Automated Source Pairing Intelligence System: A system for CTI pairing, enrichment, management, correlation and sharing

Automated Source Pairing Intelligence System: A system for CTI pairing, enrichment, management, correlation and sharing

Thanasis Chantzios

A thesis submitted in partial fulfillment
of the requirements for the MSc in
Computer Science

Committee

University of the Peloponnese
2021

Περίληψη

Ως Πληροφορίες ΚυβερνοΑπειλών (Cyber-Threat Intelligence – CTI), χαρακτηρίζουμε οποιαδήποτε πληροφορία μπορεί να χρησιμοποιηθεί για να βοηθήσει έναν οργανισμό να προσδιορίζει, αξιολογεί, παρακολουθεί και να ανταποκρίνεται σε απειλές στον κυβερνοχώρο. Σχετίζεται με οποιοδήποτε μέρος ενός οργανισμού, το οποίο είναι εκτεθειμένο στον κυβερνοχώρο (για παράδειγμα το δίκτυό του, υπολογιστικά μηχανήματα και άλλα τεχνολογικά είδη). Τα τελευταία χρόνια, λόγω της ραγδαίας αύξησης των απειλών στον κυβερνοχώρο, η διαδικασία της συλλογής και του διαμοιρασμού τέτοιας πληροφορίας, δημιουργεί ολοένα και μεγαλύτερο ενδιαφέρον τόσο ερευνητικά, όσο και ως προς την κατεύθυνση παροχής επιπρόσθετης κυβερνοασφάλειας σε οργανισμούς. Επιπλέον, αυξάνεται η ανάγκη δημιουργίας ενός συστήματος, το οποίο θα βοηθάει τους ειδικούς του κλάδου της ασφάλειας συστημάτων, να εκτιμούν το επίπεδο του κινδύνου ενός οργανισμού, μέσω τέτοιων διαθέσιμων πληροφοριών. Ωστόσο, αυτές οι πληροφορίες βρίσκονται διάχυτες στο διαδίκτυο, σε ευρεία ποικιλία πηγών, και γι΄ αυτόν το λόγο, ο σχεδιασμός ενός τέτοιου συστήματος θα πρέπει να εξασφαλίζει τη συγκεντρωτική αποθήκευση της πληροφορίας από αυτές τις πηγές, και επίσης να διευκολύνει τη διαδικασία διαμοιρασμού της. Βέβαια, η επιλογή των απαραίτητων εργαλείων για τη συλλογή και τον διαμοιρασμό της πληροφορίας αυτής, είναι μία δύσκολη διαδικασία, καθώς πρέπει να ληφθούν υπ΄ όψιν τόσο η χρηστικότητα των εργαλείων, όσο και η πληρότητα της κάλυψης των απαιτούμενων διεργασιών. Σε αυτήν τη διπλωματική εργασία, παρουσιάζεται το ASPIS, ένα σύστημα που παρακολουθεί και συλλέγει συνεχόμενα, από διαφορετικές πηγές, τις πληροφορίες αυτές που σχετίζονται με θέματα απειλών κυβερνοασφάλειας. Τέλος, τις αποθηκεύει συγκεντρωτικά και δομημένα μέσα σε μία βάση δεδομένων, εμπλουτίζοντας καθ΄ αυτόν τον τρόπο το περιεχόμενό της, παρέχοντας ταυτόχρονα τα κατάλληλα εργαλεία για τη συσχέτιση και το διαμοιρασμό των Πληροφοριών ΚυβερνοΑπειλών.

Abstract

Cyber-threat intelligence (CTI) is any information that can help an organization identify, assess, monitor, and respond to cyber- threats. It relates to all cyber components of an organization such as networks, computers, and other types of information technology. In the recent years, due to the major increase of cyber-threats, CTI gathering and CTI sharing are becoming increasingly important both as a subject of research and as a concept of providing additional security to organizations. Additionally, there is an increased need for creating a system that aids security experts to assess the risks of an organization, with regard to the CTI at hand. However, CTI is available throughout the web, by a wide variety of sources and thus, the system design should promote the collective storing of CTI and facilitate the sharing process. Furthermore, selecting the proper tools and platforms for CTI gathering and sharing, is a challenging task, that pertains to a variety of aspects, that regard both the usability of the tools and the coverability of the required tasks. This thesis presents **ASPIS**; a system that continuously monitors and gathers CTI from different sources, storing it in a structured and clustered manner, provides the tools for correlating the CTI at hand, and also facilitates the sharing process.

Acknowledgements

I would like to thank Professors Spiros Skiadopoulos, Christos Tryfonopoulos, Nicholas Kolokotronis and Costas Vasilakis for the valuable scientific guidance they have provided me throughout the course of this thesis, as well as for the immediate help they offered me whenever it was needed. In addition, I would like to thank the professors of the Department of Informatics and Telecommunications, who have given us the knowledge we need in order to become ourselves the ones who eventually will try to evolve this scientific field.

Afterwards, I would like to thank the University of Peloponnese, the Department of Informatics and Telecommunications and more specifically the Software and Database Systems Laboratory for providing me the necessary equipment to accomplish this thesis.

Finally, I would like to thank my family and my friends for their support and encouragement throughout all these years.

Contents

List of Tables vii

List of Figures x

List of Symbols xi

List of Abbreviations xv

1 Introduction 1

1.1 Problem Statement 1

1.1.1 Cyber-Threat Intelligence 1

1.1.1.1 Strategic CTI 1

1.1.1.2 Tactical CTI 2

1.1.2 Threat Information Types 2

1.1.3 CTI Sources 3

1.1.4 CTI Sharing Importance 4

1.1.5 The CTI life-cycle 5

1.2 Our Contribution 6

1.3 Thesis Organization 7

2 Related Work 9

2.1 Evaluation of CTI Sharing Standards and Platforms 10

2.2 Current State-of-the-Art CTI Sharing Solutions 12

2.2.1 CTI Sharing Tools and Platforms 12

2.2.1.1 MISP - Open Source Threat Intelligence & Open Standards for CTI Sharing 12

2.2.1.2 OpenCTI - Open Cyber-Threat Intelligence Platform 14

2.2.1.3 GOSINT - The Open-Source Threat Intelligence Gathering and Processing Framework 15

2.2.1.4 YETI - Your Everyday Threat Intelligence. 15
2.2.1.5 OpenTAXII - a Python Implementation of TAXII Services. 16
2.2.1.6 CIF - Collective Intelligence Framework. 16
2.2.2 CTI Services 17
2.2.3 CTI Platforms 17
2.3 Implementations of Frameworks, Tools and Platforms for Targeted CTI Gathering and Sharing 19

3 MISP 23
3.1 The Data Model 23
3.1.1 MISP Database Overview for Storing CTI 24
3.1.2 Events Table 25
3.1.3 Objects Table 25
3.1.4 Attributes Table 25
3.1.5 Correlations Table 25
3.2 CTI Sharing Properties and Features 30
3.3 Additional Features 31
3.4 Current State of MISP 33
3.5 General MISP Layout 33
3.5.1 Simple User 33
3.5.2 Administrator 34
3.5.3 Events 35
3.5.3.1 Creating an Event 36
3.5.3.2 List of Events 37
3.5.3.3 Events View 37
3.5.3.4 Correlation Engine 40

4 The ASPIS System 41
4.1 System Overview 41
4.2 Technology Stack and Applied Tools 44
4.3 MISP in ASPIS 44
4.3.1 MISP Objects Employed 45

4.3.1.1 The Vulnerability Object 45
4.3.1.2 The Weakness Object 46
4.3.1.3 The VulDB-Vulnerability Object 46
4.3.1.4 The ExpDB-PoC Object 46
4.3.2 MISP Correlations in ASPIS 48
4.3.2.1 MISP Correlation Engine Functionality 49
4.3.3 MISP Correlation Engine Alteration 52
4.3.3.1 Proposal for an Alternative MISP Correlation Engine 53
4.3.3.2 MISP Correlation Engine Table BCNF Decomposition 54
4.4 Monitored Sources 57
4.5 The Source Crawling and Parsing Phase 59
4.5.1 NVD Parsing 59
4.5.2 JVN Parsing 60
4.5.3 KB-Cert Crawling and Parsing 60
4.5.4 VulDB Crawling and Parsing 60
4.5.5 Exploit-DB Crawling and Parsing 61
4.6 The Object Structuring Phase 62
4.7 The Event Management Phase 62
4.7.1 Event Lookup 62
4.7.2 Event Creation 63
4.7.3 Event Modification 63
4.7.3.1 Event Enrichment with Complementary CTI from Another Source 63
4.7.3.2 Event Update due to Updated CTI 64
4.7.3.3 Exploit-DB Entry Added a Reference to a CVE ID 64
4.7.4 Events' Correlations 64
4.8 The CTI Sharing and Reviewing Phase 65
4.8.1 MISP REST API: RESTful Searches 65
4.8.1.1 List of RESTful Endpoints 65
4.8.1.2 Authorization 65
4.8.1.3 Headers 66
4.8.1.4 Search Constraints (Payload/Body) 66

4.8.1.5 cURL Example 67
4.8.2 CTI Reviewing through MISP Sightings 67
4.9 ASPIS Usage 67
4.9.1 Login 67
4.9.2 Events Browsing and Filtering 68
4.9.3 Events Inspection 69
4.9.4 CTI Sharing: MISP REST API 71
4.10 System Installation 73

5 Experimental Evaluation 75
5.1 Experimental Setup 75
5.1.1 Data Set 75
5.1.2 Query Sets 76
5.1.3 Configuration Parameters and Metrics Employed 78
5.1.3.1 CTI Storing Space Requirements 78
5.1.3.2 CTI Storing and Processing Time 78
5.1.3.3 CTI Querying Response Time 78
5.1.4 Technical Configuration 79
5.2 Experiment Results 80
5.2.1 CTI Storing Space Requirements 80
5.2.2 CTI Storing and Processing Time 80
5.2.3 CTI Querying Response Time 83

6 Conclusions and Future Work 85
6.1 Summary and Conclusions 85
6.2 Future Work 86

A Examples of MISP Objects in JSON Format 87
A.1 NVD: Vulnerability Object 87
A.2 NVD: Weakness Object 88
A.3 JVN: Vulnerability Object 88
A.4 KB-Cert: Vulnerability Object 89
A.5 VulDB: Vuldb-Vulnerability Object 91

A.6 Exploit-DB: ExpDB-PoC Object 92

B MISP Versions Changelog 93

List of Tables

2.1 CTI sharing tools and platforms 13
2.2 CTI services comparison 17
2.3 Threat intelligence platforms 18
3.1 MISP *events* table 27
3.2 MISP *objects* table 28
3.3 MISP *attributes* table 29
3.4 MISP *correlations* table 30
4.1 The *vulnerability* MISP Object 45
4.2 The *weakness* MISP Object 46
4.3 The *vuldb-vulnerability* MISP Object 47
4.4 The *expdb-poc* MISP Object 47
4.5 The resulting *correlations* table of Example 1 49
4.6 The resulting *correlations* table of Example 2 49
4.7 The resulting *correlations* table of Example 1, with highlighted redundancy 50
4.8 The resulting *correlations* table of Example 2, with highlighted redundancy 50
5.1 Query sets' average correlations per query 77

List of Figures

1.1 The CTI life-cycle 5

3.1 MISP database schema abstracted overview 24

3.2 MISP Sightings 33

3.3 MISP simple user's top bar 34

3.4 MISP administrator's top bar 34

3.5 Adding process of an Event in MISP 36

3.6 MISP layout in the List of Events 37

3.7 MISP Event View 38

3.8 MISP Event timeline 39

3.9 MISP Event Correlation graph 39

3.10 MISP Correlation Engine 40

4.1 **ASPIS** system architecture 42

4.2 **ASPIS** system dataflow example 43

4.3 The alternate equivalent (right) to the resulting *correlations* table of Example 1 (Table 4.7) (left). 53

4.4 The alternate equivalent (right) to the resulting *correlations* table of Example 2 (Table 4.8) (left). 53

4.5 **ASPIS** Login 68

4.6 **ASPIS** View Events List 68

4.7 **ASPIS** Filter Events 69

4.8 **ASPIS** Event Inspection 70

4.9 **ASPIS** Event Timeline Examination 70

4.10 **ASPIS** Event Correlations Graph Inspection 70

4.11 MISP Query Builder in Action 71

4.12 MISP REST API: cURL Querying & PyMISP Querying 72

4.13 **ASPIS** CTI for Wordpress Plugins Vulnerabilities of December 2020 72
5.1 Query sets' average correlations per query proportionally to the stored events 77
5.2 **ASPIS**$_{\mathbf{A}}$ and **ASPIS**$_{\mathbf{D}}$ CTI storing space requirements 81
5.3 Daily time elapsed for the CTI gathering and storing 82
5.4 Monthly average time elapsed for the CTI gathering and storing 82
5.5 **ASPIS**$_{\mathbf{D}}$ and **ASPIS**$_{\mathbf{A}}$ CTI querying average response times 83

List of Symbols

v	value
$\mathcal{V}$	set of common values
$AwV(v)$	number of attributes that contain the value v
ϵ	MISP Event
$\epsilon_{\mathrm{n}}AwV(v)$	number of attributes coexisting in the event ϵ_{n}, that contain the value v
$CTI_Q(\mathcal{D})$	CTI querying time for dataset $\mathcal{D}$
CTI_S	CTI storing task

List of Abbreviations

3NF	3rd Normal Form
5W3H	What, Who, Why, When, Where, How, How much, and How long
API	Application Programming Interface
ASPIS	Automated Source-Pairing Intelligence System
BCNF	Boyce-Codd Normal Form
CCE	Common Configuration Enumeration
CIDR	Classless Inter-Domain Routing
CPE	Common Platform Enumeration
CPS	Cyber-Physical Systems
CRITs	Collaborative Research Into Threats
CSIRT	Computer Security Incident Response Team
CSV	Comma-Separated Values
CTI	Cyber-Threat Intelligence
CVE	Common Vulnerability Enumeration
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DB	DataBase
DDoS	Distributed Denial-of-Service
DNS	Domain Name Server

DoS	Denial-of-Service
FQDN	Fully Qualified Domain Name
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
ID	Identifier
IDMEF	Intrusion Detection Message Exchange Format
IDS	Intrusion Detection System
IoC	Indicator of Compromise
IODEF	Incident Object Description Exchange Format
IoT	Internet of Things
IP	Internet Protocol
IPS	Intrusion Prevention Systems
ISACs	Information Sharing and Analysis Centers
JSON	JavaScript Object Notation
MDSEA	Model Driven Service Engineering Architecture
NF	Normal Form
NLP	Natural Language Processing
PDF	Portable Document Format
PoC	Proof-of-Concept
RDF	Resource Description Framework
REST	REpresentational State Transfer
RSS	Rich Site Summary
SaaS	Software as a Service
SIEM	Security Information and Event Management System
STIX	Structured Threat Information eXpression
TAXII	Trusted Automated eXchange of Indicator Information

TTPs	Tactics, Techniques and Procedures
UI	User Interface
URL	Uniform Resource Locator
UUID	Universally Unique IDentifier
VERIS	Vocabulary for Event Recording and Incident Sharing
XML	eXtensible Markup Language
YAML	YAML Ain't Markup Language

Chapter 1

Introduction

This thesis presents **ASPIS**, a system for CTI gathering and sharing. **ASPIS** is a multi-layered system aimed at collectively gathering CTI from various sources, storing it in a structured manner. Through a variety of tools and a Graphical User Interface (GUI), it provides the information in human and machine-readable formats, to both aid security experts view, process and analyze it, and facilitate the CTI sharing process. In this chapter we define the problem, discuss the current state-of-the-art approaches, outline our solution and highlight our contributions.

1.1 Problem Statement

In the recent years, due to the increasing number of cyber-threats, CTI gathering and CTI sharing is becoming increasingly important both as a subject of research and as a concept of providing additional security to organizations.

1.1.1 Cyber-Threat Intelligence

CTI is information about threats relating to networks, computers, and any other type of information technology. However, such intelligence is not just data. It is information that has been observed, analyzed, and can be actionable. CTI is divided in two main categories; strategic and tactical.

1.1.1.1 Strategic CTI

Strategic CTI is information that helps gaining an overview of threats that may affect an organization. Specifically, it is information that can provide a high-level understanding of the organization's decisions' impact, across imminent threats. The majority of strategic CTI sources are open source, and they may include:

- Policy documents from nation-states or other groups of interest.
- Local and national media.
- Industry and subject specific publications.
- Comments, online activity and articles from individuals of interest.
- Free content produced by security organizations (e.g., white papers, research reports)

1.1.1.2 Tactical CTI

Tactical CTI provides necessary information about the tactics, techniques, and procedures (TTPs) followed by threat actors so that they can achieve their goals (e.g., to compromise a network, exfiltrate data). It helps defenders gain a point of view and understand how an organization in threat, is likely to be attacked, in order to determine if appropriate detection and mitigation mechanisms exist or whether they should be implemented. Tactical CTI sources are mainly based on reports produced by security vendors. These reports focus on specific threat groups or attack campaigns and provide key tactical information such as:

- Locations and industries targeted.
- Attack vectors employed (e.g., SQL injection, DDoS)
- Tools and technical infrastructure used (e.g., botnet)

Although these reports can be valuable, they are produced for a big audience, making the majority of the CTI less relevant to any specific orgranization. A more reliable stream of tactical CTI requires active gathering processes, which monitor: open/deep/dark web, forums, marketplaces, malware analysis, social media, etc.

1.1.2 Threat Information Types

The types that are used in order to describe threat information, are distinguished in:

Indicators. Observables or technical artifacts which suggest that an attack is going to happen or that a system has already been compromised. *Indicators* can be utilized in a system to build a defense against any potential threat. Several examples of indicators are listed below:

- An IP address accompanying a suspicious command.

- A distrustful DNS domain name.
- A URL which redirects to suspicious content.
- A file hash using a malicious executable.
- A malicious email message.

Tactics, Techniques and Procedures (TTPs). TTPs are used to describe the behavior of an actor. *Tactics* describe the actor's behavior, *techniques* describe the tactics which might be followed by the actor, and finally *procedures* are detailed descriptions in the context of a technique. To sum up, TTPs describe the possibility of an actor to follow a specific attack pattern, using an attacking tool, a malware variant, exploiting a vulnerability, etc.

Security Alerts. These are also known as advisories, and they are basically a brief overview of security alerts, including human-readable notifications regarding vulnerabilities, exploits or other security issues.

Threat Intelligence Reports. These include documents which describe TTPs, system types, threat actors, target information, and any other information which best describes an incident related to cyber threats, in order to provide greater awareness to organizations.

Tool Configurations. These include recommendations for the installation and the utilization of mechanisms and methodologies followed, in order to collect, process, analyze and share CTI. Tool configuration information might include instructions on how to customize a tool to build web filter configuration files, firewall rules, etc.

1.1.3 CTI Sources

CTI is accessible through various sources, which can be categorized into three main groups:

Internal Sources. In this group of sources, CTI is collected from an intra-organization level. For example, such types of sources might be: Intrusion Prevention Systems (IPS), firewalls, anti-virus. In addition, a significant internal source of CTI derives from computer forensic analysis, providing information about application settings, running processes, services being used, system events, etc., and it could indicate adversarial behavior as well.

Community Sources. These include CTI shared within a trusted relationship circle of multiple members having common interests. This might be for example an informal group, with the community members being organizations which are in the same industrial sector. The Information Sharing and Analysis Centers (ISACs) are such an example. They are non-profit organizations, which provide a central hub for gathering CTI and allow two-way sharing between the private and the public sector.

External Sources. This group contains CTI gathered from outside an organization. There are three types of external sources:

Public. External sources, which are free of charges and publicly available. Although public feeds are freely available, they are not always credible since they might as well be based on volunteered data.

Private. These sources require payment. In order to gain access to them, an organization can subscribe to a threat feed provided by the vendor, having guaranteed data quality and credibility, based on a service level agreement. These security services include some type of CTI update mechanism that keeps the feed up-to-date.

Unindexed. These sources are sites and forums accessible only from the deep or the dark web. In most cases, CTI is gathered from unindexed chatrooms, forums, marketplaces, and so on. These sources might as well have confined access, which makes them a challenging source of CTI. In these sites, individuals exchange information which provides great insights about security issues after being analyzed.

1.1.4 CTI Sharing Importance

As stated previously, the ever increasing amount of software vulnerabilities, in addition to innovating attack techniques, increases the percentage of cyber-threat victims. Thus, the gathering and sharing of CTI amongst communities, brings great benefits to each individual member, such as:

Increased Awareness. By using shared resources, in addition to leveraging capabilities of partners, such as knowledge and experience, the security level is enhanced in a proactive way.

Improved Security Posture. CTI sharing makes it easier for organizations to identify an affected system, implement security measures for additional protection, enhance detection methods in case incidents re-occur, and recover from incidents.

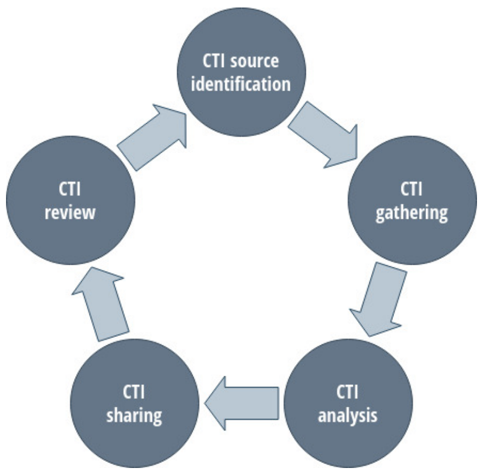


Figure 1.1: The CTI life-cycle

Knowledge Maturing. By sharing CTI, the knowledge base of security-related information is enriched with enhanced and updated information. This is achieved by enriching existing indicators, and by developing knowledge of actors' TTPs, which are associated with specific incidents, threats or threat campaigns.

Increased Defensive Agility. In order to evade detection, security controls, and to exploit new vulnerabilities, threat actors constantly adapt their TTPs. In order to reduce the probabililty of successful exploitations, via CTI sharing organizations are periodically informed about changing TTPs and can detect and respond to imminent threats rapidly.

Summing up, in order to provide additional security to organizations, there is a need for security experts to be provided with tools and systems, which aid them with the procedures of CTI monitoring, analyzing and processing, developed to support the context of the aforementioned.

1.1.5 The CTI life-cycle

The CTI cycle, as illustrated in Figure 1.1, is the process of generating and evaluating CTI. The first step of this process is CTI *source identification* (or *direction*). It pertains to the identification of threat information that needs to be collected from monitoring devices, feeds, and security repositories to support decision-making and raise cyber-security awareness. The next step, namely CTI *gathering*, is the collection of the necessary data from the identified sources, along with the tools for extracting a wide variety of information, like tactical information (infrastructure, malware, and exploits) and strategic information (revealing attackers' goals). This process requires a series of steps starting from the collection of relevant IP addresses. Moreover, it is not a one-time action, but it should be performed in a continuous manner. The main goal at this stage is to collect as much information as possible and allow correlations and further analysis. The third step is CTI *analysis* and is built upon the information that has been collected; it includes both automated and human-driven analysis. The fourth step is CTI *sharing* to the relevant stakeholders, i.e., the entities that can utilize the generated intelligence, in a form that they find to be appropriate, useful, and in many cases actionable. This makes sharing highly-dependent on the audience (e.g., tactical, operational, and strategic level). CTI *review* (also referred to as CTI *feedback*), which is the last step in the above process, constitutes the key to the continuous improvement of the generated intelligence.

1.2 Our Contribution

There is a plethora of methods, tools and platforms that facilitate the flow of the CTI life-cycle. In this thesis we present the **ASPIS** system, a combination of implemented methods, and open-source tools and platforms, which is aimed at providing a complete solution, with regard to the CTI life-cycle. **ASPIS** periodically gathers CTI from different sources (i.e. NVD [33], JVN [24], KB-Cert [26], VulDB [61] and Exploit-DB [12]). The CTI deriving from these sources is analyzed and provides information about vulnerabilities and exploits, along with a variety of other metrics, such as *exploitability*, *0day exploit price*, and so on. After gathering the CTI from the aforementioned sources, **ASPIS** system then proceeds to store it in a structured manner, using the MISP platform as the data storage, which simplifies the procedures of CTI sharing and reviewing, as it provides all information in both human and machine-readable format. Furthermore, it provides tools that support the enrichment of CTI gathered through correlating processes, as well as it enables security experts to review CTI through a user-friendly GUI.

The core functionality of **ASPIS** is implemented by utilizing MISP [28] - a free and community-driven, open source platform, helping information sharing of threat intelligence including cyber security indicators. **ASPIS** retrieves all CTI from the monitored sources, via data feeds (XML/JSON) or web scrapping techniques, and then stores it in MISP, in a nested manner, so that all CTI regarding a specific issue will be enclosed in one cluster, eliminating duplicates. That functionality is implemented through numerous scripts, written in Python, which drive the procedures of retrieving the CTI, storing it in MISP in a nested manner, and applying possible modifications (updates) in any previously stored CTI. The monitored sources can be categorized in two main categories; sources that provide datafeeds and sources that expose the CTI on web-based interfaces, in a structured manner. The sources that provide datafeeds, are parsed through standard XML/JSON parsing techniques, to extract the desired information. For the rest, the parsing methodology is augmented with crawling methods, to access all pages that may contain CTI exploit/vulnerability notes. Finally, the scripts make use of the MISP functionalities and database capabilities, to store or update the retrieved information in a nested manner. These processes are being executed periodically, every 24 hours, in order to keep **ASPIS** synchronized with all sources at hand.

1.3 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2 we present the related work of proposed CTI-related models, frameworks and systems, and discuss their architecture and implementations. In Chapter 3 we provide a detailed overview of MISP, regarding its data model, features/functionalities and UI. In Chapter 4 we present in detail the **ASPIS** system architecture, describe its functionality, and finally present the technologies and principles upon which the system was designed and built. Furthermore, in Chapter 4 we detailedfully describe an alteration of the MISP implementation in our system. Moreover, we present the necessary steps required in order to set-up the **ASPIS** system and provide a user guide. In Chapter 5 we present the experimental evaluation of our system, in comparison with the default MISP implementation. Finally, in Chapter 6 we conclude this work, reviewing its sustainability and proposing possible extensions of the developed system.

Chapter 2

Related Work

The need of assessment, detection and gathering of cyber-threat information escalated over the years; this is also demonstrated by the ENISA threat landscape reports of the previous years. Specifically, the surveys of ENISA [72–74, 76] indicate that there's a persistent increase in various cyber-threat types, such as: Malwares, Web-based Attacks, Web-application Attacks, DoS, Botnets, Phishing, Data Breaches, Information Leakage. Thus, it is prominent that there's an increased need for deploying efficient and automated proactive technologies, that are able to analyze and share heterogenous CTI related to the present systems' configurations, attacker's threats and tactics, indicators of ongoing incidents, and so on. Proactive detection of incidents is defined as the process of discovery of malicious activity in a team's constituency through internal monitoring tools or external services that publish information about detected incidents, before the affected constituents become aware of the problem. The type of tools most often evaluated as excellent, belong to systems for aggregation, correlation and visualization of logs and other event data [77]. However, the efficiency and the automation of processes that facilitate the flow of the CTI life-cycle (as shown in Figure 1.1) is a challenging task and requires the standardization of CTI formatting, the proper selection of tools, platforms and frameworks that are able to support CTI sharing, and the provision of appropriate methodologies that enable CTI evaluation.

To counter these challenges, a lot of research has been conducted in an effort to determine the most suitable CTI sharing standards, and to implement and deploy convenient tools that enable the automation of CTI gathering and sharing. In the following sections we will present related research, which showcases:

1. the evaluation of standards and platforms,
2. the current state-of-the-art CTI sharing solutions,

3. the implementation of frameworks, tools and platforms, which facilitate, automate and enhance the process of targeted CTI gathering and sharing.

2.1 Evaluation of CTI Sharing Standards and Platforms

To be able to automate the processes that enable CTI gathering and sharing, there's a wide variety of defined standards and formats, along with platforms that utilize them.

In [70], the authors provide a methodology for the evaluation of the standards and platforms of CTI. First, the research provides a review of the state-of-the-art CTI ecosystem's standards and platforms, by showcasing the main directions that the theme has been following in the recent years. In reference, [90] provides a broad overview of the CTI sharing dimensions, briefly mentioning some frameworks and standards that support the CTI sharing task. Other works, like [75], [89], [93], focus on both open-source and commercial CTI platforms and on the most common CTI sharing standards, to provide an overview of the current CTI sharing landscape. An extensive analysis on open-source and commercial CTI sharing tools, platforms and services, is conducted in [83], where the authors present an overview of their provided functionalities. Additionally, [70] presents the CTI practices that have been deployed or have great potential of being deployed in the area. To do so, the authors gain a complete overview of the CTI panorama, by searching in web research engines, such as Google Scholar [16], Springer [48], IEEE Digital Library [21], and so on. To find suitable results, the authors posed the following query: *(Threat Intelligence OR Cyber Threat Intelligence) AND (Platform OR Tools OR Standards).* Then, a selection strategy is developed, in order to limit and define which standards and platforms they will analyze, due to the extensiveness of the threat intelligence scenario. Through the selection strategy, the authors concluded that in order to evaluate the most relevant standards and platforms in the CTI field, the results, found through the searching process, were described in terms of popularity and license model. As popularity the authors define the number of times the standard or platform was mentioned in reliable works and sources, combined with collected statistics about the percentage of utilization among organizations. As license model, they include only free or open source solutions and initiatives. Finally, after eliminating the CTI standards and platforms results, the authors propose several evaluation criteria, infered through the 5W3H method, as well as the intelligence process flow (Figure 1.1). Specifically, regarding the CTI standards, the authors inspect the data model architecture in terms of whether they adopt a holistic architecture that includes parameters like threat, incident, threat actor, and defense. Additionally, they examine the intelligence process in terms of collection (common formatting), processing (structured format, low overhead, machine readability), analysis (unambiguous data model, relationship mechanisms), deployment (interoperability), and dissemination (transport mechanism, practical application). With regard to the CTI platforms, the authors inspect the data model architecture in the context of whether they have a holistic architecture, to review the use case applicability, and if they follow the 5W3H answering method to answer the platforms' capability. Furthermore, they examine the intelligence process in terms of collection (import formats, automatic gathering), processing (export format, graphic visualization), analysis (correlation, classification), deployment (integration with security systems), and dissemination (sharing method). Finally, for the CTI platforms, the authors also inspect the usability in respect of their documentation and license model. The CTI standards' evaluation concluded that STIX v2 [87, 88] & TAXII [68] is the most consolidated standard in the threat intelligence context, mainly due to its holistic approach. Concerning the CTI platforms' evaluation, the authors concluded that MISP [28] and OpenCTI [36] were considered as the most complete and flexible platforms.

Similarly, in [66], the main goal of the work carried out is to survey related tools and platforms, evaluate them and identify the most appropriate for the purposes of CTI sharing, in terms of expressiveness, flexibility, automation, and structuring. To do so, first, it provides a brief overview of a variety of CTI sharing aspects, such as the types of threat information (indicators, TTPs, security alerts, threat intelligence reports, and tool configurations), the main categories of CTI sources (internal sources, community sources, and external sources), the CTI life-cycle (Figure 1.1), and also the main CTI sharing challenges (trust establishment, achieving interoperability and automation, securing sensitive information, and enabling information sharing). Then, it comes up with a set of high-level CTI sharing mechanisms' requirements, in order to construct the evaluation criteria. Particularly, the sharing mechanism must allow CTI sharing between the platform and different stakeholders (like service providers and certified authorities), as well as between the platform and the end-users' devices. Then, the sharing mechanism and platform should be expressible, flexible, and scalable. Furthermore, the sharing mechanism and platform should allow information to be both human and machine readable and facilitate automation. Next, the sharing platform should allow storing information about the source of CTI, and also support information filtering and alerting. Finally, the sharing platform should be open source. Based on the features of the surveyed platforms (MISP [28], GOSINT [17], OpenTPX [38], YETI [62], OpenTAXII [37], and CIF [5]). Then, the work proceeds to compare them in terms of organized repository, organized community, helpful documentation, creativity, additional tools, additional benefits, and drawbacks. Finally, after comparing the surveyed platforms, it evaluates them in respect of interoperability, expressiveness, flexibility, extensibility, automation, and whether they provide intelligence in both human and machine-readable formats. The evaluation is achieved through a scoring system, where '-' means that the specified evaluation requirement is not supported, '1' means that the requirement is supported to a satisfying level, and '2' means that the requirement is supported to a high level. The evaluation concludes with MISP and GOSINT taking the lead in the platforms' race, when compared to the rest of the surveyed platforms.

2.2 Current State-of-the-Art CTI Sharing Solutions

This section reviews several solutions related to the discovery and management of CTI, and presents their main features and characteristics, with respect to a number of aspects, such as their architecture, offered services, standards' adoption, and mode of operation. Specifically, the following sections will review *CTI sharing tools and platforms*, *threat intelligence services* and *threat intelligence platforms*.

2.2.1 CTI Sharing Tools and Platforms

This section compares and evaluates six open-source platforms and tools, that implement common sharing standards, to facilitate the CTI analysis, sharing and reviewing. Specifically, it considers Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing (MISP [28]), Open Cyber-Threat Intelligence Platform (OpenCTI [36]), Open-Source Threat Intelligence Gathering and Processing Framework (GOSINT [17]), Your Everyday Threat Intelligence (YETI [62]), a Python implementation of TAXII Services (OpenTAXII [37]) and Collective Intelligence Framework (CIF [5]). Finally, the comparison of the aforementioned platforms is briefly presented in Table 2.1.

2.2.1.1 MISP - Open Source Threat Intelligence & Open Standards for CTI Sharing

MISP [92] is one of the most widespread CTI sharing platforms. Following the example of most CTI sharing platforms, MISP detects, stores and shares technical

	MISP	**OpenCTI**	**GOSINT**	**YETI**	**CIF**	**OpenTAXII**
Human & machine readable	✓	✓	✓	✓	✓	✓
API	✓	✓	✓	✓	✓	✓
Tagging mechanism	✓	✓	✓	✓	✓	✗
Correlation mechanism	automatically	manually	✗	manually	✗	✗
Data formats	CSV, XML, JSON, OpenIOC, STIX/TAXII, and more	CSV, XML, JSON, OpenIOC, STIX/TAXII, and more	STIX/TAXII, VERIS, IODEF, IDMEF	JSON	JSON, CSV	TAXII

Table 2.1: CTI sharing tools and platforms

and non-technical information about malware samples, incidents, attackers and intelligence. Moreover, MISP supports data export in STIX and OpenIoC for use in Intrusion Detection Systems (IDSs) or Security Information and Event Management Systems (SIEMSs). Additionally, MISP has an automatic correlation mechanism that is: (a) able to identify relationships between attributes/objects and indicators from malware correlation engines and (b) capable of performing advanced correlations such as fuzzy hashing (e.g., ssdeep) or CIDR block matching. Another interesting characteristic of the MISP platform is that most of the supported data models are created by MISP community. MISP stores data in a structured format (to allow for the automated use of its database for various purposes), provides extensive support of cyber-security (including fraud) indicators for different vertical sectors (e.g., financial sectors), and supports CTI sharing for both human and machine applications. Furthermore, it provides STIX support, allowing data export in STIX 1.0 and 2.0 (XML and JSON) format. More details about MISP functionalities are described in [28]. Intelligence vocabularies (MISP galaxy) can be bundled with existing threat actors, malware and ransomware or linked to events from MITRE ATT&CK [30] knowledge base. MISP objects are used since the MISP version (2.4.80) and can be also utilized by other information sharing tools. The creation of these objects and their associated attributes is based on real cyber-security use-cases and existing practices in information sharing, while object sharing is transparently supported even for MISP instances that don't have the object template. The MISP objects derived from many categories depend on the threat type; the supported attacks include: ail-leak (analysis information leak framework), ais-info (automated indicator sharing), android permission, av-signature (antivirus detection signature), bank account, cap-alert (common alerting protocol alert object), and others. Finally, MISP provides a flexible free text import tool to facilitate the integration of unstructured reports into MISP and an adjustable taxonomy to classify and tag events according to the users' own classification schemes and taxonomies. The taxonomy can be either local or shareable among different MISP instances, while MISP comes with a default set of well-known taxonomies and standard classification schemes adopted by many organizations.

2.2.1.2 OpenCTI - Open Cyber-Threat Intelligence Platform

The OpenCTI platform is an open-source solution, which allows organizations to manage CTI knowledge and observables. Its main task is to structure, store, organize and visualize both technical and non-technical artifacts about cyber-threats. OpenCTI's data structure follows a knowledge schema approach, which is based on the STIX2 standards. To gather CTI, the platform can be integrated with other tools and platforms like MISP, TheHive [52], MITRE ATT&CK and more. For the CTI storage, OpenCTI integrates ElasticSearch [11]. Finally, the stored CTI is centrally available to OpenCTI's users and submodules through the integrated GraphQL API [19]. Specifically, OpenCTI aims to be a tool, that allows users to integrate technical information (e.g. TTPs, observables) and non-technical (i.e. victimology, affected verical sectors, localization), while also linking each piece of information to its primary source (like a CTI report, a MISP event). OpenCTI makes use of connectors, which are standalone processes, executed independently of the rest of the platform. The connectors use the RabbitMQ [41] message broker to consume or push data to OpenCTI, through a dedicated queue for each connector instance, that contacts the OpenCTI API. Basically, connectors define routes to the platform's API, that enable users to customize the importing of knowledge (like enriching/updating from external or internal sources) and to export it in formats like STIX2, PDF, CSV. To achieve that, it requires the use of workers that listen to RabbitMQ, in order to call the API for the insertion or export of data. Moreover, OpenCTI allows users to correlate knowledge (entities, TTPs, threat groups, etc.) and match it with existing CTI reports. This is a task that can be achieved either programmatically through an implemented Python client, or manually through the provided UI. Specifically, during this procedure, users define relations over knowledge artifacts and OpenCTI can infer additional relations, by utilizing a predefined rule-based model, that is built upon the Grakn knowledge graph [18]. Thus, it is able to enrich CTI reports with knowledge graphs. Finally, it provides customizable dashboards that enable users to visualize entities, reports and knowledge relations to entities, indicators and observables. To sum up, OpenCTI is able to act as a unified CTI repository, for numerous tools and platforms (like MITRE CVE [6], MITRE ATT&CK, MISP, TAXII2, TheHive, VirusTotal [57], and more), while allowing users to further analyze the collected CTI.

2.2.1.3 GOSINT - The Open-Source Threat Intelligence Gathering and Processing Framework

GOSINT is another popular open-source platform that focuses on intelligence gathering and processing. It collects, processes and exports IoCs; in this way it controls the data inclusion process in the platform and enriches it with high-quality metadata. GOSINT aggregates, validates, and sanitizes indicators for consumption by other tools including CRITs and MISP, or directly into log management systems and SIEMs, while also supporting STIX 2.0/1.x, TAXII and VERIS [67]. GOSINT allows forensic experts to gather structured and unstructured data from incidents occurring at third parties1. It is developed by Cisco CSIRT and can act as a powerful aggregator of IoCs before they are passed to another analysis platform or a SIEM. GOSINT additionally supports IODEF [69] and IDMEF [78] alongside STIX/ TAXII and VERIS. GOSINT can support several actions to provide additional context to indicators in the pre-processing phase; such actions may include the identification of IoCs with systems like Cisco Umbrella, ThreatCrowd, and VirusTotal. The information returned from these services can help analysts reach a verdict on the value of the indicator, as well as tag the indicator with additional context that might be used later in the analysis pipeline. The GOSINT functionalities are described in [17]; the framework is written in Go with a JavaScript frontend. Drawbacks of the GOSINT platform are mainly related to package management and include: package managers that (a) provide out-of-date versions of the software and should be tested to ensure compatibility and (b) name packages differently depending on the package managers or OS release repository at hand.

2.2.1.4 YETI - Your Everyday Threat Intelligence.

Another open-source platform is Yeti; an open, distributed, machine- and analyst-friendly threat intelligence repository. Yeti is a platform meant to organize observables, IoCs, TTPs, and threat intelligence in a single, unified repository. Moreover, Yeti automatically enriches observables (e.g., by resolving domains and geolocating IPs) on behalf of the user and provides a (Bootstrap-based) user interface for humans and an API-based for machines so that to facilitate communication and interoperability with other CTI tools. Finally, Yeti enables users to enrich the investigations of the stored observables, by providing a user-friendly GUI for the creation of relationships between them, presenting them with relationship graphs. The Yeti functionalities are described in [62].

2.2.1.5 OpenTAXII - a Python Implementation of TAXII Services.

It is an upgraded form of TAXII Services; its architecture follows the TAXII specifications with functional units for the TAXII Transfer Unit, the TAXII Message Handler, and other back-end services. OpenTAXII is a robust Python implementation of TAXII Services that delivers a rich feature set. It provides extendable persistence and authentication layers (both via a dedicated API) and provides a collection of threat specifications. Furthermore, it provides an appropriate set of services and message exchange functionality to facilitate CTI sharing between parties. Some other characteristics of OpenTAXII include: customizable APIs, authentication, flexible logging. Furthermore, it automatically handles the data of the frameworks relied on, provides machine-readable threat intelligence, and combines network security operations data with threat intelligence, analysis and scoring data in an optimized manner. it is a large repository that consists of (meta)data of intrusions; database handling typically occurs in the same query context.

2.2.1.6 CIF - Collective Intelligence Framework.

It is a CTI management system and one of the platforms of choice of ENISA for CTI sharing. CIF helps users to parse, normalize, store, post-process, query, share and produce CTI data, while allowing them to combine known malicious threat information from many sources and utilize that information for identification (incident response), detection (IDS) and mitigation (null route). It also supports an automated form of the most common types of threat intelligence warehoused in CIF which are IP addresses and URLs that are observed to be related to malicious activity. The CIF framework aggregates various data-observations from different sources. When a user query for CTI data, the system returns a series of chronologically ordered messages; users are then able to make decisions by examining the returned results (e.g., series of observations about a particular actor) in a way similar to examining an email threat. The CIF Server consists of a few different modules including csirtg-fm, cif-worker, cif-router, cif-enricher, and ElasticSearch. The csirtg-fm module has two primary capabilities: To fetch files using http(s) to/ from the local file system and to parse files using YAML to parse regex, JSON, XML, CSV, RSS, HTML and plain text files. Moreover, the cif-worker module helps the CIF extract additional intelligence from collected threat data, the cif-router module provides a ZMQ [64] broker, the cif-enricher is responsible for enriching incoming intelligence with additional information like geolocation or FQDN while the ElasticSearch module is a data Warehouse for storing (meta)data for intrusions.

	ZeroFox	CTAC	SearchLight	Intel 471	Security Ratings	Flashpoint	BreachAlert	F5 Labs
Vulnerabilities tracking	✓	✓	✓	✓	✓	✓	✗	✓
TTPs tracking	✓	✓	✗	✓	✓	✓	✗	✓
Organisation assets' tracking	✓	✓	✓	✓	✓	✗	✓	✗
Deep/dark web monitoring	✓	✓	✓	✓	✗	✓	✓	✓
Social media monitoring	✓	✓	✓	✗	✓	✗	✓	✗
CTI reports	✓	✓	✓	✓	✓	✓	✓	✓
API	✓	✓	✓	✓	✓	✓	✓	✗
Alerting mechanism	✓	✗	✓	✓	✓	✓	✓	✗

Table 2.2: CTI services comparison

Finally, all stored CTI is accessible through CIF's REST API, which is built upon Swagger [50].

2.2.2 CTI Services

There is a plethora of threat intelligence services, that are able to support the CTI sharing life-cycle, in various ways. This subsection presents and compares several representative threat intelligence services; namely: ZeroFox [63], CTAC by Wapack Labs [7], SearchLight by Digital Shadows [44], Intel 471 [22], Flashpoint Intelligence Platform [14], Security Ratings by BitSight [2], BreachAlert by SKURIO [3] and F5 Labs [13]. The comparison reviews the services' capabilities with regard to the following aspects: *vulnerabilities, TTPs* and *organization assets' tracking*, *deep / dark web* and *social media monitoring*, *CTI reports provision*, *supported API* and *alerting mechanism.* The results of the comparison conducted are presented in Table 2.2.

All of the presented threat intelligence services provide CTI reports, as concluded by Table 2.2 results. The *organisation assets' tracking* refers to the capabilities of each threat intelligence service to monitor specific CTI, related to an organisation's assets. Finally, the *Alerting mechanism* indicates whether the threat intelligence service is able to alert its users about events that may concern them.

2.2.3 CTI Platforms

Similarly to the CTI services presented previously, there are numerous CTI platforms, which aim to organise and manage CTI in a centralized manner and also integrate them with other security solutions. In this subsection we present numerous

	Helix	**Recorded Future**	**Cyjax**	**EclecticIQ**	**Cyber Advisor**	**BloxOne**	**ThreatStream**	**ThreatQ**	**Soltra**	**ThreatConnect**	**VDMR**	**MANTIS**	**BrightCloud**
Vulnerabilities tracking	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗
TTPs tracking	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗
Organisation assets' tracking	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓
Threat monitoring (IoCs, Malicious IPs, etc.)	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓
CTI intercorrelation	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Deep/dark web monitoring	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
Social media monitoring	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
CTI reports	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Alerting mechanism	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✗	✓
Remediation proposals	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗
CTI sharing	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗
Data formats	STIX/ TAXII, XML, JSON	STIX/ TAXII	STIX/ TAXII, JSON	STIX/ TAXII, OASIS	STIX/ TAXII, JSON	STIX/ TAXII, JSON, CSV, more	STIX/ TAXII, JSON, CSV, more	STIX/ TAXII, Open-IOC, Snort, Suricata	STIX/ TAXII	STIX/ TAXII	JSON, XML	STIX/ TAXII, CybOX, Open-IOC, IODEF, JSON, more	XML
On-premises	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
SaaS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓

Table 2.3: Threat intelligence platforms

aspects of representative threat intelligence platforms; namely: Helix by FireEye [20], Recorded Future [42], Cyjax [9], EclecticIQ [10], Cyber Advisor by SurfWatch Labs [8], BloxOne by Infoblox [80], ThreatStream by Anomali [55], ThreatQ by ThreatQuotient [54], Soltra by Celerium [47], ThreatConnect [53], VDMR by Qualys [56], MANTIS by SIEMENS [51] and BrightCloud by Webroot [4].

All of the aforementioned CTI platforms provide an API and search capabilities over their CTI. Table 2.3 presents the aspects of the threat intelligence platforms. The *CTI intercorrelation* describes whether each platform is intercorrelating the discovered CTI artifacts. Next, the *Remediation proposals* indicates the ability of the platforms to propose remediation strategies and techniques over vulnerable configurations or breached assets. Finally, *SaaS* and *On-premises* present the type of the platforms, whether they are available as a web-hosted service or executed locally, respectively.

2.3 Implementations of Frameworks, Tools and Platforms for Targeted CTI Gathering and Sharing

In [81], the authors propose the employment of a cross-linked and correlated database to collect, extract, filter and visualize vulnerability data, across multiple existing repositories, whereby CPS vulnerability information is inferred. They highlight the challenge of dealing with different vulnerabilities that might contribute to threats of different levels of impact, through the variety of levels of losses in confidentiality, integrity and availability. To counter such issues, the authors propose correlated database management techniques in the vulnerability data processing, in order to discover CTI concerning CPS vulnerabilities, to gain multi-level vulnerability analysis from both component-perspective and asset-perspective, and to visualize the connection between vulnerabilities, threats and attacks. In order to realize these features, the authors propose a three-step agenda, which consists of:

1. the vulnerability database preparation,
2. the CPS asset database preparation, and
3. the correlation between asset-data and vulnerability-data.

First, the vulnerability database preparation requires the extraction of attributes of vulnerabilities from a variety of repositories into one database. The proposed methodology is to collect base reports of vulnerabilities, using the formal identifier of CVE ID as index. These base reports include data such as associated vendor records, preliminary analysis of reported vulnerability severity, using CVSS, and several other pertinent metadata, such as the publication date. Next, the CPS asset database preparation requires the extraction of information such as CPE, CCE, and CWE, via information retrieval techniques, from sources such as CVE by MITRE [6], NVD by NIST [33], Exploit-DB [12], and SecurityFocus [45], which are valuable resources for security analysis data. Finally, knowledge-based reasoning approaches are applied to automatically abstract vulnerability attributes for concept-modelling and information intercorrelation. Thus, features of different vulnerabilities are abstracted and updated, via up-to-date vulnerability repositories, which are then clustered into vulnerability instances and stored in a standardized vulnerability database. In addition to that, features of components, such as component properties, version, etc., are also abstracted and stored in an asset database. Consequently, information from the two databases are queried and correlated, to generate an asset-based vulnerability database. Similar works are carried out in [86] and [91], where the authors propose similar approaches to implement IoT- and robotics-oriented vulnerability databases, respectively. Finally, in [82], the authors present VuWaDB, a vulnerability workaround database, by gathering, extracting, analyzing and labeling CTI from numerous certified vulnerability databases. The resulting workarounds are then being organized in action-oriented categories, that provide information on how each of the gathered vulnerabilities should be addressed.

The authors of [85] present an ontology-based cybersecurity framework, which makes use of knowledge reasoning for IoT. The framework is composed of two approaches; one at the design time, and one at the run time. At the design time, the framework foresees the application of the model-driven methodology, in order to build and adopt existing security services semi-automatically, by using the same security service specifications, at a high-level of abstraction, in the development of technological components. The design time layer follows a two-step implementation; service design and adaptation, and process and service deployment. In the first step (service design and adaptation), the framework explores the model-driven development, using the MDSEA [71] methodology, to optimize the service development. During this phase, any deployment aspects are still abstracted, to focus only on the functionality. In the second step (process and service deployment), the framework enables the implementation of the designed security services, within the business process. The implemented services are then made available to the knowledge base, via the integration of the IoTSec ontology [23, 85], for future requests to address the same types of security issues. The IoTSec ontology stands at the integration layer, which links the design-time layer, with the run-time. The run-time layer of the proposed framework is responsible for identifying and classifying known threats, from a knowledge base, to provide appropriate security service and prevent future occurrences. This layer follows three main steps; monitoring, data integration, and knowledge provisioning. During the first phase (monitoring), the run-time layer analyzes business processes and technological assets, in order to detect threats and vulnerabilities for the IoT system. It entails specific monitoring tools to identify threats such as IP-tables/Netfilter [34], Snort [46], Prelude [39], Suricata [49], and vulnerabilities such as the Retina Network Community [43]. Security alerts that are generated by the aforementioned monitoring tools, follow the IDMEF standard. Raised alerts are used to classify threats and vulnerabilities from the IoTSec ontology. Then, the proposed framework uses SPARQL language, to perform queries on the ontology, to gather suitable information from potential threats in the IoT environment. In the second step (data integration), the framework provides cybersecurity information from distinct data sources, using the Ontop framework [65], which enables the instantiation of a knowledge base from IoTSec ontology. The data population also uses Ontop, to support data access through a conceptual layer, rewriting the SPARQL queries, over the virtual RDF graph, to SQL queries. Finally, during the third step of the run-time layer, knowledge provisioning derives from the IoTSec ontology, by using the SPARQL language to perform queries and check all information in the knowledge base. Due to language flexibility, correlations between ontology classes can be used to cross information regarding the IoT environment.

In [84], the work carried out aims on the information gathering task of IoT-related CTI and presents a novel architecture that is able to provide a crawling infrastructure for a variety of CTI sources in the clear, social and dark web. The approach followed employs a thematically focused crawler, for directing the crawl towards websites of interest to the CTI gathering task. It is a combinatory approach, which makes use of machine learning techniques for the open domain crawling, and a regex-based link filtering for structured domains, such as forums. Any retrieved content is then stored in a NoSQL datastore and then it is inspected, in order to decide whether it is useful to the task. This is achieved by employing statistical language modelling techniques, which allow the training of a language model, that is able to:

1. capture and exploit the most salient words for the given task by building upon user conversations,
2. compute the semantic relatedness between the crawled content and the task at hand by leveraging the identified salient words, and
3. classify the crawled content according to its relevance/usefulness, based on its semantic similarity to the CTI gathering task.

The architecture presented in this work is entirely designed and developed, using open-source software. Namely, it uses ACHE Crawler [1] for the focused crawling, gensim [15], which is an implementation of words embeddings for the latent topic modeling, and the NoSQL database MongoDB [31], which acts as the storage of the topic models and the crawled content. By utilizing NLP techniques for named entity recognition, CTI can be extracted from the relevant harvested content. That CTI can then be correlated with existing knowledge encompassed within a vulnerability database, in order to enrich its content.

Extending the work carried out in [84], the inTIME system presented in [83], aims to combine the aforementioned crawling procedure, with social media monitoring techniques and the utilization of MISP [28]. By doing so, the system provides a zero-administration, open-source, integrated framework, that is able to support security analysts to: deploy various data acquisition services, automatically rank the collected content, identify and extract CTI artifacts with the use of NLP processes, further investigate the identified CTI and finally, manage, share and collaborate on the stored CTI, via open standards and intuitive tools. It fully supports the complete CTI life cycle, providing a holistic CTI gathering and sharing approach.

Chapter 3

MISP

In Chapter 2 we have seen that MISP [28] takes the lead in the platforms' race, when compared to the rest of the surveyed platforms, for the purposes of the CTI life-cycle support [66, 70]. Thus, it is the platform of choice for the CTI management and sharing of our system. Specifically, **ASPIS** will use, extend, enhance and modify MISP, in order to enrich and optimize its storing capabilities (as described in Section 4.3). In the rest of this chapter, we will describe the essential details of MISP that regard its (i) data model, (ii) CTI sharing properties and features, (iii) additional features and (iv) UI layout design.

3.1 The Data Model

The data model of MISP follows a simple approach, while at the same time it enables more complex functionalities. The main objective is to have a minimum viable data format, which can be extended according to the needs of additional complexity, instead of trying to capture all possible future requirements in advance.

A new entry in MISP is called an *event* object, which is defined by a set of characteristics, along with all kinds of respective descriptions for indicators, including attachments. These characteristics are called *attributes* in MISP, and they provide all useful information to the event, such as an IoC date, threat level, comments, organization that created it, and so on. Attributes are mainly described by two fields; *category* and *type*. The main difference is that the *category* field describes what the attribute represents, such as network activity, financial fraud, etc., while the *type* field describes how the attribute represents the chosen category. For example, an attribute type might be a checksum, a filename, a hostname, an ip-address, and so on. The actual payload of the attribute is stored in the *value* field.

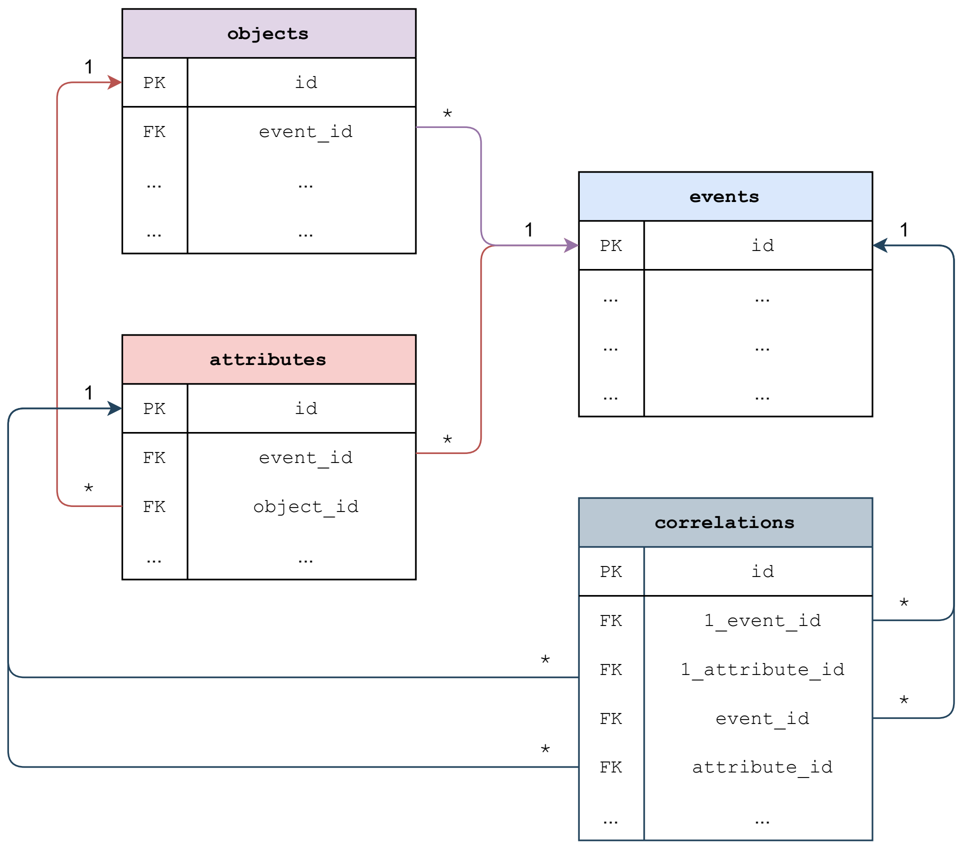


Figure 3.1: MISP database schema abstracted overview

3.1.1 MISP Database Overview for Storing CTI

MISP provides an extensive database schema, which is capable of supporting a multi-user environment, with complex data management, for a wide variety of vertical sectors. This is achieved through the implemented MISP attribute *categories* and *types*, *objects*, and *taxonomies*. **ASPIS** stores all CTI artifacts in clusters, in the MISP platform. In this section we will mainly focus on the database structure of MISP.

Any CTI artifact, such as a CVE ID of a vulnerability, is stored in the MISP database in the form of *attributes*. Multiple attributes can be grouped to form an *object*, which mainly consists a bigger CTI artifact, like a vulnerability report. Both attributes and objects must be attached to *events*, which basically serve as the records of the artifacts storage. Finally, MISP enables an event to be correlated with other events, through matching techniques over their attributes. Each *correlation* that may occur between events serves as a bond, which also indicates the matching attribute. In Figure 3.1, we present an abstract overview of the database schema part, which is used for storing the CTI. In the following subsections, we will also provide an extended description of the presented tables and their columns.

3.1.2 Events Table

The *events* table is a meta-structure scheme, where attributes, objects and meta-data are embedded to compose a sufficient set of indicators, that is able to describe a specific case, like a vulnerability report. An event can be composed from an incident, a security analysis report or a specific threat actor analysis. The meaning of an event derives solely from the information embedded within it. In our case, one event is a collection of objects that are used to describe the CTI artifacts. Table 3.1 presents a detailed description of the *events* table.

3.1.3 Objects Table

Objects serve as a contextual bond between a list of attributes within an event. Their main purpose is to describe more complex structures than can be described by a single attribute. Each object is created using an *Object Template* and carries the meta-data of the template used for its creation within. Objects belong to a meta-category and are defined by a name. The schema used is described by the template_uuid and template_version fields. Table 3.2 provides a detailed description of the *objects* table.

3.1.4 Attributes Table

Attributes are used to describe the indicators and contextual data of an event. The main information contained in an attribute is formed by category-type-value triplets, where the category and type give meaning and context to the value. Through the various category-type combinations, a wide range of information can be conveyed. Table 3.3 presents a detailed description of the *attributes* table.

3.1.5 Correlations Table

Correlations serve as a bonding system between the stored events. Their main purpose is to describe any artifacts' matching that may have occurred between the events through the MISP Correlation Engine (which is briefly discussed in Section 3.5.3.4). Table 3.4 provides a detailed description of the *correlations* table.

From the correlations table definition, as presented in Table 3.4, having the following set of attributes: {`id, value, 1_event_id, 1_attribute_id, event_id, attribute_id, org_id, distribution, a_distribution, sharing_group_id, a_sharing_group_id, date, info`}, we realize the following dependencies for the *correlations* relation:

FD1. `id` → `value 1_event_id 1_attribute_id event_id attribute_id org_id distribution a_distribution sharing_group_id a_sharing_group_id date info`

FD2. `1_event_id 1_attribute_id event_id attribute_id org_id` → `id`

FD3. `1_event_id 1_attribute_id` → `value`

FD4. `event_id attribute_id` → `value`

Particularly, from **FD1**, `id` is the unique identifier of the *correlations* relation, and hence it is able to identify all attributes. Then, one organization of MISP can generate only one correlation entry for two correlated events on a common valued set of attributes. Thus, from **FD2**, `1_event_id 1_attribute_id event_id attribute_id org_id` is able to provide the unique identifier of each correlation entry. Finally, from **FD3** and **FD4**, `event_id attribute_id` refers to a particular correlated value, and hence that is dependent of the two IDs.

id INT(11)	The human-readable identifier associated to the event for a specific MISP instance.
uuid VARCHAR(40)	The Universally Unique IDentifier (UUID) of the event. The uuid must be preserved for any updates or transfer of the same event.
published TINYINT(1)	The event publication state. If the event was published, the published value is true (1). In any other publication state, the published value is false (0).
info TEXT	It represents the information field of the event. It is a free-text value to provide a human-readable summary of the event.
threat_level_id INT(11)	The threat level. [4: *Undefined*, 3: *Low*, 2: *Medium*, 1: *High*].
analysis TINYINT(4)	The analysis level. [0: *Initial*, 1: *Ongoing*, 2: *Complete*].
date DATE	A reference date to the event in the *YYYY-MM-DD* format. This date corresponds to the date that the event was generated.
timestamp INT(11)	A reference time when the event was created or last updated/edited on the instance. It is expressed in seconds since 1st of January 1970 (Unix timestamp).
publish_timestamp INT(11)	A reference time when the event was published on the instance. It is expressed in seconds since 1st of January 1970 (Unix timestamp). If the event was never published, the published_timestamp is set to 0. If it is present but the published flag is set to false, then it represents the previous publication timestamp.
org_id[1] INT(11)	A human-readable identifier referencing an Org object of the organisation which generated the event.
orgc_id[1] INT(11)	A human-readable identifier referencing an Orgc object of the organisation which created the event.
attribute_count UNSIGNED INT(11)	The number of attributes in the event.
distribution TINYINT(4)	The basic distribution rules of the event. The system must adhere to the distribution setting for access control and for dissemination of the event. [0: *Your Organization Only*, 1: *This Community Only*, 2: *Connected Communities*, 3: *All Communities*, 4: *Sharing Group*].
sharing_group_id INT(11)	A human-readable identifier referencing a Sharing Group object that defines the distribution of the event, if distribution level 4 is set. (Sharing Groups can be defined in the MISP UI).
extends_uuid VARCHAR(40)	Which event is extended by this event. It is described as a Universally Unique IDentifier (UUID), with the UUID of the extended event.
disable_correlation TINYINT(1)	A setting that allows an event's attributes to be correlated with attributes from other events.

Table 3.1: MISP *events* table

[1]MISP stores separately the org_id from orgc_id, to support syncing actions between different instances. Thus, in such a case, the organization that generated an event in an instance, may have synced it from another instance, so the creator should remain the same.

`id` INT(11)	The human-readable identifier associated to the attribute for a specific MISP instance.
`uuid` VARCHAR(40)	The Universally Unique IDentifier (UUID) of the object. The uuid must be preserved for any updates or transfer of the same object.
`name` VARCHAR(255)	The human-readable name of the object describing the intent of the object package. (I.e. vulnerability, vuldb-vulnerability, weakness, and so on. The MISP Objects employed are presented in Subsection 4.3.1).
`meta-category` VARCHAR(255)	The sub-category of objects that the given object belongs to. meta-categories are not tied to a fixed list of options but can be created on the fly. (I.e. a vuldb-vulnerability object has a *vulnerability* meta-category, as we defined it in MISP.)
`description` TEXT	Human-readable description of the given object type, as derived from the template used for creation.
`template_uuid` VARCHAR(40)	The Universally Unique IDentifier (UUID) of the template used to create the object. It must remain the same during a transfer or an update, in order to preserve the object's association with the correct template used for creation.
`template_version` INT(11)	A numeric incrementing version of the template used to create the object. It is used to associate the object to the correct version of the template and together with the template_uuid forms an association to the correct template type and version.
`event_id` INT(11)	The human-readable identifier of the event that the object belongs to on a specific MISP instance.
`timestamp` INT(11)	A reference time when the object was created or last modified. It is expressed in seconds since 1st of January 1970 (Unix timestamp).
`distribution` TINYINT(4)	The basic distribution rules of the object. The system must adhere to the distribution setting for access control and for dissemination of the object. [0: *Your Organization Only*, 1: *This Community Only*, 2: *Connected Communities*, 3: *All Communities*, 4: *Sharing Group*].
`sharing_group_id` INT(11)	A human-readable identifier referencing a Sharing Group object that defines the distribution of the object, if distribution level *4* is set. (Sharing Groups can be defined in the MISP UI).
`comment` TEXT	A contextual comment field.
`deleted` TINYINT(1)	A setting that allows attributes, within the object, to be revoked. Revoked attributes are not actionable and exist merely to inform other instances of a revocation.
`first_seen` BIGINT(20)	A reference time when the object was first seen. It is expressed as a datetime up to the micro-second with time zone support.
`last_seen` BIGINT(20)	A reference time when the object was last seen. It is expressed as a datetime up to the micro-second with time zone support.

Table 3.2: MISP *objects* table

id INT(11)	The human-readable identifier associated to the attribute for a specific MISP instance.
uuid VARCHAR(40)	The Universally Unique IDentifier (UUID) of the attribute. The uuid must be preserved for any updates or transfer of the same attribute.
type VARCHAR(100)	The means through which an attribute tries to describe the intent of the attribute creator, using a list of pre-defined attribute types. (I.e. text, link, datetime, float)
category VARCHAR(255)	The intent of what the attribute is describing as selected by the attribute creator, using a list of pre-defined attribute categories. (I.e. External analysis, Network activity, Vulnerability)
to_ids TINYINT(1)	Whether the attribute is meant to be actionable. Actionable defined attributes that can be used in automated processes as a pattern for detection in Local or Network Intrusion Detection System, log analysis tools or even filtering mechanisms.
event_id INT(11)	A human-readable identifier referencing the Event object that the attribute belongs to.
object_id INT(11)	A human-readable identifier referencing the MISP Object that the attribute belongs to.
object_relation VARCHAR(255)	The attribute of the MISP Object that the attribute describes. (The MISP Object attributes employed are presented in Subsection 4.3.1).
distribution TINYINT(4)	The basic distribution rules of the attribute. The system must adhere to the distribution setting for access control and for dissemination of the attribute. [0: *Your Organization Only*, 1: *This Community Only*, 2: *Connected Communities*, 3: *All Communities*, 4: *Sharing Group*, 5: *Inherit Event*].
timestamp INT(11)	A reference time when the attribute was created or last modified. It is expressed in seconds since 1st of January 1970 (Unix timestamp).
comment TEXT	A contextual comment field.
sharing_group_id INT(11)	A human-readable identifier referencing a Sharing Group object that defines the distribution of the attribute, if distribution level 4 is set. (Sharing Groups can be defined in the MISP UI).
deleted TINYINT(1)	A setting that allows attributes to be revoked. Revoked attributes are not actionable and exist merely to inform other instances of a revocation.
value TEXT	The payload of an attribute. The format of the value is dependent on the type of the attribute.
first_seen BIGINT(20)	A reference time when the attribute was first seen. It is expressed as a datetime up to the micro-second with time zone support.
last_seen BIGINT(20)	A reference time when the attribute was last seen. It is expressed as a datetime up to the micro-second with time zone support.
disable_correlation TINYINT(1)	A setting that allows an attribute of a specific object_relation to be correlated with attributes from other events with the same object_relation.

Table 3.3: MISP *attributes* table

`id` `INT(11)`	It represents the human-readable identifier associated to the correlation for a specific MISP instance.
`value` `TEXT`	The payload of the correlated attribute. The format of the value is dependent on the type of the attribute.
`1_event_id` `INT(11)`	The human-readable identifier of the event at hand that the correlation refers to on a specific MISP instance.
`1_attribute_id` `INT(11)`	The human-readable identifier of the attribute at hand that the correlation refers to on a specific MISP instance.
`event_id` `INT(11)`	The human-readable identifier of the correlating event to the one at hand that the correlation refers to on a specific MISP instance.
`attribute_id` `INT(11)`	The human-readable identifier of the correlating attribute to the one at hand that the correlation refers to on a specific MISP instance.
`org_id` `INT(11)`	A human-readable identifier referencing an Org object of the organisation which generated the correlation.
`distribution` `TINYINT(4)`	The basic distribution rules of the correlation. The system must adhere to the distribution setting for access control and for dissemination of the event. [0: *Your Organization Only*, 1: *This Community Only*, 2: *Connected Communities*, 3: *All Communities*, 4: *Sharing Group*].
`a_distribution` `TINYINT(4)`	The basic distribution rules of the correlated attribute. The system must adhere to the distribution setting for access control and for dissemination of the event. [0: *Your Organization Only*, 1: *This Community Only*, 2: *Connected Communities*, 3: *All Communities*, 4: *Sharing Group*]. This is useful for attributes that may contain sensitive information.
`sharing_group_id` `INT(11)`	A human-readable identifier referencing a Sharing Group object that defines the distribution of the correlation, if distribution level 4 is set. (Sharing Groups can be defined in the MISP UI).
`a_sharing_group_id` `INT(11)`	A human-readable identifier referencing a Sharing Group object that defines the distribution of the correlated attribute, if distribution level 4 is set. (Sharing Groups can be defined in the MISP UI). This is useful for attributes that may contain sensitive information.
`date` `DATE`	A reference date to the correlation in the *YYYY-MM-DD* format. This date corresponds to the date that the correlation was generated.
`info` `TEXT`	The information field of the correlated event at hand. It is a free-text value to provide a human-readable summary of the event.

Table 3.4: MISP *correlations* table

3.2 CTI Sharing Properties and Features

There are two main aspects that regard the sharing model of MISP. First, MISP enables its users to select the *sharing level* of the information stored in the platform. For example, the sharer can disseminate the information at hand with a specific organization, a community of organizations, interconnected communities, all participants of MISP, or even define a sharing group manually. The next main aspect of MISP, is the *proposals* feature. While the modification of events is only permitted to member of the creating organization, proposals allow users to make suggestions for changes to an event, created by another organization. A proposal is reported back to the original creator of the event, who may accept the change or discard it. Then, the outcome of the creator's decision will be propagated to all interconnected instances. An example of this feature is the reporting of false positives to the event creator, asking for an error correction.

3.3 Additional Features

Furthermore, MISP provides various features, including:

PyMISP: A Python Library for the Implementation of MISP API. PyMISP provides users with fetching, adding, updating, deleting and searching capabilities over the stored events/attributes or *samples.* A full documentation of PyMISP is provided in [40].

The Free-Text Import Tool. It enables users to copy and paste raw data (in free-text format) into a single data field, that through a heuristic algorithm matches the attributes. The resulting attributes are then presented to the user who proceeds to validate the findings.

MISP Tagging Mechanism. It enables users to define customizable tags, through which they can later filter the events and classify the encompassed information. Furthermore, the tags can also be exportable, hence allowing the reusing of the same tags from other MISP instances.

MISP Taxonomies. A *taxonomy* is a triplet of tags, which is described by a *namespace*, a *predicate* and a *value.* Through the utilization of taxonomies' repository, organizations have a common format for describing incidents. Furthermore, if the predefined taxonomies do not fit the description of an event, users can define their own.

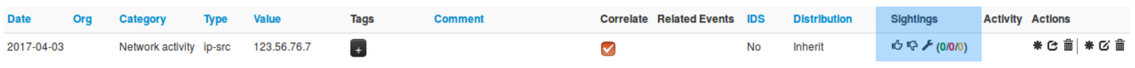
MISP Instances' Syncing. MISP is provided with a synchronization protocol, which supports four main features; *pull*, *push*, *cherry-picking*, and the *feed* system. The *pull* feature allows a MISP instance to discover available and accessible events on a connected instance and download any new or modified events. The *push* mechanism allows a MISP instance to convert events to a JSON format that is transferable to remote instances. The *cherry-picking* feature is an alternative to the pull method, which allows users to decide which events should be pulled to the local instance. Finally, the *feed* mechanism allows a MISP instance to generate a dump of JSON files, which derive from a selection of events that an organization was to publish. Then, the output can be served via a web server, through which other MISP instances can access and retrieve the contents via the UI, similarly to the cherry-picking.

MISP Sightings. MISP provides a sighting system, which allows users to react on attributes on an event. Originally, it was designed to provide an easy method for users to verify a given attribute, hence raising its credibility. Since the MISP version 2.4.66, sightings have been improved to provide a method to signal false positives, but also to give an expiration date for some attributes [29]. As stated previously, MISP Sightings are a way for users to state that they have seen or noticed an attribute and also confirm its validity. An attribute may be spotted several times by the same user, and thus a single user can use sighting several times on a single attribute. Sometimes, some attributes may be considered as false positives, and similarly to the previous case, users can signal a single attribute as a false positive several times. There is also the case of some attributes being valid for a certain period of time (for instance, in case of a phishing campaign that is assumed to be up for only one week). In this case, users can assign an expiration date to an attribute, but this time, there can only be one valid expiration date per organization of the MISP instance. Finally, as shown in Figure 3.2, a *sighting* is applied to every attribute, under the column *Sightings*, identifiable by its colored numbers. This column contains three icons and three values. The three values show respectively:

- the number of true positives detected with the attribute, in green,
- the number of times the attribute has been marked as false positive, in red, and
- the number of different expiration dates that have been affected on this attribute.

Finally, concerning the three icons:

- the first (Thumb up) allows to add a sighting (true positive) on an attribute,
- the second one (Thumb down) allows to mark the attribute as a false positive, and



Date	Org	Category	Type	Value	Tags	Comment	Correlate	Related Events	IDS	Distribution	Sightings	Activity	Actions
2017-04-03		Network activity	ip-src	123.56.76.7	+		✓		No	Inherit	(0/0/0)		

Figure 3.2: MISP Sightings

- the third one (Tool) opens a popup for advanced sightings, showing sightings details and allowing different actions.

3.4 Current State of MISP

MISP is an active open-source platform, which is enhanced, fixed and introduced with additional support, approximately on a monthly basis. During the timeframe of the work carried out, as showcased in the thesis, we used the latest version of MISP (2.4.132), which had been released on September 21, 2020.

Appendix B provides a brief summary of the most notable changes that fulfil our expectations and illustrate additional MISP capabilities, as they were extracted from its release page.

Generally, it is strongly suggested to keep MISP up to date in accordance with the latest version published, in order to fully exploit the platform's improvements and fixes.

3.5 General MISP Layout

The MISP layout differentiates whether the end-user is a simple user or the administrator of the instance.

3.5.1 Simple User

The top bar of a simple user's interface (as shown in Figure 3.3) includes the tabs described below.

Home tab guides the user to the initial profiling interface of the application.

Event Actions gives access to all users to functionalities that are related to creation, modification, deletion, publishing, searching and listing of the events and attributes.

Galaxies guide the user to the list of MISP *Galaxies* (supported vertical sectors' groups of objects) on MISP.

Input Filters define the type of data that enter in each instance. The tab *Input filters* has a drop-down list with various options. *Import Regexp* allows the

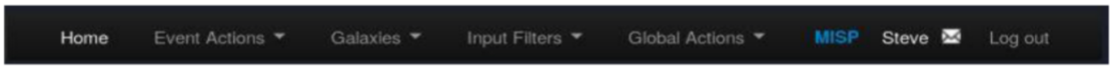


Figure 3.3: MISP simple user's top bar

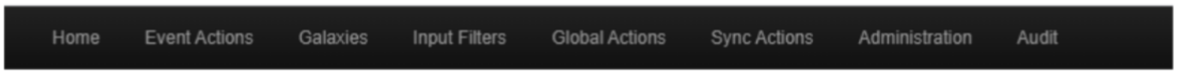


Figure 3.4: MISP administrator's top bar

admin of the system to view the Regular Expression rules which define the data that entered into the system.

Global Actions allows the user to have access to information regarding MISP and a specific instance, also has the capability to view and modify the profile, receive a manual of MISP. Some options include information regarding the latest MISP news, the sharing groups that the organisation communicate, organisation role permissions etc. Also, administrator can view and manage profiling details, can view organisations that exists on a specific instance as well as the statics which are referred to the users and the data on this instance.

MISP tab provides a link that leads to the *baseURL*, which refers to the MISP hostname.

User. In Figure 3.3 the simple user's tab is the *Steve* tab, which is auto generated from the user email address of current logged in user.

The Envelop Icon guides the user to the User Dashboard, which contains the latest information of the account's management such as, notifications, modifications of the account etc.

Log Out leads the user out of the system.

3.5.2 Administrator

The top bar of an administrator's interface (as shown in Figure 3.4) includes the tabs described below.

Home tab guides the administrator to the initial profiling interface of the application.

Event Actions gives access to all functionalities related to creation, modification, deletion, publishing, searching and listing of the events and attributes.

Galaxies guide the administrator to the list of MISP Galaxies and enables the updating the galaxy as well.

Input Filters has a drop-down list with various options. *Import Regexp* allows theadmin of the system to view the Regular Expression rules which define the data that are inserted into the system. Therefore, a site administrator or a user with regex permissions can edit the rules. *Signature Whitelist* includes the kind of information that should be forbidden by the system, and the site administrator can edit this list. *List warninglists* includes indicators for potential false positives, errors or mistakes. The warning lists are integrated in MISP to display an info/warning box at the event and attribute level.

Global Actions allows the administrator to have access to information regarding MISP and a specific instance, also has the capability to view and modify the profile, receive a manual of MISP. Some options include information regarding the latest MISP news, the sharing groups that the organisation communicates, organisation role permissions etc. Also, the administrator can view and manage profiling details, can view organisations that exist on a specific instance as well as the statistics, which are referred to the users and the data on this instance.

Sync Actions prerequires administrator's access rights, then the admin can visualize the instances connections. Sync Actions includes *List Servers* and *List Feeds.*

Audit needs permission to be accessible. The administrator can visualize organization logs (or for site admins for the entire system) and search targeted the logs of a specific event.

MISP tab provides a link that leads to the baseURL.

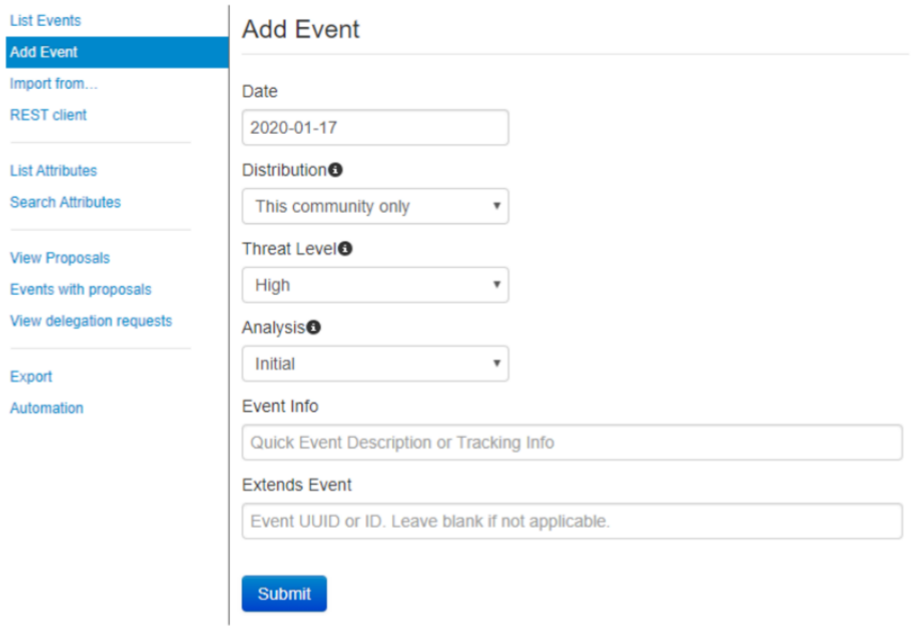
Admin can handle user's information. More specifically, view, modify, delete and add users in the instance. For coordination issues or in case of any problem in user's accounts, the admin has the capability to contact the current and future users and provide them temporary passwords. The admin has the same capabilities as before, towards the organisations.

The Envelop Icon guides the user to the User Dashboard, which contains the latest information of the account management such as notifications, modifications of the account etc.

Log Out leads you out of the system.

3.5.3 Events

As referred previously, *Event Actions* gives access to all users, to functionalities that are related to the creation, modification, deletion, publishing, searching and



Add Event

Date: 2020-01-17

Distribution: This community only

Threat Level: High

Analysis: Initial

Event Info: Quick Event Description or Tracking Info

Extends Event: Event UUID or ID. Leave blank if not applicable.

Figure 3.5: Adding process of an Event in MISP

listing of the events and attributes. Some of the aforementioned functionalities are presented below.

3.5.3.1 Creating an Event

In order to create an event, users need to make three actions:

Generation of the event itself (Figure 3.5). This means that the basic event will be created without any actual attributes and will store general information, such as the description, time, and risk level of the incident.

Populating the event with attributes and attachments by clicking on the tab *Add Event* and completing the particular form.

Publishing the event.

Every user needs to complete the fields with the exact information. The user should pay attention through the data completion, since they are consisting vital elements of the incident's description.

Date indicates the date of the incident.

Distribution is a setting control, that reveals who can see the event, once it becomes published and eventually when it is pulled. Also, you can control whether the event will be shared to other servers too or not.

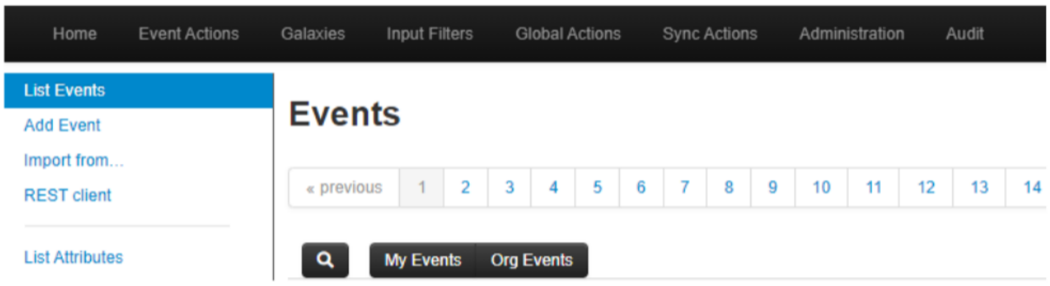


Figure 3.6: MISP layout in the List of Events

Threat Level indicates the risk level of an event. Incidents can be classified into three threat categories; namely low, medium and high. Also, this field can remain unclarified.

Analysis specifies the event's stage of the analysis; more specifically (a) Initial, (b) Ongoing, (c) Completed.

Event Info gives information regarding the malware/incident, with a brief description starting with the internal reference.

3.5.3.2 List of Events

In this list, users are presented with information regarding the interface of MISP, that allows the user to view, search for events and attributes of events, that are already stored in the system in various ways (Figure 3.6). The menu, through the tab "List events", allows for the creation of a list with the 60 last events in the instance, without presenting the attributes, and the pagination of the rest events into lists of 60 events, with regard to the time they were created on the MISP instance.

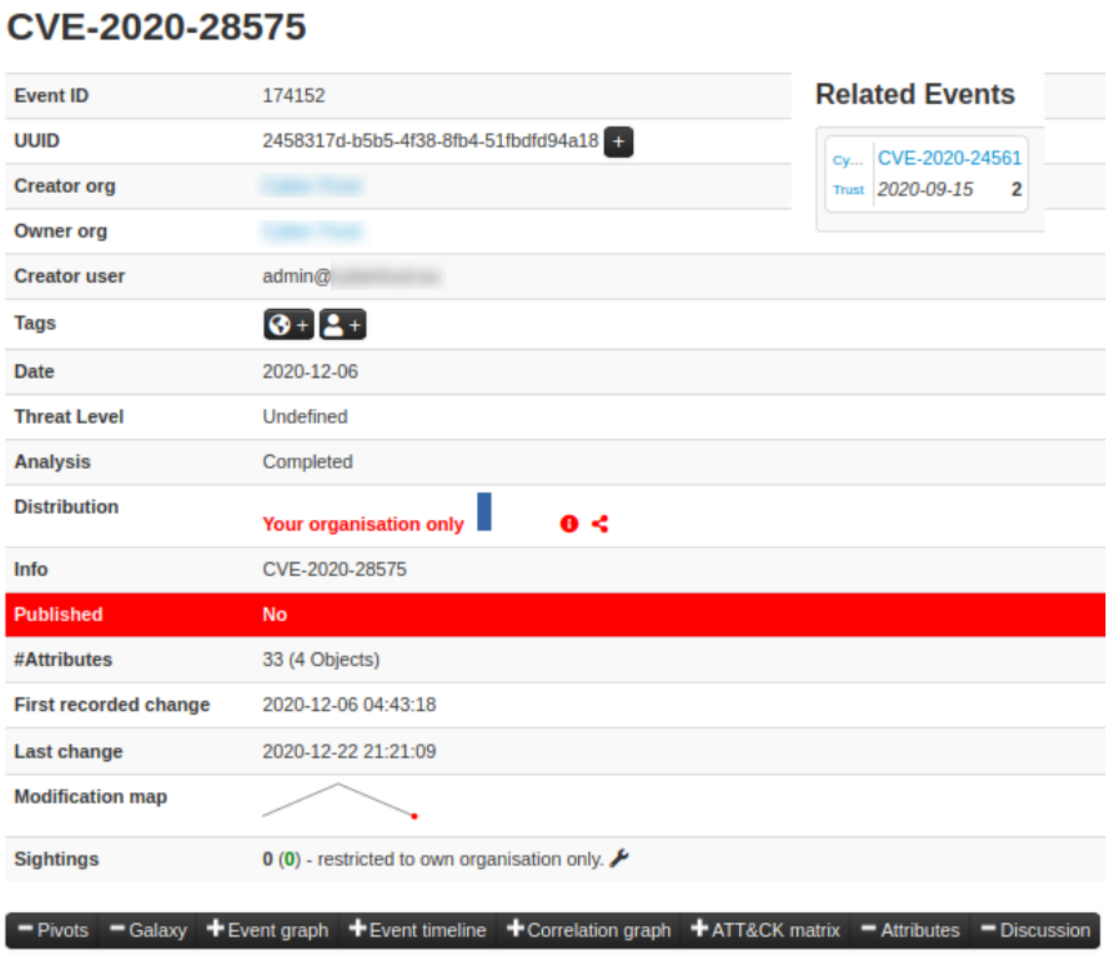
3.5.3.3 Events View

An event in the MISP UI is a tab that encompasses various characteristics of the MISP event structure, along wtih its attributes and objects. Specifically, such characteristics are indicated below:

ID shows the ID of the event.

UUID provided in order to avoid collisions between events and attributes. A UUID is assigned to each one of them separately to uniquely identify them.

Org refers to the organization that has either generated the event (Creator org) on this instance or created the event (Owner org). A string that represents the organization is also shown next to it.



CVE-2020-28575

Event ID	174152
UUID	2458317d-b5b5-4f38-8fb4-51fbdfd94a18 +
Creator org	[redacted]
Owner org	[redacted]
Creator user	admin@[redacted]
Tags	+ +
Date	2020-12-06
Threat Level	Undefined
Analysis	Completed
Distribution	Your organisation only
Info	CVE-2020-28575
Published	No
#Attributes	33 (4 Objects)
First recorded change	2020-12-06 04:43:18
Last change	2020-12-22 21:21:09
Modification map	
Sightings	0 (0) - restricted to own organisation only.

Related Events

Cy...	CVE-2020-24561	
Trust	2020-09-15	2

Figure 3.7: MISP Event View

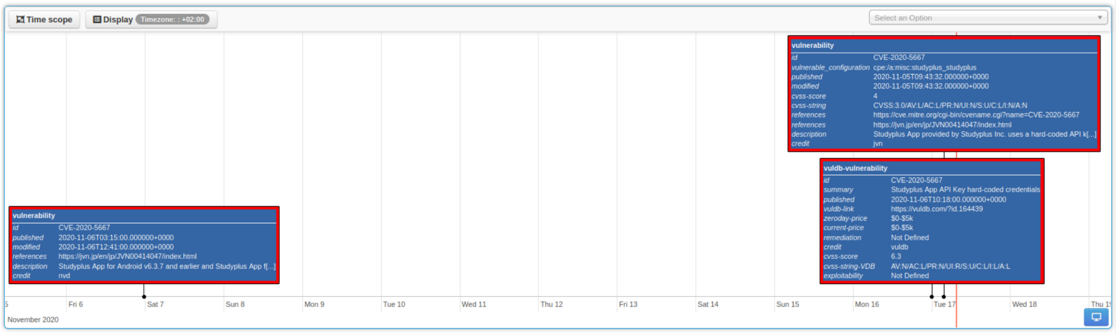
Tags shows a list of tags associated with the event. Clicking a tag will show a list of events with the same tag attached. The little x next to each tag allows the users to remove the tag from the event, whilst the + button allows them to assign a tag. For the latter two options to be visible, the users should have tagging permissions.

Date indicates the date of detection, set by the user that created the event. (Not to be confused with the creation date of the event as provided in the *First recorded change*).

Threat Level indicates the assigned threat level of the event.

Analysis provides the current status of the analysis.

Distribution shows the distribution rules applied to this event, controlling whether only the authoring organization can see (Your organization only) it, or everyone on the instance (This community only). The two remaining settings allow the event to be propagated to organizations on remote connected instances.

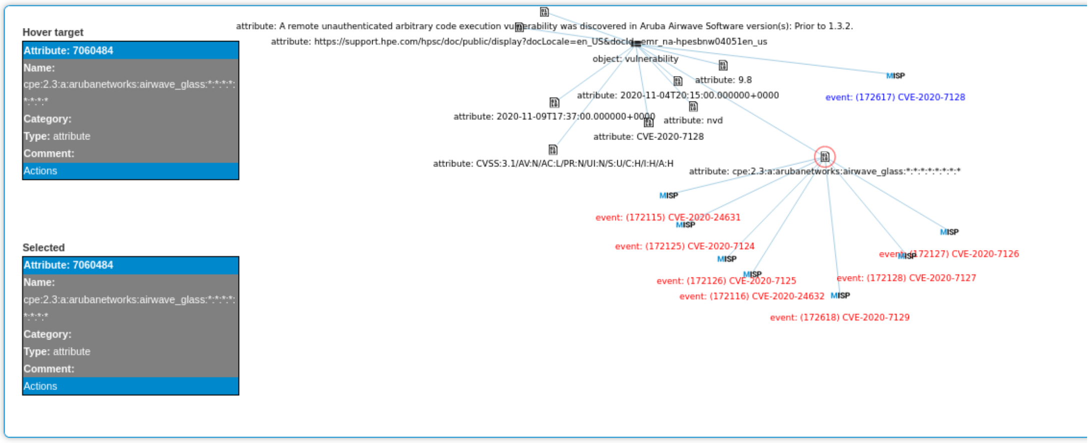


vulnerability	
id	CVE-2020-5667
vulnerable_configuration	cpe:/a:misc:studyplus_studyplus
published	2020-11-05T09:43:32.000000+0000
modified	2020-11-05T09:43:32.000000+0000
cvss-score	4
cvss-string	CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
references	https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5667
references	https://jvn.jp/en/jp/JVN00414047/index.html
description	Studyplus App provided by Studyplus Inc. uses a hard-coded API k[...]
credit	jvn

vuldb-vulnerability	
id	CVE-2020-5667
summary	Studyplus App API Key hard-coded credentials
published	2020-11-06T10:18:00.000000+0000
vuldb-link	https://vuldb.com/?id.164439
zeroday-price	$0-$5k
current-price	$0-$5k
remediation	Not Defined
credit	vuldb
cvss-score	6.3
cvss-string-VDB	AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L
exploitability	Not Defined

vulnerability	
id	CVE-2020-5667
published	2020-11-06T03:15:00.000000+0000
modified	2020-11-06T12:41:00.000000+0000
references	https://jvn.jp/en/jp/JVN00414047/index.html
description	Studyplus App for Android v6.3.7 and earlier and Studyplus App f[...]
credit	nvd

Figure 3.8: MISP Event timeline


Hover target

Attribute: 7060484
Name: cpe:2.3:a:arubanetworks:airwave_glass:*:*:*:*:*:*:*:*
Category:
Type: attribute
Comment:
Actions

Selected

Attribute: 7060484
Name: cpe:2.3:a:arubanetworks:airwave_glass:*:*:*:*:*:*:*:*
Category:
Type: attribute
Comment:
Actions

attribute: A remote unauthenticated arbitrary code execution vulnerability was discovered in Aruba Airwave Software version(s): Prior to 1.3.2.
attribute: https://support.hpe.com/hpsc/doc/public/display?docLocale=en_US&docId=emr_na-hpesbnw04051en_us
object: vulnerability
attribute: 9.8
MISP
event: (172617) CVE-2020-7128
attribute: 2020-11-04T20:15:00.000000+0000
attribute: 2020-11-09T17:37:00.000000+0000
attribute: nvd
attribute: CVE-2020-7128
attribute: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
attribute: cpe:2.3:a:arubanetworks:airwave_glass:*:*:*:*:*:*:*:*
MISP
event: (172115) CVE-2020-24631
MISP
event: (172125) CVE-2020-7124
MISP
MISP
event: (172127) CVE-2020-7126
MISP
event: (172126) CVE-2020-7125
event: (172128) CVE-2020-7127
event: (172116) CVE-2020-24632
MISP
event: (172618) CVE-2020-7129

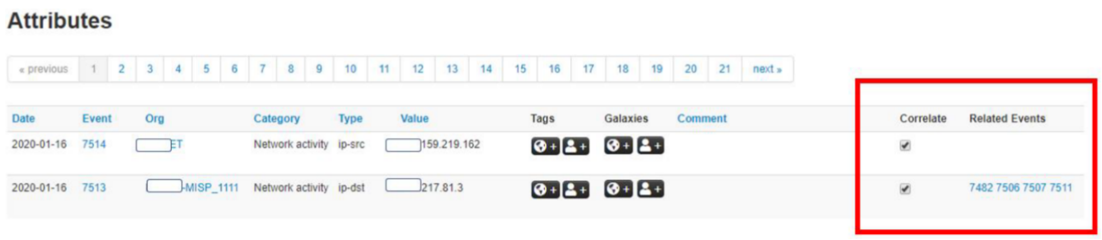
Figure 3.9: MISP Event Correlation graph

Info gives a short description of the event itself.

Published gives information on whether the event has been published or not. Publishing allows the attributes of the event to be used for all eligible exports and it notifies users that have subscribed to the event alerts. Also, a publication initiates a push to all eligible instances.

Additionally, there is a list of related events. This list refers to the relations that are shown on the right side of the general event information. Events can be related by having one or more attributes that are exact matches. For example, if two events both contain a source IP of 1.1.1.1, then they are related. The list of events that are related to the one at hand, are listed under *Related Events*, as links (titled by the event's date and info) to the events themselves.

Finally, as shown in Figure 3.7, through the Event View, users are also provided with the *Event timeline* (as shown in Figure 3.8) and the *Correlation graph* (as shown in Figure 3.9); two useful functionalities for the purposes of the CTI reviewing procedure.



Attributes

« previous | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | next »

Date	Event	Org	Category	Type	Value	Tags	Galaxies	Comment	Correlate	Related Events
2020-01-16	7514	[redacted]ET	Network activity	ip-src	[redacted]159.219.162				✓	
2020-01-16	7513	[redacted]-MISP_1111	Network activity	ip-dst	[redacted]217.81.3				✓	7482 7506 7507 7511

Figure 3.10: MISP Correlation Engine

3.5.3.4 Correlation Engine

The correlation engine of MISP encompasses all the correlations between attributes and more advanced correlations like fuzzy hashing correlation (e.g., ssdeep) or CIDR block matching. Correlations can be both enabled or disabled, for each event per attribute. The *value* field of the attribute is the main payload of the attributes, which is described by the category and type columns, and it is used by the correlation engine to find relations between events.

Chapter 4

The ASPIS System

In this chapter we present **ASPIS**, a full-stack system that was developed, aiming to provide a complete proactive methodology for the CTI gathering and sharing. Particularly, **ASPIS** gathers CTI from different identified sources, which usually store common information that needs to be parsed and consolidated. To this end, **ASPIS** parses each identified CTI source, merging all parsed artifacts into clusters. The resulting clustered information is then stored and managed through the utilization of MISP (presented in Chapter 3). Furthermore, **ASPIS** uses MISP to inter-correlate the CTI clusters, to find possible matches between the stored artifacts. The system is able to present all gathered information in human-readable formatting, through the MISP web-application, which enables users to further edit, analyze, enrich, and share the stored CTI. To fully accommodate MISP to our needs, we customize and extend its *taxonomies*, defining new *objects*, and also improve its *correlation engine* storage requirements, altering the corresponding database's structures.

In the rest of this chapter we present the architecture of **ASPIS** and extendedly describe all of the system's functionalities, along with the utilization and customizations of MISP.

4.1 System Overview

The **ASPIS** system provides a CTI life-cycle-complete solution. To this end, we first need to identify various CTI sources (e.g., NVD [33], JVN [24], KB-Cert [26], VulDB [61], and Exploit-DB [12]). Such sources may store vulnerability and exploit data and contain analyzed CTI, in the form of vulnerability and exploit reports. These reports embody a plethora of useful and actionable intelligence about the vulnerabilities and exploits, such as a description of the vulnerability at hand, an exploit

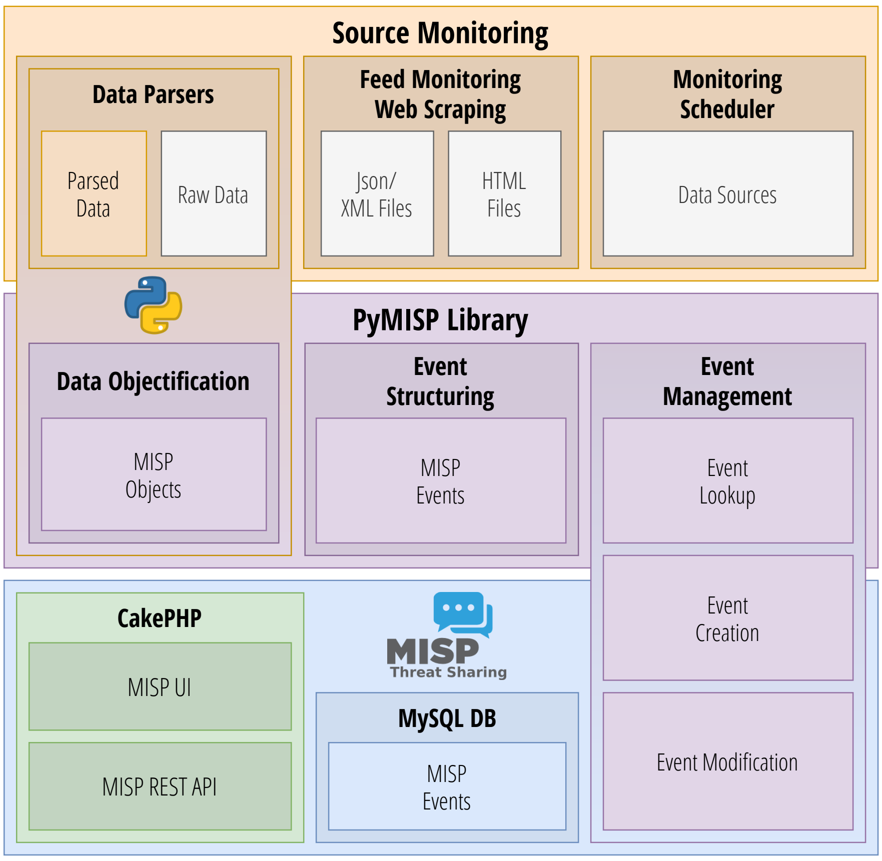


Figure 4.1: **ASPIS** system architecture

proof-of-concept, a list of the affected products' configurations (CPEs), metrics that provide an impact factor for the affected product (CVSS), publication and modification dates, references to similar reports, and a unique identifier that has been assigned to the vulnerability at hand (CVE ID). However, while the aforementioned sources often provide reports about the same unique CVE ID, these reports tend to differ. This happens due to the dynamicity of available information at the time of the analysis. Thus, analyses that occurred at a different time, may provide different metrics in the final reports. To overcome this issue, we gather all publicly available reports from the monitored sources, we parse them, one-by-one, to extract the CTI provided, and then we store the parsed CTI, in a clustered manner. The selected platform for storing and disseminating the gathered CTI is MISP. These clusters are called *events* in the MISP platform and the clustering of the reports occurs at the event management phase. MISP provides the information stored in its database, in both human and machine-readable formats, and allows users to access it either through a GUI or via a REST API. Finally, MISP has implemented various tools, available in the GUI, that enable UI users to review CTI gathered and eliminate false positives or comment on the artifacts, and further analyze and enrich CTI through correlation processes. Figure 4.1 presents an abstracted view of the **ASPIS**

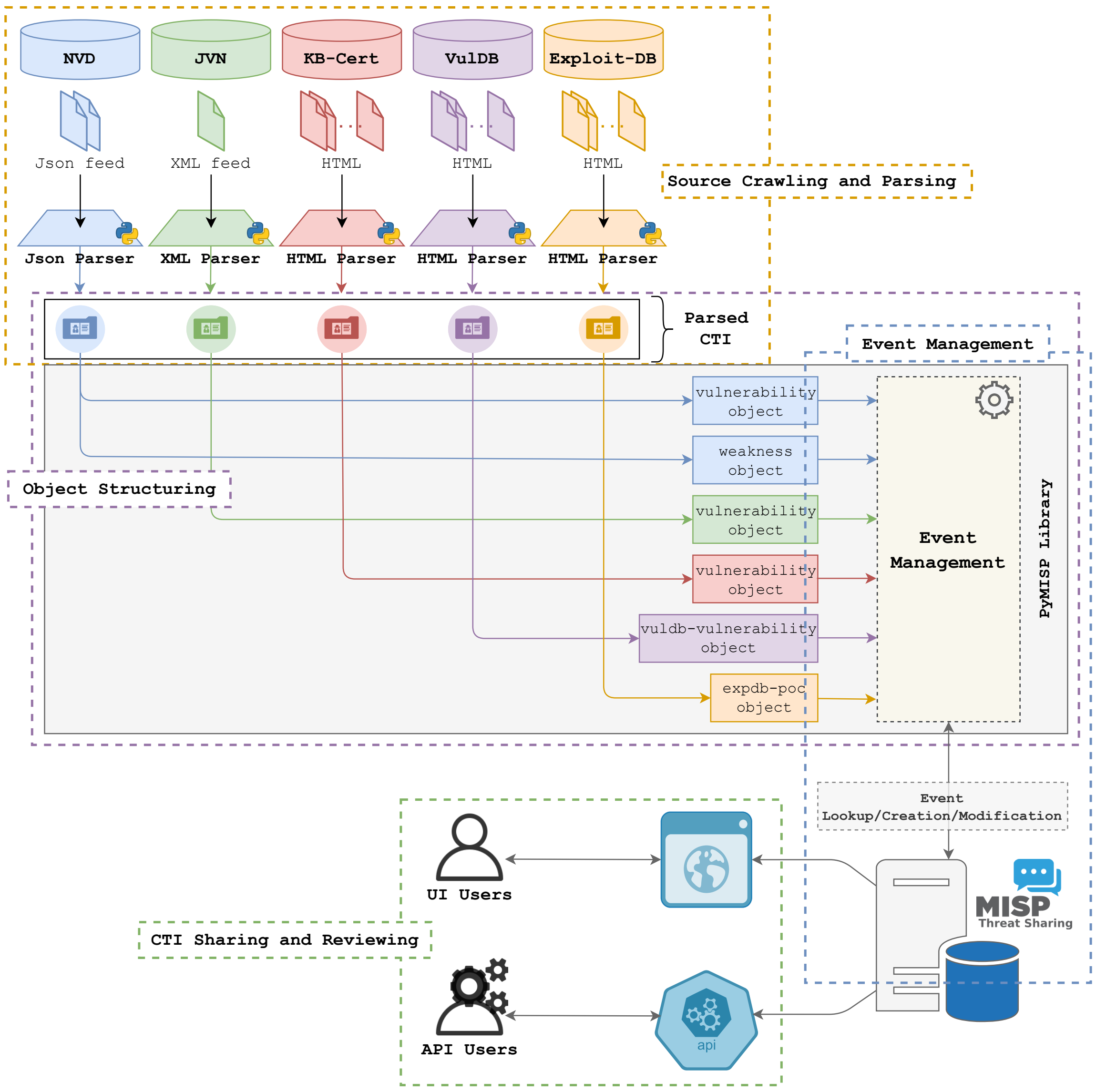


Figure 4.2: **ASPIS** system dataflow example

architecture.

In the following sections, we present the technology stack and the applied tools that are used in **ASPIS** to support the implementation of MISP (Section 4.2). Next, we present the extensions and customizations of MISP in **ASPIS** (Section 4.3). Then, we provide a summary for each one of the monitored sources (Section 4.4). In Figure 4.2, we provide an example of the dataflow of **ASPIS**. Each phase of the system dataflow; namely the *Source Crawling and Parsing*, *Object Structuring*, *Event Management*, and *CTI Sharing and Reviewing* phases are described in detail in Sections 4.5, 4.6, 4.7, 4.8, respectively. Finally, we provide an overview of the usage of the **ASPIS** system (Section 4.9).

4.2 Technology Stack and Applied Tools

As mentioned in Section 3.4, we currently work with MISP v2.4.132, which is the latest version of MISP published. MISP is built upon programming frameworks like CakePHP and PHP for the UI, and MariaDB/MySQL for the data storage. **ASPIS** uses the following technologies to support the implementation of MISP v2.4.132.

- **CakePHP 2.10.19.** CakePHP is an open-source web framework, which follows the Model-View-Controller (MVC) approach and is written in PHP. MISP is built upon CakePHP v2.10.19, which supports PHP 7.0+, and it also makes use of the CakeResque plugin of CakePHP, which enables the creation of background jobs that can be processed offline.
- **PHP 7.2.** **ASPIS** uses PHP version 7.2; CakePHP employs PHP to build the MISP platform, and connect the web application with the data storage.
- **MariaDB 10.1.** MariaDB is a variation of MySQL and it acts as the data storage of MISP in **ASPIS**.

Additionally, to support the gathering and sharing of CTI, **ASPIS** uses the following technologies:

- **PyMISP 2.4.132.** PyMISP is a Python programming language library that provides access to the MISP platform via its REST API. It enables users to fetch, add, update, delete and search events/attributes or samples. Furthermore, it facilitates the creation of scripts that enable other components to easily interact with MISP. Particularly, through the utilization of the PyMISP library, we create and update events, each time we gather new CTI from our monitored sources. PyMISP 2.4.132 is supported by Python 3.6+ versions.
- **Python 3.6.** **ASPIS** uses Python 3.6 to automate the process of collecting, storing, and managing CTI from the monitored sources. Through the implemented python scripts, **ASPIS** is able to gather and parse CTI from different sources (like NVD, JVN, VulDB, KB-Cert, and Exploit-DB). All gathered CTI is then structured into MISP objects (in JSON format), through the PyMISP library, and finally inserted into MISP to become available for further processing by **ASPIS**.

4.3 MISP in ASPIS

As stated in Chapter 3, MISP is the platform of choice for the CTI management and sharing of **ASPIS**. However, to fully accommodate MISP to our needs, we

Attribute	Type	Description
`created`	`datetime`	First time when the vulnerabililty was discovered.
`credit`	`text`	Who reported/found the vulnerability such as an organization, person or nickname.
`cvss-score`	`float`	Score of the CVSSv3.
`cvss-string`	`text`	String of the CVSSv3.
`description`	`text`	Description of the vulnerability.
`id`	`vulnerability`	Vulnerability ID (generally CVE, but not necessarely).
`modified`	`datetime`	Last modification date.
`published`	`datetime`	Initial publication date.k
`references`	`link`	External references.
`state`	`text`	State of the vulnerability. A vulnerability can have multiple states depending on the current actions performed. [‘Published’, ‘Embargo’, ‘Reviewed’, ‘Vulnerability ID Assigned’, ‘Reported’, ‘Fixed’].
`summary`	`text`	Summary of the vulnerability.
`vulnerable_configuration`	`text`	The vulnerable configuration is described in the CPE format.

Table 4.1: The *vulnerability* MISP Object

make use of the platform's provided tools to define custom objects that are able to fully encompass the CTI artifacts of the monitored sources (as presented in Section 4.3.1). Furthermore, in this section we discuss in detail the MISP correlation engine (Section 4.3.2) and propose an alternative correlation engine (in Section 4.3.3), which improves the storage requirements.

4.3.1 MISP Objects Employed

To best describe the CTI artifacts parsed by **ASPIS**, we need to store them in MISP in the most suitable objects; namely, the *vulnerability*, and *weakness* objects. Additionally, MISP provides a method for creating custom MISP objects, which we use to create two custom objects for **ASPIS**; namely, the *vuldb-vulnerability* and *expdb-poc* objects, which enrich the attributes of *vulnerability* and *exploit-poc* objects respectively. The aforementioned objects are described in the following sections in detail.

4.3.1.1 The Vulnerability Object

Vulnerability objects describe CVEs, which refers to published, unpublished, or under review vulnerabilities for software, equipment or hardware. An overview of the *vulnerability* object is provided in Table 4.1. In our case, the vulnerability object is used for three of the monitored sources; namely, NVD, JVN and KB-Cert, since it is considered as the most suitable MISP object to describe the parsed CTI.

Attribute	Type	Description
`description`	`text`	Description of the weakness.
`id`	`text`	Weakness ID (generally CWE).
`name`	`text`	Name of the weakness.
`status`	`text`	Status of the weakness. [‘Incomplete’, ‘Deprecated’, ‘Draft’, ‘Usable’].
`weakness-abs`	`text`	Abstraction of the weakness. [‘Class’, ‘Base’, ‘Variant’].

Table 4.2: The *weakness* MISP Object

4.3.1.2 The Weakness Object

Weakness objects describe CWEs which refer to usable, incomplete, draft or deprecated weaknesses for software, equipment of hardware. CWE serves as a common language, a measuring technique for security tools, and as a baseline for weakness identification, mitigation, and prevention efforts. An overview of the *weakness* object is provided in Table 4.2. Finally, although CWE can be obtained from both NVD and JVN, we use it only for CTI parsed from NVD, since it is considered to be the most credible source, to avoid storing duplicate objects within the same event.

4.3.1.3 The VulDB-Vulnerability Object

The *vuldb-vulnerability* object is an enriched version of the *vulnerability* object, describing a CVE which can describe published, unpublished, under review or embargo vulnerability for software, equipments or hardware. Additionally, it provides all proper attributes to store supplementary CTI parsed, such as the price estimations, CVSS strings from external sources, exploitability/remediation status, and so on. A more detailed overview of the *vuldb-vulnerability* object is presented in Table 4.3. In our case, this object is used to store CTI parsed from VulDB.

4.3.1.4 The ExpDB-PoC Object

The *expdb-poc* object is a differentiated version of the *exploit-poc* object, describing a proof-of-concept or exploit of a vulnerability. This object has often a relationship with a vulnerability object. In Table 4.4 we present the additional attribute of the *expdb-poc* object, which differentiates it from *exploit-poc*. In our case, Exploit-DB does not provide CTI that regard CPE, and thus we do not use the *vulnerable_configuration* attribute. Furthermore, instead of downloading and storing all exploit proof-of-concepts, we point towards the link of the Exploit-DB raw code PoC, through *references*, and hence the *poc* attribute is not used by **ASPIS**. Finally, since all MISP Objects that were used in our system, have a credit field, we make use of it for storing the source of the parsed CTI.

Attribute	Type	Description
`id`	`text`	Vulnerability ID (generally CVE, but not necessarely).
`summary`	`text`	Summary of the vulnerability.
`description`	`text`	Description of the vulnerability.
`published`	`text`	Initial publication date.
`status`	`text`	Status of the vulnerability aproval. (Typical VulDB verification of a reported vulnerability)
`cvss-score`	`float`	Score of the CVSSv3. This is a Meta score, calculated by VulDB. The calculation method is described in the attribute's comment. [e.g., CVSSof(vuldb+nvd)/2].
`cvss-string-VDB`	`text`	String of the CVSSv3 of VulDB security analysts.
`cvss-string-NVD`	`text`	String of the CVSSv3 of NVD security analysts.
`cvss-string-Vend`	`text`	String of the CVSSv3 of the Vendor's security analysts.
`cvss-string-Res`	`text`	String of the CVSSv3 of a researcher who analyzed it for VulDB.
`vuldb-link`	`link`	The link to the VulDB report.
`zeroday-price`	`text`	Vuldb analysts are monitoring exploit markets and are in contact with vulnerability brokers. The range indicates the observed or calculated exploit price to be seen on exploit markets. A good indicator to understand the monetary effort required for and the popularity of an attack. This is the price range of the exploit for the 0day exploitation of the vulnerability. (e.g., \$0-\$5k)
`current-price`	`text`	Similarly to the *zeroday-price*. This is the current estimation of the price range for the exploitation of the vulnerability.
`exploitability`	`text`	The status of an exploit for the vulnerability. (e.g., 'Functional', 'Proof-of-Concept')
`remediation`	`text`	A status of whether there is a remediation for this vulnerability. (e.g., 'Workaround (Alternative)', 'Official Fix (Upgrade)')
`credit`	`text`	Who reported/found the vulnerability such as an organisation, person or nickname.

Table 4.3: The *vuldb-vulnerability* MISP Object

Attribute	Type	Description
`description`	`text`	Description of the exploit - proof of concept.
`poc`	`attachment`	Proof of Concept or exploit (as a script, binary or described process).
`references`	`link`	External references.
`vulnerable_configuration`	`text`	The vulnerable configuration described in CPE format where the exploit/proof of concept is valid.
`author`	`text`	Author of the exploit - proof of concept.
`credit`	`text`	Source of the exploit - proof of concept. (e.g., expdb)

Table 4.4: The *expdb-poc* MISP Object

Specifically, the *credit* attribute of the aforementioned objects used in our case, may contain one of the following values: *nvd*, *jvn*, *kb-cert*, *vuldb*, *expdb*. Any reference to an actor that has been involved in discovering, conducting or reporting a vulnerability/exploit is provided in the *description* attribute of *vulnerability* and *vuldb-vulnerability*, and in the *author* attribute of *expdb-poc*.

4.3.2 MISP Correlations in ASPIS

As described in Section 3.5.3.4, MISP provides a correlation mechanism, which is able to generate correlations between different MISP Events, with regard to their encompassed attributes. Specifically, after each event creation, the correlation engine of MISP, scans through the database for exact matches of the event's correlatable attributes' `value`, within the rest of the events, with regard to the attributes' `category` and `type`. By default, the MISP correlation engine generates correlations solely between different events. For each exact match, MISP proceeds to store two correlation entries in the database; one that points from the recently inserted event, to the previously stored and one that points to the recently inserted event, from the previously stored, through their unique IDs (ϵ_{j}, ϵ_{i}), along with their corresponding attributes' unique IDs (α_{j}, α_{i}) and the matched value v. The MISP correlation engine utilizes the *correlations* relation (presented in Table 3.4) to store the generated correlations' information. Particularly, from *correlations*, it utilizes `1_event_id` and `event_id` to store the corresponding event IDs, `1_attribute_id` and `attribute_id` to store the corresponding attribute IDs, and `value` to store the matched value. So, for example, to store a correlation between ϵ_{j} and ϵ_{i}, MISP correlation engine will store two rows in the *correlations* table, like:

`1_event_id`	`1_attribute_id`	`event_id`	`attribute_id`	`value`
ϵ_{j}	α_{j}	ϵ_{i}	α_{i}	v
ϵ_{i}	α_{i}	ϵ_{j}	α_{j}	v

where, in the first row, ϵ_{j} points to ϵ_{i}, through their corresponding α_{j} and α_{i} attributes' common `value` v, and vice-versa in the second row.

ASPIS uses that mechanism, to further enrich the stored CTI, with correlations that concern the affected products, through the `vulnerable_configuration` attribute of the events' objects. Thus, the system is able to realize which vulnerabilities and exploits affect a specific product. With this information at hand, security experts gain a better overview of the products' threat landscape.

Currently, the way that correlations' information is stored in the MISP database is very space demanding. To illustrate an example, in a setting that stored ∼83K events, that encompassed ∼5M attributes, the correlations' information consists of ∼1.2B records that occupied over 315GB of hard disk storage. To counter that, the correlations' table of the MISP database, has to be restructured, by also implementing that alteration in the MISP correlation engine's internal code. The proposed solution is extendedly presented in Section 4.3.3.

`1_event_id`	`1_attribute_id`	`event_id`	`attribute_id`	`value`
ϵ_1	α_1	ϵ_2	α_2	v_1
ϵ_1	α_1	ϵ_3	α_3	v_1
ϵ_1	β_1	ϵ_2	β_2	v_2
ϵ_2	α_2	ϵ_1	α_1	v_1
ϵ_2	α_2	ϵ_3	α_3	v_1
ϵ_2	β_2	ϵ_1	β_1	v_2
ϵ_3	α_3	ϵ_1	α_1	v_1
ϵ_3	α_3	ϵ_2	α_2	v_1

Table 4.5: The resulting *correlations* table of Example 1

`1_event_id`	`1_attribute_id`	`event_id`	`attribute_id`	`value`
ϵ_1	α_1	ϵ_2	α_2	v_1
ϵ_1	β_1	ϵ_2	α_2	v_1
ϵ_2	α_2	ϵ_1	α_1	v_1
ϵ_2	α_2	ϵ_1	β_1	v_1

Table 4.6: The resulting *correlations* table of Example 2

4.3.2.1 MISP Correlation Engine Functionality

To describe the MISP correlation engine functionality, we consider the following examples.

Example 1. *Let us consider 3 events:*

event ϵ_1, with attributes: α_1, β_1,

event ϵ_2, with attributes: α_2, β_2 and

event ϵ_3, with attribute: α_3.

Let us also assume that the attributes α_1, α_2, and α_3 store the value v_1, while β_1 and β_2 store the value v_2, in field `value` *respectively. When the MISP correlation engine considers the events ϵ_1, ϵ_2, and ϵ_3, it produces the correlation table presented in Table 4.5.*

Example 2. *Let us consider 2 events:*

event ϵ_1, with attributes: α_1, β_1 and

event ϵ_2, with attribute: α_2,

Let us also assume that the attributes α_1, β_1, and α_2 store the value v_1 in field `value`*. When the MISP correlation engine considers the events ϵ_1 and ϵ_2, it produces the correlation table presented in Table 4.6.*

Both Table 4.5 and Table 4.6 store redundant information. To provide a better view, in Table 4.7 and Table 4.8 we highlight the redundant information of Table 4.5 and Table 4.6 respectively. It is important to note that the numbers displayed at

	1_event_id	1_attribute_id	event_id	attribute_id	value
1	ϵ_1	α_1	ϵ_2	α_2	v_1
2	ϵ_1	α_1	ϵ_3	α_3	v_1
3	ϵ_1	β_1	ϵ_2	β_2	v_2
4	ϵ_2	α_2	ϵ_1	α_1	v_1
5	ϵ_2	α_2	ϵ_3	α_3	v_1
6	ϵ_2	β_2	ϵ_1	β_1	v_2
7	ϵ_3	α_3	ϵ_1	α_1	v_1
8	ϵ_3	α_3	ϵ_2	α_2	v_1

Table 4.7: The resulting *correlations* table of Example 1, with highlighted redundancy

	1_event_id	1_attribute_id	event_id	attribute_id	value
1	ϵ_1	α_1	ϵ_2	α_2	v_1
2	ϵ_1	β_1	ϵ_2	α_2	v_1
3	ϵ_2	α_2	ϵ_1	α_1	v_1
4	ϵ_2	α_2	ϵ_1	β_1	v_1

Table 4.8: The resulting *correlations* table of Example 2, with highlighted redundancy

the beginning of each row of Table 4.7 and Table 4.8 are only used for reference and they are not stored in the tables.

The redundancy of the generated correlation tables of Example 1 and Example 2 is further discussed in the following observations.

Observation 1. Reviewing Table 4.7, we realize that half of the rows stored (highlighted in dark gray) are redundant, since the same information is already captured by other rows of the table. For instance, the first and the fourth row represent the same information, because they both define the correlation between ϵ_1 and ϵ_2 through their attributes α_1 and α_2, on their common value v_1. Furthermore, we observe that cells of information (highlighted in light gray) already exist in the *correlations* table as well. For example, in the first row of Table 4.7, we observe that ϵ_1 is correlated with ϵ_2 on their corresponding attributes α_1 and α_2 that have the value v_1. Thus, the database already contains the knowledge that α_1 of ϵ_1 is correlated with another attribute, on the value v_1. So, the highlighted cells of the second row of Table 4.7 can be inferred from the information of the first row, since `1_event_id`, `1_attribute_id`, and `value` columns' data of both rows, are the same (ϵ_1, α_1, and v_1). Similarly, in the fifth row, `1_event_id`, `1_attribute_id`, and `value` columns' data (ϵ_2, α_2, and v_1) are already stored in `event_id`, `attribute_id`, and `value` of the first row. Additionally, in the fifth row, `event_id`, `attribute_id`, and `value` columns' data (ϵ_3, α_3, and v_1) are the same with the corresponding columns' data of the second row.

Observation 2. In Example 2, we notice that although attributes α_1 and β_1 of event ϵ_1 contain the same value, they are not intercorrelated by the MISP correlation engine, and hence no information regarding such correlations is stored in Table 4.6. As described in Section 4.3.2, the correlation engine of MISP only produces correlations between different events. Finally, as described in Observation 1, in Table 4.8 half of the rows stored (highlighted in dark gray) are redundant and the cells of information highlighted in light gray can be inferred.

Observing the tables generated by the MISP correlation engine for Example 1 (Table 4.5) and Example 2 (Table 4.6), we infer the following lemma.

Lemma 1. Let $\mathcal{A}$ be the set of stored attributes and $\mathcal{V}$ be the set of common values, so that $\forall_{(\mathrm{i}\neq\mathrm{j})}\ \alpha_\mathrm{i},\ \alpha_\mathrm{j} \in \mathcal{A}$, with $\alpha_\mathrm{i}.v = \alpha_\mathrm{j}.v$, then $v \in \mathcal{V}$. Additionally, let $AwV(v)$ be the number of attributes having value v. Then, the size of the *correlations* table (S_{CORR}), computed by the MISP correlation engine, is bounded by the following expression:

$$S_{\mathrm{CORR}} \leq \Sigma_{v\in\mathcal{V}}\big(AwV(v)\cdot(AwV(v)-1)\big).$$

Example 3. *Let us apply Lemma 1 to the events of Example 1. We have a set $\mathcal{V}$ of two common values; v_1 and v_2. So, from Lemma 1, S_{CORR} would be:*

$S_{CORR} \leq \Sigma_{v\in\mathcal{V}}\big(AwV(v)\cdot(AwV(v)-1)\big)$, *where:* $AwV(v_1) = 3$ *and* $AwV(v_2) = 2$. *Thus,*

$$S_{CORR} \leq \Sigma_{v\in\mathcal{V}}\big(AwV(v)\cdot(AwV(v)-1)\big) =$$
$$= AwV(v_1)\cdot(AwV(v_1)-1) + AwV(v_2)\cdot(AwV(v_2)-1) =$$
$$= 3\cdot(3-1)+2\cdot(2-1) = 3\cdot2+2\cdot1 = 6+2 = 8 \text{ rows (as presented in Table 4.5).}$$

As realized, Example 3 is able to validate the equality of the calculated sum with the S_{CORR} provided by Lemma 1. However, that sum consists an upper-bound of the *correlations* table size. This happens because it does not take under consideration the attributes that are not intercorrelated, due to the fact that MISP correlations occur solely between different events. To better understand that, we provide the following example.

Example 4. *Let us apply Lemma 1 to the events of Example 2. We have a set $\mathcal{V}$ of one common value; v_1. So, from Lemma 1, S_{CORR} would be:*

$S_{CORR} \leq \Sigma_{v\in\mathcal{V}}\big(AwV(v)\cdot(AwV(v)-1)\big)$, *where:* $AwV(v_1) = 3$.

Thus,

$$S_{CORR} \leq \Sigma_{v\in\mathcal{V}}\big(AwV(v)\cdot(AwV(v)-1)\big) =$$

$= AwV(v_1) \cdot (AwV(v_1) - 1) = 3 \cdot (3-1) = 3 \cdot 2 = 6$ *rows (which is larger than the rows presented in Table 4.6, that are actually 4).*

Therefore, to calculate the exact value of S_{CORR}, correlations resulting from different attributes that coexist within one event and contain the same value, should be excluded from the calculated sum. That leads to the following lemma.

Lemma 2. Let $\mathcal{A}$ be the set of stored attributes and $\mathcal{V}$ be the set of common values, so that $\forall_{(\mathrm{i}\neq\mathrm{j})}\ \alpha_{\mathrm{i}}, \alpha_{\mathrm{j}} \in \mathcal{A}$, with $\alpha_{\mathrm{i}}.v = \alpha_{\mathrm{j}}.v$, then $v \in \mathcal{V}$. Additionally, let $AwV(v)$ be the number of attributes having value v. Next, let $\mathcal{E}$ be the set of stored events, and $\epsilon_{\mathrm{n}}AwV(v)$ be the number of attributes of event ϵ_{n}, that have the value v, where $\epsilon_{\mathrm{n}} \in \mathcal{E}$ and $\mathrm{n} \leq |\mathcal{E}|$, $\mathrm{n} \in \mathbb{Z}^{+}$. Then, the size of the *correlations* table (S_{CORR}), computed by the MISP correlation engine, would be:

$$S_{\mathrm{CORR}} =$$
$$= \Sigma_{v\in\mathcal{V}}\big(AwV(v) \cdot (AwV(v) - 1)\big) - \Sigma_{n=1}^{|\mathcal{E}|}\Big(\Sigma_{v\in\mathcal{V}}\big(\epsilon_{\mathrm{n}}AwV(v) \cdot (\epsilon_{\mathrm{n}}AwV(v) - 1)\big)\Big).$$

Example 5. *Let us apply Lemma 1 to the events of Example 2, we have a set $\mathcal{V}$ of one common value; v_1. Additionally, there are 2 attributes that exist in the event ϵ_1; α_1 and β_1, both containing value v_1. So, from Lemma 2, S_{CORR} would be:*

$$S_{CORR} =$$
$$= \Sigma_{v\in\mathcal{V}}\big(AwV(v) \cdot (AwV(v) - 1)\big) - \Sigma_{n=1}^{|\mathcal{E}|}\Big(\Sigma_{v\in\mathcal{V}}\big(\epsilon_{n}AwV(v) \cdot (\epsilon_{n}AwV(v) - 1)\big)\Big),$$

where: $AwV(v_1) = 3$, $|\mathcal{E}| = 2$, $\epsilon_1 AwV(v_1) = 2$, and $\epsilon_2 AwV(v_1) = 0$.

Thus,

$$\Sigma_{v\in\mathcal{V}}\big(AwV(v) \cdot (AwV(v) - 1)\big) - \Sigma_{n=1}^{2}\Big(\Sigma_{v\in\mathcal{V}}\big(\epsilon_{n}AwV(v) \cdot (\epsilon_{n}AwV(v) - 1)\big)\Big) =$$
$$= AwV(v_1) \cdot (AwV(v_1) - 1) - \epsilon_1 AwV(v_1) \cdot (\epsilon_1 AwV(v_1) - 1) =$$
$$= 3 \cdot (3-1) - 2 \cdot (2-1) = 3 \cdot 2 - 2 \cdot 1 = 6 - 2 = 4$$ *rows (as presented in Table 4.6).*

As realized, Example 5 is able to validate the equality of the calculated sum with S_{CORR} provided by Lemma 2. The same analysis as above can also be conducted for Example 1, to validate the correlations table size calculation expressed in Lemma 2. However that would be redundant, since Example 1 does not contain any duplicate attribute values within the same event, and that would lead to the S_{CORR} calculation of Lemma 1, as analyzed in Example 3.

4.3.3 MISP Correlation Engine Alteration

In this section we propose an alternative MISP correlation engine, to eliminate the redundancy indicated by Observation 1 and Observation 2. Then, we check the validity of the proposed alteration, based on the standard BCNF decomposition algorithm.

1_event_id	1_attribute_id	event_id	attribute_id	value
ϵ_1	α_1	ϵ_2	α_2	v_1
ϵ_1	α_1	ϵ_3	α_3	v_1
ϵ_1	β_1	ϵ_2	β_2	v_2
ϵ_2	α_2	ϵ_1	α_1	v_1
ϵ_2	α_2	ϵ_3	α_3	v_1
ϵ_2	β_2	ϵ_1	β_1	v_2
ϵ_3	α_3	ϵ_1	α_1	v_1
ϵ_3	α_3	ϵ_2	α_2	v_1

$\leftrightarrow$

event_id	attribute_id	value
ϵ_1	α_1	v_1
ϵ_1	β_1	v_2
ϵ_2	α_2	v_1
ϵ_2	β_2	v_2
ϵ_3	α_3	v_1

Figure 4.3: The alternate equivalent (right) to the resulting *correlations* table of Example 1 (Table 4.7) (left).

1_event_id	1_attribute_id	event_id	attribute_id	value
ϵ_1	α_1	ϵ_2	α_2	v_1
ϵ_1	β_1	ϵ_2	α_2	v_1
ϵ_2	α_2	ϵ_1	α_1	v_1
ϵ_2	α_2	ϵ_1	β_1	v_1

$\leftrightarrow$

event_id	attribute_id	value
ϵ_1	α_1	v_1
ϵ_1	β_1	v_1
ϵ_2	α_2	v_1

Figure 4.4: The alternate equivalent (right) to the resulting *correlations* table of Example 2 (Table 4.8) (left).

4.3.3.1 Proposal for an Alternative MISP Correlation Engine

Based on the redundancy observed in Observation 1 and Observation 2, we propose to alter the *correlations* table in such a manner, that it only contains information which is not redundant.

Specifically, in Figure 4.3 we present how the *correlations* table of Example 1 (Table 4.7) can be altered to uniquely store the correlations' knowledge. Specifically, in both tables of Figure 4.3 we uniquely highlight each cell of information (that is not considered as redundant by Observations 1 and 2), to illustrate how the alternate equivalent *correlations* table derives from Table 4.7. Similarly, in Figure 4.4 we present how the *correlations* table of Example 2 (Table 4.8) can be altered to remove redundancy.

Additionally, we notice that Table 4.7 and Table 4.8 can also be produced by their alternate equivalent if we perform a join operation ($\bowtie$) over their common `values`, on different `event_ids`. Particularly, to reproduce the default *correlations* table from its alternate equivalent (*correlationsNF*):

Let *correlationsNF*`(event_id, attribute_id, value)`.

Then,

correlations=

$$=\rho(\texttt{1\_event\_id, 1\_attribute\_id, value})(\mathit{correlationsNF}) \bowtie_{\text{1\_event\_id} \neq \text{event\_id}} \mathit{correlationsNF}.$$

From the alternate equivalent tables presented in Figure 4.3 and Figure 4.4, we infer the following lemma.

Lemma 3. Let $\mathcal{A}$ be the set of stored attributes and $\mathcal{V}$ be the set of common values, so that $\forall_{(\mathrm{i}\neq\mathrm{j})}\ \alpha_\mathrm{i},\ \alpha_\mathrm{j} \in \mathcal{A}$, with $\alpha_\mathrm{i}.v = \alpha_\mathrm{j}.v$, then $v \in \mathcal{V}$. Additionally, let $AwV(v)$ be the number of attributes having value v. Then, the size (S_{CORRNF}) of the proposed alternate *correlations* table would be:

$S_{\mathrm{CORRNF}} = \Sigma_{v\in\mathcal{V}}\big(AwV(v)\big)$.

Example 6. *Let us apply Lemma 3 to the events of Example 1. We have a set* $\mathcal{V}$ *of two common values;* v_1 *and* v_2*. So, from Lemma 3,* S_{CORRNF} *would be:*

$S_{CORRNF} = \Sigma_{v\in\mathcal{V}}\big(AwV(v)\big)$, *where:* $AwV(v_1) = 3$ *and* $AwV(v_2) = 2$.

Thus,

$\Sigma_{v\in\mathcal{V}}\big(AwV(v)\big) = AwV(v_1) + AwV(v_2) = 3 + 2 = 5$ *rows (as presented in Figure 4.3).*

As realized, Example 6 is able to validate the equality of the calculated sum with S_{CORRNF} provided by Lemma 3. Moreover, as we will present in Example 7, the S_{CORRNF} of Lemma 3 is unaffected by attributes that may contain the same value, coexisting in the same event, and hence it consists a precise calculation of the *correlations* table size, for the proposed alteration.

Example 7. *Let us apply Lemma 3 to the events of Example 2. We have a set* $\mathcal{V}$ *of one common value;* v_1*. So, from Lemma 3,* S_{CORRNF} *would be:*

$S_{CORRNF} = \Sigma_{v\in\mathcal{V}}\big(AwV(v)\big)$, *where:* $AwV(v_1) = 3$.

Thus, $\Sigma_{v\in\mathcal{V}}\big(AwV(v)\big) = AwV(v_1) = 3$ *rows (as presented in Figure 4.4).*

4.3.3.2 MISP Correlation Engine Table BCNF Decomposition

From the observations and the cases presented in Section 4.3.2.1 and Section 4.3.3.1, we realize the redundancy of the *correlations* relation produced by the MISP correlation engine. The most common type of redundancy in relational databases is based on the functional dependencies. A relational schema avoids functional dependency based redundancy, when it is in Boyce-Codd Normal Form (BCNF) or in 3rd Normal Form (3NF). These forms are defined in Definition 1 and Definition 2 respectively.

Definition 1. A relational schema R is in BCNF **iff** for every one of its dependencies X $\rightarrow$ Y, at least one of the following conditions hold:

- X $\rightarrow$ Y is a trivial functional dependency (Y $\subseteq$ X),
- X is a superkey for schema R.

Definition 2. A relational schema R is in 3NF **iff** for every one of its dependencies X $\rightarrow$ Y, at least one of the following conditions hold:

- X $\rightarrow$ Y is a trivial functional dependency (Y $\subseteq$ X),
- X is a superkey for schema R,
- Y is part of a candidate key.

As expressed in Section 3.1.5, the functional dependencies of the *correlations* relation are:

FD1. `id` $\rightarrow$ `value 1_event_id 1_attribute_id event_id attribute_id org_id distribution a_distribution sharing_group_id a_sharing_group_id date info`

FD2. `1_event_id 1_attribute_id event_id attribute_id org_id` $\rightarrow$ `id`

FD3. `1_event_id 1_attribute_id` $\rightarrow$ `value`

FD4. `event_id attribute_id` $\rightarrow$ `value`

From the above functional dependencies, the candidate keys deriving from the *correlations* relation are:

- `id`
- `1_event_id 1_attribute_id event_id attribute_id org_id`

Thus, correlations is not in BCNF, since the left set of **FD3** and **FD4** functional dependencies are not superkeys. Specifically, the functional dependency **FD3**,

`event_id attribute_id` $\rightarrow$ `value`

is not trivial and the attributes `event_id, attribute_id` do not consist a superkey. As a result, a pair of attributes' IDs, that provide the same value, along with their corresponding event IDs, create duplicate entries in the *correlations* relation. That occurs during the correlation engine's procedure, which will produce a correlation entry each time it encounters a matching pair of attributes' values, within different events. To counter that, we need to restructure the correlations table, while also altering the MISP correlation engine to implement this alteration.

Furthermore, the *correlations* relation is neither in BCNF nor in 3rd Normal Form (3NF), since the left set of **FD3** and **FD4** functional dependencies are not superkeys and *value* (on the right set of **FD3** and **FD4**) is not part of a candidate key. Since the *correlation* relation employed by MISP is not in BCNF or 3NF, to reduce redundancy, we apply the decomposition algorithm [79].

The decomposition algorithm considers in turn all functional dependencies that violate the BCNF and 3NF conditions. The first such dependency is **FD3**.

Let A={1_event_id, 1_attribute_id} and B={value}. According to the decomposition algorithm, the *correlations* relation is decomposed into relations R1 and R2 as follows:

Relation	Attributes	Functional Dependencies
R1	A$^+$={1_event_id, 1_attribute_id, value}	1_event_id 1_attribute_id → value
R2	*correlations*-(A$^+$-B)= ={id, value, event_id, attribute_id, org_id, distribution, a_distribution, sharing_group_id, a_sharing_group_id, date, info}	id → value, event_id, attribute_id, org_id, distribution, a_distribution, sharing_group_id, a_sharing_group_id, date, info event_id attribute_id → value

Following the same procedure for **FD4**, we are resulted with:

Relation	Attributes	Functional Dependencies
R3	A$^+$={event_id, attribute_id, value}	event_id attribute_id → value
R4	*correlations*-(A$^+$-B)= ={id, value, 1_event_id, 1_attribute_id, org_id, distribution, a_distribution, sharing_group_id, a_sharing_group_id, date, info}	id → value, 1_event_id, 1_attribute_id, org_id, distribution, a_distribution, sharing_group_id, a_sharing_group_id, date, info 1_event_id 1_attribute_id → value

Summing up, we are resulted with R1, R2, R3, and R4, where R1 and R3 provide the same information, thus we may discard one of them. Similarly, R2 and R4 provide the same information, thus we may discard one of them too. From that, we have:

R2: {id, value, event_id, attribute_id, org_id, distribution, a_distribution, sharing_group_id, a_sharing_group_id, date, info}, with functional dependencies:

- id → value, event_id, attribute_id, org_id, distribution, a_distribution, sharing_group_id, a_sharing_group_id, date, info
- event_id attribute_id → value

R3: {event_id, attribute_id, value}, with functional dependencies:

- event_id attribute_id → value

Finally, R3 consists only a subset of R2, and thus it may be discarded as well, leading to one final relationship:

R2: {`id, value, event_id, attribute_id, org_id, distribution, a_distribution, sharing_group_id, a_sharing_group_id, date, info`}, which is the BCNF equivalent of *correlations*. We will refer to this relation by *correlationsNF*.

To implement that alteration in the MISP correlation engine of **ASPIS**, the table *correlationsNF* is created in the MISP database, as follows:

```sql
CREATE TABLE correlationsNF AS
SELECT DISTINCT value, event_id, attribute_id, org_id, distribution,
    a_distribution, sharing_group_id, a_sharing_group_id, date, info
FROM correlations;
```

Next, after removing the *correlations* table, a *view* is created, in order to support the engine's functionality. The view is created in the following manner:

```sql
CREATE VIEW misp.correlations AS SELECT c1.value AS value, c2.event_id AS
    1_event_id, c2.attribute_id AS 1_attribute_id, c1.event_id AS event_id,
    c1.attribute_id AS attribute_id, c1.org_id AS org_id, c1.distribution AS
    distribution, c1.a_distribution AS a_distribution, c1.sharing_group_id AS
    sharing_group_id, c1.a_sharing_group_id AS a_sharing_group_id, c1.date AS
    date, c1.info AS info
FROM correlationsNF c1, correlationsNF c2
WHERE c1.value = c2.value AND c1.event_id != c2.event_id;
```

Then, we add the required indices in the *correlationsNF* table:

```sql
ALTER TABLE correlationsNF ADD INDEX (value(255), event_id, attribute_id, org_id,
    sharing_group_id);
```

Finally, these changes are implemented in the internal code of the MISP correlation engine as well.

4.4 Monitored Sources

As mentioned in Section 4.1, we have identified five certified CTI sources, which are either vulnerability or exploit databases and contain analyzed CTI, in the form of vulnerability and exploit reports. In this section, we present the example sources that are being monitored by **ASPIS**, along with the CTI they provide.

NVD is the U.S. government repository of standards-based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. The NVD includes databases of security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics. NVD provides CTI information about vulnerabilities and their corresponding weaknesses, in the CVE and CWE formats. Specifically, NVD shares CTI like CVE IDs, CPEs, vulnerability entries' publication and modification dates, CVSS base metrics, external references and vulnerability entries' descriptions, along with the corresponding CWE IDs, CWE names and CWE descriptions.

JVN stands for *the Japan Vulnerability Notes*, and it is a vulnerability information portal site designed to help ensure internet security by providing vulnerability information and their solutions for software products used in Japan. JVN iPedia is the database of vulnerability countermeasure information published on JVN, in Japan and abroad. JVN provides CTI information about vulnerabilities, in the CVE format. Specifically, JVN shares CTI like CVE IDs, CPEs, vulnerability entries' publication and modification dates, CVSS base metrics, external references and vulnerability entries' descriptions.

KB-Cert vulnerability notes database is run by the CERT Division, which is part of the Software Engineering Institute, a federally funded research and development center operated by Carnegie Mellon University. It provides information about software vulnerabilities, such as summaries, technical details, remediation information, and lists of affected vendors. KB-Cert provides CTI information about vulnerabilities, in the CVE format. Specifically, KB-Cert shares CTI like CVE IDs, vulnerability entries' publication and modification dates, CVSS base/temporal/environmental metrics, external references and vulnerability entries' descriptions.

VulDB is a community-driven vulnerability database, which is consisted of a variety of teams (for analyzing reported threats, conducting the reports, moderating reports that may contain wrong or inaccurate data gathered from other sources, etc.), and it documents all security vulnerabilities that got published in a plethora of sources. Such examples may be vendor sites, mailing lists, vulnerability contributors, vulnerability databases, code repositories, news sites and blogs, social networks, marketplaces, darknet, as well as internal testing (through white-hat exploiting techniques). VulDB provides CTI information and meta-data about vulnerabilities, enriching the standard CVE format. Specifically, it shares CTI like CVE IDs, CPEs, vulnerability entries' publication and modification dates, CVSS base metrics, external references, vulnerability entries' summaries, zero day and current exploitation price range estimations, exploitability levels and remediation levels.

Exploit-DB is maintained by Offensive Security (OffSec), an information security training company that provides various information security certifications as well as high end penetration testing services. However, Exploit-DB is a non-profit project that is provided as a public service by the company. It is a CVE-compliant archive of public exploits and the corresponding vulnerable software, developed for use by penetration testers and vulnerability researchers. Exploit-DB provides exploits gathered through direct submissions, mailing lists, and other public sources.

It serves as a repository for exploits and proof-of-concepts. Exploit-DB provides CTI information about exploits. Specifically, it shares CTI like CVE IDs, Exploit-DB IDs (for the yet not discovered CVEs), exploits' descriptions, exploits' authors' alias and the raw scripts for each exploit stored.

4.5 The Source Crawling and Parsing Phase

During this phase, **ASPIS** downloads and starts parsing, one-by-one, the required files from the aforementioned sources, in order to extract only the encompassed CTI. To achieve that, we have coded five Shell and Python scripts, accordingly tailored for each one of the five sources at hand, which are executed daily to update the **ASPIS** indexed content. In the following sections we will describe in detail the parsing process followed for each source. The first two sources, NVD and JVN, provide data feeds of their entries, in JSON and XML respectively, which are updated daily. The last three sources, KB-Cert, VulDB and Exploit-DB, do not provide data feeds, and hence we created scrapping processes to extract the CTI of new entries, which are specifically tailored for each source and will be textually described in the following sections.

4.5.1 NVD Parsing

NVD provides public access to a plethora of data feeds of its contents [35]; vulnerability data feeds using the JSON format, CPE match data feed using the JSON format, RSS vulnerability feeds, and official vendor comments on existing vulnerabilities. The vulnerability data feeds contain both feeds grouped by the year of the CVE ID, like nvdcve-1.1-2020.json, and the most recent entries (nvdcve-1.1-recent.json), along with the most recent modifications of past entries (nvdcve-1.1-modified.json).

During this phase, **ASPIS** downloads the two latter aforementioned data feeds, which regard only the recent and recently modified entries of NVD. Then, through a custom-built Python JSON parser, we extract the CTI of each entry of the JSON files. Specifically, from these entries we extract the following: CVE ID, CPEs, publication and modification dates of the report, base CVSS metrics (strings and scores), references to other advisories (along with the type of the reference, e.g., *Vendor* - for an official announcement of the affected product's vendor), description of the vulnerability, CWE ID, CWE name and CWE description.

4.5.2 JVN Parsing

JVN provides public access to various data feeds of the JVN iPedia contents [25], using the XML format; vulnerability data feeds, CPE match data feed, and a Vendor name (product developer) list. The vulnerability data feeds contain both feeds grouped by the year of the CVE ID (i.e., jvndb_detail_2020.rdf), and the most recent entries (jvndb_new.rdf), along with the most recent modifications of past entries (jvndb.rdf).

During this phase, **ASPIS** downloads the two latter data feeds, which regard recent entries and the modifications of the past entries of JVN. Then, similarly to the NVD Parsing, through a custom-built Python XML parser, we extract the CTI of each entry of the XML files. Namely, from the XML entries, we extract the following: CVE ID, CPEs, publication and modification dates of the report, base CVSS metrics (strings and scores), references to other advisories, and description of the vulnerability.

4.5.3 KB-Cert Crawling and Parsing

KB-Cert provides a publicly available list of their notes [27]. During this phase, **ASPIS** first crawls all the links of this list, and stores them internally in a list structure in the Python script implemented. After it has finished crawling the vulnerability notes' links, it proceeds to parse them one-by-one as an HTML file. For the purposes of this task, we have built a tailored HTML parser in Python, which targets specific fields of the HTML file, that contain the useful CTI. The fields parsed contain information, such as: an overview of the vulnerability, a description of the vulnerability, a description of the impact of this vulnerability on the affected products, a description of the solution to mitigate the issues caused by the vulnerability, acknowledgements to the reporter of the vulnerability and the note conductor, base, temporal and environmental CVSS metrics (strings and scores), references to other advisories, CVE ID, and a publication and modification date of the note at hand.

4.5.4 VulDB Crawling and Parsing

VulDB provides a limited access to its data. However, most data are accessible on the public website (via [58–60]), providing up to 50 entries per day. The publicly available CTI in VulDB is presented concisely in tables. To daily parse all available CTI from VulDB, we have employed a custom-built Python script, which acts as an HTML parser for three separate pages of VulDB. First, it visits the *recent* page of VulDB with regard to the current date, like `https://vuldb.com/?recent.20200909`, for the daily entries at hand. From the *recent* page, we parse the following CTI, row-by-row from the provided table: publication date of the vulnerability report, summary of the vulnerability, base meta-CVSS score, zero day and current price range estimations (which derive from marketplaces' monitoring), exploitability label (i.e., an available exploit proof-of-concept), remediation level (i.e., an official-fix proposal, like upgrading), and CVE ID. By following this approach, we are able to crawl for more information or for entries that were above the limit of 50 in *recent*, and are encompassed in the *cvssv3* and *exploits* pages. To do so, for each entry of the *recent* page, there is a set of url links that are parsed and stored in Python lists, to be visited after parsing *recent*. This set is provided by two attributes of each entry; the CVSS base score, and the price range estimations, which are accompanied by a hyperref respectively, that lead to the aforementioned pages (i.e., `https://vuldb.com/?cvssv3.20200915` and `https://vuldb.com/?exploits.20200915`). Similarly to *recent*, we then proceed to parse the *cvssv3* page; row-by-row we extract the following CTI: base meta-CVSS score (which occurs as a result of the average score of all other available CVSS strings and scores), base VulDB CVSS string and score, base NVD CVSS string and score, base Vendor CVSS string and score (as resulted by the Vendor's analysis), and base Researcher CVSS string and score (as provided by a researcher to the VulDB), summary of the vulnerability, and CVE ID. Finally, the same procedure is followed for the *exploits* page. The employed script parses the provided table row-by-row, extracting the following CTI: publication date of the vulnerability report, zeroday and current day price range estimations (as they derive from marketplaces' monitoring), summary of the vulnerability, exploitability label, a link to the source of the vulnerability's exploit, and CVE ID.

4.5.5 Exploit-DB Crawling and Parsing

Exploit-DB provides a publicly available list of their exploits and proof-of-concepts [12]. However, the contents of the list load dynamically and do not exist in the HTML file. Thus, in order to parse the available CTI, we have created a Shell script, that starts parsing each page of the exploits of Exploit-DB. So, the first time it executes, it would start by `https://www.exploit-db.com/exploits/1`, and while it reaches the final entry, it keeps the first (unstored) Exploit-DB ID which hadn't a CVE ID assigned, in a file locally, in order to avoid parsing the same entries repeatedly. The fields parsed from these pages contain information as: Exploit-DB ID, CVE ID, description of the exploit, author alias, and finally the raw script of the exploit.

4.6 The Object Structuring Phase

After extracting all actionable CTI from the parsing procedure described in the previous section, **ASPIS** proceeds to structure it in the format of the suitable MISP objects, in accordance to the objects and attributes described in Section 4.3.1, with the use of the PyMISP library, as presented in Figure 4.2. To achieve that, the system generates the MISP objects in JSON format, as sets of attributes' <`field, value, comment`>, with the values extracted from the parsing phase. The *comment* field is used to store enriching information to the *value*. In example, declaring the source of a reference, whether it is from the affected vendor, or an other vulnerability notes' source. Examples of the occurring format, are presented in Appendix A, for each source.

4.7 The Event Management Phase

The event management phase executes in parallel to the source crawling and parsing phase as described in the previous section. What actually happens during this phase, is either the creation of new events, each time new CTI arrives to the **ASPIS** system, or the modification of previously stored events, due to updated CTI artifacts. This is also, the phase during which the clustering of the gathered CTI occurs. In the following sections, we describe the process followed in order to achieve that.

4.7.1 Event Lookup

First of all, in order to determine whether the CTI which arrived, is uncatalogued by the system or not, **ASPIS** queries the MISP instance, with the CTI's unique identifier at hand, which may be either a CVE ID, or an Exploit-DB ID, as described in Section 4.5. So, through the use of PyMISP, **ASPIS** queries MISP, for any event that regards the currently parsed CTI's unique ID, by looking into the events' *info* field, which is used by the system for storing such identifiers. The result of the query can lead to two possible outcomes; (a) the parsed CTI ID doesn't exist within the **ASPIS** database, and therefore a new event should be created (see Section 4.7.2), or (b) the parsed CTI ID exists, and therefore one or more existing events should be modified (see Section 4.7.3). For the second case, the system returns the corresponding MISP Event in JSON format, through PyMISP, and it also temporarily stores the corresponding MISP Event ID, as it is stored in the MISP instance.

4.7.2 Event Creation

If the parsed CTI is unindexed by **ASPIS**, then through PyMISP, the system follows a three-step approach, to catalogue it.

Step 1. It generates a new event in the MISP instance, with the following predefined event characteristics (as presented in Section 3.5.3.3): *Distribution - Your organisation only*, *Threat Level - Undefined*, *Analysis - Completed*. Additionally, it sets the event's *info* field, to match the parsed CVE ID, or the Exploit-DB ID (in case a CVE ID is not defined in the Exploit-DB entry at hand).

Step 2. It generates the required MISP Objects (with regard to the specifications of each monitored source), from the constructed JSON structures of the object structuring phase (as described in Section 4.6). Additionally, the generated objects' validity is checked both locally, through the PyMISP library's objects' definitions, and externally, through a PyMISP request of the MISP instance objects' definitions. Both definitions must be the same for this step to succeed, and they are expressed in the form of JSON files, in the PyMISP library's files and the MISP instance's files.

Step 3. It attaches the generated MISP Objects to the event that was generated in the first step, on the MISP instance.

4.7.3 Event Modification

An event modification may occur in three cases; the system parsed CTI which is (a) unstored by the system, but it regards an existing CVE ID entry (which happens due to overlapping CTI from different sources), (b) an updated version of previously stored CTI, (c) an updated version of previously stored CTI from Exploit-DB, where the old entry did not contain a reference to a CVE ID, and the new one does. Any modification that occurs during this phase, makes use of the previously stored MISP Event, which derives from the Event Lookup phase (Section 4.7.1), through its ID, that points on the MISP instance, through PyMISP, the event that is going to be modified. In the following sections, we describe the process followed for each one of the aforementioned cases.

4.7.3.1 Event Enrichment with Complementary CTI from Another Source

In this case, the system encountered unstored CTI deriving from the parsing phase, which regards an existing event on the MISP instance. **ASPIS** proceeds to generate the required MISP Objects, as described in the second step of Section 4.7.2, and then it attaches them to the existing event. To achieve that, the system checks the *credit* field of each object within the event at hand. If there is no match, the system proceeds with the process described.

4.7.3.2 Event Update due to Updated CTI

In this case, the system encountered previously stored CTI deriving from the parsing phase, which regards an existing outdated entry on the MISP instance. Similarly to the previous case, to achieve that, the system generates the corresponding MISP Objects and checks the *credit* field of each object within the event at hand. If there is a match, the system proceeds to check the *modified* attribute of the matching object, which regards the modification date of the CTI encompassed. Iff the modification date of the newly parsed CTI is more recent than the previously stored one, the system deletes the stored object, and proceeds to attach the newly generated object, to the MISP Event at hand.

4.7.3.3 Exploit-DB Entry Added a Reference to a CVE ID

Exploit-DB is an exceptional source, since it specifically regards exploit CTI (not vulnerability CTI, as the rest of the sources do). Thus, it is possible to encounter CTI in Exploit-DB, which does not encompass a reference to a CVE ID. In this case, it is stored in the **ASPIS** database, with the Exploit-DB ID at the event's *info* field, as mentioned in the first step of the event creation phase (Section 4.7.2). However, there is a chance that an Exploit-DB entry is updated, and it then encompasses a reference to a CVE ID. In this case, the system proceeds to check if the returned matching event of the event lookup phase, does not contain a CVE ID in its *info* field. Thus, if the parsed CTI of Exploit-DB contains a reference to a CVE ID, and the aforementioned event that does not regard a CVE ID, but encompasses the aforementioned CTI, the system proceeds to (a) delete the existing event, and then (b) attach the generated MISP Object to the corresponding MISP Event (of the referenced CVE ID, found in the parsed CTI).

4.7.4 Events' Correlations

Finally, it is important to note that, after each event creation/modification, **ASPIS** proceeds to recalculate the correlations, through the MISP Correlation Engine, since there is a possibility that the newly stored CTI may regard the same affected products, as other events (as described in Section 4.3.2). After this process, the event at hand points to all related events, as depicted in Figures 3.7, 3.10.

4.8 The CTI Sharing and Reviewing Phase

After gathering all publicly available CTI from the monitored sources, **ASPIS** is then able to proceed to the CTI sharing and reviewing phase. The sharing of the encompassed CTI may occur in two ways. The first, is to share CTI through the sharing features of MISP, as described in Section 3.2. The second method, is to query **ASPIS** through the provided MISP REST API, using the required authorization credentials. In the following section, we provide a detailed overview of how this may be achieved.

4.8.1 MISP REST API: RESTful Searches

As mentioned earlier, MISP provides the option to search its embedded database, via the provided REST API. In this section, we focus on RESTful searches over the stored CTI of the **ASPIS** database. As mentioned in Section 3.4, MISP is able to export CTI in various CTI sharing standards such as JSON, XML, OpenIOC, Suricata, Snort, STIX, and more. Thus, it is possible to query the MISP REST API, for information regarding a specific entry, and receive a response in the requested format. For these purposes, there are two REST endpoints; one that regards information on event level, and one for the attribute level. In the first case, a user may retrieve all related CTI to the posed query, while in the second case, the user may retrieve all related attributes of the stored CTI, which match the posed query (e.g., a vulnerability's description). Both of these endpoints use the POST HTTP method to query the MISP REST API. In the following sections, we break down the structure of the POST request (endpoint, headers, payload/body), that is required to retrieve the desired CTI from the MISP API.

4.8.1.1 List of RESTful Endpoints

As mentioned earlier, there are two REST endpoints to retrieve the desired CTI from MISP. Below, we present their URLs:

- Events' endpoint: `https://<misp_url>/events/restSearch`
- Attributes' endpoint: `https://<misp_url>/attributes/restSearch`

4.8.1.2 Authorization

Automation functionality is designed to automatically feed other tools and systems with the data of the MISP repository. To make this functionality available for automated tools, an authentication key is used. Thus, in order to gain access to the REST API of MISP, the users should include their uniquely generated key (as a header in the request).

4.8.1.3 Headers

The headers that should be included in the request are the following:

```
Authorization: <authorization_key>
Accept: application/json
Content-type: application/json
```

4.8.1.4 Search Constraints (Payload/Body)

Finally, in the RESTful search request, the user should also include the search constraints of choice, in the following manner:

```
{
  "returnFormat": "format", //(required)
  "value": "search_value", //(required)
  "type": "type_of_value_to_search",
  "last": "last_x_amount_of_time",
  "page": page_number,
  "limit": number_of_results_per_page
}
```

Where:

- The `format` may be one of: `json`, `xml`, `csv`, `openioc`, `stix2`, `suricata`, `yara`, and `more`.
- The `search_value` may be either a specific string (e.g., `cpe:/a:apache:solr:6.4.2`), or it may include wildcards, which are expressed by the `%` character in MISP (e.g., `%solr%`).
- The `type_of_value_to_search` specifies the types of attributes, that the search should focus on. (e.g., a CPE is stored in an attribute of `text` type, while a reference url is stored in an attribute of `link` type.[1])
- The `last_x_amount_of_time` is used to request events published within the last `x` amount of time, where `x` can be defined in days, hours or minutes (for example: `5d` or `12h` or `30m`).
- The `page_number` serves as the paginator of the results, where the user can request the n^{th} page of the results.

[1]See `https://www.circl.lu/doc/misp/categories-and-types/#types` for all available types.

- The `number_of_results_per_page` is used to define the number of results the user would like to receive per page.

4.8.1.5 cURL Example

With all of the aforementioned at hand, a user could pose a RESTful search, via the curl command on the terminal. For example:

```
curl \
-d '{"returnFormat": "json", "value": "%ibm%", "page": 1, "limit": 10}' \
-H "Authorization: auth_key" \
-H "Accept: application/json" \
-H "Content-type: application/json" \
-X POST https://misp_url/events/restSearch
```

In this case, the user searches and retrieves the 10 latest CTI entries stored in **ASPIS**, which refer "ibm" in any of their attributes, having the search output be in JSON format.

4.8.2 CTI Reviewing through MISP Sightings

Finally, for the reviewing of the encompassed CTI, **ASPIS** makes use of the MISP Sightings mechanism (as described in Section 3.3), which allows users to declare whether an artifact is true positive or false positive, with regard to the vulnerabilities and exploits that are stored in **ASPIS**.

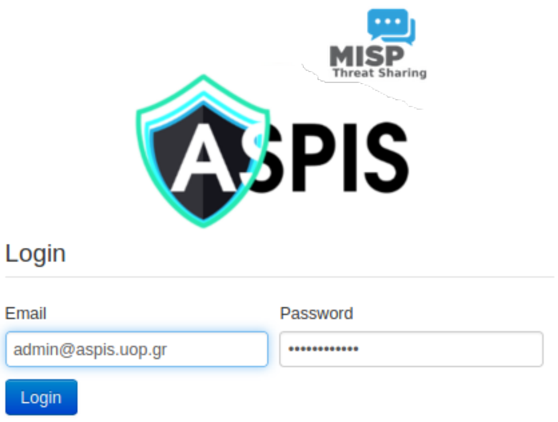
4.9 ASPIS Usage

This section presents the main usage of **ASPIS**. Specifically, it showcases the main functionalities of MISP, that provide a security expert with browsing, filtering, inspecting and sharing capabilities, over the **ASPIS** indexed artifacts.

4.9.1 Login

The first page an **ASPIS** user comes across, is the login page, in order to gain access to the CTI contents. As presented in Figure 4.5, the two fields required for logging in, are an e-mail and a password.[2]

[2]In a fresh setup of MISP, these are provided to the system administrator during the installation. The e-mail isn't required to be a valid e-mail. Instead, it is a credential in the form of e-mail, and it is created and provided by the system administrator, to each user.

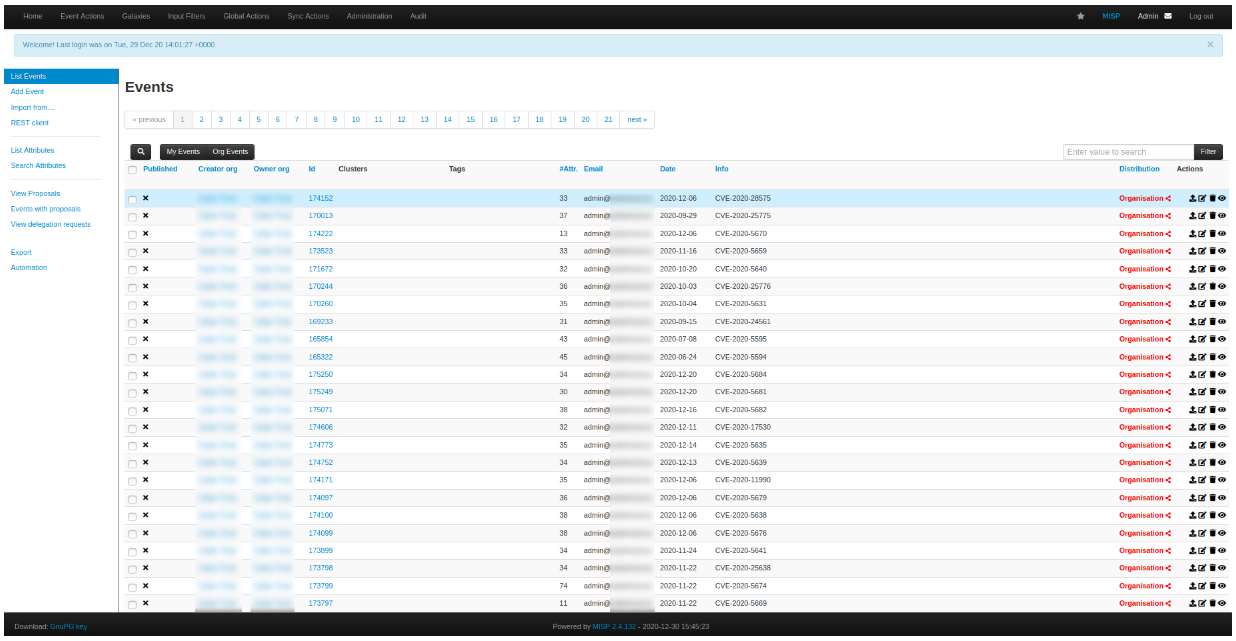


Email: admin@aspis.uop.gr

Password: ••••••••••••

Login

Figure 4.5: **ASPIS** Login



Welcome! Last login was on Tue, 29 Dec 20 14:01:27 +0000

Events

Published	Creator org	Owner org	Id	Clusters	Tags	#Attr.	Email	Date	Info	Distribution
×	[redacted]	[redacted]	174152			33	admin@[redacted]	2020-12-06	CVE-2020-28575	Organisation
×	[redacted]	[redacted]	170013			37	admin@[redacted]	2020-09-29	CVE-2020-25775	Organisation
×	[redacted]	[redacted]	174222			13	admin@[redacted]	2020-12-06	CVE-2020-5670	Organisation
×	[redacted]	[redacted]	173523			33	admin@[redacted]	2020-11-16	CVE-2020-5659	Organisation
×	[redacted]	[redacted]	171672			32	admin@[redacted]	2020-10-20	CVE-2020-5640	Organisation
×	[redacted]	[redacted]	170244			36	admin@[redacted]	2020-10-03	CVE-2020-25776	Organisation
×	[redacted]	[redacted]	170260			35	admin@[redacted]	2020-10-04	CVE-2020-5631	Organisation
×	[redacted]	[redacted]	169233			31	admin@[redacted]	2020-09-15	CVE-2020-24561	Organisation
×	[redacted]	[redacted]	165854			43	admin@[redacted]	2020-07-08	CVE-2020-5595	Organisation
×	[redacted]	[redacted]	165322			45	admin@[redacted]	2020-06-24	CVE-2020-5594	Organisation
×	[redacted]	[redacted]	175250			34	admin@[redacted]	2020-12-20	CVE-2020-5684	Organisation
×	[redacted]	[redacted]	175249			30	admin@[redacted]	2020-12-20	CVE-2020-5681	Organisation
×	[redacted]	[redacted]	175071			38	admin@[redacted]	2020-12-16	CVE-2020-5682	Organisation
×	[redacted]	[redacted]	174606			32	admin@[redacted]	2020-12-11	CVE-2020-17530	Organisation
×	[redacted]	[redacted]	174773			35	admin@[redacted]	2020-12-14	CVE-2020-5635	Organisation
×	[redacted]	[redacted]	174752			34	admin@[redacted]	2020-12-13	CVE-2020-5639	Organisation
×	[redacted]	[redacted]	174171			35	admin@[redacted]	2020-12-06	CVE-2020-11990	Organisation
×	[redacted]	[redacted]	174097			36	admin@[redacted]	2020-12-06	CVE-2020-5679	Organisation
×	[redacted]	[redacted]	174100			38	admin@[redacted]	2020-12-06	CVE-2020-5638	Organisation
×	[redacted]	[redacted]	174099			38	admin@[redacted]	2020-12-06	CVE-2020-5676	Organisation
×	[redacted]	[redacted]	173899			34	admin@[redacted]	2020-11-24	CVE-2020-5641	Organisation
×	[redacted]	[redacted]	173798			34	admin@[redacted]	2020-11-22	CVE-2020-25638	Organisation
×	[redacted]	[redacted]	173799			74	admin@[redacted]	2020-11-22	CVE-2020-5674	Organisation
×	[redacted]	[redacted]	173797			11	admin@[redacted]	2020-11-22	CVE-2020-5669	Organisation

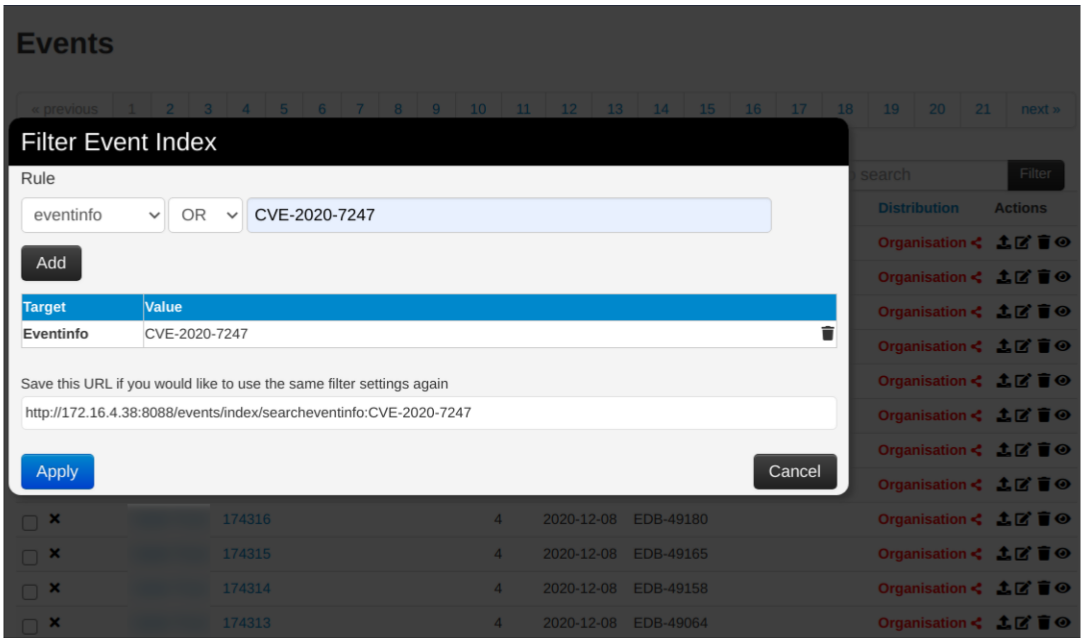
Download: GnuPG key

Powered by MISP 2.4.132 - 2020-12-30 15:45:23

Figure 4.6: **ASPIS** View Events List

4.9.2 Events Browsing and Filtering

By the time the users are logged in to the **ASPIS** MISP instance, they are provided with a list of stored events, sorted in a chronologically descending order (as presented in Figure 4.6. Through this page, the users are able to browse through the list of the stored events. Furthermore, this page provides the users with filtering capabilities. By clicking on the magnifying glass button, a window will pop up and as shown in Figure 4.7, the users can search for a specific CVE ID, by adding a filtering *rule* on the *eventinfo* field. Finally, through this page, if the users click on an event ID, they will be redirected to the *Event View* page (presented in Section 3.5.3.3).


Events

« previous 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 next »

Filter Event Index

Rule

eventinfo | OR | CVE-2020-7247

Add

Target	Value
Eventinfo	CVE-2020-7247

Save this URL if you would like to use the same filter settings again

http://172.16.4.38:8088/events/index/searcheventinfo:CVE-2020-7247

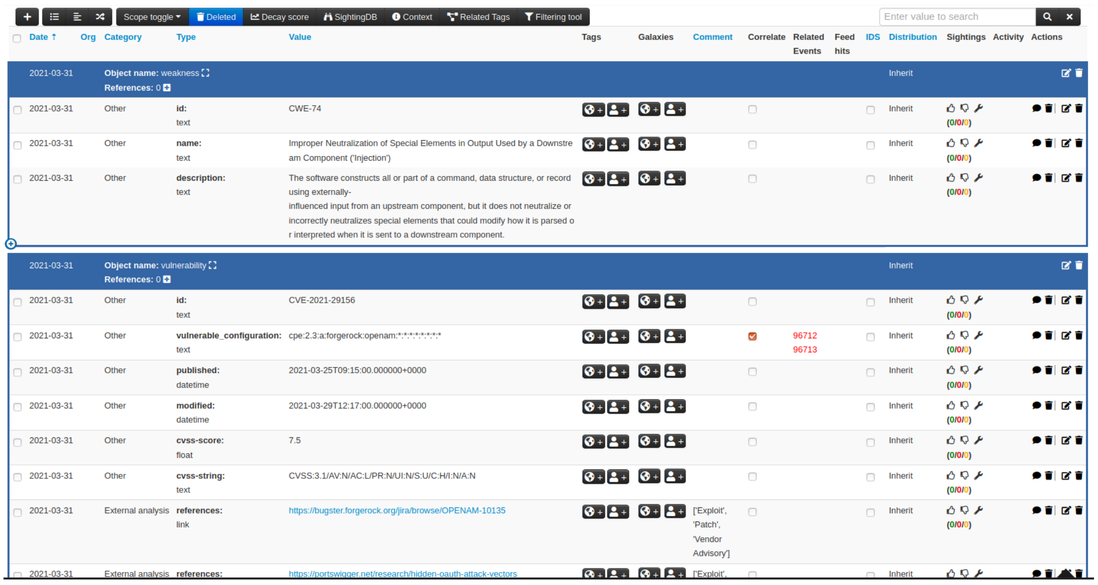
Apply | Cancel

✕	[redacted]	174316	4	2020-12-08	EDB-49180	Organisation	
✕	[redacted]	174315	4	2020-12-08	EDB-49165	Organisation	
✕	[redacted]	174314	4	2020-12-08	EDB-49158	Organisation	
✕	[redacted]	174313	4	2020-12-08	EDB-49064	Organisation	

Figure 4.7: **ASPIS** Filter Events

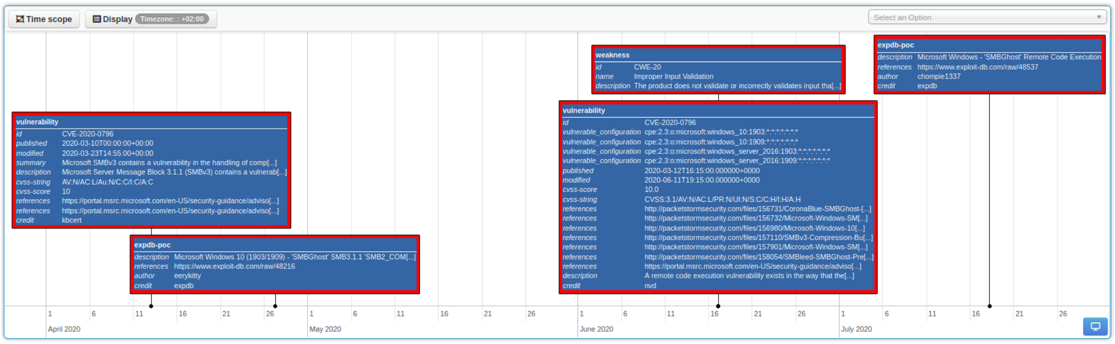
4.9.3 Events Inspection

The *Event View* page enables **ASPIS** users to further inspect an event. First, as presented in Figure 4.8, users can view, modify and review the event's stored artifacts. Moreover, the *Event View* page provides an *Event Timeline* view (Figure 4.9), through which users are able to examine the timeline of any modification that may have occurred to the event. In **ASPIS**, users may use the Event Timeline, to gain an overview of the artifacts' discovery timeline. Finally, users are provided with a correlation graph, which enables them to investigate any correlation that may have occurred between the event at hand and the rest of the events, as illustrated in Figure 4.10.



Date ↑	Org	Category	Type	Value	Tags	Galaxies	Comment	Correlate	Related Events	Feed hits	IDS	Distribution	Sightings	Activity	Actions
2021-03-31		**Object name:** weakness **References:** 0										Inherit			
2021-03-31		Other	**id:** text	CWE-74				☐			☐	Inherit	(0/0/0)		
2021-03-31		Other	**name:** text	Improper Neutralization of Special Elements in Output Used by a Downstre am Component ('Injection')				☐			☐	Inherit	(0/0/0)		
2021-03-31		Other	**description:** text	The software constructs all or part of a command, data structure, or record using externally- influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify how it is parsed o r interpreted when it is sent to a downstream component.				☐			☐	Inherit	(0/0/0)		
2021-03-31		**Object name:** vulnerability **References:** 0										Inherit			
2021-03-31		Other	**id:** text	CVE-2021-29156				☐			☐	Inherit	(0/0/0)		
2021-03-31		Other	**vulnerable_configuration:** text	cpe:2.3:a:forgerock:openam:*:*:*:*:*:*:*:*				☑	96712 96713		☐	Inherit	(0/0/0)		
2021-03-31		Other	**published:** datetime	2021-03-25T09:15:00.000000+0000				☐			☐	Inherit	(0/0/0)		
2021-03-31		Other	**modified:** datetime	2021-03-29T12:17:00.000000+0000				☐			☐	Inherit	(0/0/0)		
2021-03-31		Other	**cvss-score:** float	7.5				☐			☐	Inherit	(0/0/0)		
2021-03-31		Other	**cvss-string:** text	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N				☐			☐	Inherit	(0/0/0)		
2021-03-31		External analysis	**references:** link	https://bugster.forgerock.org/jira/browse/OPENAM-10135			['Exploit', 'Patch', 'Vendor Advisory']	☐			☐	Inherit	(0/0/0)		
2021-03-31		External analysis	**references:**	https://portswigger.net/research/hidden-oauth-attack-vectors			['Exploit',	☐			☐	Inherit			

Figure 4.8: **ASPIS** Event Inspection



vulnerability

id	CVE-2020-0796
published	2020-03-10T00:00:00+00:00
modified	2020-03-23T14:55:00+00:00
summary	Microsoft SMBv3 contains a vulnerability in the handling of comp[...]
description	Microsoft Server Message Block 3.1.1 (SMBv3) contains a vulnerab[...]
cvss-string	AV:N/AC:L/Au:N/C:C/I:C/A:C
cvss-score	10
references	https://portal.msrc.microsoft.com/en-US/security-guidance/adviso[...]
references	https://portal.msrc.microsoft.com/en-US/security-guidance/adviso[...]
credit	kbcert

expdb-poc

description	Microsoft Windows 10 (1903/1909) - 'SMBGhost' SMB3.1.1 'SMB2_COM[...]
references	https://www.exploit-db.com/raw/48216
author	eerykitty
credit	expdb

weakness

id	CWE-20
name	Improper Input Validation
description	The product does not validate or incorrectly validates input tha[...]

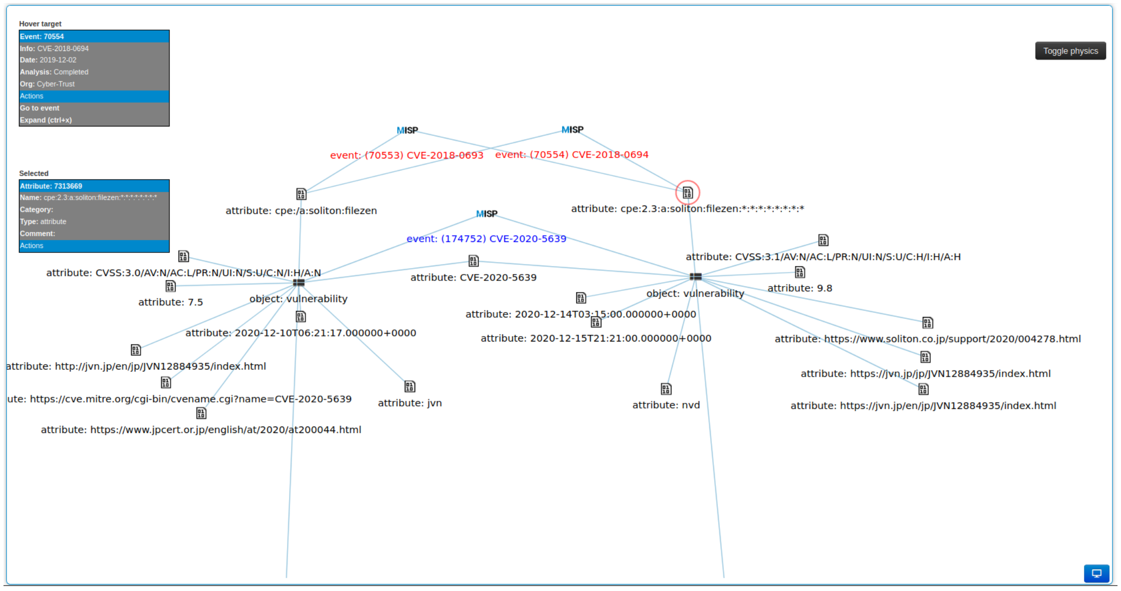
vulnerability

id	CVE-2020-0796
vulnerable_configuration	cpe:2.3:o:microsoft:windows_10:1903:*:*:*:*:*:*:*
vulnerable_configuration	cpe:2.3:o:microsoft:windows_10:1909:*:*:*:*:*:*:*
vulnerable_configuration	cpe:2.3:o:microsoft:windows_server_2016:1903:*:*:*:*:*:*:*
vulnerable_configuration	cpe:2.3:o:microsoft:windows_server_2016:1909:*:*:*:*:*:*:*
published	2020-03-12T16:15:00.000000+0000
modified	2020-06-11T19:15:00.000000+0000
cvss-score	10.0
cvss-string	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
references	http://packetstormsecurity.com/files/156731/CoronaBlue-SMBGhost-[...]
references	http://packetstormsecurity.com/files/156732/Microsoft-Windows-SM[...]
references	http://packetstormsecurity.com/files/156980/Microsoft-Windows-10[...]
references	http://packetstormsecurity.com/files/157110/SMBv3-Compression-Bu[...]
references	http://packetstormsecurity.com/files/157901/Microsoft-Windows-SM[...]
references	http://packetstormsecurity.com/files/158054/SMBleed-SMBGhost-Pre[...]
references	https://portal.msrc.microsoft.com/en-US/security-guidance/adviso[...]
description	A remote code execution vulnerability exists in the way that the[...]
credit	nvd

expdb-poc

description	Microsoft Windows - 'SMBGhost' Remote Code Execution
references	https://www.exploit-db.com/raw/48537
author	chompie1337
credit	expdb

Figure 4.9: **ASPIS** Event Timeline Examination



Toggle physics
MISP
MISP
MISP
event: (70553) CVE-2018-0693
event: (70554) CVE-2018-0694
event: (174752) CVE-2020-5639
attribute: cpe:/a:soliton:filezen
attribute: cpe:2.3:a:soliton:filezen:*:*:*:*:*:*:*:*
attribute: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N
attribute: 7.5
object: vulnerability
attribute: CVE-2020-5639
attribute: 2020-12-10T06:21:17.000000+0000
attribute: http://jvn.jp/en/jp/JVN12884935/index.html
ute: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5639
attribute: https://www.jpcert.or.jp/english/at/2020/at200044.html
attribute: jvn
attribute: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
attribute: 9.8
object: vulnerability
attribute: 2020-12-14T03:15:00.000000+0000
attribute: 2020-12-15T21:21:00.000000+0000
attribute: nvd
attribute: https://www.soliton.co.jp/support/2020/004278.html
attribute: https://jvn.jp/jp/JVN12884935/index.html
attribute: https://jvn.jp/en/jp/JVN12884935/index.html

Hover target

Event: 70554
Info: CVE-2018-0694
Date: 2019-12-02
Analysis: Completed
Org: Cyber-Trust
Actions
Go to event
Expand (ctrl+x)

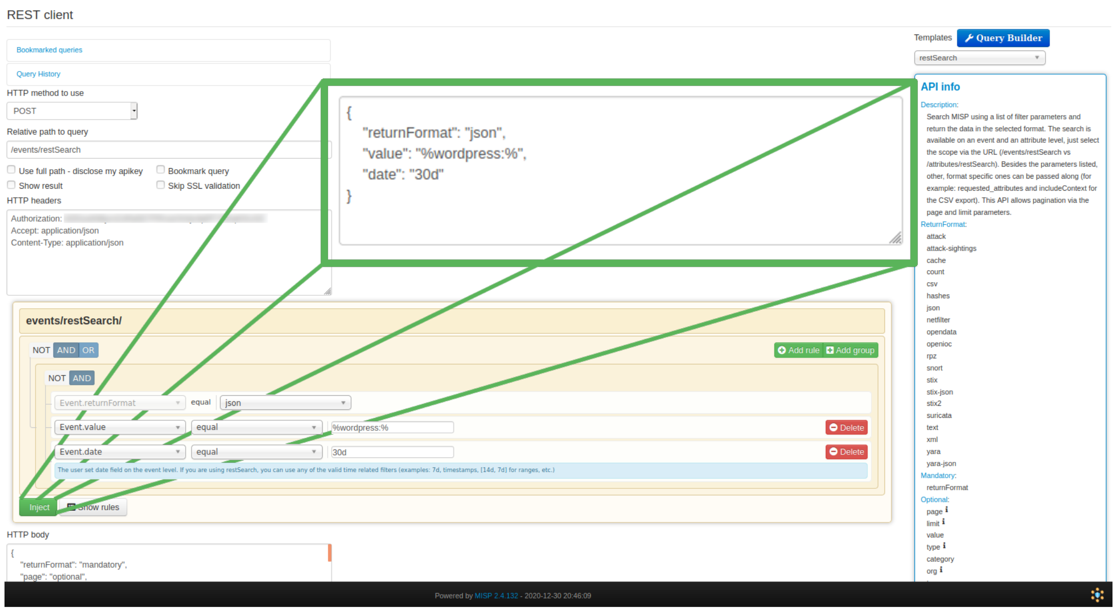
Selected

Attribute: 7313669
Name: cpe:2.3:a:soliton:filezen:*:*:*:*:*:*:*:*
Category:
Type: attribute
Comment:
Actions

Figure 4.10: **ASPIS** Event Correlations Graph Inspection

4.9.4 CTI Sharing: MISP REST API

With regard to the CTI sharing capabilities of **ASPIS**, it utilizes the MISP REST API, to enable users share and retrieve any information that is stored in the system's database. To facilitate the process of CTI sharing, MISP provides a GUI that enables users formulate their queries; namely the MISP *Query Builder*. Specifically, MISP Query Builder provides various *templates* to best support each user's querying needs (i.e., Search based on events' metadata or events' attributes). After formulating their queries, by adding the corresponding filtering rules, users can *inject* that set of rules to an HTTP request. In example, in Figure 4.11, a user is willing to retrieve or share any CTI artifacts that regard *Wordpress* and are discovered within the past month. Next, users can execute the formulated query to the MISP REST API, to verify its validity in terms of the MISP REST API querying syntax. After success, as presented in Figure 4.12, users are provided with two possible ways of querying MISP, without using the provided GUI; through cURL or PyMISP. Following the aforementioned example, with simple PyMISP scripting, **ASPIS** users are able to request from MISP any CTI artifact that regards *Wordpress*, discovered within the past month, such as the CPE, CVSS score and description, in order to stay alert over cyber-threats that may concern them (Figure 4.13).



HTTP method to use: POST

Relative path to query: /events/restSearch

☐ Use full path - disclose my apikey ☐ Bookmark query
☐ Show result ☐ Skip SSL validation

HTTP headers

```
Authorization: [redacted]
Accept: application/json
Content-Type: application/json
```

```
{
    "returnFormat": "json",
    "value": "%wordpress:%",
    "date": "30d"
}
```

events/restSearch/

Event.returnFormat	equal	json
Event.value	equal	%wordpress:%
Event.date	equal	30d

The user set date field on the event level. If you are using restSearch, you can use any of the valid time related filters (examples: 7d, timestamps, [14d, 7d] for ranges, etc.)

HTTP body

```
{
    "returnFormat": "mandatory",
    "page": "optional",
```

Templates: restSearch

API info

Description:
Search MISP using a list of filter parameters and return the data in the selected format. The search is available on an event and an attribute level, just select the scope via the URL (/events/restSearch vs /attributes/restSearch). Besides the parameters listed, other, format specific ones can be passed along (for example: requested_attributes and includeContext for the CSV export). This API allows pagination via the page and limit parameters.

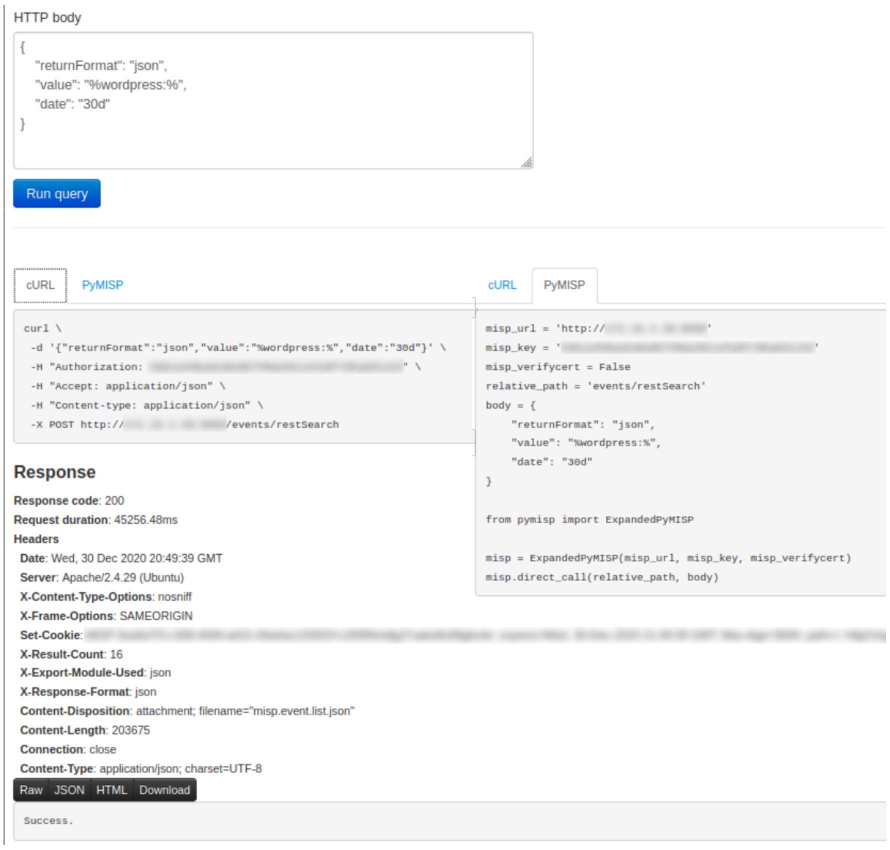
ReturnFormat:
attack
attack-sightings
cache
count
csv
hashes
json
netfilter
opendata
openioc
rpz
snort
stix
stix-json
stix2
suricata
text
xml
yara
yara-json

Mandatory:
returnFormat

Optional:
page ℹ
limit ℹ
value
type ℹ
category
org ℹ

Powered by MISP 2.4.132 - 2020-12-30 20:46:09

Figure 4.11: MISP Query Builder in Action


HTTP body

```
{
    "returnFormat": "json",
    "value": "%wordpress:%",
    "date": "30d"
}
```

```
curl \
  -d '{"returnFormat":"json","value":"%wordpress:%","date":"30d"}' \
  -H "Authorization: [redacted]" \
  -H "Accept: application/json" \
  -H "Content-type: application/json" \
  -X POST http://[redacted]/events/restSearch
```

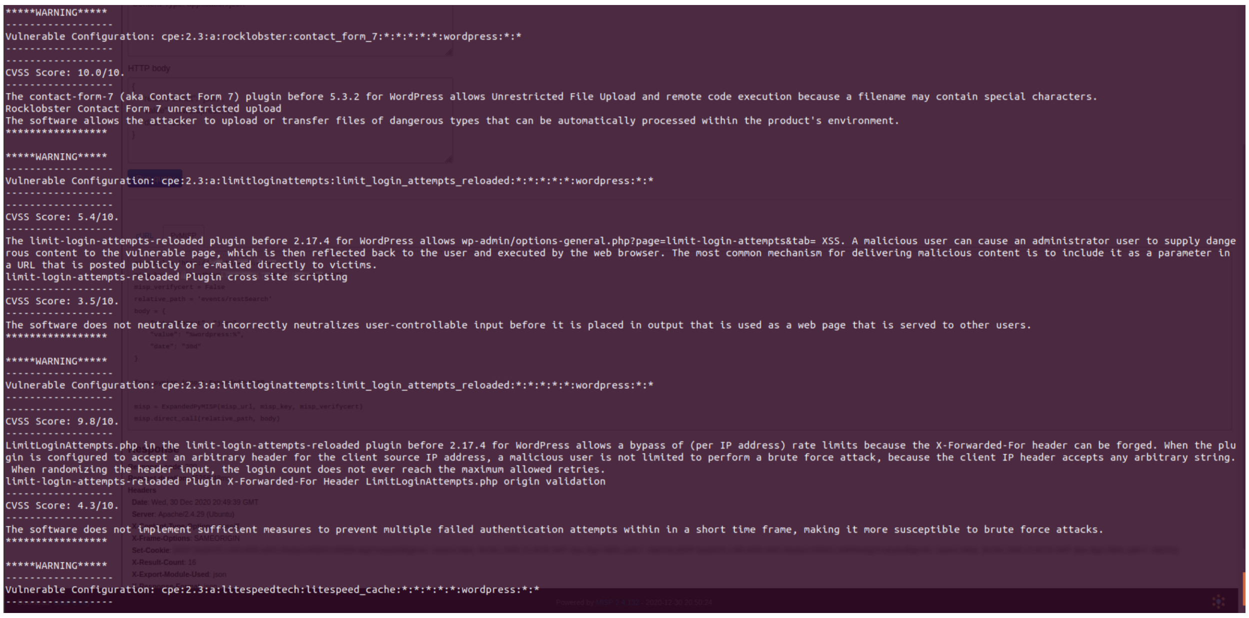
```
misp_url = 'http://[redacted]'
misp_key = '[redacted]'
misp_verifycert = False
relative_path = 'events/restSearch'
body = {
    "returnFormat": "json",
    "value": "%wordpress:%",
    "date": "30d"
}

from pymisp import ExpandedPyMISP

misp = ExpandedPyMISP(misp_url, misp_key, misp_verifycert)
misp.direct_call(relative_path, body)
```

Response

Response code: 200
Request duration: 45256.48ms
Headers
Date: Wed, 30 Dec 2020 20:49:39 GMT
Server: Apache/2.4.29 (Ubuntu)
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Set-Cookie: [redacted]
X-Result-Count: 16
X-Export-Module-Used: json
X-Response-Format: json
Content-Disposition: attachment; filename="misp.event.list.json"
Content-Length: 203675
Connection: close
Content-Type: application/json; charset=UTF-8

```
Success.
```

Figure 4.12: MISP REST API: cURL Querying & PyMISP Querying



```
*****WARNING*****
-----------------
Vulnerable Configuration: cpe:2.3:a:rocklobster:contact_form_7:*:*:*:*:*:wordpress:*:*
-----------------
-----------------
CVSS Score: 10.0/10.
-----------------
The contact-form-7 (aka Contact Form 7) plugin before 5.3.2 for WordPress allows Unrestricted File Upload and remote code execution because a filename may contain special characters.
Rocklobster Contact Form 7 unrestricted upload
The software allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment.
*****************

*****WARNING*****
-----------------
Vulnerable Configuration: cpe:2.3:a:limitloginattempts:limit_login_attempts_reloaded:*:*:*:*:*:wordpress:*:*
-----------------
-----------------
CVSS Score: 5.4/10.
-----------------
The limit-login-attempts-reloaded plugin before 2.17.4 for WordPress allows wp-admin/options-general.php?page=limit-login-attempts&tab= XSS. A malicious user can cause an administrator user to supply dange
rous content to the vulnerable page, which is then reflected back to the user and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in
a URL that is posted publicly or e-mailed directly to victims.
limit-login-attempts-reloaded Plugin cross site scripting
-----------------
CVSS Score: 3.5/10.
-----------------
The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.
*****************

*****WARNING*****
-----------------
Vulnerable Configuration: cpe:2.3:a:limitloginattempts:limit_login_attempts_reloaded:*:*:*:*:*:wordpress:*:*
-----------------
-----------------
CVSS Score: 9.8/10.
-----------------
LimitLoginAttempts.php in the limit-login-attempts-reloaded plugin before 2.17.4 for WordPress allows a bypass of (per IP address) rate limits because the X-Forwarded-For header can be forged. When the plu
gin is configured to accept an arbitrary header for the client source IP address, a malicious user is not limited to perform a brute force attack, because the client IP header accepts any arbitrary string.
 When randomizing the header input, the login count does not ever reach the maximum allowed retries.
limit-login-attempts-reloaded Plugin X-Forwarded-For Header LimitLoginAttempts.php origin validation
-----------------
CVSS Score: 4.3/10.
-----------------
The software does not implement sufficient measures to prevent multiple failed authentication attempts within in a short time frame, making it more susceptible to brute force attacks.
*****************

*****WARNING*****
-----------------
Vulnerable Configuration: cpe:2.3:a:litespeedtech:litespeed_cache:*:*:*:*:*:wordpress:*:*
-----------------
```

Figure 4.13: **ASPIS** CTI for Wordpress Plugins Vulnerabilities of December 2020

4.10 System Installation

The **ASPIS** system installation is a process that can be achieved in three steps. First, a MISP docker should be setup on the user's machine, according to the instructions found in MISP docker's github (`https://github.com/MISP/misp-docker`). After finalizing the MISP docker setup, the user must navigate to the project folder and copy the *ASPIS_scripts* folder to the same machine that hosts the MISP docker. The Python scripts found in this folder support the functionality described in Sections 4.5-4.7. Next, the user should connect to the MISP instance and navigate to <misp_url>/events/automation, to copy the automatically generated authorization key and paste it into the aforementioned folder's *keys.py*, in the *misp_key* field. Then, the user should install a crontab to daily execute the *extract_from.sh* shell script, which executes the required Python scripts that monitor the identified CTI sources.

Chapter 5

Experimental Evaluation

In Section 4.3.2 we presented that the correlation engine of MISP stores redundant information in the *correlations* relation. To counter that, in Section 4.3.3 we proposed an alternative representation of the *correlations* relation, which we referred to as *correlationsNF*. In this chapter we evaluate **ASPIS** performance, with regard to the space and time requirements for storing the gathered CTI, as well as the queries' response time. To this end, we have conducted a series of experiments that are analyzed in the following sections.

5.1 Experimental Setup

In this section we will describe the setup of the experiments conducted, to acquire the measurements that evaluate the performance of the two **ASPIS** variations. The first variation, namely **ASPIS**$_{\mathbf{D}}$, utilizes the default MISP correlation engine. The second, **ASPIS**$_{\mathbf{A}}$, implements the proposed alteration of the MISP correlation engine, as described in Section 4.3.3. The experimental evaluation aims to examine both space and time requirements for storing CTI, and queries' response times of the two **ASPIS** variations. The experiments will review the effects of the complexity of the multiple join operations, and the redundancy elimination of the stored information, implemented in **ASPIS**$_{\mathbf{A}}$, in comparison with **ASPIS**$_{\mathbf{D}}$. In the following, we will discuss the utilized data set, the generated query sets, the metrics employed in our evaluation and the technical configuration of the machine that was used for the experiments.

5.1.1 Data Set

The data set that is utilized for the purposes of the experimental evaluation, contains all vulnerability entries, enlisted in the NVD dataset. Specifically, the data set consists of all yearly JSON data feeds provided by NVD, until CVE-2020, as they're offered in [35]. For reference purposes, in the rest of this chapter the data set will be noted as `DATA`. The utilized elements of the JSON data feeds are presented in Section 4.5.1.

`DATA` stores information of ∼158K CVE entries, that contain ∼307K *objects* (approximately two *objects* per CVE entry, namely *vulnerability* and *weakness*, as described in Section 4.3.1), and ∼3.8M *attributes*.

5.1.2 Query Sets

To evaluate the performance of the two **ASPIS** variations, we need to construct a query load that is representative of the intended usage. In a real use-case scenario, queries may use a CVE ID, a publication date and/or a CPE, to request matching CTI entries and their related correlations. Queries that request entries with a high number of correlations, are considered as *expensive queries*. Specifically, the response time of such queries is increased proportionally to the number of correlations of the requested entries.

To construct the query sets for the experimental evaluation, we first store the vulnerability entries of the input dataset `DATA`, in **ASPIS**, in batches of $40K$ entries, chronologically (with regard to their publication dates). Storing the vulnerability entries in **ASPIS** results in the calculation and generation of correlations for each stored entry. Then, we sample the stored CTI, to generate three query sets for each batch of $40K$ stored entries, to be provided with a dynamic view of the query response times, in a constantly growing database. Thus, for $40K$ vulnerability entries indexed and correlated by **ASPIS**, we are resulted with $C_{\text{INT}}(40K)$, $C_{\text{RAND}}(40K)$, $C_{\text{LOW}}(40K)$; the three generated query sets. Similarly, for $80K$ vulnerability entries indexed and correlated, we have $C_{\text{INT}}(80K)$, $C_{\text{RAND}}(80K)$, $C_{\text{LOW}}(80K)$, and so on. Each query of the generated query sets basically contains one CVE ID, and it requests the corresponding stored CTI entry and its related correlations, from **ASPIS**. Particularly, the formulated queries are generated from three CVE ID lists (namely, C_{INT}, C_{RAND} and C_{LOW}), and distinguished into three query sets, with regard to the number of correlations related to each corresponding CTI entry. The CVE ID lists' generation methodology is described in the following.

$C_{\textbf{INT}}$ consists of the 20% CVE IDs of the indexed entries of **ASPIS**, that have the higher number of correlations.

$C_{\textbf{RAND}}$ consists of all CVE IDs of the indexed entries of **ASPIS**.

$C_{\textbf{LOW}}$ consists of the 20% CVE IDs of the indexed entries of **ASPIS**, that have the lower number of correlations (having at least one correlation).

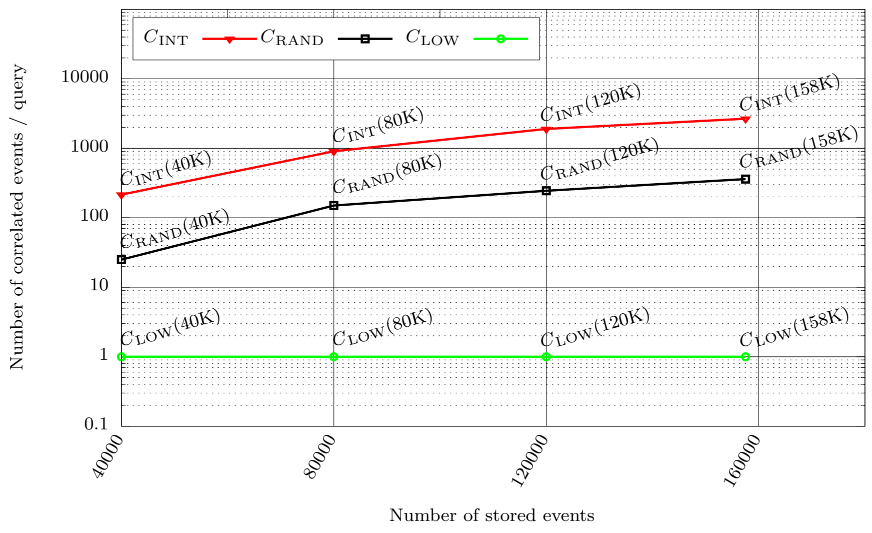


Figure 5.1: Query sets' average correlations per query proportionally to the stored events

Query set	**`AVG`(correlations per query)**
$C_{\text{INT}}(40K)$	215
$C_{\text{RAND}}(40K)$	25
$C_{\text{LOW}}(40K)$	1
$C_{\text{INT}}(80K)$	905
$C_{\text{RAND}}(80K)$	150
$C_{\text{LOW}}(80K)$	1
$C_{\text{INT}}(120K)$	1902
$C_{\text{RAND}}(120K)$	245
$C_{\text{LOW}}(120K)$	1
$C_{\text{INT}}(158K)$	2660
$C_{\text{RAND}}(158K)$	360
$C_{\text{LOW}}(158K)$	1

Table 5.1: Query sets' average correlations per query

In Figure 5.1 and Table 5.1 we provide an overview of the resulting *correlations per query* distribution for each query set. To calculate that, we produced a query load of 100 queries for each query set. Each query requested from **ASPIS** the total number of correlations per CVE ID searched. Thus, we were able to extract an average *correlations per query* distribution for each query set. Finally, the numbers presented in Figure 5.1 and Table 5.1 are averaged over 10 executions of the aforementioned methodology. E.g., for $\sim 158K$ stored CVE entries, a highly correlated CVE entry would return 2660 correlations on average ($C_{\text{INT}}(158K)$), while a random CVE entry would be intercorrelated with 360 entries ($C_{\text{RAND}}(158K)$). Finally, while the generated query sets of C_{INT} and C_{RAND} provide an adequate overview of the *expensive queries'* response times, C_{LOW} query sets provide an overview of the minimal effect of the *correlations* relation over the *expensive queries'* response time, since they request one correlation per CTI entry, which is the minimal number required to examine the effect of the *correlations* relation over the CTI querying times.

5.1.3 Configuration Parameters and Metrics Employed

To evaluate **ASPIS**, we evaluate the required space for storing the incoming CTI, and the time required for the CTI storing and processing and the CTI querying tasks.

5.1.3.1 CTI Storing Space Requirements

Following, we compare the space required by **ASPIS**$_\mathbf{A}$ and **ASPIS**$_\mathbf{D}$, to store the incoming CTI. Since **ASPIS**$_\mathbf{A}$ and **ASPIS**$_\mathbf{D}$ only differ in the *correlations* relation, we focus on the space required to store this relation. Specifically, to achieve that comparison, we study the *correlations* table size, with respect to the *data* and *index size* (in MB). That information is provided by the MySQL *information* schema [32]. The aforementioned metrics presented in this section, are monitored for each step of 100 CVEs insertions or modifications that occured, for both configurations.

5.1.3.2 CTI Storing and Processing Time

As described in Section 4.5, **ASPIS** gathers CTI from the monitored sources daily. So, it is important for the system to be able to process the incoming CTI in less than $24h$ to function properly, because otherwise it would not be able to keep up with the sources' CTI dissemination rate. The processing and storing of the incoming CTI is mainly affected by the number of correlations of the events. Thus, the more events that are being stored in the system, the more correlations are going to be generated, which leads to an increase in the incoming CTI processing and storing time. So, the first metric that we are going to monitor is the *CTI storing and processing time*. This metric is the wall-clock time elapsed by the moment the system begins processing the gathered CTI until it is stored, indexed and correlated by **ASPIS**.

As mentioned in Section 5.1.1, `DATA` organizes all CVE entries by the year they were discovered. So, to monitor that metric in a realistic manner, we simulate the daily execution of the CTI storing task, by using the `publication` and the `modification dates` provided by `DATA`. We use both `dates` to cumulatively count the total time required for each day's CTI publications' indexing by **ASPIS**. The daily execution simulation of the CTI storing task is described in Algorithm 1.

5.1.3.3 CTI Querying Response Time

The final metric we compare is the CTI querying response time of **ASPIS**. Specifically, we query both variations, using the query sets presented in Section 5.1.2, and compare the resulting *average query response time* for each query set. Furthermore, we measure the same metric for **ASPIS**, at the same time it executes the CTI

Algorithm 1: Daily CTI storing task simulation

```
Result: The daily time required by ASPIS for handling incoming CTI
daily_storing_time = {};
/*daily_storing_time is a list that stores the simulated elapsed time for each date*/
for each CVE_entry in DATA do
    start_time = time.now();
    Store or update CVE_entry and correlate with existing entries of ASPIS;
    elapsed_time = time.now() - start_time;
    if CVE_entry.publication_date = CVE_entry.modification_date then
        /*CVE_entry was never modified after publication*/
        daily_storing_time[CVE_entry.publication_date] += elapsed_time;
    else
        /*CVE_entry was published on publication_date and modified on
         modification_date*/
        daily_storing_time[CVE_entry.publication_date] += elapsed_time;
        daily_storing_time[CVE_entry.modification_date] += elapsed_time;
    end
end
```

storing task, to capture the effects it would have over the CTI queries' response times, providing an overview of a realistic use-case scenario, where users can query the system's CTI at any given point in time. As mentioned in Section 5.1.3.2, during the CTI storing task, **ASPIS** handles incoming CTI and recalculates correlations, which are tasks that occupy the system's resources, burdening its performance. To this end, for both experiments we count the wall-clock time required for the queries' responses. Finally, while the first experiment is conducted for all query sets, the second is conducted for the last two triplets of query sets of Section 5.1.2 (for $120K$ and $158K$ indexed entries in **ASPIS**), to capture the *average query response time* on instances that already contain high amounts of correlations, which makes the CTI storing a challenging task for the system.

5.1.4 Technical Configuration

For the experiments conducted, in order to get the measurements that evaluate the two variations of **ASPIS**, we use an off-the-rack PC with a Xeon W-1250 $3.3GHz$ CPU, a total of $16GB$ RAM capacity ($2 \times 8GB$ DDR-4 $3200MHZ$) and a $512GB$ NVMe SSD (Read & Write Speeds: $3GB/s$ & $1.8GB/s$), running Ubuntu Linux 18.04. The results of each experiment presented in Section 5.2 are averaged over 10 runs.

5.2 Experiment Results

In this section, we review the experiment results for the CTI storing and querying space and time requirements of **ASPIS**. The experiments conducted, can be divided in three parts. The first regards the CTI storing space requirements of **ASPIS**. The second part regards the CTI storing time requirements of the system. Finally, the third part regards the CTI queries' response times of the system. All experiments are conducted for both **ASPIS**$_{\mathbf{A}}$ and **ASPIS**$_{\mathbf{D}}$ variations, to compare them in terms of overall space and time requirements, for the tasks of CTI storing and CTI querying.

5.2.1 CTI Storing Space Requirements

Regarding the system's space requirements for the task of CTI storing, we compare the *correlations* relation's required space of **ASPIS**$_{\mathbf{A}}$ and **ASPIS**$_{\mathbf{D}}$ variations, used to store the corresponding information. Particularly, as stated in Section 5.1.3.1, we monitor the *correlations* table size of the two variations, with respect to the *data* and *index size* (in MB). Figure 5.2 illustrates how the two aformentioned metrics increase for both variations, with regard to the number of stored CTI entries. Specifically, the x axis of the illustrated graphs represents the number of unique CTI entries, stored in the system, while the y axis indicates the storage space that was used to store the *correlations* relation's information in the database.

Reviewing the results, we observe that for **ASPIS**$_{\mathbf{D}}$, the occupied space escalates exponentially with regard to the number of the stored CTI entries, while in **ASPIS**$_{\mathbf{A}}$, it increases logarithmically after $\sim 50K$ entries. This difference is entirely attributed to the redesign of the *correlations* relation discussed in Section 4.3.3. Our proposal optimizes storage requirements, reducing it by an order of magnitude. Specifically, for $\sim 158K$ CTI entries, **ASPIS**$_{\mathbf{D}}$ required $\sim 40GB$, and **ASPIS**$_{\mathbf{A}}$ required $\sim 350MB$.

5.2.2 CTI Storing and Processing Time

Regarding the CTI storing and processing task's execution times, as stated in Section 5.1.3.2, we monitor the *daily* and the *monthly average CTI storing and processing time*. For the *daily CTI storing and processing time*, we simulated the daily execution of the CTI storing task, by using the `publication` and `modification dates` provided by the input dataset, `DATA`, according to the simulation methodology presented in Algorithm 1. With regard to the *monthly average CTI storing and processing time*, we monitor a moving average for a window of 30 *days*, over the stored dates and times provided by the *daily CTI storing and processing time*.

To this end, Figure 5.3 presents the simulations' elapsed daily times for storing

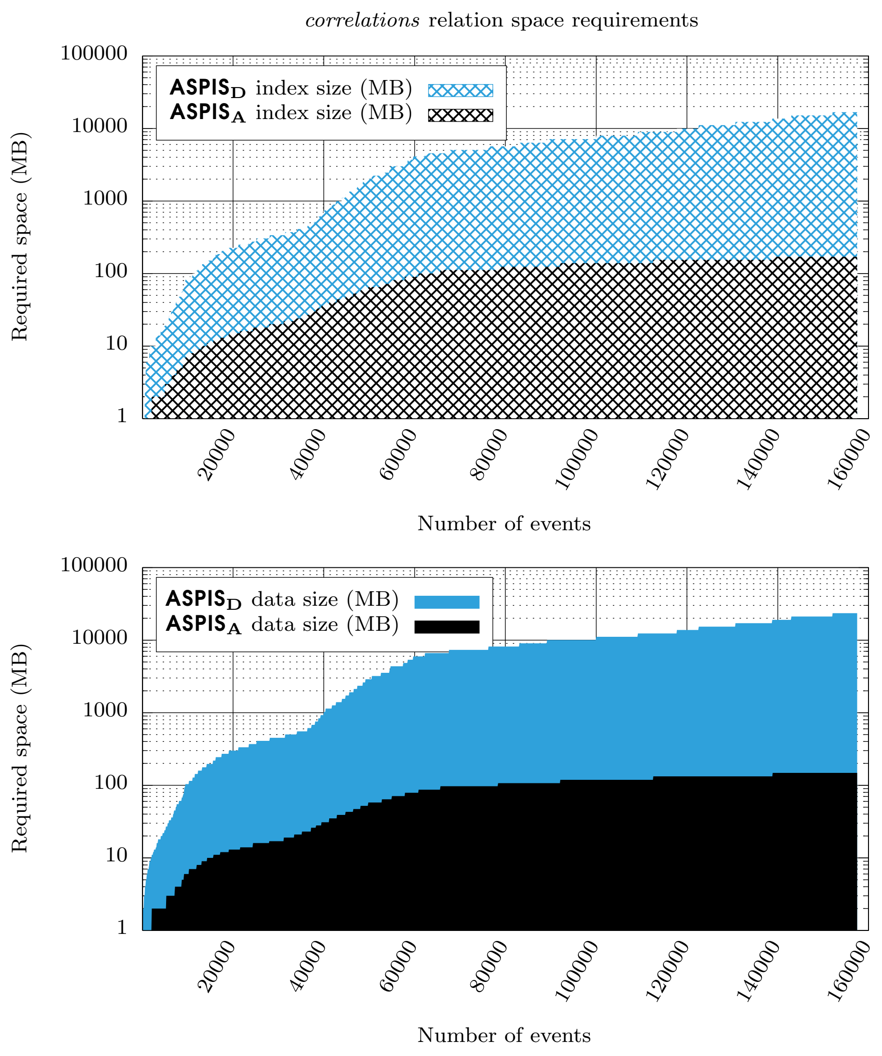
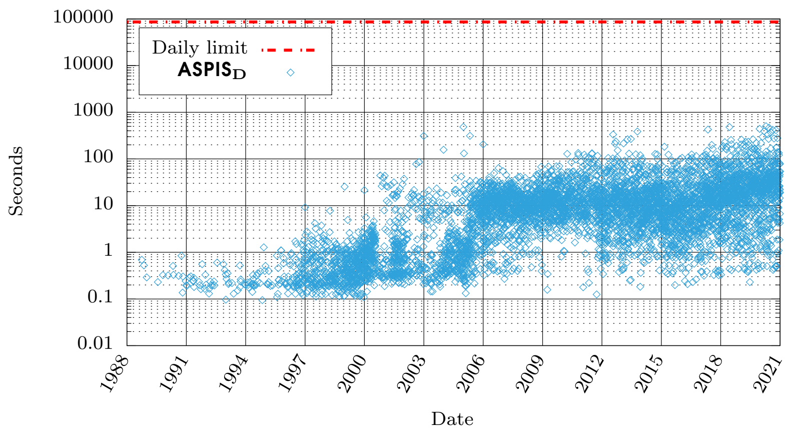


Figure 5.2: **ASPIS**$_A$ and **ASPIS**$_D$ CTI storing space requirements

the CTI for both **ASPIS**$_D$ and **ASPIS**$_A$. To provide a clearer view of the elapsed times for the CTI gathering and storing, we present the calculated monthly moving average of the daily times, in Figure 5.4. From that, we realize that the MISP correlation engine alteration implemented in **ASPIS**$_A$, requires approximately 150% more CTI processing and storing time, than the default correlation engine of **ASPIS**$_D$. This occurs due to the internal correlation mechanism of MISP, which reproduces the default *correlations* relation through multiple join operations over *correlationsNF*, each time it stores new CTI, to calculate and store new CTI correlations, as described in Section 4.3.3 and Section 4.3.3.2. However, it doesn't interfere with the system's functionality, since it doesn't exceed the CTI processing and storing time margin of $24h$.



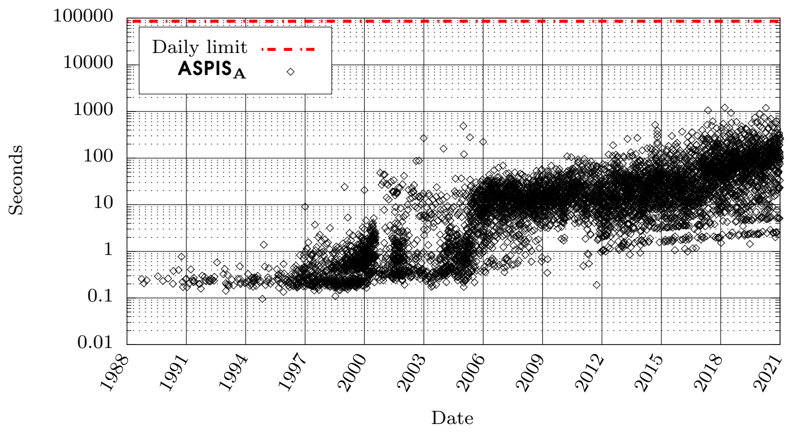


Figure 5.3: Daily time elapsed for the CTI gathering and storing

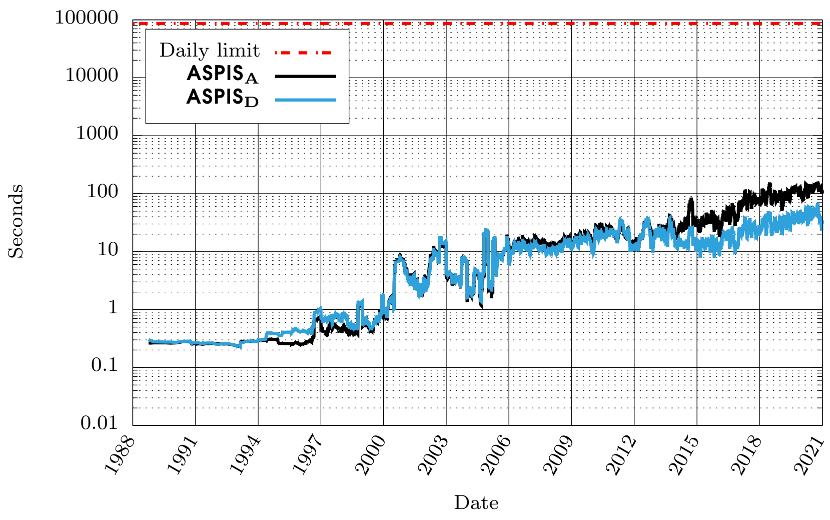
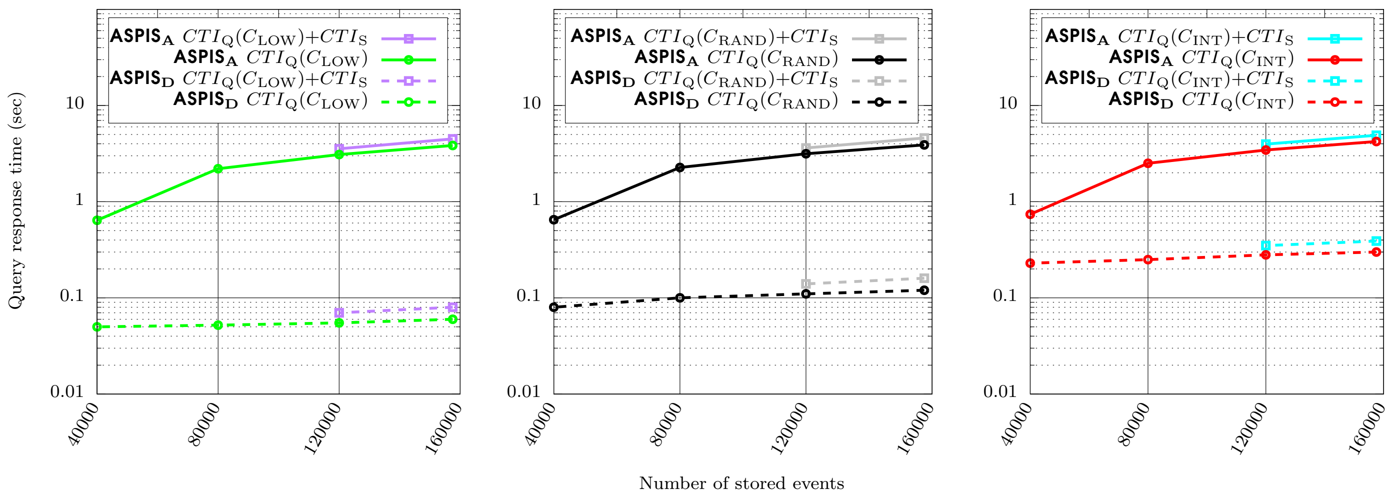


Figure 5.4: Monthly average time elapsed for the CTI gathering and storing



(a) C_{LOW} query sets **(b)** C_{RAND} query sets **(c)** C_{INT} query sets

Figure 5.5: **ASPIS$_D$** and **ASPIS$_A$** CTI querying average response times

5.2.3 CTI Querying Response Time

Using the query sets presented in Section 5.1.2, we compare the resulting *average query response times* of the two variations of **ASPIS**, in seconds, for each query set. To this end, Figure 5.5 presents the resulting *average query response times* for each query set, for both variations. Particularly, Figure 5.5a presents the resulting *average query response times* of **ASPIS$_A$** and **ASPIS$_D$**, for queries that request CTI entries, with a low amount of correlations, while the results illustrated in Figure 5.5b and Figure 5.5c refer to the queries that request random CTI entries and CTI entries with a high number of correlations, respectively. The resulting times are provided in relation to the number of stored events in **ASPIS**, and as realized, the number of the stored events and their corresponding correlations affect the system's CTI querying performance. Furthermore, we realize that queries that request CTI entries with a higher amount of correlations, tend to result in longer response times, compared to the queries that request CTI entries with lower numbers of correlations. Finally, Figure 5.5 presents how the CTI querying response times are affected by the simulatenous execution of the CTI storing task. Particularly, for each examined query set, the CTI storing task results in ~20% increased querying response times.

Chapter 6

Conclusions and Future Work

In the previous chapter, we presented the **ASPIS** system's experimental evaluation, comparing the two correlation engines; the one provided by MISP, and the proposed alteration. In this chapter, we sum up the thesis and conclude the experimental evaluation results. Finally, we discuss our future plans regarding both the extensibility and the longevity of **ASPIS**.

6.1 Summary and Conclusions

In this thesis we put our focus on facilitating the CTI life-cycle, by utilizing the appropriate open-source tools, for automating the CTI gathering and sharing tasks. Specifically, this thesis presented **ASPIS**, an Automated Source Pairing Intelligence System, which stores valuable cyber-security-oriented information in an eVDB, for the CTI gathering and sharing. With the development and automation of crawling and parsing procedures, **ASPIS** manages to monitor five different sources; NVD, JVN, KB-Cert, VulDB and Exploit-DB, and retrieve CTI daily. The proposed CTI sharing platform for the purposes of the work carried out in this thesis, was MISP. With the utilization of the PyMISP library and the use of Python scripting, **ASPIS** manages to fully automate the CTI gathering task. Particularly, through the proposed architecture, the system is able to gather and store CTI from different sources, in a unified manner. At the same time it enables its users to complete the CTI life-cycle, by providing all required functionalities that regard the CTI sharing and reviewing.

Regarding the CTI analysis, we used the MISP correlation engine, to provide additional context to the stored CTI. Characteristically, our proposed system uses the MISP correlation engine, to generate correlations between the stored CTI, with regard to the vulnerable tools, platforms, and hardware, providing connections between related vulnerability events. However, when we put the default MISP correlation engine to use, we realized that it led to space deficiency issues, due to its non-optimized database schema. To counter that, we proposed an alternative correlation engine schema, following the standard BCNF decomposition methodology.

Finally, through the experimental evaluation conducted, we presented that **ASPIS** not only provides a sufficient solution to the space deficiency issues of the MISP correlation engine, but it also consists a sustainable solution for the tasks of CTI gathering, storing and sharing.

6.2 Future Work

In this section we discuss the possibility of future extensions of **ASPIS**, and also the available directions that we plan to focus on in our future work.

Our future plans focus on automating and enhancing the complete CTI life-cycle, and optimizing the system's procedures by:

- automating the process of source identification, which may be achieved with the use of focused crawling techniques, which are able to target CTI-related content,
- providing a ranking methodology for the credibility of the identified sources,
- utilizing more flexible data structuring techniques, in order to be able to dynamically enrich existing knowledge with additional context,
- establishing a more user-friendly UI, for the purposes of the contents' presentation, along with a user-friendly publish/subscribe mechanism,
- extending the MISP correlation engine with additional functionality, to provide supplementary context to the correlated stored information,
- optimizing the query execution for **ASPIS**$_\mathbf{A}$, by fine-tuning the MISP internal code.

Appendix A

Examples of MISP Objects in JSON Format

A.1 NVD: Vulnerability Object

```
[
  {"id": {"value": "CVE-2020-5659"}},
  {"vulnerable_configuration":
    {"value": "cpe:2.3:a:riken:xoonips:*:*:*:*:*:xoops:*:*"}
  },
  {"published":
    {"value": "2020-11-16T05:15:00.000000+0000"}
  },
  {"modified":
    {"value": "2020-11-20T14:54:00.000000+0000"}
  },
  {"cvss-score": {"value": 8.8}},
  {"cvss-string":
    {"value": "CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"}
  },
  {"references":
    {"value": "https://jvn.jp/en/vu/JVNVU92053563/index.html"},
    {"comment": "[‘Third Party Advisory’]"}
  },
  {"references":
    {"value":
   "https://xoonips.osdn.jp/modules/news/index.php?page=article&
storyid=13"},
    {"comment": "[‘Release Notes’, ‘Vendor Advisory’]"}
  },
  {"description":
    {"value": "SQL injection vulnerability in the XooNIps 3.49 and
   earlier allows remote authenticated attackers to execute
   arbitrary SQL commands via unspecified vectors."}
  },
  {"credit":  {"value": "nvd"}}
]
```

A.2 NVD: Weakness Object

```
[
  {"id": {"value": "CWE-89"}},
  {"name":
    {"value": "Improper Neutralization of Special Elements used in
   an SQL Command ('SQL Injection')"}
  },
  {"description":
    {"value": "The software constructs all or part of an SQL
   command using externally-influenced input from an upstream
   component, but it does not neutralize or incorrectly
   neutralizes special elements that could modify the intended SQL
   command when it is sent to a downstream component."}
  }
]
```

A.3 JVN: Vulnerability Object

```
[
  {"id": {"value": "CVE-2020-5659"}},
  {"vulnerable_configuration":
    {"value": "cpe:/a:riken:xoonips"}
  },
  {"published":
    {"value": "2020-11-09T06:10:44.000000+0000"}
  },
  {"modified":
    {"value": "2020-11-09T06:10:44.000000+0000"}
  },
  {"cvss-score": {"value": 6.3}},
  {"cvss-string":
    {"value": "CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L"}
  },
  {"references":
    {"value":
   "https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5659"}
  },
  {"references":
    {"value":
   "https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5662"}
  },
  {"references":
    {"value":
   "https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5663"}
  },
  {"references":
    {"value":
   "https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5664"}
  },
  {"references":
    {"value": "https://jvn.jp/en/vu/JVNVU92053563/"}
  },
  {"description":
    {"value": "XOOPS module "XooNIps" contains multiple
   vulnerabilities listed below. * SQL injection (CWE-89) -
   CVE-2020-5659 * Reflected cross-site scripting (CWE-79) -
   CVE-2020-5662 * Stored cross-site scripting (CWE-79) -
   CVE-2020-5663 * Deserialization of untrusted data (CWE-502) -
   CVE-2020-5664 stypr of Flatt Security Inc. reported this
   vulnerability to the developer and coordinated on his own.
   After coordination was completed, this case was reported to
   JPCERT/CC, and JPCERT/CC coordinated with the developer for the
   publication."}
  },
  {"credit":  {"value": "jvn"}}
]
```

A.4 KB-Cert: Vulnerability Object

```
[
  {"id": {"value": "CVE-2019-9507"}},
  {"vulnerable_configuration":
    {"value": "cpe:2.3:a:riken:xoonips:*:*:*:*:*:xoops:*:*"}
  },
  {"published":
    {"value": "2019-04-12T00:00:00+00:00"}
  },
  {"modified":
    {"value": "2020-03-30T17:38:00+00:00"}
  },
  {"summary":
    {"value": "The Vertiv Avocent Universal Management Gateway
   Model UMG-4000 is a data center management appliance. The web
   interface of the UMG-4000 is vulnerable to command injection,
   stored cross-site scripting (XSS), and reflected XSS, which may
   allow an authenticated attacker with administrative privileges
   to remotely execute arbitrary code."}
  },
  {"description":
    {"value": "The Vertiv Avocent UMG-4000 contains multiple
   vulnerabilities that could allow an authenticated attacker with
   administrative privileges to remotely execute arbitrary code.
   The web interface does not sanitize input provided from the
   remote client, making it vulnerable to command injection,
   stored cross-site scripting, and reflected cross-site
   scripting. CVE-2019-9507 - CWE-95 ... \nImpact: An
   authenticated remote attacker could inject arbitrary scripts or
   persistently store malicious scripts on the web server that
   could be used to collect and exfiltrate sensitive
   information.\nSolution: Apply an update Vertiv Avocent has
   addressed these issues in the below versions: Non-Trellis
   customers are encouraged to install Universal Management
   Gateway firmware version 4.2.2.21 or higher to address these
   vulnerabilities, located here. Trellis users of the Universal
   Management Gateway running firmware version 4.2.0.23 that are
   operating Trellis versions 5.0.2 through 5.0.6 should install
   the update patch located here. Trellis users of the Universal
   Management Gateway that are operating Trellis versions 5.0.6
   and later should install Universal Gateway firmware version
   4.3.0.23 located here.\nAcknowledgements: This document was
   written by Laurie Tyzenhaus."}
  },
  {"cvss-string":
    {"value": "AV:N/AC:M/Au:S/C:C/I:C/A:C"},
    {"comment": "Temp [E:POC/RL:OF/RC:C] / Env [
   CDP:ND/TD:ND/CR:ND/IR:ND/AR:ND]"}
  },
  {"cvss-score": {"value": 8.5}, {"comment": "Temp [6.7] / Env [
   6.7]"}},
  {"references":
    {"value": "https://jvn.jp/en/vu/JVNVU92053563/index.html"},
    {"comment": "[‘Third Party Advisory’]"}
  },
  {"references":
    {"value":
   "https://www.vertiv.com/globalassets/documents/firmware/
universal-management-gateway-release-notes-v4.3.0.23_vertiv_
update.pdf"}
  },
  {"references":
    {"value":
   "https://www.vertiv.com/en-us/support/software-download/
software/trellis-enterprise-and-quick-start-solutions-software-
downloads/"}
  },
  {"references":
    {"value": "https://cwe.mitre.org/data/definitions/95.html"}
  },
  {"references":
    {"value": "https://cwe.mitre.org/data/definitions/79.html"}
  },
  {"references":
    {"value":
   "https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)#
Stored_and_Reflected_XSS_Attacks"}
  },
  {"credit": {"value": "kbcert"}}
]
```

A.5 VulDB: Vuldb-Vulnerability Object

```
[
  {"id": {"value": "CVE-2020-35364"}},
  {"summary":
    {"value": "Beijing Huorong Internet Security 5.0.55.2
   injection"}
  },
  {"published":
    {"value": "2020-27-12T10:05:00.000000+0000"}
  },
  {"vuldb-link":
    {"value": "https://vuldb.com/?id.166845"}
  },
  {"zeroday-price": {"value": "$5k-$25k"}},
  {"current-price": {"value": "$0-$5k"}},
  {"remediation":
    {"value": "Not Defined"},
    {"comment": "https://vuldb.com/?advisory_url.166845"}
  },
  {"exploitability":
    {"value": "Proof-of-Concept"},
    {"comment": "https://vuldb.com/?exploit_url.166845"}
  },
  {"cvss-score": {"value": 6.3}, {"comment": "VulDB [5.3] / NVD [
   8.8] "}},
  {"cvss-string-VDB":
    {"value": "AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N"}
  },
  {"cvss-string-NVD":
    {"value": "AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H"}
  },
  {"credit":  {"value": "vuldb"}}
]
```

A.6 Exploit-DB: ExpDB-PoC Object

```
[
  {"description":
    {"value": "Gitlab 11.4.7 - Remote Code Execution"}
  },
  {"references": {"value":
   "https://www.exploit-db.com/raw/49257"}},
  {"author": {"value": "Fortunato Lodari"}},
  {"credit": {"value": "expdb"}}
]
```

Appendix B

MISP Versions Changelog

MISP 2.4.95 (2018-09-06)

- The search API in MISP has been refactored to be consistent among the various export formats (JSON, XML, OpenIOC, Suricata, Snort, and the text export); particularly, regarding the filtering process. String searches are by default exact lookups, but the search API allows the use of "%" wildcards to perform substring searches.
- A complete REST client has been added in the MISP interface, to enable MISP users query the API from the instance at hand.
- A debug functionality has been added in any API query to quickly show the SQL queries performed.

MISP 2.4.96 (2018-10-09)

- All MISP export APIs have been unified into the restSearch APIs, with an improved query format.
- A pagination system has been introduced, allowing users to easily paginate over search result sets and limit the output.
- The search results in the MISP UI can be directly downloaded in any of the supported formats available in MISP.
- Event/attribute data fetching performance increased, with the use of an internal pagination and caching mechanism, which scales with the amount of memory given to the PHP process, and hence reducing the chance of running into memory limit issues.
- The freetext import is now delegated to a background process for large imports.

MISP 2.4.100 (2018-12-31)

- Improvements to the UI, API, import and export.
- Addition of a new query builder, available through the REST client interface, that facilitates users to create JSON queries.

MISP 2.4.101 (2019-01-20)

- Improvements to the UI, import and export.
- Enabling/Disabling correlations is now accessible when creating/modifying an attribute.

MISP 2.4.103 (2019-03-04)

- Improvements to the UI.
- Implementation of a new attribute filtering tool to the event view, that allows for complex filtering rules.

MISP 2.4.106 (2019-04-25)

- Performance improvements for events with large numbers of attributes and objects.

MISP 2.4.108 (2019-06-04)

- Added object_relation as a filter for both the event/attribute restSearch functions.

MISP 2.4.109 (2019-06-13)

- Added date as a new restSearch filter, with a variety of accepted syntax options, such as:
  - time ranges in the shorthand format (7d or 24h, etc.)
  - Timestamps
  - fallback parsing for other formats (2019-01-01, "fortnight ago", etc.)
  - date ranges using lists [14d, 7d]

MISP 2.4.112 (2019-08-02)

- New parameters added to attributes/restSearch to include additional context.

  - includeCorrelations: includes the correlations to other attributes (includes a light-weight event object with each attribute)
  - includeContext: includes the additional event fields in the attributes/restSearch results (in JSON format) (e.g. UUID)

- Added ”weakness” object. It describes a weakness through the Common Weakness Enumeration (CWE) format.

MISP 2.4.119 (2019-12-02)

- Enhanced database diagnostics with the integration of a new sub-system that compares the current state of the MISP database to the reference DB schema, highlighting potential issues or divergences. Additionally, it allows users to generate SQL queries that would rectify the potential issues.
- Improved timestamp filtering in MISP. It now provides 4 different timestamp filters on the following levels: event, attribute, attribute and event, and event publish.
- Added tracking of the API deprecations, warning users of their state.

MISP 2.4.120 (2020-01-21)

- Extended the data-model by adding first_seen and last_seen values at the attribute and object levels.
- Added the event timeline feature, through which users can (a) quickly view the overall timeline of attributes and objects within an event, (b) zoom in and out in the timeline, (c) edit and change the first_seen and last_seen by moving the attributes/objects directly on the timeline.

MISP 2.4.123 (2020-03-11)

- Added a dashboard system, which is accessible directly in MISP and fully customizable by its users.
- Users can select their home page, to land to any page in MISP

MISP 2.4.127 (2020-06-19)

- Added event ID to the page table.
- Refactored correlation saving to (a) always show other correlating value, (b) make correlation saving faster (by moving more work to database, and not fetching unnecessary fields), (c) fixed minor bugs.

MISP 2.4.129 (2020-07-28)

- Merge events functionality improved to (a) correctly handle objects, tags and sharing groups, (b) be enabled through API (which will directly merge all contents of the source event into the target event).

MISP 2.4.130 (2020-08-21)

- Speed improvements:
  - Updating of correlations in one query. (Before, for every event saving action, four queries for updating correlations were generated.
  - Faster loading of related attributes (from the correlations).
- Show event preview before merging two events.
- The free-text import tool converts [at] to @ and hxtp and htxp to http.

MISP 2.4.131 (2020-09-08)

- Added a *count* returnFormat for the REST API, which simply counts the number of attributes/events found (on each respective scope).
- The API GET requests on restSearch with no parameters are no longer allowed. The users of the GET queries are warned with posted JSON bodies.

References

[1] ACHE Crawler by ViDA-NYU. https://github.com/ViDA-NYU/ache. Accessed: 2021-04-06.

[2] BitSight Security Ratings. https://www.bitsight.com/security-ratings. Accessed: 2021-02-20.

[3] BreachAlert, SKURIO. https://skurio.com/solutions/breach-alert/. Accessed: 2021-02-20.

[4] BrightCloud, Webroot. https://www.brightcloud.com/. Accessed: 2021-02-20.

[5] CIF: Collective Intelligence Framework by csirtgadgets. https://csirtgadgets.com/collective-intelligence-framework. Accessed: 2021-04-06.

[6] Common Vulnerabilities and Exposures (CVE) by MITRE. https://cve.mitre.org. Accessed: 2021-04-06.

[7] CTAC, Wapack Labs. https://www.wapacklabs.com/ctac. Accessed: 2021-02-20.

[8] Cyber Advisor, SurfWatch Labs. https://www.surfwatchlabs.com/threat-intelligence-products/cyber-advisor. Accessed: 2021-02-20.

[9] Cyjax. https://www.cyjax.com/cyber-threat-services/. Accessed: 2021-02-20.

[10] EclecticIQ. https://www.eclecticiq.com/platform. Accessed: 2021-02-20.

[11] ElasticSearch (a distributed, RESTful search and analytics engine). https://www.elastic.co/elasticsearch/. Accessed: 2021-04-06.

[12] Exploit-DB by OffSec Services. https://www.exploit-db.com. Accessed: 2021-04-06.

[13] F5 Labs. https://www.f5.com/labs. Accessed: 2021-02-20.

[14] Flashpoint Intelligence Platform. https://www.flashpoint-intel.com/platform/. Accessed: 2021-02-20.

[15] Gensim: Topic modeling for humans (a free Python library). https://radimrehurek.com/gensim/. Accessed: 2021-04-06.

[16] Google Scholar. https://scholar.google.com. Accessed: 2021-05-19.

[17] GOSINT: A Framework for Collecting, Processing, and Exporting Indicators of Compromise (IoC). https://gosint.readthedocs.io/en/latest/. Accessed: 2021-04-06.

[18] Grakn: The Knowledge Graph. https://grakn.ai/. Accessed: 2021-04-06.

[19] GraphQL: A query language for APIs. https://graphql.org/. Accessed: 2021-04-06.

[20] Helix Security Platform, FireEye. https://www.fireeye.com/products/helix.html. Accessed: 2021-02-20.

[21] IEEE Digital Library. https://ieeexplore.ieee.org/Xplore/home.jsp. Accessed: 2021-05-19.

[22] Intel 471. https://intel471.com/. Accessed: 2021-02-20.

[23] IoTSec Ontology github page. http://iotsec.brunomozza.com/. Accessed: 2021-04-06.

[24] JVN iPedia - Vulnerability Countermeasure Information Database. https://jvndb.jvn.jp/en/. Accessed: 2021-04-06.

[25] JVN vulnerability data-feeds. https://jvndb.jvn.jp/en/feed/. Accessed: 2021-04-06.

[26] KB-Cert by Carnegie Mellon University. https://www.kb.cert.org/vuls/. Accessed: 2021-04-06.

[27] KB-Cert vulnerability notes. https://www.kb.cert.org/vuls/bypublished/desc/. Accessed: 2021-04-06.

[28] MISP - Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing. https://www.misp-project.org/. Accessed: 2021-04-06.

[29] MISP Sightings remodeling and extensions. https://www.misp.software/2017/02/16/Sighting-The-Next-Level.html. Accessed: 2021-04-06.

[30] MITRE ATT&CK TTPs knowledge base. https://attack.mitre.org/. Accessed: 2021-04-06.

[31] mongoDB: The database for modern applications (a NoSQL database). https://www.mongodb.com/. Accessed: 2021-04-06.

[32] MySQL information schema. https://dev.mysql.com/doc/mysql-infoschema-excerpt/8.0/en/information-schema-tables-table.html. Accessed: 2021-04-06.

[33] National Vulnerability Database (NVD) by NIST. https://nvd.nist.gov/. Accessed: 2021-04-06.

[34] netfilter: firewalling, NAT, and packet mangling for linux. https://www.netfilter.org. Accessed: 2021-04-06.

[35] NIST NVD vulnerability data-feeds. https://nvd.nist.gov/vuln/data-feeds. Accessed: 2021-04-06.

[36] OpenCTI: Open Cyber-Threat Intelligence platform. https://www.opencti.io/en/. Accessed: 2021-04-06.

[37] OpenTAXII: a robust Python implementation of TAXII Services. https://opentaxii.readthedocs.io/en/stable/. Accessed: 2021-04-06.

[38] OpenTPX - Threat Partner eXchange by LookingGlass. https://github.com/Lookingglass/opentpx. Accessed: 2021-04-06.

[39] Prelude SIEM - Smart Security. https://www.prelude-siem.com/en/prelude-siem-en/. Accessed: 2021-04-06.

[40] PyMISP documentation. https://pymisp.readthedocs.io/en/latest/. Accessed: 2021-04-06.

[41] RabbitMQ (an open-source message broker). https://www.rabbitmq.com/. Accessed: 2021-04-06.

[42] Recorded Future. https://www.recordedfuture.com/. Accessed: 2021-02-20.

[43] Retina Network Community by BeyondTrust. https://www.beyondtrust.com/vulnerability-management. Accessed: 2021-04-06.

[44] SearchLight, Digital Shadows. https://www.digitalshadows.com/searchlight/. Accessed: 2021-02-20.

[45] SecurityFocus Vulnerability Database. https://www.securityfocus.com. Accessed: 2021-04-06.

[46] Snort: The open-source intrusion prevention system. https://www.snort.org. Accessed: 2021-04-06.

[47] Soltra, Celerium. https://www.celerium.com/automate. Accessed: 2021-02-20.

[48] Springer. https://www.springer.com. Accessed: 2021-05-19.

[49] Suricata - Open Source Intrusion Detection System (IDS) / Intrusion Prevention System (IPS) / Network Security Monitoring (NSM) engine. https://suricata-ids.org. Accessed: 2021-04-06.

[50] Swagger: API Development. https://swagger.io/. Accessed: 2021-04-06.

[51] The MANTIS Cyber Threat Intelligence Management Framework, SIEMENS. https://django-mantis.readthedocs.io/en/latest/readme.html. Accessed: 2021-02-20.

[52] TheHive - Security incident response for the masses. https://thehive-project.org/. Accessed: 2021-04-06.

[53] ThreatConnect. https://threatconnect.com/. Accessed: 2021-02-20.

[54] ThreatQ, ThreatQuotient. https://www.threatq.com/. Accessed: 2021-02-20.

[55] ThreatStream, Anomali. https://www.anomali.com/products/threatstream. Accessed: 2021-02-20.

[56] VDMR, Qualys. https://www.qualys.com/apps/vulnerability-management-detection-response/. Accessed: 2021-02-20.

[57] VirusTotal (online file virus analysis). https://www.virustotal.com/gui/. Accessed: 2021-04-06.

[58] VulDB recent CVSS metrics. https://vuldb.com/?cvssv3. Accessed: 2021-04-06.

[59] VulDB recent exploits. https://vuldb.com/?exploits. Accessed: 2021-04-06.

[60] VulDB recent vulnerabilities. https://vuldb.com/?recent. Accessed: 2021-04-06.

[61] VulDB: The community-driver vulnerability database. https://vuldb.com/. Accessed: 2021-04-06.

[62] YETI - Your Everyday Threat Intelligence. https://yeti-platform.github.io/. Accessed: 2021-04-06.

[63] ZeroFox. https://www.zerofox.com/. Accessed: 2021-02-20.

[64] ZeroMQ (an open-source messaging library). https://zeromq.org/. Accessed: 2021-04-06.

[65] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8, 02 2016.

[66] T. Chantzios, P. Koloveas, S. Skiadopoulos, N. Kolokotronis, C. Tryfonopoulos, V.G. Bilali, and D. Kavallieros. The Quest for the Appropriate Cyber-threat Intelligence Sharing Platform. In Slimane Hammoudi, Christoph Quix, and Jorge Bernardino, editors, *Proceedings of the 8th International Conference on Data Science, Technology and Applications, DATA 2019, Prague, Czech Republic, July 26-28, 2019*, pages 369–376. SciTePress, 2019.

[67] VERIS Community. *Vocabulary for Event Recording and Incident Sharing*, 2014. `http://veriscommunity.net/`. Accessed: 2020-08-14.

[68] J. Connolly, M. Davidson, and C. Schmidt. *The Trusted Automated eXchange of Indicator Information (TAXII™)*, 2014. `http://taxii.mitre.org/about/documents/Introduction_to_TAXII_White_Paper_May_2014.pdf`. Accessed: 2020-08-14.

[69] R. Danyliw. *The Incident Object Description Exchange Format Version 2*, 2016. `https://tools.ietf.org/html/rfc7970`. Accessed: 2020-08-14.

[70] A. de Melo e Silva, J.J. Costa Gondim, R. de Oliveira Albuquerque, and L.J. García-Villalba. A Methodology to Evaluate Standards and Platforms within Cyber Threat Intelligence. *Future Internet*, 12(6):108, 2020.

[71] Y. Ducq, C. Agostinho, D. Chen, G. Zacharewicz, R. Jardim-Gonçalves, and G. Doumeingts. Generic methodology for service engineering based on service modelling and model transformation. *Manufacturing Service Ecosystem. Achievements of the European 7th FP FoF-ICT Project MSEE: Manufacturing SErvice Ecosystem (Grant No. 284860). Eds. S. Weisner, C. Guglielmina, S. Gusmeroli, G. Doumeingts*, pages 41–49, 2014.

[72] ENISA. *ENISA Threat Landscape 2015*, 2016. `https://www.enisa.europa.eu/publications/etl2015/at_download/fullReport`. Accessed: 2020-08-14.

[73] ENISA. *ENISA Threat Landscape 2016*, 2017. `https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2016/at_download/fullReport`. Accessed: 2020-08-14.

[74] ENISA. *ENISA Threat Landscape 2017*, 2018. `https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2017/at_download/fullReport`. Accessed: 2020-08-14.

[75] ENISA. *Exploring the Opportunities and Limitations of Current Threat Intelligence Platforms*, 2018. `https://www.enisa.europa.eu/publications/exploring-the-opportunities-and-limitations-of-current-threat-intelligence-platforms/at_download/fullReport`. Accessed: 2020-08-14.

[76] ENISA. *ENISA Threat Landscape 2018*, 2019. `https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018/at_download/fullReport`. Accessed: 2020-08-14.

[77] ENISA. *ENISA Proactive Detection - Survey Results*, 2020. `https://www.enisa.europa.eu/publications/proactive-detection-survey-results/at_download/fullReport`. Accessed: 2020-08-14.

[78] B. Feinstein, D. Curry, and H. Debar. *The Intrusion Detection Message Exchange Format (IDMEF)*, 2007. `https://tools.ietf.org/html/rfc4765`. Accessed: 2020-08-14.

[79] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database systems - the complete book (international edition).* Pearson Education, 2002.

[80] Infoblox. BloxOne Threat Defense Advanced: Strengthen and Optimize Your Security Posture from the Foundation. `https://www.infoblox.com/wp-content/uploads/infoblox-datasheet-bloxone-threat-defense-advanced.pdf`, 2020. Accessed: 2020-10-30.

[81] Y. Jiang, Y. Atif, and J. Ding. Cyber-Physical Systems Security Based on a Cross-Linked and Correlated Vulnerability Database. In S. Nadjm-Tehrani, editor, *Critical Information Infrastructures Security - 14th International Conference, CRITIS 2019, Linköping, Sweden, September 23-25, 2019, Revised Selected Papers*, volume 11777 of *Lecture Notes in Computer Science*, pages 71–82. Springer, 2019.

[82] A. Khazaei, G. Mohammad, and M. Christoph. VuWaDB: A Vulnerability Workaround Database. *International Journal of Information Security and Privacy (IJISP)*, 12(4):24–34, 2018.

[83] P. Koloveas, T. Chantzios, S. Alevizopoulou, S. Skiadopoulos, and C. Tryfonopoulos. inTIME: A Machine Learning-Based Framework for Gathering and Leveraging Web Data to Cyber-Threat Intelligence. *Electronics*, 10(7), 2021.

[84] P. Koloveas, T. Chantzios, C. Tryfonopoulos, and C. Skiadopoulos. A Crawler Architecture for Harvesting the Clear, Social, and Dark Web for IoT-Related Cyber-Threat Intelligence. In C. K. Chang, P. Chen, M. Goul, K. Oyama, S. Reiff-Marganiec, Y. Sun, S. Wang, and Z. Wang, editors, *2019 IEEE World Congress on Services, SERVICES 2019, Milan, Italy, July 8-13, 2019*, pages 3–8. IEEE, 2019.

[85] B.A. Mozzaquatro, C. Agostinho, D. Gonçalves, J. Martins, and R. Jardim-Gonçalves. An Ontology-Based Cybersecurity Framework for the Internet of Things. *Sensors*, 18(9):3053, 2018.

[86] M. Nerwich, P. Gauravaram, H. Paik, and S. Nepal. Vulnerability Database as a Service for IoT. In L. Batina and G. Li, editors, *Applications and Techniques in Information Security. ATIS 2020. Communications in Computer and Information Science*, volume 1338. Springer, Singapore, 2020.

[87] OASIS. *STIX Version 2.0. Part 1: STIX Core Concepts*, 2017. `http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part1-stix-core.html`. Accessed: 2020-08-14.

[88] OASIS. *STIX Version 2.0. Part 2: STIX Objects*, 2017. `http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part2-stix-objects.html`. Accessed: 2020-08-14.

[89] P. Poputa-Clean and M. Stingley. *Automated Defense-Using Threat Intelligence to Augment Security*, 2015. `https://www.sans.org/reading-room/whitepapers/threats/automated-defense-threat-intelligence-augment-35692`. Accessed: 2020-08-14.

[90] F. Skopik, G. Settanni, and R. Fiedler. A problem shared is a problem halved: A survey on the dimensions of collective cyber defense through security information sharing. *Computers & Security*, 60:154–176, 2016.

[91] V.M. Vilches, L.U.S. Juan, B. Dieber, U.A. Carbajo, and E. Gil-Uriarte. Introducing the Robot Vulnerability Database (RVD), 2020.

[92] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody. MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform. In S. Katzenbeisser, E. R. Weippl, E.O. Blass, and F. Kerschbaum, editors, *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, WISCS 2016, Vienna, Austria, October 24 - 28, 2016*, pages 49–56. ACM, 2016.

[93] T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah. Cyber threat intelligence sharing: Survey and research directions. *Computers & Security*, 87:101589, 2019.