fc1369bdb",
      "created":"2016-08-23T18:05:49.307Z",
      "first_observed":"2018-03-26T14
          :21:25.283Z",
      "last_observed":"2018-03-26T14
          :21:25.283Z",
      "modified":"2018-03-26T14:21:25.283Z
          ",
      "number_observed":1,
      "objects":{
        "0":{
          "name":"Canonical Ubuntu Linux
              16.04 LTS",
          "type":"software",
          "vendor":"Canonical",
          "version":"7.0",
          "cpe":"cpe:2.3:o:canonical:
              ubuntu_linux:16.04:*:*:*:lts
              :*:*:*"
        },
        "1":{
          "name":"iptables Firewall",
          "type":"software",
          "vendor":"Linux",
          "version":"1.8.0"
        }
      }
    }
  ]
}
```

2. After completing all required Self-Healing operations, the mitigation rule is remotely applied on the virtual machine. Furthermore, Self-Healing composes a human-readable mitigation which is displayed to the user.
3. The Rule Applicator subcomponent of the Self-Healing enriches the STIX package with a Course of Action SDO which contains information about the mitigation rule that was generated by the Self-Healing component.

```
...
    {
      "type":"course-of-action",
      "id":"course-of-action--8e2e2d2b-17d4
          -4cbf-938f-98ee46b3cd3f",
      "created":"2016-08-31T11:37:49.307Z",
      "modified":"2016-08-31T11:37:49.307Z
          ",
      "name":"mitigation",
      "description":"iptables -A CSAWARE-IN
            -s 123.183.209.131 -j REJECT"
    }
```

4. The CSAWARE-IN chain now contains the mitigation rule that was generated by the Self-Healing component.

6 CONCLUSIONS

The extensive exposure of organisations to existing and emerging cyberthreats has forced them to invest on mechanisms that efficiently consume shared threat intelligence information and reduce response times in adapting their security posture. Self-Healing mechanisms provide the means for administrators to address the complexity of systems management and mitigate potential system faults. In this paper we proposed a Self-Healing solution that has been designed in the context of the CS-AWARE project to address the needs of local public administrations that typically do not have the expertise or the resources to manage security and other appliances. The proposed solution provides a method to appropriately mitigate cyber threats, while still allowing the system administrator to have control over these actions.

ACKNOWLEDGEMENTS

This work has been supported by the EU research program CS-AWARE (A cybersecurity situational awareness and information sharing solution for local public administrations based on advanced big data analysis) project. Call: DS-02-2016, Grand Agreement No.: 740723.

REFERENCES

CS-AWARE (2018). CS-AWARE framework. Deliverable D2.4. Available online: https://cs-aware.eu/2019/03/28/d2-4-cs-aware-framework/.

Dean, D. J., Nguyen, H., and Gu, X. (2012). UBL: unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems. In *Proceedings of the 9th international conference on Autonomic computing - ICAC '12*, page 191, San Jose, California, USA. ACM Press.

Elgenedy, M. A., Massoud, A. M., and Ahmed, S. (2015). Smart grid self-healing: Functions, applications, and developments. In *2015 First Workshop on Smart Grid and Renewable Energy (SGRE)*, pages 1–6, Doha, Qatar. IEEE.

Keromytis, A. D. (2007). Characterizing software self-healing systems. In Gorodetsky, V., Kotenko, I., and Skormin, V. A., editors, *Computer Network Security*, pages 22–33, Berlin, Heidelberg. Springer Berlin Heidelberg.

Psaier, H. and Dustdar, S. (2011). A survey on self-healing systems: approaches and systems. *Computing*, 91(1):43–73.

Rantos, K., Spyros, A., Papanikolaou, A., Kritsas, A., Ilioudis, C., and Katos, V. (2020). Interoperability Challenges in the Cybersecurity Information Sharing Ecosystem. *Computers*, 9(1):18.

Schaberreiter, T., Kupfersberger, V., Rantos, K., Spyros, A., Papanikolaou, A., Ilioudis, C., and Quirchmayr, G. (2019a). A quantitative evaluation of trust in the quality of cyber threat intelligence sources. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, pages 83:1–83:10, New York, NY, USA. ACM.

Schaberreiter, T., Röning, J., Quirchmayr, G., Kupfersberger, V., Wills, C. C., Bregonzio, M., Koumpis, A., Sales, J. E., Vasiliu, L., Gammelgaard, K., Papanikolaou, A., Rantos, K., and Spyros, A. (2019b). A Cybersecurity Situational Awareness and Information-Sharing Solution for Local Public Administrations based on Advanced Big Data Analysis: The CS-AWARE Project. In Bernabe, J. B. and Skarmeta, A., editors, *Challenges in Cybersecurity and Privacy – the European Research Landscape*, RIVER PUBLISHERS SERIES IN SECURITY AND DIGITAL FORENSICS, pages 149–180. River Publishers, Netherlands.

Schneider, C., Barker, A., and Dobson, S. (2015). A survey of self-healing systems frameworks: A SURVEY OF SELF-HEALING SYSTEMS. *Software: Practice and Experience*, 45(10):1375–1398.

STIX (2017). Structured threat information expression (STIX) version 2.0. OASIS standard https://www.oasis-open.org/standards#stix2.0.

Zidan, A. and El-Saadany, E. (2012). A cooperative multi-agent framework for self-healing mechanisms in distribution systems. *Smart Grid, IEEE Transactions on*, 3:1525–1539.