Enhancing Cyber Threat Intelligence with STIX-Shifter: An Analysis of ACT and STIX Integration

Erik Sørli
Benjamin Jørgensen

Thesis submitted for the degree of
Master in Informatics: Information Security
60 credits

Department of Informatics
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2023

Enhancing Cyber Threat Intelligence with STIX-Shifter: An Analysis of ACT and STIX Integration

Erik Sørli
Benjamin Jørgensen

Enhancing Cyber Threat Intelligence with STIX-Shifter: An Analysis of ACT and STIX Integration

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

As the cyber threat landscape continues to evolve, organizations are turning to automated systems to address challenges posed by the complexity and volume of cyber threats. Such challenges include proactively detecting, prioritizing, and contextualizing threats. Further, improve incident response and decision making, and utilize collaboration and automation to improve the organization's resilience. These automated systems aggregate threat data from multiple sources, providing organizations with accurate and actionable intelligence to protect their assets and address these challenges. However, organizations offer their own events and logs scattered along different platforms. Based on the variation in their origin from other security devices and reliance on different programming languages composes an obstacle as they cannot be interpreted in the same manner. [1].

This thesis provides a distinct perspective on a contribution to developing a new component that helps translate and exchange Cyber Threat Intelligence (CTI) between different security devices. The current study executes it by withdrawing data from the Semi-Automated Cyber Threat Intelligence (ACT) platform, a CTI platform developed by the Managed Security Service Provider (MSSP) mnemonic AS. To display ACTs data in a standardized format, the Structured Threat Information eXpression (STIX) data language was incorporated to convey data about CTI in a common language. STIX is an XML programming language used to visually represent data in a JSON format to quickly and easily be read by users and machines. The contribution fosters collaboration and threat detection, bridges the gap between disparate security devices, and nurtures greater cooperation within the cyber security community.

Acknowledgements

First and foremost, this master thesis would not be possible without the help, guidance, and research ideas from our co-supervisor Mateusz Zych, main supervisor Associate Professor Vasileios Mavroeidis from the University of Oslo, and supervisor Dr. Siri Bromander from mnemonic AS. Secondly, we would like to thank Fredrik Borg, also from mnemonic AS for his valuable information and guidance. Finally, we would like to thank friends and family for all their love and support. Without all these people, this thesis would never have seen the light of day.

Division of Responsibilities for Joined Thesis

Following the findings presented in this thesis, the division of responsibilities between Erik Sørli and Benjamin Jørgensen was systematically organized. Erik Sørli was entrusted with managing the connector's translation component, while Benjamin Jørgensen was designated responsible for overseeing the transmission component of the connector. Nevertheless, both individuals maintained a close working relationship throughout the project to ascertain the equal workload distribution and accomplishment of the project objectives. Both parties have agreed to distribute tasks as evenly as possible and give information to one another where there might be too much to do or too little. Various digital tools were employed to facilitate effective communication and collaboration, including Discord, Messenger, and Trello. These platforms facilitated the monitoring of progress and the coordination of collective efforts. Moreover, the collaborative process utilized Overleaf's cooperative mode, enabling both parties to work on a shared document simultaneously. In the context of the development of code and version control, Git and GitHub were used accordingly. This approach streamlined the review process and ensured that the final submission met a coherent and consistent quality that could be delivered to the supervisor and co-supervisor.

Contents

List of Figures

List of Tables

List of Listings

Glossary and Acronyms

ACT (Semi-Automated Cyber Threat Intelligence): Open Source Threat Intelligence Platform by mnemonic. i, vii–xi, 1, 6, 8–12, 14, 15, 17, 31, 34–40, 42–47, 49–66, 68, 69, 71–98, 100–102, 105–121

ANTLR (Another Tool for Language Recognition): A powerful parser generator for reading, processing, executing, or translating structured text or binary files. 1, 57

API (Application Programming Interface): A set of rules and conventions for building and interacting with software applications. x, 1, 10, 15, 30, 34, 42, 43, 45, 47, 51, 53, 58–62, 65, 66, 72, 73, 78, 79, 89, 90, 93

AS (Autonomous Systems): Large networks operated by organizations that are responsible for routing traffic between different networks. 1

ASN (Autonomous System Number): A unique identifier assigned to each autonomous system in the global internet. 1, 38, 63, 75, 83, 95, 99, 107

BGP (Border Gateway Protocol): A protocol used to exchange routing information between routers in different autonomous systems. 1

CIDR (Classless Inter-Domain Routing): An IP addressing scheme that improves the allocation of IP addresses. 1, 63, 79

CLI (Command-Line interface): Interacting with a device or computer programs in text lines. 1, 32, 61, 70

CTI (Cyber Threat Intelligence): A field in cybersecurity focused on collecting and analyzing information about potential cyber threats. i, 1, 5, 6, 8, 14, 15, 17, 19, 21, 24, 26–31, 34, 35, 47, 49, 53, 57, 62, 68, 72, 91, 97, 98

EICAR (European Institute for Computer Antivirus Research): Anti-malware test file to test the correct operation of malware detection scanners.. 1, 68

FQDN (Full Qualified Domain Name): A domain name that includes all the necessary components, such as the hostname and top-level domain. 1, 38, 47, 68, 69, 77, 80, 90, 107

GNN (Graph Neural Networks): A type of neural network architecture that directly operates on graph data structures, enabling the learning and prediction of graph-structured data, widely used in various applications modeling of objects and their relationships. 1, 14

IDS (Intrusion Detection Systems): Systems designed to monitor network traffic for malicious activity and report detected threats. 1, 27

MSSP (Managed Security Service Provider: An MSSP offers outsourced management and monitoring of security devices and systems.. i, 1, 6

OCA (Open Cybersecurity Alliance): A consortium of cybersecurity vendors working together to create open and interoperable cybersecurity tools. 1, 15

OSINT (Open-Source Intelligence): Intelligence gathered from publicly available sources, such as websites, social media, and news articles. 1, 5

REGEX (Regular Expression): A sequence of characters that specifies a search pattern, mainly for use in pattern matching. 1

REST (Representational State Transfer): An architectural style for designing networked applications that focuses on resources and their representations. 1, 42, 43, 47

RFC (Request for Comments): A publication from the Internet Engineering Task Force that describes a proposed standard, protocol, or procedure. 1, 22, 69, 82

SCO (STIX Cyber-observable): Objects in STIX that represent specific cybersecurity-related observations, such as file hashes or IP addresses. ix, x, 1, 20–24, 28, 32, 51, 54, 56, 68, 69, 72, 78–85, 89, 90, 95, 98, 101

SDO (STIX Domain Objects): Objects in STIX that describe high-level constructs, such as campaigns, threat actors, and attack patterns. x, xi, 1, 20, 21, 24, 26, 27, 71, 72, 85, 87, 89, 90, 98, 101, 102, 105, 106, 108–120

SIEM (Security Information and Event Management): Software or hardware solutions that collect, analyze, and manage security events in real-time. 1, 27, 31, 34

SQL (Structured Query Language): A domain-specific language used in programming for managing relational databases. 1, 62

SRO (STIX Relationship Object): A type of STIX Domain Object that describes relationships between other SDOs and SCOs. 1, 20–22, 24, 26, 27, 53, 72, 90, 91, 98

STIX (Structured Threat Information eXpression): A standardized language and serialization format used to exchange cyber threat intelligence. i, vii, ix–xi, 1, 6–12, 14–17, 19–24, 26–34, 50–52, 54–60, 62, 68–72, 74, 75, 77–84, 87, 88, 90, 92, 94, 97, 98, 102, 105–118, 120, 121

TAXII (Trusted Automated eXchange of Intelligence Information): Defines how cyber threat information can be transported via services and message exchanges.. 1, 19, 20, 30

URI (Uniform Resource Identifier): A string of characters that identifies a name or a resource on the internet. 1, 37–39, 59, 66, 68, 69, 83, 88

UUID (Universally Unique Identifier): A 128-bit number used to uniquely identify information in computer systems. 1, 44, 47, 53, 60, 61, 71, 88, 102

Chapter 1

Introduction

The constantly evolving landscape of cyber threats and increased numbers presents a formidable challenge for organizations to keep up with the rapid technological changes [2]. Traditional manual processes for detecting and remediating threats can be time-consuming and may not address the full spectrum of cyber threats. Cyber threats can be categorized into various types based on their tactics, techniques, and procedures. To mention a few, it includes malware, phishing, social engineering, DDoS attacks, and insider threats. To tackle these challenges, many organizations are turning to automation and CTI platforms to help identify and respond to threats more promptly and precisely [3]. A CTI platform is a knowledge management system that collects and disseminates information about potential and existing cyber threats. CTI platforms aggregate threat data from multiple sources, including Open-Source Intelligence (OSINT), proprietary data feeds, manual data analysis, and industry-specific threat intelligence [4]. CTI platforms provide organizations, especially analysts, with accurate, timely, and actionable intelligence that can help them identify, understand, and defend against various cyber threats and attacks. The current state of the cyber threat landscape is characterized by the accumulation of valuable and actionable intelligence from multiple organizations. Organizations often manage a multitude of assets containing data, which, when analyzed, generate insightful intelligence on how to protect their assets. These assets could significantly contribute to understanding and remediating cyber threats. How-

ever, the inability to share this information creates a barrier that hinders the optimal utilization of these resources. In addressing this barrier, the emergence of STIX as a standardized language enables the representation and exchange of CTI among various security tools and platforms [5]. By employing STIX, organizations can foster collaboration, enhance threat detection, and streamline their response to cyber security incidents. This study investigates STIX-Shifter, an open-source Python library that plays a vital role in this context. STIX-Shifter is an open-source Python library allowing software to connect to products that house data repositories by using STIX patterning and return results as STIX observations [6]. The current process is executed by developing a connector for the specific security product or software. A fully developed connector creates a way of interoperability, striving to minimize information loss during the conversion. This study aims to develop a connector for the CTI platform called ACT, designed by the MSSP mnemonic AS. Integrating the ACT platform with STIX-Shifter aims to facilitate sharing of CTI among multiple organizations. This integration enables organizations to detect and respond more effectively to threats, bridging the gap between disparate security tools and nurturing greater cooperation within the cyber security community.

1.1 Problem Statement

The variety of security tools available presents a significant challenge for organizations striving to efficiently protect their assets from being compromised. A primary obstacle stems from the multiple native languages these tools employ, which demands considerable time and resources to be utilized efficiently. The present study addresses this issue by contributing to STIX-Shifter, conveying data about cyber threats in a common language. While connectors are currently available and actively contribute to the STIX-Shifter library, the exchange of CTI across different platforms remains an area with potential for further coverage. By actively incorporating additional connectors to the STIX-Shifter library, the

breadth of the knowledge domain can be substantially increased, thereby covering intelligence gaps specific organizations withhold to be available in a siloed security solution.

Research done by IBM shows that an average large organization may have as many as 80+ security products from up to 40 different organizations [7]. Another article from IBM shows a complexity that hinders response across multiple categories of the threat life cycle in companies attending the survey. The survey explains the negative impact of organizations using 50+ security tools, as they ranked themselves 8% lower in their ability to detect and 7% lower in their ability to respond to an attack than organizations with fewer tools. These research papers show that simplicity makes for greater security [8]. If it is possible to migrate existing solutions and tools to all use STIX, and its housing STIX-Shifter, it removes the need to learn specifics such as query language to utilize the current tool.

A logical rationale for the absence of collaboration within threat intelligence sharing might be attributed to the high R&D costs and is not considered a true core business declared in an article from Oxford Academic [9]. These vendors often maintain an alternative product line of importance or utilize third-party threat intelligence feeds to supplement their primary cyber security services. Also stated in the same article, "In addition, market statistics show the lack of a significant revenue stream generated by threat intelligence" [9]. This is one of several reasons for the lack of collaboration and exchange of critical information between security organizations. This hinders the collective ability to effectively identify, analyze and respond to emerging threats on time within an organization. Consequently, the industry may face difficulties in precisely understanding the evolving threat landscape and devising robust defensive strategies.

1.2 Research Questions

This thesis aims to develop a connector for ACT. To effectively address this overall objective, it is beneficial to decompose a main research question into two subquestions. This approach clarifies the process of answering the overarching question by managing individual aspects of the problem.

Research Question: How feasible is developing a new STIX-Shifter module that allows the ACT platform to share its knowledge base comprising CTI?

This thesis aims to answer the following research questions:

- To what extent is the sharing of CTI possible from ACT to STIX?
- To what extent will data be lost in the convergence of results returned from the ACT platform?

1.3 Research Methodology

This thesis examines the cyber security domain, a specialized sector within the broader field of computer science. At its inner core, the thesis adheres to a scientific research approach [10][11]. Therefore experimental research methodology is deemed applicable to the nature of the research questions. Further practices within design and development for software are utilized, such as requirement analysis [12], design, and development. Moreover, to understand and manage the development, knowledge acquisition is needed in the form of a literature review [13]. The research methodology comprises the following:

- **Literature review** - Comprehensive literature review on STIX 2.0 Standard, the ACT platform, and existing connectors in the STIX-Shifter library. This will address requirements, challenges, and best practices for developing a new connector. Furthermore, an investigation of previous research regarding sharing of CTI with ACT will be examined.

- **Requirement analysis** - Identify functional and non-functional requirements for the STIX-Shifter connector for the ACT platform.
- **Design and development** - Adopt a design approach according to the requirements and specifications in STIX-Shifter for developing a new connector.
- **Experimentation** - Test the developed STIX-Shifter connector in a controlled environment. Design experiments to assess the connector's performance, data loss, and validity.
- **Evaluation** - Evaluate the results of the experiments to determine if the connector meets requirements.

According to Science and the Global Environment [14], a genuine experimental design necessitates the examination of relationships between and among variables. Typically, the independent variable is controlled to measure its influence on the dependent variable in a systematic and controlled manner. Now to apply this to the hypothesis:

Hypothesis (*H)*: ACT Data Model and the STIX language and serialization format can be mapped appropriately to construct a connector for the STIX-Shifter library with minimal data loss.

This hypothesis claims that the ACT Data Model and the STIX language and serialization format can be mapped appropriately to construct a connector for the STIX-Shifter library with minimal data loss. The variable measured will be the effectiveness of the connector in terms of data loss minimization during the mapping process. Data loss minimization is crucial for ensuring the accuracy and reliability of the information exchanged between the ACT Data Model and the STIX language and serialization format.

The independent variable in this scenario is the mapping strategy employed to create the connector between ACT and STIX. The dependent variable, in this case, is the data that disappears throughout the mapping process. This study

aims to establish a mapping strategy that minimizes data loss, resulting in an accurate connector for the STIX-Shifter library. The challenges with mapping will solely be based on values in both platforms. Data loss will be measured by comparing the information present in the original Data Models (STIX and ACT) and the information retained after the mapping process. A successful mapping strategy will have a low data loss rate, preserving more of the original data's integrity. A supplement to the data loss check will be compared to the existing MySQL connector.

1.4 Technical Approach

Creating a new connector for the ACT platform poses several challenges:

- **Understanding target Data Model**: Developing a new connector entails an extensive understanding of the target Data Model's architecture, data format, and query language. This requires developing knowledge about the existing research documentation, documentation of standards and specifications, and further on examining target APIs and its connectivity.
- **Mapping to STIX Data Model**: Mapping the target Data Model to the STIX Data Model is a key challenge. This analysis will ascertain the compatibility of the mapping between STIX and the target Data Model. Identifying the data sources in the target Data Model and mapping them to existing STIX data sources is important to determine compatibility. Furthermore, challenges and limitations in the mapping process must be identified. Potential challenges and limitations may include data loss during the conversion between STIX and the target Data Model and if the data sources adhere to the same standards and taxonomy.
- **Handling query translations and transmission**: The completed connector must be able to translate STIX pattern expressions into the target data source's native query language, which can be intricate due to discrepancies in semantics and linguistic structures between the two languages. The translation process of the connector should account for

discrepancies and ensure accurate and efficient queries. Moreover, the connector must also address the transmission aspect, which effectively exchanges information. This entails the transfer of queries between the data source and the STIX-Shifter library [15].

- **Error handling and resilience**: A resilient connector should be able to manage errors adeptly, ensuring any issues during data acquisition or translation processes are suitably addressed. Such resiliency may require implementing extensive error handling.
- **Testing and validation**: Testing the connector is imperative to define the accuracy and functionality. Further, validating the precision of the query translations, data mappings, and error-handling mechanisms is critical to guarantee reliable outcomes for the end user.
- **Documentation**: Developing a simple connector to maintain, extend and troubleshoot is essential for enduring a successful high-quality connector. This adheres to best practices regarding software development, clear code organization, comprehensive documentation, code comments, etc. Further on, it is important to configure clear instructions on how to set up and configure the connector and define any constraints, and acknowledged issues of importance.

1.5 Thesis Outline

This section offers a synopsis of the thesis chapters and their respective contents:
Chapter 2: Previous Research - This chapter reviews the current literature about the research topic, enlightening how the present study aims to expand upon and add to the existing expertise in the field.

Chapter 3: Background - Covers the core elements of the thesis. Such as taxonomy, security automation, and models within cyber security. In-depth analysis of the ACT platform and how STIX works.

Chapter 4: Research - This chapter focuses on mapping between STIX and ACT and the STIX-Shifter connector's functionality.

Chapter 5: Challenges and Interpretations - Discusses the challenges and interpretations associated with the employed connector and the current state of the technology.

Chapter 6: Knowledge Acquisition - In this chapter, further exploration into the functionality was undertaken.

Chapter 7: Results - The result chapter explains the comprehensive mapping between STIX and ACT and the result of the created bundles. In addition, an explanation of challenges and potential data loss in the convergence of ACT objects to a STIX bundle of observed objects will be explained.

Chapter 8: Discussion - This chapter will discuss findings from the Results chapter, an extensive exposition of outcomes obtained from the newly created ACT connector. It gives highlights on different edge cases, data loss, and limitations.

Chapter 9: Validation - This chapter addresses how the newly established connector is tested in a controlled environment and how a STIX bundle is validated.

Chapter 10: Conclusion - The thesis was concluded by incorporating the analysis and discussions presented throughout the preceding sections, thereby addressing the research question and the adhering subquestions posited at the beginning of this thesis.

Chapter 11: Future Work - Recommendations for future work are addressed in this chapter.

Chapter 12: Appendices - This chapter withholds appendices such as addi-

tional tables and figures.

Chapter 2

Previous Research

This chapter will address the current research on developing a STIX-Shifter connector in general and research articles found applicable to this thesis. Furthermore, this chapter will explore a specific implementation of an existing connector to understand how it works and its functionalities and apply it to integrating a new connector for the ACT platform.

2.1 Literature Review

Regarding STIX and ACT, previous research has made significant findings in various areas. Martin K. J Heggem investigated the use of Graph Neural Networks (GNN), specifically the SEAL framework, to predict new relations in the ACT graph, finding that the inclusion of the "mention" fact led to better performance in some cases, but not evidently across the graph [16]. Mari Grønberg developed an ontology for CTI based on concepts found in the STIX sharing standard, presenting the potential of using ontologies for CTI and highlighting the need for further development in creating a shared language [17]. At last, Siri Bromander's research paper introduces the CTI model, which enables professionals within cyber security to explore threat intelligence capabilities and comprehend their position against the constantly moving threat landscape. Bromander also emphasizes that the cyber security community lacks a comprehensive ontology that covers the complete spectrum

of threat intelligence and delves into if standardization is used. The paper also discusses some of the greatest difficulties in working towards an ontology for CTI, such that it is a vaguely defined terminology, lack of standardized representation of relevant information, and lack of coherent relationship between layers abstraction in ontologies. By addressing these obstacles, the presented ontology would better facilitate knowledge sharing and ultimately enhance the effectiveness of CTI [18], such as nurturing the standardization this current thesis will strive for.

2.2 STIX-Shifter Connectors

Sharing information with STIX is already performed by several other security products, including Splunk, Qradar, and CyberReason [19]. The documentation available for implementing a connector is in the GitHub repository of STIX-Shifter [20], managed by Open Cybersecurity Alliance (OCA). It consists of a basic outline of how to create a connector, but according to their documentation, an understanding of the product's query language and API is recommended. Further on, understanding observable objects and STIX patterning is a prerequisite to developing a connector [15].

2.3 Previous Work Regarding ACT Connector

There have been no efforts to establish a connector for ACT in STIX-Shifter. Nevertheless, there is some documentation regarding the Data Model of ACT available on the ACT platform's website [21]. As a result, it is both logical and necessary to examine existing connectors for inspiration to design and implement this connector effectively. These connectors do not have any research papers on how the actual mapping between the source fields of a security product and STIX is created. Therefore, the research done in this paper is based on the documentation of STIX-Shifter and an assumption of how previous products have integrated the functioning connectors.

2.4 Composition of the MySQL Connector

This section provides a step-by-step overview of the documentation on how the MySQL connector is composed, found in the Connector Coding Lab [22]. The selection of MySQL as the focus of this section is due to its more in-depth descriptions and actions on creating a new connector. This connector is also the one with the least advanced functionality according to the official GitHub repository [22]. The connector coding lab serves as a guide for implementing a new connector module in STIX-Shifter. The main purpose of this lab is to develop practical experience for creating a functional connector. It should be noted that it provides a limited perspective on creating the functionality, as it primarily focuses on printing functionality directly from the existing MYSQL connector. Generally, the lab guide goes through the following:

- Clone the STIX-Shifter GitHub repository
- Make a copy of stix_shifter_modules/demo_template
- Create module EntryPoints
- Implement input configuration of the connector
- Implement STIX translation module
- Implement STIX transmission module
- Implement data source results to STIX translation
- Test the end-to-end query flow

The guide specifies that it will not delve into how the connector's functionality is created. Instead, it aims to provide an overall impression of what development needs. To effectively develop the functionality, one must examine existing connectors and employ an iterative, trial-and-error approach to develop a new connector.

Chapter 3

Background

In the forthcoming chapter, a broad analysis will be conducted on several essential topics to provide an overall understanding of the core elements of this thesis. This chapter elucidates taxonomy concepts, security automation, and CTI. Additionally, an in-depth examination of ACT and STIX, along with an exploration of their dependent technologies. Given the complexity and significance of these topics, this chapter is considered sizable in length. Yet, these subjects are crucial for understanding the functionality of both STIX-Shifter and the ACT platform. A deep comprehension of these elements will enable the design of an accurate connector to be seamlessly incorporated.

3.1 Cyber Threat Intelligence

In cyber security, threat actors constantly try to outmaneuver different defenses and find new ways to exploit a system or gain access to an infrastructure. More proactive detection and tailored defenses are in demand to handle the exposure of exponentially increasing threats properly. This can be achieved by sharing and collecting analysis of data from multiple sources. This proactive approach is known as Cyber Threat Intelligence (CTI).

Gartner has provided a definition CTI as follows: "Threat Intelligence is evidence-based knowledge (e.g., context, mechanisms, indicators, implications,

and action-oriented advice) about existing or emerging menaces or hazards to assets" [23]. In other words, it is about collecting data, including context, indicators, implications, and preventative actions that enrich and provide value for an organization.

However, the question is, when can information be defined as intelligence? According to SANS: "Intelligence is the *collection*, *processing*, and *analysis* of information about a competitive entity and its agents, needed by an organization or group for its security and well-being" [2]. The Joint Intelligence model Relationship of Data, Information, and Intelligence, depicted in Figure 3.1, visualizes the progressive refinement of raw data into actionable intelligence. It demonstrates how data, when collected and processed, becomes information. As this information is further analyzed and contextualized, it transforms into intelligence, which enables informed decision-making and strategic planning. This model highlights the importance of filtering, analysis, and synthesis in extracting valuable insights from vast data. Adhering to the definition of intelligence, one could affirm that information transforms into intelligence upon contextualization. Ergo, when data is collected, it gains an intelligent condition as soon as it is processed and analyzed.

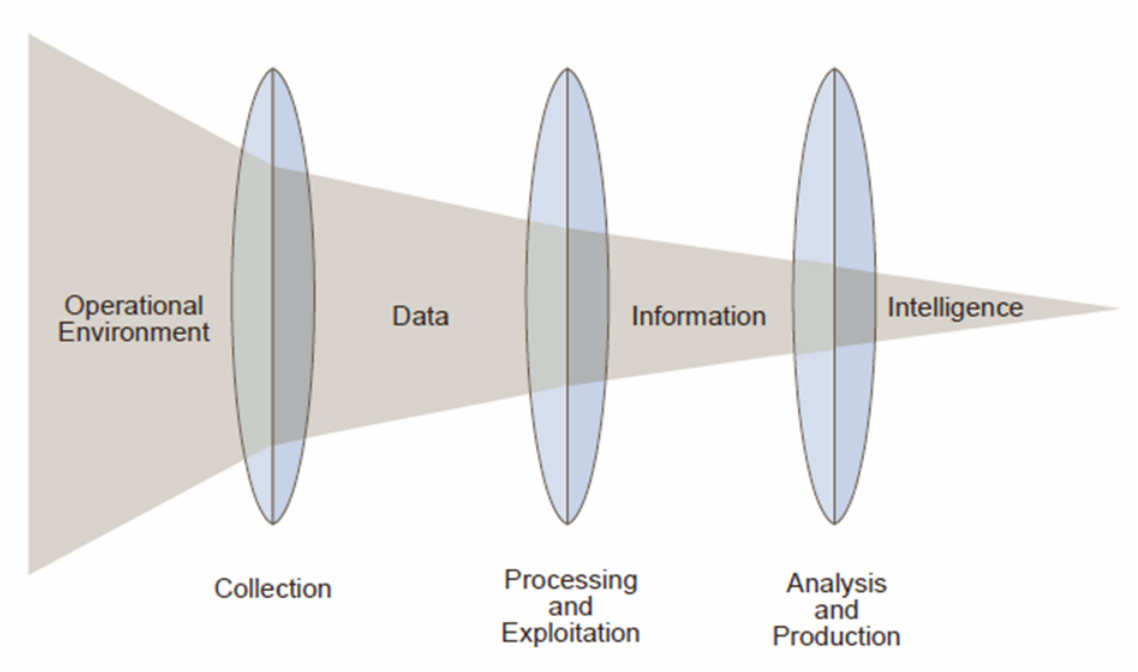

Figure 3.1: Relationship between data, information, and intelligence [24].

3.2 Security Automation

The application of security automation brings a significant advancement in the cyber security field, as it allows CTI to be automatically prioritized and handled in an automated process. This procedure optimizes critical functions like monitoring and detection and elevates the accuracy and effectiveness of incident response by providing data enrichment. Moreover, integrating security automation has substantially improved business continuity [25] [6]. Therefore, creating a connector enabled by STIX and STIX-Shifter will contribute to more data-enriching existing automation tools. A STIX-Shifter connector enhances security automation with Trusted Automated eXchange of Intelligence Information (TAXII) by integrating diverse security tools and platforms. The TAXII protocol

was designed specifically for exchanging CTI across systems [26]. Moreover, the STIX-Shifter connector acts as a bridge between the security product and the TAXII ecosystem, allowing them to communicate using STIX and TAXII standards. This enables security tools to query and retrieve threat intelligence data from TAXII servers, perform real-time analysis, and take automated actions to respond to the identified threats. This advances the detection and remediation process and assures that security teams are working with the latest and most precise threat information.

3.3 Structured Threat Information eXpression

STIX is a standardized language that is designed to deliver data about cyber security threats in a simple manner. The purpose of STIX is to make it human and machine-readable. Achieving this objective involves the standardization of the way one communicates about threats. STIX allows for effective communication between security professionals and facilitates the development of technologies that can protect different organizations or individuals from these threats [6][27].

STIX was created with many different use cases in mind. For instance, utilized by security analysts to review cyber threats and cyber threat activity. With STIX, you can identify patterns and behavior directly related to cyber threats. The STIX data produced can be used in decision-making and by operational employees to aid cyber threat response activities. These responses include prevention, detection, and reaction. Further sharing this information within and with other organizations benefits from previous knowledge and improves robustness and scaling. Using STIX provides a powerful norm for intelligence sharing, creating more effective and accurate cyber threat detection.

3.3.1 STIX Core Objects

This subchapter will explore the intricacies of the three primary objects employed in STIX. These core objects are SCO, SDO, and STIX Relationship Object (SRO). Through an in-depth examination of each core object's purpose,

structure, and function, this subchapter aims to provide the reader with an understanding of how these elements contribute to STIX.

STIX Cyber Observable (SCO)

STIX defines a way to characterize host-based, network, and related entities; this is called the SCO [28]. Each object defined corresponds to a data point commonly represented in CTI, including, amongst others, IP-address, file hashes, domain names, and email addresses. These objects define the facts about a security incident on a network or a host. This could be information regarding a file on a host or the network traffic between two IPs. Together with SDO, these objects will help create a higher-level understanding of a threat landscape. This could help an organization understand why a particular type of intelligence is relevant to them.

STIX Domain Object (SDO)

SDO are used to describe the various characteristics and attributes of these objects and can be used to provide detailed information about specific cyber threats and related entities. For example, a SDO representing a specific piece of malware might include information about the malware's file hash, the target operating systems it is designed to infect, and the tactics it uses to evade detection. SDO are an important part of the STIX language because it provides a structured and consistent way to represent and exchange information about cyber threats and related entities.

STIX Relationship Object (SRO)

The SRO are the objects connecting SDOs together, as well as connecting SCOs together. These can also create a relationship between an SDO and a SCO for a more complete and complex understanding of a threat landscape.

Unfortunately, STIX version 2.1 is not yet fully supported in STIX-Shifter [29]. These values will, as for now, be displayed in their deprecated version 2.0.

Structure of STIX Cyber Observable Objects

SCO is supported by both STIX version 2.0 and 2.1. This is a defined structured representation of observable objects and their properties. These can describe data in different functional domains [30]. The SCOs for version 2.0 consists of two sets of different properties: common and object-specific.

The common properties are attributes or characteristics that **MUST** be present in every STIX object, regardless of its type. They help standardize information representation across various STIX objects. Common properties include the type of the object. This property defines the specific type of STIX object, such as IPv4-address, mutex, or network traffic.

The object-specific properties are attributes and values for a specific STIX object. These properties are unique to each type SCO, providing more detailed information. For example, an artifact object has object-specific properties describing its mime-type, payload_bin, and URL. The common properties themselves are optional to display the JSON serialization of a SCO. In the documentation of STIX 2.0 [30], the "optional" keyword for a property is interpreted as described in Request for Comments (RFC) 2119. This standard states that a value is truly optional. This means that a vendor may choose to include it if required or enhance the product according to the vendor. At the same time, others may omit the same item [31]. However, if they are included, the property name and strings **MUST** be the same. Some of the SCO for STIX version 2.0 includes properties that may specify a reference to another SCO. These values must not be confused with SROs, as they are mere references to another object and only exists for specific SCO such as the IPv4 address object and domain-name object. This means the relations can not be included by any other SCOs and are specific to these objects.

To summarize required common properties provide a consistent framework for representing cybersecurity information across various STIX objects. In contrast, object-specific properties allow for detailed descriptions of the particular object

type.

The file structure of STIX-Shifter consists of JSON objects that represent different types of cyber security information, such as domain names, IP addresses, and observables. Each object has a set of attributes that provide information about the object, as well as relationships to other objects.

```
{
    "0": {
        "type": "domain-name",
        "value": "example.com",
        "resolves_to_refs": [
        "1"
        ]
    },
    "1": {
        "type": "ipv4-addr",
        "value": "198.51.100.2"
    }
}
```

Listing 1: example of a SCO.

The STIX format consists of a graph-based Data Model, which represents information as objects with attributes and relationships to other objects. In Listing 1, there is a domain object with a value of "example.com" and a relationship to an IP object with a value of "198.51.100.2". These objects are represented using the JSON format. The STIX-Shifter package allows for manipulating and querying this data to facilitate the exchange and analysis of threat intelligence information.

3.3.2 STIX Graph-Based Model

STIX possesses a graph composed of interconnected nodes and edges. The graph's nodes are defined by SDOs and SCOs, and its edges are defined by SRO, which include both external and embedded STIX Relationship Objects. This graph-based language supports flexible, modular, structured, and consistent representations of CTI while adhering to standard analysis methodologies [28]. STIX employs a graph-based model to represent the various types of CTI data and their interrelationships. The entities in the graph are represented as nodes, and the relationships between them are represented as edges.

In the context of STIX the following exemplify a few STIX relationships SRO:

- Attribution illustrated in Figure 3.2:
  - A relationship between a threat actor node and a malware node indicating that the threat actor is known to use that specific malware.
  - Example: There could be a relationship between a threat actor node representing the APT29 group, and a malware node representing the Hammertoss malware, indicating that APT29 is known to use Hammertoss [32].

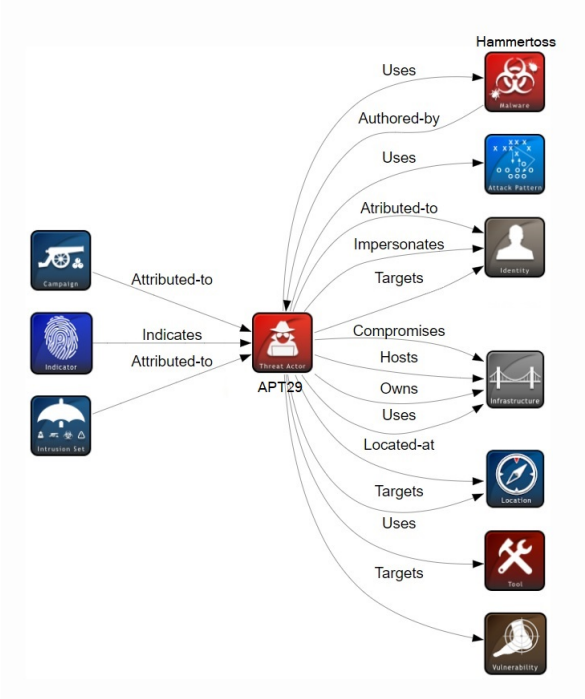

Figure 3.2: APT29 Group relationship Hammertoss.

- Campaigns illustrated in Figure 3.3:
  - A relationship between a threat actor node and a campaign node indicates that the threat actor is known to be involved in that specific campaign.
  - Example: There could be a relationship between a threat actor node representing the APT10 group, and a campaign node representing the Operation Cloud Hopper, indicating that APT10 is known to be involved in the Operation Cloud Hopper campaign [33].

Figure 3.3: APT10 Group relationship Operation Cloud Hopper.

These are just a few examples of the types of relationships that can be represented in STIX. The STIX Language Specification defines a set of standard objects and properties for representing different types of CTI data and relationships, allowing for the consistent representation and exchange of CTI data between different systems and organizations. Figure 3.2 and 3.3 are created by using Visualized SDO Relationships [34] and modifying them in an image editor.

3.3.3 STIX 2.0 and STIX 2.1

The main difference between STIX 2.1 and 2.0 lies in introducing new objects and enhancements to existing objects. STIX 2.1 introduced the new object SROs and enhancement to already existing SDOs. In STIX 2.1, a new SRO called "Sighting" was introduced that had the purpose of expressing and observation of an SDO, such as Indicator, Malware, Threat Actor in a specific context.

The main purpose is to offer a more standardized way to represent real-world occurrences of CTI entities, enabling improvement in tracking and correlation, located in sections 4.14 and 5. in the official documentation [35]. Moreover, STIX 2.1 added new properties and relationships that enhance the already existing SDOs by providing more expressiveness and flexibility. By this, an Indicator object in STIX 2.1 allows for the use of granular patterns and has more support for representing a level of confidence and threat actor profiles. To summarize, the main difference between STIX 2.1 and 2.0 is the introduction of new SROs, and enhancement of SDOs, by supplementing methods for more expressive and versatile ways of conveying CTI. For this project, the final results will be displayed as STIX Cyber Observable Objects in version 2.0 due to STIX-Shifter currently not supporting 2.1 [20].

3.3.4 STIX Patterning

STIX patterning was introduced as part of the STIX 2.0 language to meet the increasing need for a standardized and comprehensive method to express patterns, promoting more effective sharing and analysis of CTI across organizations and tools [36]. Before STIX patterning, various Intrusion Detection Systems (IDS) and security tools relied on their custom rule languages or syntaxes, such as Snort or Yara, for articulating threat patterns. The opportunity to further develop these was considered unattainable due to licensing issues [36].

As a result, STIX patterning was developed as another abstraction layer that could optimize the serialization and correlation rules already utilized by existing Security Information and Event Management (SIEM) systems and other platforms used for cyber security. Each of these platforms is dependent on query languages. Therefore, offering one standardized language to articulate cyber threat patterns across different platforms, all speaking the same language, helps to optimize information gathering and sharing of the total picture.

The vital part of STIX patterning is responsible for the uniform language behind the operation of STIX-Shifter connectors. The connectors function as an

intermediary between security platforms, allowing them to translate queries based on STIX patterning into the native query and languages for the respective platform. Patterning ensures platform compatibility, ultimately enhancing efficiency and sharing regarding CTI.

In order to comprehend this concept, a basic example of a STIX pattern can involve the use of Comparison Expression, which uses a single property of a SCO to a given constant using a Comparison Operator. For example, a Comparison Expression within an Observation Expression can match against an IPv4 address:

```
1 [ipv4-addr:value = '192.0.2.44/32']
```

The next level of complexity, as shown in Figure 3.4, includes a truncated version of the STIX patterning. This displays the pattern, and its Observation Expressions, which consist of one or more Comparison Expressions combined with a Boolean Operator and enclosed by square brackets. These expressions refine the selection of Cyber Observable data that matches the pattern.

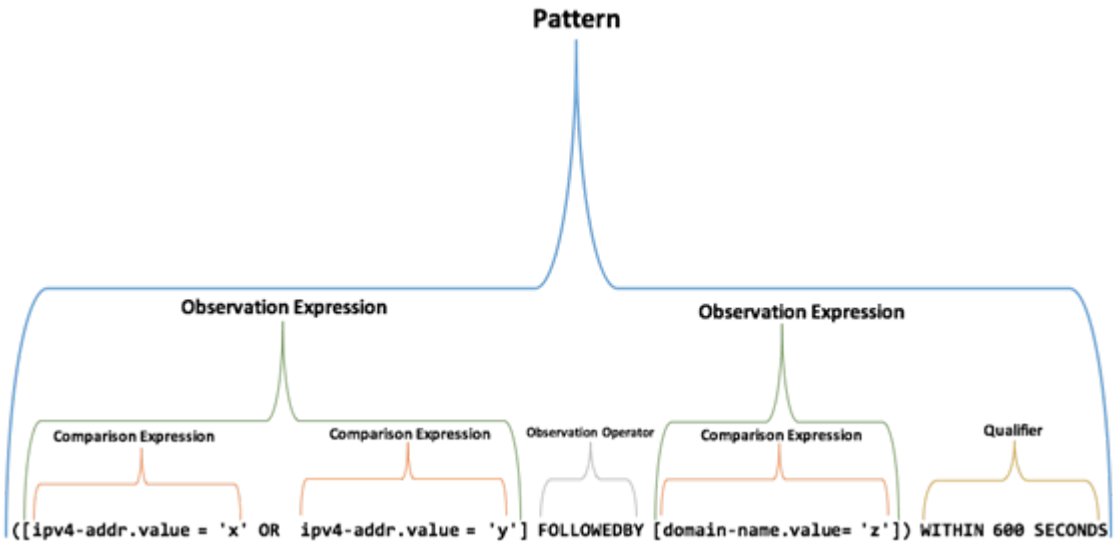

Figure 3.4: Truncated version of STIX patterning components [37].

The following STIX pattern example matches against both IPv4 and IPv6

addresses:

```
[ipv4-addr:value = '192.0.2.44/32' OR ipv4-addr:value = '
   198.51.100.77/32' OR ipv6-addr:value = 'e05f:943e:38ef:36
   d6:e016:43f1:806a:e6db/128']
```

Qualifiers can also be added to Observation Expressions, allowing additional restrictions on the data matching the pattern. For instance, a Qualifier can specify that the IP addresses must be observed repeatedly:

```
[ipv4-addr:value = '198.51.100.1/32' OR ipv4-addr:value = '
   203.0.113.33/32' OR ipv6-addr:value = 'e05f:943e:38ef:36d6
   :e016:43f1:806a:e6db/128'] REPEATS 5 TIMES
```

To finalize, STIX patterns are essential to the STIX language, allowing for the accurate specification of observable patterns in CTI. By utilizing the hierarchical structure of building blocks and various expressions, STIX patterns can accurately depict intricate cyber threat situations through a methodical approach. To delve even deeper into the composition of STIX patterning, please refer to the official documentation, which this section is solely based on [36].

3.3.5 STIX2 Pattern Python Package

The Python-STIX2 package is a powerful tool for interacting with STIX 2 content. This package provides users with broad capabilities that enable them to read and write STIX 2 content. Furthermore, the ability to interact with the STIX 2 content in an intuitive and user-friendly way makes the Python-STIX2 package a valuable tool for any organization looking to incorporate STIX 2 into their security operations [38].

The package is divided into three logical layers as depicted in Figure 3.5, each representing a different level of abstraction that is useful in several types of scripts and applications. The lowest layer, the Object Layer, is where Python objects representing STIX 2 data types are created and can be serialized and deserialized to and from JSON representation. This layer is appropriate for stand-alone scripts that produce or consume STIX 2 content or can serve as a

low-level data API for larger applications that need to represent STIX objects as Python classes.

The Environment Layer introduces several elements that simplify the handling of STIX 2 data as a component of a bigger application and as a component of a broader ecosystem for CTI. It consists of Data Source objects, which stand in for places where STIX data can be retrieved, like a TAXII server, database, or local disk, which stand in for places where STIX data can be delivered. All newly formed objects can have common properties added to them using an object factory. This layer can be used separately or in combination with other layers to form an Environment, allowing various users of a multi-user application to use different settings.

The Workbench Layer is the top layer created for a single user in a highly interactive analytical setting, like a Jupyter Notebook. It advances the STIX's lowest levels while concealing the majority of its complexity. This layer makes it simple to immediately interact with STIX data from several sources without having to build and execute one-off Python scripts because it is intended to be used directly by end users [38].

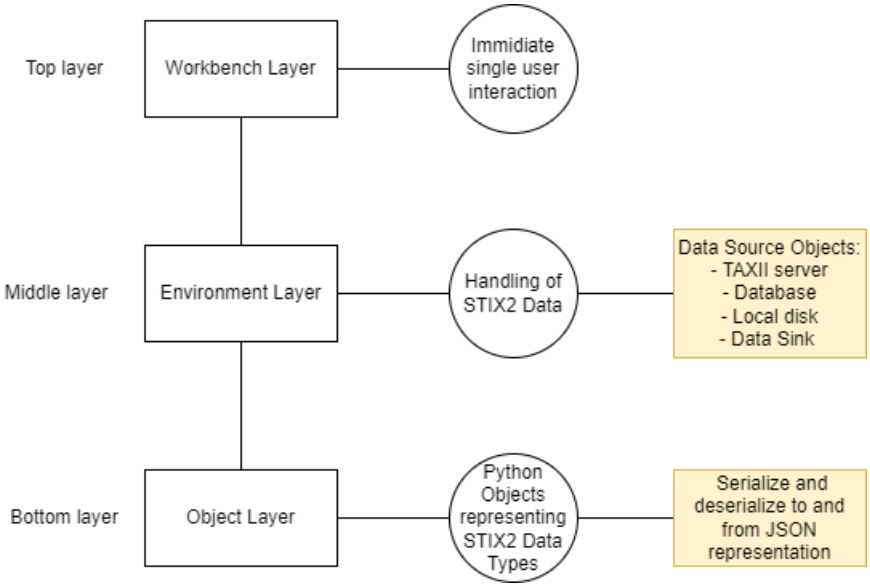

Figure 3.5: Architecture of STIX2 Pattern Python Package.

The Python package can be further utilized to write STIX2 content and map from the ACT Data Model to the STIX2 Data Model and vice versa. In this project, a mapping was created between each object in the ACT model that had a related Cyber Observable Object from the STIX documentation [30]. More information on how the mapping process was facilitated can be found in Chapter 7

3.4 STIX-Shifter

STIX-Shifter is an open-source Python library, enabling users to establish connections to various SIEM products containing data repositories, using the STIX patterning language and return results as STIX observables. The library facilitates the exchange of CTI between the different systems, regardless of the data source format or structure, and displays it in a standardized format. Using the STIX pattern language, STIX-Shifter enables users to create complex queries, retrieving specific information from different data sources. These complex queries are displayed as a STIX bundle of observable objects, conveying information regarding information observed on systems, networks,

and hosts using SCO. The library's architecture allows the integration of new connectors to extend and customize its functionality. Additionally, the library provides tools and utilities to perform testing, debugging, and troubleshooting for queries to ensure the reliability and accuracy of results from its supporting connectors.

3.4.1 Functionalities in STIX-Shifter

STIX-Shifter is, as mentioned, an open-source Python library and is available to download through pip. Once installed, STIX-Shifter can be utilized to query and manipulate STIX data. This includes querying, converting, and searching for specific patterns from supported sources.

Figure 3.6 on the next page describes the architecture behind how the STIX-Shifter Command-line Interface (CLI) works. The model depicts the three distinct functions STIX-Shifter provides: translation, transmission, and execution.

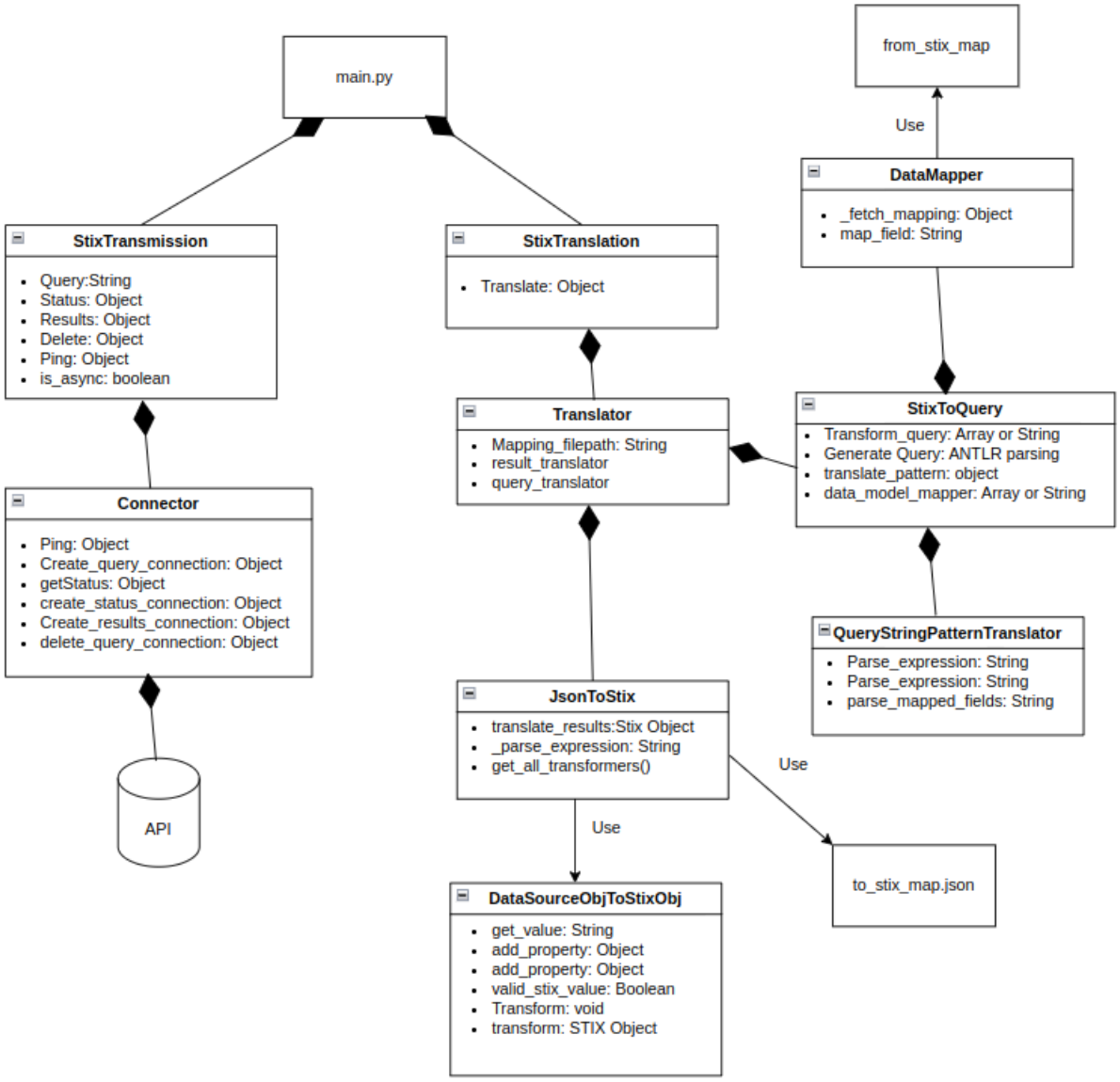

Figure 3.6: Simplified representation of the STIX-Shifter architecture, the full version can be found in their GitHub repository [39].

Figure 3.6 depicts each file and function of the STIX-Shifter library, which the main Python file depends on. The translation command allows a user to create data source queries for different supported sources and display the results from a query as a standardized STIX bundle of observable objects. The command takes a STIX pattern as an argument and creates a query for any supported con-

nectors. The result of these source queries can then be translated into a STIX observation object, displaying CTI information in a standardized format for an end user.

The transmit command allows STIX-Shifter to connect with various products that house repositories of cybersecurity data. Connection and authentication credentials are passed to the source API of the connecter where STIX-Shifter can ping a data source, make and delete queries, and check and fetch query status and results.

The execute command is used for the transmit and translation functions to work in sequence.

3.4.2 STIX-Shifter Connectors

STIX-Shifter connectors are modules within STIX-Shifter, enabling communication between different SIEM data sources. Creating a new STIX-Shifter connector expands on the data sources that the platform supports. Each connector will have a transmission and translation function that allows each data source to create a native query for each data source. Each of these queries can then be executed for each supported connector. The results are then translated back into a STIX bundle, allowing for uniform data harmonization.

3.5 ACT Platform

ACT is an open-source research project led by mnemonic AS, with the main objective of developing a platform for CTI to detect threats, espionage, and sabotage [40].

One key aspect of the ACT platform is the collection of intelligence from multiple sources. Therefore, a strict and effective Data Model must integrate this intelligence on a single platform. Furthermore, this Data Model will ensure the various intelligence sources can coexist and be analyzed cohesively. Overall, the

ACT project represents a significant effort to advance the field of Cyber Threat Intelligence and improve an organization or user's ability to detect and defend against cyber threats.

3.5.1 ACT Data Model

The ACT Data Model is created to collect CTI from different sources and gather them on one platform to remove repetitive tasks for a security analyst. Figure 3.7 displays the Data Model of ACT. This Data Model represents an object and how their relationships connect them.

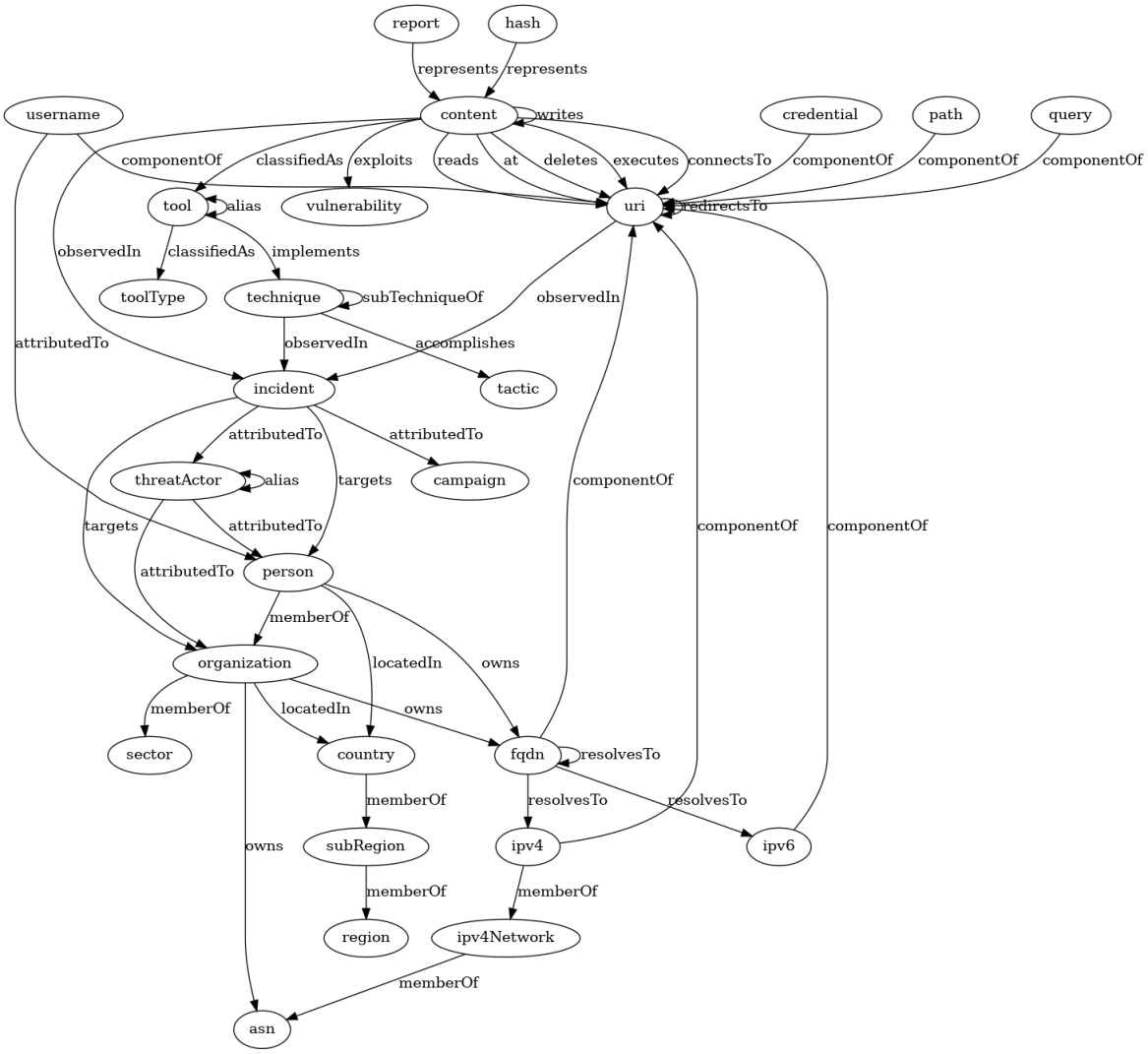

Figure 3.7: The ACT Data Model without the mentions fact [21]. The full module with mentions is a comprehensive model and not feasible to display.

3.5.2 Objects

The Data Model illustrated in Figure 3.7, represents objects and facts in a graph structure. Nodes within the graph signify objects, and the facts are the graphs creating relationships between each node. These nodes can represent a wide range of entities, like, tools, approaches, organizations, and Uniform Resource Identifier (URI)s. Each object is related to one another, as shown by the relationships that connect them, known as facts. Each object in the Data Model primarily consists of two characteristics: a name and a value. For example, an IPv4 address in the graph will have an object named "IPv4" and the value "192.168.1.1". It is important to note that objects in the ACT model are considered immutable. Objects have a permanent existence once created and cannot be altered or removed. However, a placeholder can represent an object in the graph structure when it is unknown. One can change this placeholder value with an actual value [41]. The concept of a placeholder is beneficial in cases where the values of two related objects (A and C) are known. Still, the value of an intermediary object (B) is not. In such situations, the unknown object (B) can be represented by a placeholder, which can eventually be updated with relevant information when discovered.

Objects in ACT are intended to provide a common language and definitions for describing and analyzing cyber threats. They facilitate the exchange of information about threats between different organizations and systems. They are an essential component of the ACT framework and are used to support the development of effective cyber defense strategies and identify emerging threats.

3.5.3 Facts

Facts, within the circumstance of the ACT graph, stand for the relationship among various objects, demonstrated by *componentOf* or *ObservedIn*, as illustrated in Figure 3.7. These facts define the interconnections between diverse objects within the ACT platform. A fact can be related to single or multiple objects by establishing a relationship between them. Each fact has been validated and assigned a confidence level to assure non-repudiation [42].

Facts are also immutable, but in the case of removal or need for change, it is possible with "retraction." This fact type asserts that a referenced fact is invalid or retracted.

As mentioned in Figure 3.7, another data module exists with a fact called *mentions* [21]. This fact is depicted with several outgoing edges from report and connects to almost every object in ACT. *Mentions* signifies a large quantity of data; nevertheless, these data points are generic in isolation. The particular fact merely illustrates the presence of an object mentioned in a given report.

Explanation of specific facts

Every object in ACT exists with a representing fact. This section will describe each entry used later in this thesis to understand better the information retrieved from the ACT platform.

- *componentOf* - This fact connects to the URI objects, and describes different components for a URI. This fact contains the host(domain/IP), path, and query.
- *memberOf* - This fact connects to the IPv4, IPv4Network and Autonomous System Number (ASN), describing members of an IPv4. This fact contains information on how an IPv4 can be a member of an IPv4Network, and which ASN the network is a member of.
- *represents* - This fact connects the hash to a content, describing which content a hash represents. *represents* includes, amongst others, that a hash could represent a stream segment, file, text string, or a part of content found in memory.
- *resolvesTo* - This fact connects Full Qualified Domain Name (FQDN) and IPv4, describing the process of converting a domain name to an IPv4. *resolvesTo* displays which IPv4 resolves to a domain.

- *redirectsTo* - This fact links a URI with itself, describing how a URI can redirect a user to another page.
- *deletes*, *at*, *connectsTo* and *execute* - These facts link content to an URI. The *deletes* fact describes if a file is deleting another file locally on a host. at describes "seen at" and "downloaded from." *ConnectsTo* represents if the content is seen connecting to a URI. Lastly, *execute* describes if a content executes another file, spawning another file and the like.
- *mentions* - This fact represents the relations from report to most other object types. *Mentions* describes if something is mentioned in a report, such as userAgent or Mutex.

One-legged Fact

In the context of ACT, a "one-legged fact" is defined as a fact with a single reference or association with another object. Although some one-legged facts may have two relationships to different objects, they do not inherit relationships from other facts such as the ones in Figure 3.7. Instead, they can only point to a second or, in some cases, a third fact. For example, as depicted in Figure 3.8 the path only points to *basename*. But for the URI it points to both *port* and *scheme*; for more documentation on all one-legged facts, refer to Figure 1 in Appendices. One-legged facts are distinct from other facts in that they do not interconnect with other facts within the actual ACT Data Model, Figure 3.7. Instead, they are incorporated to provide the end user additional information about the fact. The one-legged fact itself encapsulates the value.

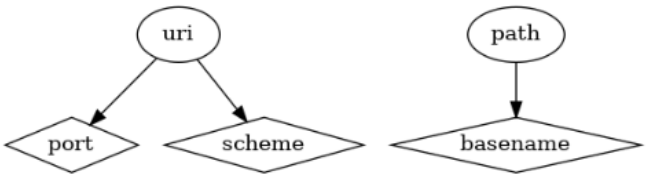

Figure 3.8: One-legged Facts: URI and Path.

Fact-Chains

The concept of "Fact Chains" is currently an experimental idea that involves a sequence of facts, in which certain elements within the chain may be represented as unknown or placeholder values. These unknowns are designated using the value "*" and are subsequently assigned a unique value "[placeholder[*]]," in which the HASH is calculated based on the incoming and outgoing paths from the placeholder. An example of this can be observed in the provided code snippet below in Listing 2, where a series of facts are created and subsequently added to a chain. The documentation notes that this feature is experimental, subject to change, and implemented on the client side. The backend of the ACT platform does not yet have the capability to understand the concept of fact chains. Additionally, it is essential to note that adding facts to a chain, as demonstrated in the example, is not atomic and may lead to inconsistencies if specific facts fail validation in the backend. Further, it might lead to empty values in the representation of data in the ACT Platform, as one can, for instance, see in Figure 3.9, where IP addresses, Tools, and Technique is empty. A quick fix in the representation of this data could be a filter option in the client, such as one can do on the main query illustrated in Figure 3.10. This paragraph is based on what is posted on mnemonic's GitHub repository [43].

```
{
facts = (
        c.fact("observedIn").source("uri", "http://uri.no").destination("incident", "*"),
        c.fact("targets").source("incident", "*").destination("organization", "*"),
        c.fact("memberOf").source("organization", "*").destination("sector", "energy"),
    )
chain = act.api.fact.fact_chain(*facts)
for fact in chain:
        fact.add()
}
```

Listing 2: FactChain example code.

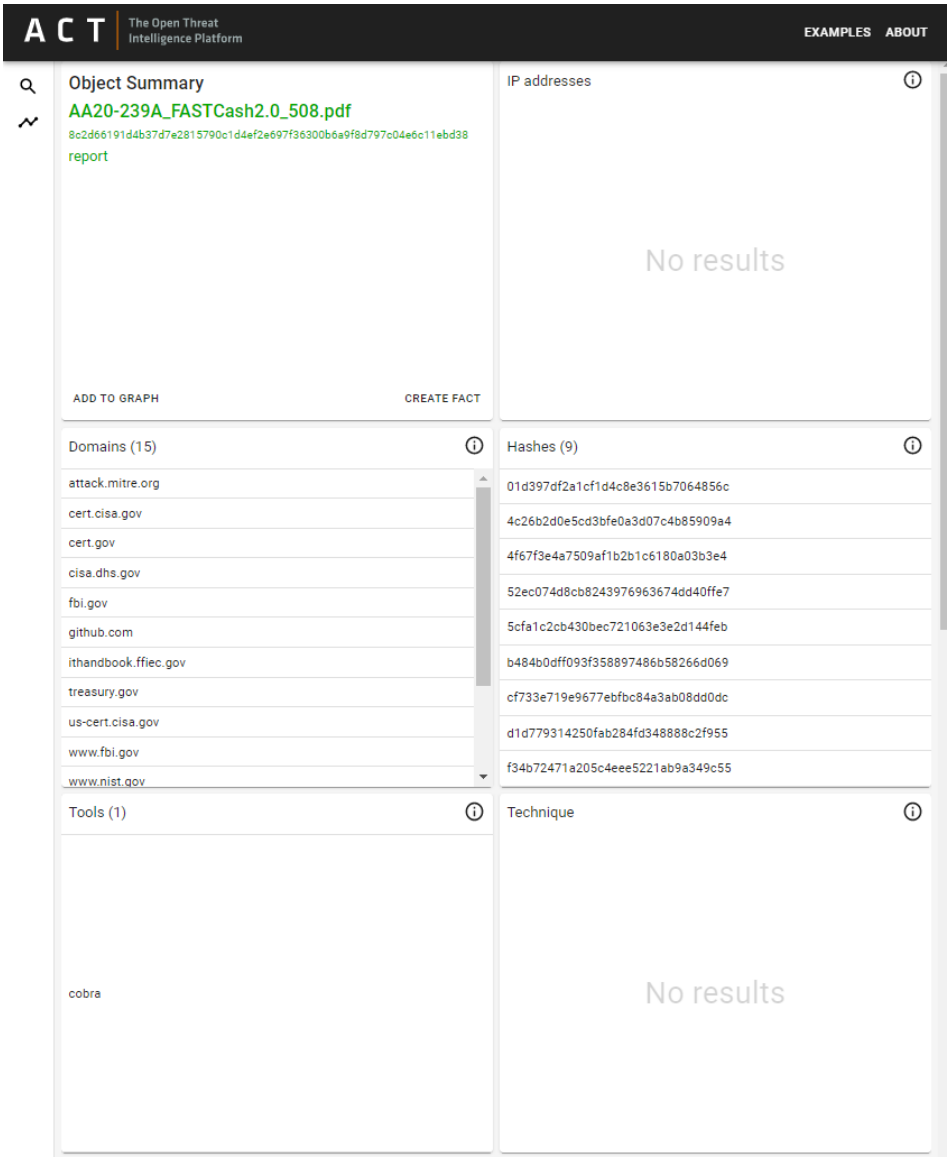

Figure 3.9: Fact-Chains with empty values.

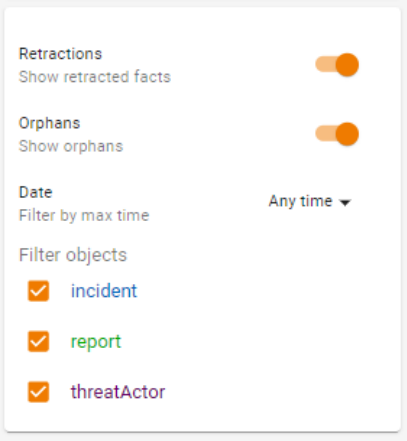

Figure 3.10: Filter option on the main query.

3.5.4 ACT API

The ACT API provides a set of functions and tools that allow developers to access and manipulate data from the platform. It allows an end-user to query, retrieve, and update information in the ACT platform.

The API utilizes a Representational State Transfer (REST) architecture, which means that it uses standard methods to make requests and receive responses. These requests allow the user to return a JSON file with the queried object, fact, or both. In addition, users can add new objects and facts with the API. These features make it an attractive option for organizations to incorporate threat intelligence data into their security infrastructure. For example, Figure 3.11 represents how to add a new fact to the platform.

```
curl -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'ACT-User-ID: 1' -d '{
  "type": "resolve",
  "sourceObject": "ip/140.82.118.4",
  "destinationObject": "fqdn/www.github.com",
  "bidirectionalBinding": true
}' 'http://localhost/v1/fact'
```

Figure 3.11: The visual representation of the generation of a new fact within the ACT platform.

3.5.5 Gremlin and Swagger

The ACT API is documented with Swagger. Swagger is an open-source tool that helps programmers develop, design, document, and use a REST API. This tool is built around OpenAPI and describes the structure and requirements to help users of the ACT API fully understand its functionality. This documentation includes each possible API operation for the ACT platform. In addition, Section 3.5.6 describes more information on specific operations.

The Swagger documentation describes operations for adding and searching for specific objects and facts in ACT. However, it is possible to traverse the entire Data Model by utilizing Apache TinkerPop. Apache TinkerPop is an open-source graph computing framework that provides a unified API for accessing and working with graph databases. Apache TinkerPop can be queried with Gremlin, a graph traversal language designed for querying and manipulating graph data.

Gremlin is a declarative language, meaning that queries are expressed in terms of the desired outcome rather than the steps needed to achieve that outcome. This allows users to focus on the problem they are trying to solve rather than the mechanics of how to solve it.

3.5.6 How to Use the ACT API

To effectively utilize the ACT API, it is necessary to employ the following objects, designed to facilitate seamless connectivity to the API. This object exposes a wide range of essential operations inherent to the API:

```
1 {c = act.api.Act("https://act-eu1.mnemonic.no", user_id = 1, log_level = "warning")}
```

The API has several operations; their Swagger documentation is available online [44], each serving a specific purpose. The students further explored only a few of the API operations. A short description is available below:

- `POST /v1/fact/uuid/{id}`: This operation is a get request that returns a fact type identified by its Universally unique identifier (UUID).
- `POST /v1/fact/search`: This operation searches for facts and returns the result with its source and destination object.
- `POST /v1/object/search`: This operation searches for an object and returns the value of the object.
- `POST /v1/fact/uuid/{fact}/meta`: This operation creates and returns a new metafact, a fact directly referencing another existing fact.

End-users can employ these operations to retrieve data from the ACT platform.

API error message

When utilizing the Gremlin language, the ACT platform will return a nested file structure in JSON format. This format will return six top-level fields containing the following:

```
{
    "responseCode": 200,
    "limit": 10000,
    "count": 1263,
    "messages": null,
    "data": []
    "size" : 1263
}
```

Within the context of the returned dataset, the initial field is the response code. This field determines the success or failure of a given query. To elaborate further, the return code is thoroughly documented in the Swagger documentation [45]. As such, these codes serve as a vital reference point for end-users, seeking to interpret the results of their search queries.

200	Query returned normally.
400	User could not be authenticated.
403	User is not allowed to perform this operation.
404	Fact does not exist.
412	Any parameter has an invalid format.

Table 3.1: Response codes in the ACT platform.

Within this project, the search queries executed responded with two distinct response codes, 200 and 403. As documented in Table 3.1, a response code 200 indicates that the query is executed successfully and returns the expected results. On the other hand, 403 implies that the user who submitted the query does not possess the necessary permissions to perform the requested operations. It is worth mentioning that the other response codes in Table 3.1 pertain to different aspects of the search query process, such as user authentication, invalid parameters, and the non-existence of the desired fact.

408	The performed graph traversal query timed out

Table 3.2: Graph query timeout.

An undocumented 408 response code also appeared during the research phase. This specific response code displayed in Table 3.2 was returned due to time out. Depicting that the ACT API could not retrieve all the values, as the graph nodes retrieved were too large.

An additional crucial field is the data field. This field contains all the graph information in a nested structure. The resulting data is presented in a JSON format and provided to give a clear understanding of which fact is connected to each object:

```
        {
            "id": "0a88082c-e84d-4851-aaae-49698dd2656e",
            "type": {
                "id": "5545881c-82ad-4b7f-9649-64c9b6ab58a0",
                "name": "resolvesTo"
            },
            "value": null,
            "inReferenceTo": null,
            "origin": {
                "id": "1dee9c92-c4b0-4fd4-84e4-9753d16811ab",
                "name": "mnemonic-pdns"
            },
            "trust": 0.8,
            "confidence": 1.0,
            "accessMode": "Public",
            "timestamp": "2022-09-30T01:21:01.256Z",
            "lastSeenTimestamp": "2022-09-30T01:21:01.256Z",
            "sourceObject": {
                "id": "abc6031e-47d9-43a4-b23a-c2b416a9df62",
                "type": {
                    "id": "8b713f00-05f8-4c20-871b-b05b5f3cdfb4",
                    "name": "fqdn"
                },
                "value": "natsumi.world"
            },
            "destinationObject": {
                "id": "f1da74bd-976f-4bab-b755-36482eef8250",
                "type": {
                    "id": "8adfa5e5-6a76-4e10-9810-a16fb1feb7aa",
                    "name": "ipv4"
                },
                "value": "192.168.0.1"
            },
            "bidirectionalBinding": false,
            "flags": [],
            "certainty": 0.8
        },
```

Listing 3: Codesnippet of full ACT fact object.

Listing 3 displays the fact *resolvesTo*, for a search on the IPv4 address "192.168.1.1". The first five lines depict the id and name of the fact, then from line 7, each object connected to the IP with the fact *resolvesTo*. Each object has a "sourceObject" and a "destinationObject." This represents how the objects are connected, meaning that the FQDN, which in this case is "natsumi[.]world" resolves to the IPv4 "192.168.0.1".

3.5.7 Querying the ACT API

There are two ways to query the ACT platform, utilizing Apache TinkerPop Gremlin language or searching for specific facts or objects by using ACT's API, and the documentation in Swagger [44].

Querying CTI from ACT by utilizing Gremlin works by breaking down queries into small, easy-to-understand steps that can be combined to explore any part of the graph. The challenge with these queries is that they are slow and return a list of each individual object related to the object. This query returns all objects around the arbitrary origin node and the object and facts for each node traversing the whole Data Model, seen related to that specific object.

The ACT API can also be queried by utilizing the REST API documented in Swagger. The Swagger documentation has several functionalities, although this thesis focuses on fact search and fact UUID search as described in Section 3.5.6. The fact UUID will return an item of class Fact. An example of returned values is displayed with its source object, representing fact and destination object:
(ipv4/185.117.73.66) -[memberOf]-> (ipv4Network/185.117.73.0/24)
More information on how these queries work will be described in Section 4.4.

3.5.8 Taxonomy

Taxonomy refers to a scheme of classification [46] which involves systematic organization and categorization of content into distinct groups or classes, an essential aspect of structuring content provided by the ACT platform. A well-

defined taxonomy facilitates efficient organization and retrieval of information [47]. In addition, taxonomies will enable the creation of mappings that illustrate common characteristics and their interconnections. The uses of taxonomies are essential in the mapping section of this thesis Chapter 7.

Chapter 4

Research

This chapter will describe the operational mechanism of the newly created ACT connector. The purpose of the new connector is for ACT to be able to share its CTI in a standardized way for other security tools or users to access the data. This chapter will primarily explain how the students created this connector and its core functionalities, shown in Figure 4.1, and explain some of the specific choices made when creating this connector. Each functionality will also briefly describe how the authors coded it. The code for this project can be found in the GitHub repository [48].

4.1 The New ACT Connector

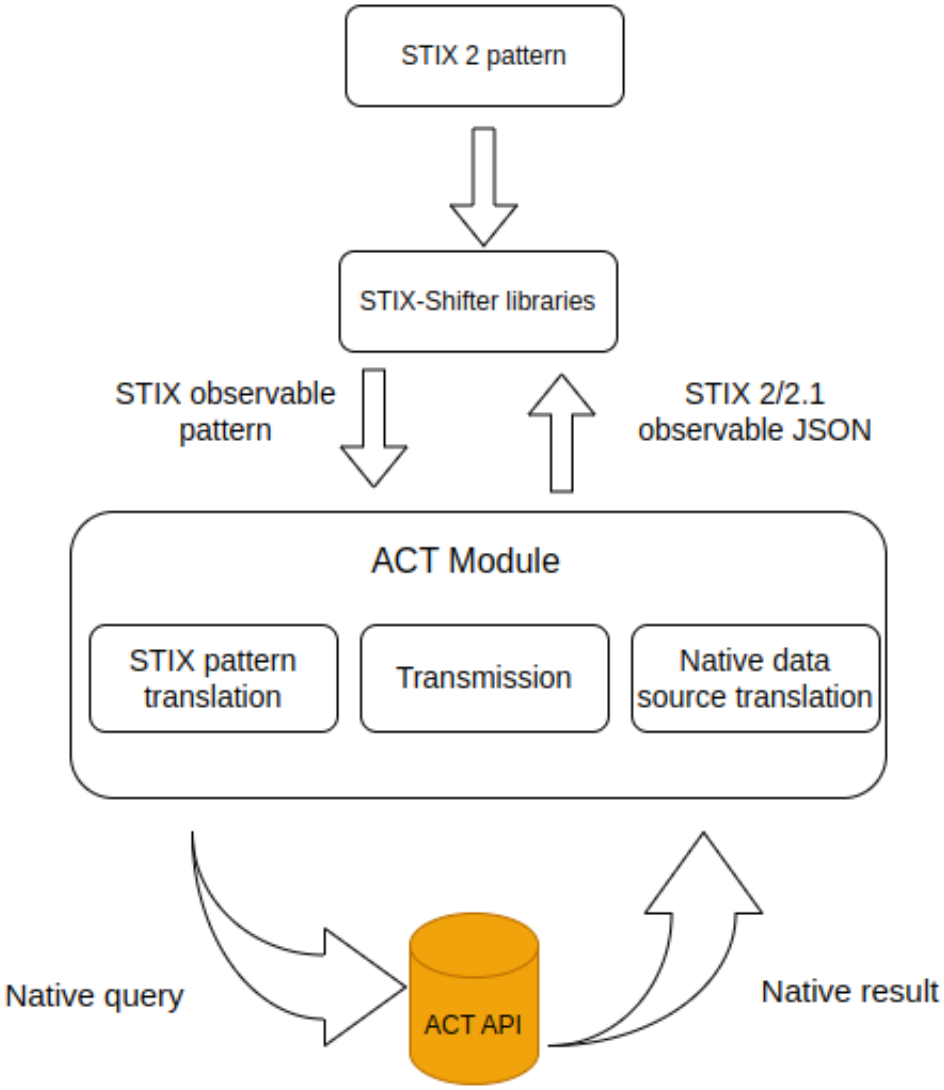

Figure 4.1: Representation of data flow from a STIX pattern to a native data source query in ACT, and back into a STIX bundle.

Figure 4.1 illustrates the entire data flow for the new ACT module in STIX-Shifter. The Figure is a specific instance representing the functionalities of the ACT connector and was created from a similar Figure found in [49]. The Figure portrays how the new connector accepts a STIX 2.0 pattern as an argument and

converts the data into an ACT native query. Next, the query is transmitted to the ACT API, which detects data matching the patterns in the query. Finally, the connector takes the native result obtained and converts it into an observable object. The conversion allows an end user utilizing ACT to display any information from the ACT API in the unified format of a STIX observable object.

4.2 Mapping Between STIX and ACT

Before creating the connector, the students mapped the objects in STIX and ACT. The mapping process entails mapping a STIX pattern to data source fields in ACT to generate a data source query. Once a new query is executed, the results obtained from the data source fields in ACT are transformed into a bundle of STIX observations. This was accomplished by mapping ACT's data source fields to their respective values in STIX version 2.0 cyber observable objects [30].

It is important to note that only objects from the ACT Data Model 3.7 are mapped to their corresponding STIX patterns, as some SCOs currently do not exist in ACT. To facilitate the mapping process, Figures 4.2 and 4.3 were created to display how an ACT object and fact is retrieved from the platform.

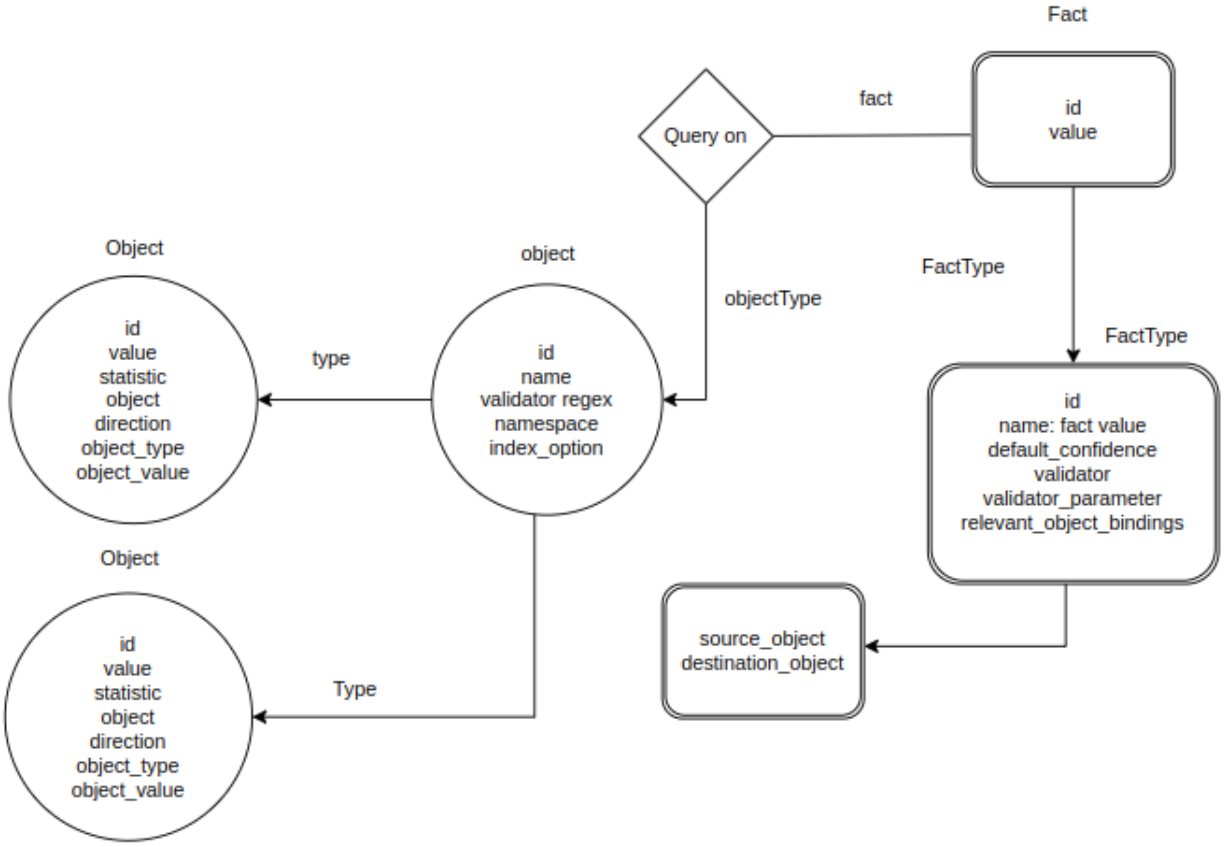

Figure 4.2: Visualization of an ACT source object and destination object, as well as its representing fact.

Figure 4.2 illustrates an in-depth representation of an ACT object and its associated fact. This Figure provides a complete overview of the data contained within a source object, a destination object, and a fact. In addition, the Figure showcases the data obtained from a query by searching for a factType, object, or objectType from the swagger documentation explained in Section 3.5.6.

In this thesis, the students decided to base queries in ACT on specific facts for objects, or as depicted in Figure 4.2, the factType. This decision was made based on thoroughly examining the query mechanisms in ACT. Firstly, querying ACT using Gremlin is known to be slow as it explores every part of the graph, as outlined in Section 3.5.7. As a result, the data returned by this method is often excessive and does not translate well into STIX bundles due to unmapped data

or the requirement for mapping with SRO to indicate self-relatedness. Querying by an object was another option; however, this approach only returns the value and fact of a specific object, resulting in a lack of overview regarding the threat landscape. After exploring all options, it was determined that querying based on the fact of an object was the most suitable solution.

By this means, the exploration of the factType and all its values are explained below:

- **id** - the UUID of the factType, this is created for each object as displayed in Figure 4.2.
- **name** - The name of the fact.
- **default_confidence** - is the confidence of where the value is returned from; it represents the source's trustworthiness.
- **validator** - This represents the regular expression that validates if a value is in the correct format. This value checks, for example, an IPv4 address and that it conforms to the correct standard.
- **relevant_object_bindings** - This value represents the bindings to source and destination object for an object value in ACT, as displayed in the Figure 4.2.

Figure 4.3 was created to depict a less complex version of a threat landscape in ACT. This representation includes the values parsed from the resultset when querying the ACT API.

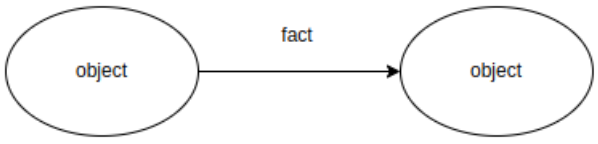

Figure 4.3: representation of two objects and their fact in ACT.

Figure 4.3 provides an illustration of a typical scenario for sharing CTI in ACT. The diagram displays the relationships between objects through their associated

fact. The search query utilized in this thesis presents comparable information. Thus, this diagram provides an overview of the data that is used to transform the ACT data source fields into a bundle of STIX observations, as well as the transformation of a STIX pattern into a data source query. The diagram was used with the full Data Model illustrated in Figure 3.7, to map entries from ACT to their respective SCO in STIX. The finished mapping can be found in the result section in Table 7.1 and 7.2.

4.3 Implementation of the Mapping

To incorporate the mapping procedures outlined in this chapter, the successful mapping found in the two tables 7.1 and 7.2 will be added to their individual JSON files in the new ACT connector, respectively called from_stix_map and to_stix_map. These files were used to transform STIX patterns to their respective data sources in ACT, and the results from a ACT query into a JSON structure of STIX properties that was used to create a STIX bundle. Listing 4 depicts the JSON representation of how a STIX pattern was translated into an ACT query.

```
1     "common-property-type":
2    {
3        "fields":
4        {
5            "value": ["act-object"]
6        }
7    },
```

Listing 4: Model of STIX pattern to an ACT object.

Listing 4 illustrates transforming a STIX pattern and its associated values into specific data source fields in ACT. The common-property-type features the patterns object portrayed in Table 7.1. Each object contains a field depicting which

ACT object the values should be mapped to. For example, if an IPv4-addr pattern from STIX is to be translated into an ACT data source, the common-property-type will be IPv4-addr, the value '192.168.1.1' and the act-object IPv4. This file only displays how the specific properties in STIX are generated into a JSON object of ACT particular properties. More information on how the actual queries in ACT are generated and look like can be found in Section 4.4.

To generate a JSON format that is compatible with STIX, Listing 5 depicts how the returned result from an ACT query is turned into a STIX JSON format that later can be used to create a STIX bundle of observed objects.

```
"ACT-Object":
[
      {
     "fields": {
         "key": "stix-object.stix_object_property",
         "object": "stix_object",
         "references": "another_stix_object"
     },
     "fields": {
         "key": "x_custom_object.property",
         "cybox": false
     }
 }
 ]
```

Listing 5: JSON representation of creating a STIX JSON format from an ACT query result.

Listing 5 demonstrates the process of converting the results from an ACT query to a STIX JSON format, which can be utilized to create a STIX bundle of observed objects. The ACT-Object is obtained from Table 7.2 and represents

the ACT object value. Listing 5 also displays two fields; each contains a key element representing a STIX object and its properties. In the key element, the STIX object represents the SCO object, and the stix_object_property depicts the mapping to the specific property value. The "object" is what holds the name of the SCO, this is only a description and can be used to refer to a different observable object if it is supported by the specific SCO. Finally, a custom object can be depicted with x_custom_object.property, but must have the cybox set to false. It is important to note that this is not a visualization of a STIX bundle. However, this can be found in the next Section 4.4.

4.4 Core Functionalities for the New ACT Connector

The new ACT connector supports all functionalities in STIX-Shifter, which includes translate, transmit, and execute. The Tables 4.1 and 4.2 respectively present a list of parameters associated with each functionality in the newly created ACT connector.

Transmit	
<connector name>	Defines the connector to be used, in this thesis, this will be ACT.
<connection object>	Contains information on how to connect to a specific data source. This needs to contain a URI or IP address.
<configuration object>	Contains authentication keys that store authentication information for the data source.
<method name>	Defines functionalities in the transmit function: ping, query, status, results, delete.

Table 4.1: List of all parameters in the transmit function.

Translate	
<connector name>	Defines the connector to be used, in this thesis, this will be ACT.
<"query" or "result">	Query: Create new data source query. Result: Translate native data source results from a query to a STIX observable object.
<STIX identity object>	An object defining which data source is being queried. This is incorporated into the finished bundle.
<STIX pattern> OR <list of JSON result>	<STIX pattern>: The pattern from STIX the user wants to translate into an ACT object. <list of JSON> result: Takes the result from a native data source query.
<options>	Takes optional arguments: stix_2.1, stix_validator.

Table 4.2: List of all parameters in the translate function.

Tables 4.1 and 4.2 explain each parameter for the two functionalities transmit and translate. The last function execute is dependent on all of these parameters to be able to run, as this function allows an end user to execute the full cycle of a connector.

Translate "query"

The "query" functionality aims to facilitate the translation of STIX patterns into ACT queries for searching the ACT platform for specific CTI. The process involves the use of Another Tool for Language Recognition (ANTLR) parsed objects, Data Model mappings, and the JSON objects from Listing 4 to map STIX object types to their corresponding fields. The translate "query" functionality consists of two scripts, one of which returns the ACT object and its corresponding fact. The second script facilitates creating the ACT query. This process aims to enable effective and efficient STIX-to-ACT query translation, enhancing the value of STIX bundle objects for threat intelligence analysis. One can create a data source query with the following command:

```
main.py translate ACT "query" '' "<STIX pattern>" '<options>'
```

The Python command takes STIX pattern(s) and options as parameters and translates the pattern into specific data source fields in ACT.

```
main.py translate act "query" '' "[ipv4-addr:value='192.168.1.1']" '
```

In this example, the STIX object IPv4-addr will create data source field entries for ACT and returns the queries depicted in Listing 6.

```
{
    "queries": [
        [
            "c.fact_search(fact_type=\"memberOf\", object_value
               =\"192.168.1.1\", limit=1)",
            "c.fact_search(fact_type=\"componentOf\", object_value
               =\"192.168.1.1\", limit=1)"
        ]
    ]
}
```

Listing 6: ACT query based on the search result from the STIX pattern ipv4-addr:value='192.168.1.1'.

To comprehend the functionality of the queries presented in Listing 6, the first step was to establish a connection to the API. The object, c, performs this. This connection can then employ the fact_search function to execute a query on a particular object, as well as all outgoing facts related to it. There is also added parameter, "limit," as this will limit the number of returned objects.

Translate "result"

The translate "result" function takes a JSON string as input and translates it into a STIX observable object. The code implementation of the translate "result" function comprises two distinct scripts. The initial script is responsible for formatting the raw data from a string into a JSON object. The second script transforms the data into the correct format; this needs to be performed if a value, for example, an integer, must be represented. For example, the following command executes the translate "result" function:

```
python3 main.py translate ACT "results" <STIX identity object> <list of JSON result>
```

This function takes the parameter STIX identity object and a list of JSON results. The STIX identity object represents which data source the data comes from, as well as its first object that gets added to a STIX bundle during result translation. In addition, the command takes the JSON formatted values from the ACT platform, retrieved from the API calls. This will then display the result as a STIX observable object.

Transmit function

The transmit command allows STIX-Shifter to connect with products that house repositories of cyber security data. By passing connection and authentication credentials to the data source APIs, STIX-Shifter can efficiently perform various actions, including pinging the data source, making queries, deleting queries, checking query status, and retrieving query results. When working with the ACT connector, it's important to note that the functions such as transmit status and transmit delete are designed to operate in an asynchronous environment. However, the ACT API is synchronous, which implies that these functions are neither applicable nor implemented for this connector. However, an explanation for these functions will still be provided and considered important.

Transmit "ping"

The ping function tests the connectivity of the ACT API to determine whether the user is connected and the endpoint is active. The ping function sends a GET method to the ACT API in this connector and waits for a response. If the response is received, the connection is working correctly. For example, the following command executes the ping function:

```
python3 main.py transmit act <connection object> "ping"
```

This command invokes the ping function and specifies the necessary parameters, including the API connection details, such as URI and credentials. Credentials in ACT are optional, as it is only required if an instance is behind a reverse proxy [43]. Overall, the ping function is a fundamental component of STIX-Shifters functionality, enabling reliable testing of API connectivity. It also

provides the user with the necessary information to determine the operational status of the API.

Transmit "query"

The transmit "query" function takes a query string as input and returns a UUID for a specific fact. This query is based on providing a data source query in ACT as an input, which then processes the query and returns a UUID, used by the "results" function in 4.4. The command can be executed as depicted below:

```
python main.py transmit act 'connection object' query "<ACT Query String>"
```

This command initiates the transmit "query" function, which will execute a query against the ACT API. This query is based on the factType search described in the API functionalities found in Section 3.5.6. The function then processes the query, returned from the ACT API, and fetches the UUID of a fact, which later can be used to retrieve queries.

Transmit "status"

The status query is designed to get the status of a specific async search. When a search is initiated, it will be assigned a unique search ID. From here, you can check the progress of the status. Which can be "COMPLETED," "EXECUTING," "CANCELLED," "SORTING," "ERROR," or "WAIT" according to the documentation in STIX-Shifter [50].

Transmit "results"

The transmit "results" is utilized to retrieve the results of a query. The function accepts a UUID from ACT as input, and a range of how many results the end-user wants, by specifying two parameters, x, and y. The query result will be sent to the translate "results" function, which displays the result in the standardized format of a STIX bundle. The query UUID is generated from the function transmit "query." The results function can retrieve the results of a previous query. For instance, in an asynchronous query scenario where an end-user requests

100 entries but only receives 50 entries, the results query can be executed to retrieve the remaining entries, starting from 50 to 100. This enables an end-user to retrieve the remaining entries of a query.

As previously stated, the ACT API employs a synchronous search process. Specifically, this means that once a search id is returned from the transmit "query" function, each query will be executed orderly if multiple is defined. This approach ensures that the execution of each id is completed before moving on to the next, guaranteeing the search process's reliability and consistency. For example, this function is executed with the following:

```
python main.py transmit act 'connection object' results "<UUID from transmit "query">"
```

Transmit "delete"

The delete function deletes the query stored in the API. It takes a query UUID as input and deletes the data corresponding to that query UUID. The delete function can delete unwanted data or clean up a previous query's results. This is irrelevant for the ACT API since it does not store previous queries.

isAsync

The isAsync function is predefined and not a query itself. This function is a boolean test to determine whether the search process of an API is synchronous or asynchronous. If the function is executed asynchronously, the user must wait for the results to be returned. It is a synchronous query in the case of the ACT API.

Execute

The last function created was the execute function. This function depends on all the functionality of both translation and transmission. This allows an end user to run the entire end-to-end flow of the ACT connector from their CLI. This command can be utilized by running the following:

```
python3 main.py execute act act <stix identity object> '
<connection-object>
' <configuration object> <STIX pattern>
```

The command line interface enables the transmission and translation module to work in a coordinated manner. The primary functionality of the command is to translate a STIX pattern into one or more native data source queries. The API processes the queries and returns a result set. During this process, a script within STIX-Shifter checks if the ACT API is either synchronous or asynchronous. Given that the API is a synchronous query, the function invokes the transmit "result" operation to consolidate the outcome of each query into a list. This list is then sent to translate "result," translating the JSON results into a bundle of STIX objects.

In summary, this function serves as a tool that facilitates the sequence of each operation created in the new ACT connector. Furthermore, the successful implementation of this function allows an end user unfamiliar with ACT to represent its CTI in a standardized format.

4.5 Utilizing Comparison Operators for Enhanced Information Acquisition

STIX-Shifter is a crucial component in modern cyber security, allowing for several platforms to share CTI information in a standardized way for other security tools to display and access additional data. As a result, a supplementary feature of the tool called comparison operators enables the mapping of STIX pattern operators to data source query operators. Comparison operators are used within a comparison expression against a set of constants [38]. This feature is typically utilized in Structured Query Language (SQL) queries, where comparison operators evaluate whether two expressions are equivalent and can filter, sort, and otherwise manipulate large data sets to retrieve specific information [51]. Although a challenge with ACT is that it does not offer any comparison operators

natively, it was added in this thesis to allow an end-user to generate multiple queries for different objects and thereby enable the retrieval of additional information.

For this project, two new operators were added to enhance the functionality of a query in ACT. The operators, namely 'OR' and 'AND,' were designed to enable a more enhanced search for additional results.

The 'OR' operator was included to facilitate the creation of distinct queries for a single IPv4/IPv6 address or network. This operator enables the user to specify multiple search criteria for a single query or define whether an IPv4 is a single address or a network address with Classless Inter-Domain Routing (CIDR) notation. This feature greatly enhances the versatility of the query system, enabling users to conduct more complex searches with ease.

The 'AND' operator was also introduced to enable the search for additional information to gain further knowledge about a specific object. In addition, the operator allows the user to search for multiple objects simultaneously, thus providing a broader understanding of the data. This includes, among others, enabling search for information about a single IPv4 address and its associated ASN by utilizing the following pattern:

```
ipv4-addr:value='196.10.119.0' AND ipv4-addr:value='196.10.119.0/24' AND
autonomous-system:number='37610'
```

This will grant the user the following observable objects:

```
"objects": {
    "0": {
        "type": "autonomous-system",
        "number": 37610,
        "name": "NTFC-ASN"
        },
    "1": {
        "type": "ipv4-addr",
        "value": "196.10.119.0"
        },
    "2": {
        "type": "ipv4_cidr",
        "value": "196.10.119.0/24",
        "belongs_to": "0"
        }
    }
```

Listing 7: Listing of the objects for a search with operators.

This feature enables users to conduct more sophisticated searches, providing more detailed and precise results. Overall, introducing these two new operators represents a significant advancement in the query system's functionality.

4.6 Testing

Two unit tests were added to ensure the proper functionality in the newly integrated ACT connector. These unit tests verify both queries and that a freshly created bundle contains the correct information.

To validate queries, the unit tests confirm the presence of all relevant facts for an object. For instance, when assessing the validity of an IPv4 object, the test verifies that the new query for IPv4 contains its corresponding fact *memberOf*. This process is carried out with the following test function, verifying each fact,

demonstrated in Listing 8.

```
for substring in expected_substrings:
    assert any(substring in query for query in queries),
    f"{substring} not found in queries"
```

Listing 8: Test function verifying facts.

The code in Listing 8 operates by iterating through each fact or outgoing edge from an object. First, it verifies that the new list created by the "translate" query script contains each fact representing the object, raising an error if not all facts are present.

The second unit test validates connection to the ACT API and verifies that a new bundle created by either translate or execute contains the necessary information. This unit test only validates that a newly created bundle includes the identity object and observed data. Asserting that a bundle contains the proper values and fields. To assess the validity of each observed data entry, the validator script provided by STIX-Shifter was employed. A detailed description of this script was further described in Section 5.4. By utilizing these unit tests, the validation that a newly created bundle contains the information necessary was further evaluated.

Chapter 5

Challenges and Interpretations

5.1 Interpretation of "path"

A path within the context of ACT comprises two distinct data sets. These include the full path of an URI and the path of a Windows system or Linux system. Regrettably, at present, ACT lacks any example values of how a path is depicted if it represents a path to a Windows or Linux system. As a result, an assumption was made by the students concerning how this should be addressed. Specifically, the new bundle was created with example data for the connector, without generating an actual query. This entailed assuming that the path always contains the basename and a hash. However, further testing with actual data from the ACT API should be performed to verify the validation of the returned data set.

```
 {
"type": "bundle",
"id": "bundle--4960da25-8e39-47c2-92bb-b3e39eb35cdf",
"objects": [
{
    "id": "observed-data--de78259e-dfb0-4d29-81f4-f1fdf686cfbf",
    "type": "observed-data",
    "created_by_ref": "identity--3532c56d-ea72-48be-a2ad-1a53f4c9c6d3",
    "created": "2023-04-19T18:28:22.817Z",
    "modified": "2023-04-19T18:28:22.817Z",
    "objects": {
        "0": {
        "type": "file",
        "hashes": "b42ec8b47deb2dc75edebd01132d63f8e8d4cd08e5d26d8bd366bdc5"
        },
        "1": {
        "type": "directory",
        "path": "/home/user/Downloads"
        },
        2": {
        "type": "file",
        "name": "eicar.txt"
        }
    },
    "first_observed": "2023-04-19T18:28:22.817Z",
    "last_observed": "2023-04-19T18:28:22.817Z",
    "number_observed": 1
    }
    ],
"spec_version": "2.0"
}
```


Listing 9: Depiction of EICAR test file to verify the mapping of path.

Listing 9 depicts the verification of the object path to its representing SCO. This was performed with the European Institute for Computer Antivirus Research (EICAR) test file, an anti-malware test file used to test that a malware detection scanner is functioning correctly [52].

5.2 Interpretation of "content"

The *content* fact in ACT stores additional information regarding a file hash or a report. As this fact is vague, a more precise description of the fact *content* is important to describe to enable sharing of information to STIX. To comprehend what *content* is in this the scope of this analysis, this report will refer to the work of Siri Bromander in "Investigating Sharing of Cyber Threat Intelligence and Proposing A New Data Model for Enabling Automation in Knowledge Representation and Exchange". In the context of CTI, information being handled is limited to files and segments of data streams, text strings, and parts of the content found in memory. This is all classified as *content* [18], but should not be all classified as files. Further on, when considering files, it is important to note that their uniqueness is based on multiple attributes. A unique file is characterized by a combination of the filename, the file's actual content, and the content's location [18]. These factors define a file's distinctiveness when referred to as unique. To illustrate this, consider the example of a path on two different Linux systems, such as /etc/home is the same in a given situation, specifically their name and content, but they are still not the same file since they are associated with separate machines.

5.3 URI and FQDN Standards in ACT

In certain scenarios, SCOs within STIX must adhere to distinct standards or requirements. Within the scope of this thesis, two specific SCOs, namely domain-name, and URL, are subjected to conform with designated standards. As ACT does not contain any information on which standards it follows for URI and FQDN, a thorough examination of the source code for ACT was performed.

When searching through the different repositories in GitHub, the codesnippet depicted in Listing 10 was found on how a URI was retrieved from different third parties:

```
my_uri = urllib.parse.urlparse(uri)
scheme = my_uri.scheme
```

Listing 10: visualiztion of function dependent on the import urllib.parse [53].

Listing 10 displays a function dependent on the import function urllib.parse. This module defines how to break down a URI string into different components, such as path, addressing scheme, network locations, and many others [54]. This module's documentation states: "Changed in version 3.3: The fragment is now parsed for all URL schemes (unless allow_fragment is false), in accordance with RFC 3986. Previously, an allowlist of schemes that support fragments existed." [54]. As this is stated in the documentation and is utilized by ACT, this means that the mapping between a URI object in ACT and the SCO URL is possible, as both conform to the standard RFC 3986. This standard specifies the URI syntax and process for resolving URI references that might be in relative form, along with guidelines and security considerations for the use of URI on the internet [55].

For a domain name in STIX, the value of a domain name **MUST** conforms to the standard RFC 1034 [56], and for sub-domains RFC 5890 [57]. This comprised a challenge when translating an FQDN from ACT into a STIX domain-name object, as retrieving information from ACT does not conform to any standards and may contain an IPv4 address and not the FQDN. This aspect was further elaborated upon in Discussion 8.2.1, emphasizing the necessity of incorporating a validator within ACT to effectively ensure the validation of the FQDN object.

5.4 Validation Script for STIX 2.x

STIX-Shifter includes a script for validating a created STIX bundle, found in their repository [58]. This validator script checks that the created bundle from

STIX-Shifter conforms to the requirements specified in both STIX 2.0 and 2.1. Additionally, it checks conformance with requirements that cannot be specified in a JSON schema and establishes "best practices." This script validates two types of requirements:

- mandatory "**MUST**" requirements - Specific Python functions to check if the JSON schema meets the mandatory "**MUST**" requirements [59].
- recommended "**SHOULD**" best practice requirements - requirements checked by Python, these can be ignored, but are all a part of the "best practices" for creating a new bundle[59].

In cases where a bundle fails to meet these mandatory and recommended requirements for a STIX bundle type 2.0, these requirements are printed in a CLI. However, executing this script was faced with a challenge. The validator script includes a Python module not compatible with the latest Python version. To overcome this obstacle, the deprecated 3.9 version of Python was installed, and the bundle_validator script was modified to use the deprecated version instead. Additionally, each pip requirement was also downloaded with the deprecated 3.9 version.

Chapter 6

Knowledge Acquisition

6.1 Understanding Identity Object in a Bundle

When generating a new bundle in STIX-Shifter, it is required to add the parameter <STIX identity object>, described in Table 4.2. This object describes the connector responsible for creating the bundle, and Listing 11 illustrates this:

```
{
    "type": "identity",
    "id": "identity--3532c56d-ea72-48be-a2ad-1a53f4c9c6d3",
    "name": "ACT",
    "identity_class": "observed-data"
},
\label{identity object in \gls{stix}}
```

Listing 11: Code snippet of the identity-object in a STIX bundle.

This object is visualized with its type, identity, a UUID, the connector's name, which here is ACT, and the identity class, containing information that the objects here are observed data. This representation composes a challenge, as when validating a STIX 2.0 bundle of observed data, the identity class is interpreted as an SDO and returns the errors in the Figure 6.1:

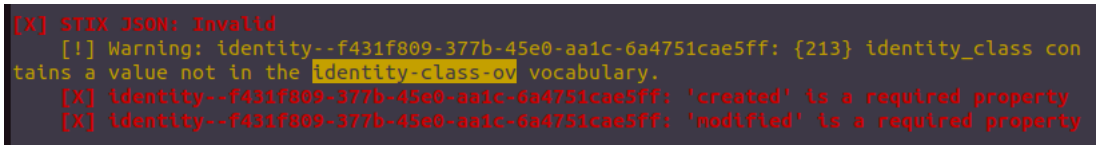

Figure 6.1: Error message from the validator script.

In STIX, the identity object represents actual individuals, organizations, or groups to define the target of the attack, information sources, and object creators [60]. The error message in Figure 6.1 states that when creating an identity object, it must contain the common properties, created and modified, representing when the identity object was created and last modified. However, in this thesis, and for STIX-Shifter connectors, an identity object is the representation of the source creating the query, as well as a representation that the objects are observed data of the type SCO. The students, therefore, decided to omit this value, as this thesis focuses on the observed data and not SRO and SDO. However, it is worth noting that this is present when creating a new bundle. Still, this value was omitted for validation to verify that the observed data is in the correct format and contains valid data.

6.2 Utilization of STIX Python Library and the ACT API

As part of the learning process for coding related to this thesis, the students developed several scripts to experiment with the STIX Python library [61] and the ACT API [43]. Despite not being incorporated into the final product, it served to gain a deeper understanding of how to extract and manipulate CTI data from various sources. It also gave a deeper understanding of how the ACT API works and how the STIX Python library works. The script uses the ACT API to query the graph database written in Gremlin language for information on a given fact and the STIX Python library to parse and process the results. An example of this could be "("memberOf," "154.213.21.0/24", "ipv4Network")" that makes a call to the ACT API and returns a Gremlin result like displayed in Listing 12.

```
(ipv4Network/154.213.21.0/24) -[memberOf]-> (asn/136933)
(ipv4Network/154.213.21.0/24) -[memberOf]-> (asn/136933)
(ipv4/154.213.21.73) -[memberOf]-> (ipv4Network/154.213.21.0/24)
(ipv4/154.213.21.73) -[memberOf]-> (ipv4Network/154.213.21.0/24)
(ipv4/154.213.21.70) -[memberOf]-> (ipv4Network/154.213.21.0/24)
(ipv4/154.213.21.70) -[memberOf]-> (ipv4Network/154.213.21.0/24)
```

Listing 12: Examples of Gremlin Queries in ACT.

```
c = act.api.Act("https://act-eu1.mnemonic.no", user_id = 1, log_level = "warning")
    path = c.object("ipv4Network", "154.213.21.0%2F24").traverse('g.bothE("memberOf")
    .bothV().path().unfold()')
```

Listing 13: Snippet of Python script querying the ACT API.

Based on the following code snippet from [act_objects.py] in Listing 13. An idea was to further develop and manipulate the output based on the three inputs "ipv4Network, 154.213.21.0/24, memberOf", that in the script is called *act_type* which is the *memberOf* value related to the mapping in the ACT Data Model, Object which is the initial value being queries, which in this case is an ipv4-network address, lastly the metaFact is what type of Object the query withholds, therefore the ipv4Network value to gives information to the ACT API what data is withdrawn displayed in Table 6.1.

Name	**Input value**
act_type	memberOf
Object	154.213.21.0/24
metaFact	ipv4Network

Table 6.1: Table that displays the name of placeholders and their related input values

The idea was to make [act_backend.py] return a JSON object that could be used to make STIX bundle objects, where the first three values, type, value, and id, are what is being queried. The following values are type and value, which is other instances of relationships to the first entry:

```
[
    {
        "type": "ipv4Network",
        "value": "154.213.21.0/24",
        "id": "affb7536-26f7-40e9-92a5-4a6a18952df6"
    },
    {
        "type": "ipv4-addr",
        "value": "154.213.21.70"
    },
    {
        "type": "ipv4-addr",
        "value": "154.213.21.73"
    },
    {
        "type": "autonomous-system",
        "value": "136933"
    },
    {
        "type": "ipv4-addr",
        "value": "154.213.21.207"
    }
]
```

Listing 14: JSON object that returns data from ACT that could be used to make STIX Bundle Objects.

The following code [stixbundle.py] took this JSON object and turned it into a STIX bundle object containing the relevant STIX objects and relationships. The

code function extracts relevant data from the input, such as the observable type, observable value, and related IPv4 addresses. For relevant data, it creates, for instance, an Identity object for the Threat Actor. Otherwise, it creates an Identity object for the network. The function then generates a list of relationships and populates it with information showing that each related IPv4 address is either linked to the network or the Threat Actor. If the input data includes an ASN, the function adds a "belongs-to" relationship to the list of relationships and creates an Identity object for the ASN. The result is that it parses input data, creates relevant STIX objects, and defines relationships among them. The resulting STIX bundle can represent a cyber security threat or incident.

6.3 MYSQL and ACT Connector

Connectors may exhibit different modalities when it comes to approaches to handling data. This underscores the importance of participating in sharing information with the same language by adopting STIX. In this chapter, a comparative analysis will be conducted between the final result from the newly developed ACT connector, with the existing MySQL connector. The MySQL connector is documented and tested in Section 2.4.

6.3.1 Bundle in ACT and MySQL

To compare the two connectors, a MySQL database was set up accordingly and propagated with example data from the .csv file found in the STIX-Shifter GitHub repository [62]. This analysis will provide examples to demonstrate the functionality of the ACT connector and compare it to the MySQL connector. Furthermore, this comparison aims to view the differences and similarities between the two connectors.
The executed query for the ACT connector is represented as follows:

```
python3 main.py execute act act '{"type": "identity," "id": "identity--
f431f809-377b-45e0-aa1c-6a4751cae5ff","name": "ACT","identity_class":
"observed-data"}' '{"host": "https://act.mnemonic.no"}' '{"auth": {"user_id":
```

```
1, "log_level": "warning"}}' "[ipv4-addr:value='192.124.249.137' AND domain-
name:name='example.com' AND autonomous-system:number='11955' AND
file:hashes='230f5db4a9f3b09b85d8b66171f4b73e']"
```

The executed query for the MySQL connector is represented as follows:

```
python3 main.py execute mysql mysql '{"type": "identity","id": "identity--
f431f809-377b-45e0-aa1c-6a4751cae5ff","name": "mysql","identity_class":
"system"}' '{"host": "localhost", "database":"new_database", "options":
{"table":"my_table"}}' '{"auth": {"username":"sammy", "password":"password"}}'
"[ipv4-addr:value = '213.213.142.5']"
```

Data	**ACT Connector**	**MySQL Connector**
url	"type": "domain-name", "value": "www.example.net /download/path/file.exe"	"type": "url", "value": "www.example.net"
hashes	"type": "file", "hashes": "230f5db4a9f3b09b 85d8b66171f4b73e"	"type": "file", "name": "calendar.doc", "hashes": { "SHA-256": "a8db77b872512df 0fd15943a79efb4e16c745cd81 22efaf948b3c56d463e4b70", "MD5": "230f5db4a9f3b09b 85d8b66171f4b73e"
ipv4	"type": "ipv4-addr", "value": "213.213.142.5"	"type": "ipv4-addr", "value": "213.213.142.5"
fqdn	"type: "domain-name", "value": "example.net"	"type": "url", "value": "www.example.net"

Table 6.2: Table comparing the ACT connector's resulting bundle with MySQL.

The query executed for both ACT and MySQL is compared in Table 6.2. As lis-

ted in the first entry for "URL," the data bears a resemblance. The only notable difference is the minor discrepancy in how ACT stores the complete URL and MySQL only stores the FQDN.

The data entry for hashes slightly varies in which the two data sources store data. For instance, the ACT connector does not provide information on what type of cryptographic algorithm is employed, whereas the MySQL connector does, as well as another entry. Concerning the third entry in the Table, IPv4, the storing of IPv4 addresses is identical for both connectors.

Finally, for the last entry, there is only a minor difference in the type, whereas, for ACT, it is represented as domain-name, and in MySQL, it is represented as URL. It is important to note that this analysis represents a limited data sample and may not accurately reflect the complete picture. However, it serves as a clarification of the different connectors on how they store information.

In this situation, sharing information highlights the need to alleviate matters about how two different Data Models preserve, retrieve, and display data. The information can be easily accessed and efficiently utilized in the analysis using STIX-Shifter and STIX to ensure that all parties involved communicate using the same language. Furthermore, this removes the need to understand multiple syntaxes to attain helpful information from different organizations.

Chapter 7

Results

This thesis highlights the successful implementation of a new module in STIX-Shifter, which facilitates the creation of a query for the ACT API and generates a STIX bundle of observable objects from the resultant data source. The module is created as instructed in Chapter 4. The new module successfully created a new entry for each query returned by the ACT API. This means that for a single query in ACT, a new STIX bundle object was successfully created for both the source and destination objects. Furthermore, operators were incorporated to generate supplementary queries in ACT, thereby improving the representation of a STIX bundle of observed objects. When utilizing the translate functions described in Chapter 4, a new query is created, but the user must then execute these queries. However, employing the execute function allows a user to run the entire cycle of the new ACT connector. This will result in a new STIX bundle.

7.1 Mapping STIX Pattern to Data Source Queries in ACT

Table 7.1, depicts the successful mapping of STIX patterns to data source fields in ACT, which is utilized for creating queries for the ACT API. As explained in Section 4.2, some patterns and SCOs in STIX does not exist in ACT. The mapping represented in the two Tables 7.1 and 7.2 is, therefore, only a depiction of the objects in ACT that have a matching pattern and SCO in STIX.

Number	**STIX**	**ACT**	
	Pattern	Object	Fact
1	IPv4-addr:value='ipv4 address'	["ipv4", "ipv4Network"]	memberOf componentOf
2	ipv6-addr:value='ipv6 address'	["ipv6"]	resolvesTo
3	url:value='URL'	["uri"]	redirectsTo accomplishes
4	directory:path='path'	["path"]	componentOf
5	file:hashes='hash'	["hash"]	represents
6	domain-name:value='domain'	["fqdn"]	resolvesTo componentOf
7	autonomous-system:number='as'	["number"]	name
8	mutex:name='mutex'	["mutex"]	mentions
9	directory:path='OS path'	["path"]	componentOf
10	user-account:user_id='user-id'	["username"]	componentOf

Table 7.1: Representation of mapping between a STIX pattern and its corresponding ACT values.

The Table 7.1 comprises a set of numerical entries, each representing a STIX pattern that is successfully translated to its representing ACT data source. This resulted in a functional translation of STIX patterns to data source queries in ACT. Each entry represents a single pattern, its value, and its corresponding object in ACT. The depiction translates the pattern into an object in ACT, which adds its outgoing fact to create a data source query for the ACT API.

The initial entry in the Table depicts the mapping between an IPv4-addr pattern to an ACT object. Notably, as the SCO accommodates both IPv4 and IPv4Network with a CIDR notation in a single object, a list is developed to establish a correspondence between these values. This enabled the creation of two observation expressions for IPv4 and IPv4_CIDR, joined by a Boolean 'OR' expression and enclosed within square brackets to enable ACT to create either an IPv4 or IPv4 network query.

The second and eighth entries, IPv6 and mutex, do not comprise any outgoing edges as displayed in Figure 3.7, thus posing a challenge for the query method implemented in this thesis. These specific ACT objects are queried by their incoming facts to address this issue.

In the sense of IPv6, a missing functionality was identified. IPv4 object in the Data Model 3.7, composes a relationship for its representing network. However, for IPv6 this is not the case. This is further discussed in the future work Section 11.1. However, until this functionality is developed, an IPv6 object is now queried by its incoming fact *resolvesTo* to depict which FQDN the IPv6 address object resolves to.

7.2 Mapping of Data Source Results from ACT to STIX

ACT queries are returned as a *actresultset*, which contains data displayed in Figure 4.2. However, in this thesis, the included information is the source and destination objects, as well as the fact representing their relationship. To transform these data source results into a STIX bundle, a mapping of each object in ACT to its corresponding value in SCO was performed. Table 7.2, depicts the comprehensive mapping of the data source object in ACT to its corresponding values in SCO. The Table emphasizes how a specific data source object in ACT was translated into a JSON formatted text, used by STIX-Shifter to create a STIX bundle of different objects. It is important to note that this mapping is one-to-one and does not portray how a finalized STIX bundle of objects is depicted.

Number	**ACT**		**STIX**	
	object	Facts	object	properties
1	ipv4	memberOf componentOf	ipv4-addr	type value
2	ipv4Network	memberOf	ipv4_cidr	type value belongs_to_ref
3	Hash	represents	file	type hashes(hashes)
4	FQDN	resolvesTo componentOf	domain-name	type value resolves_to_refs
5	ASN	name	autonomous-system	type number
6	Mutex		mutex	type name
7	URI	redirectsTo accomplishes	URL	type name
8	Path	componentOf	Directory	type Name
9	username	componentOf	user-account	type user_id
10	ipv6		ipv6-addr	type value
12	userAgent	mentions	http-request-ext	key request_header

Table 7.2: Representation of mapping between ACT data source results and its corresponding SCOs in STIX.

Number	**ACT**		**STIX**	
	object	Facts	object	properties

1	content	at deletes connectsTo executes	x_act.content	type cybox property

Table 7.3: Representation of mapping between ACT data source results and custom objects.

Table 7.2 comprises a collection of numerical entries that represent an ACT object and its corresponding SCO in STIX. Each listed entry includes the respective ACT object and its associated facts, as well as the corresponding SCO object and its properties as described in the STIX version 2.0 specification document [30]. It is worth mentioning that SCO are characterized by many object-specific properties. Fortunately, the majority of the object-specific properties are OPTIONAL in STIX. This made it possible to omit specific values when representing a SCO, allowing for flexibility in presenting the objects without all the optional properties.

Each object in Table 7.2 is depicted as one-to-one. The Table presents how a single object in ACT was translated into its corresponding SCO and its value. However, some objects in ACT do not have an existing corresponding SCO. These values are therefore represented as source-specific objects or custom objects in Table 7.3. This value was mapped by setting the *cybox* to false, and a key with an identifier as a custom object. this is represented in the Table as *x_act.<act-object>*. When *cybox* was set to false, the STIX-Shifter will interpret the object as a custom object, and not a cyber observable expression.

One of the challenges when mapping object-specific properties in SCO, was that they must conform to certain standards to be represented in a STIX bundle. First, the fourth and seventh entries, domain name, and URL must be represented by conforming to the RFC 1034 and RFC 3986 standards. The standard followed by ACT is undocumented. However, this matter is addressed in Section 5.3.

The *content* object represents the instance custom object. Further details regarding the interpretation and taxonomy of *content* and its respective mapping possibilities are provided in Section 5.2. Following the STIX 2.0 documentation, a file's content is represented as a mime-type in the SCO file [30].

7.3 Mapping of "one-legged facts"

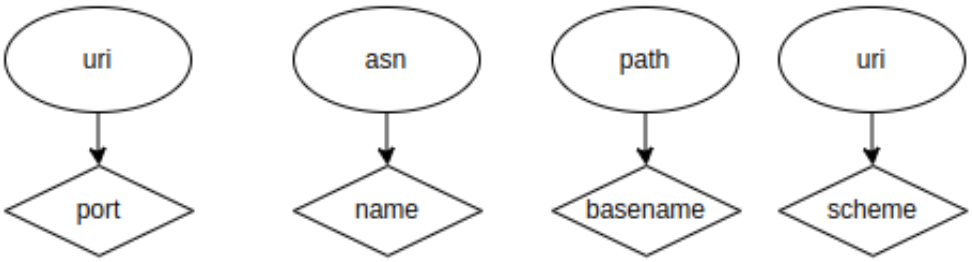

Figure 7.1: One-legged facts in ACT.

ACT contains different "one-legged facts", which are singular pieces of information that cannot be queried directly. Consequently, it was determined that incorporating these one-legged facts all in one section would be optimal. In addition, these one-legged facts contain useful supplementary information for other objects. The one-legged facts in ACT that were mapped to STIX are the following: *basename*, *port*, *scheme*, and *name*, as displayed in Figure 7.1. The *basename* is derived from a *path* and contains the filename from a specific path. The *name* is derived from several ACT objects, including *ASN*. As only *ASN* was mapped by these values, this is the only one depicted. Lastly, the *port* was derived from a *URI*. Table 7.4 depicts the mapping performed between one-legged facts in ACT and its corresponding object and properties for a SCO.

Number	**ACT**	**STIX**	
	One-legged fact	Object	Property
1	port	x_act.port	port

2	name	autonomous-system	name
3	basename	file	name
4	scheme	x_act.protocol	protocol

Table 7.4: Representation of mapping between ACTs one-legged facts and its corresponding SCOs in STIX.

Table 7.4 comprises the collection of one-legged facts in ACT and how they were mapped to their corresponding object and properties in STIX. Specifically, the fourth entry pertains to the property *scheme*, which has a corresponding SCO property called *protocols*. However, a *scheme* in ACT does not necessarily conform to the standard.

The fourth entry *scheme* could potentially be mapped to *protocols*. However, this requires adherence with the IANA service name, and the default scheme in ACT is a network, which does not conform with the standard. Therefore, the *scheme* entry was added as a custom object to minimize information loss. Another issue with mapping *scheme* to *protocol* was that protocols are a property in the STIX object network-traffic, which requires the inclusion of a source IP or destination IP. ACT does not store this information, and it is therefore not possible to translate *scheme* to *protocols*, as defining network traffic with only a port is not meaningful and will not be validated.

Additionally, when mapping the one-legged fact *port* from ACT to STIX, the same issue as *scheme* was detected. It is possible as the ACT platform stores *port* as a fact. However, when validating the bundle, it fails as a network-traffic object requires either the destination or source ip of the traffic.

7.4 Mapping Coverage

Within the scope of this thesis, the effective execution of mapping SCO is based on the Data Model from the official documentation from ACT displayed in Figure 3.7. Furthermore, this adapted model is further reconstructed to

simplify visualization with color coding. Within the visual representation in Figure 7.2, green nodes are designated as successful implementation of mappings, while yellow nodes are used to distinguish those who are SDO and not implemented as of yet. As addressed earlier, this is not supported in STIX-Shifter, and potential solutions will be discussed in Section 8.2.2. Nonetheless, every possible SCO and their representative values in ACT is mapped and implemented in the newly created connector.

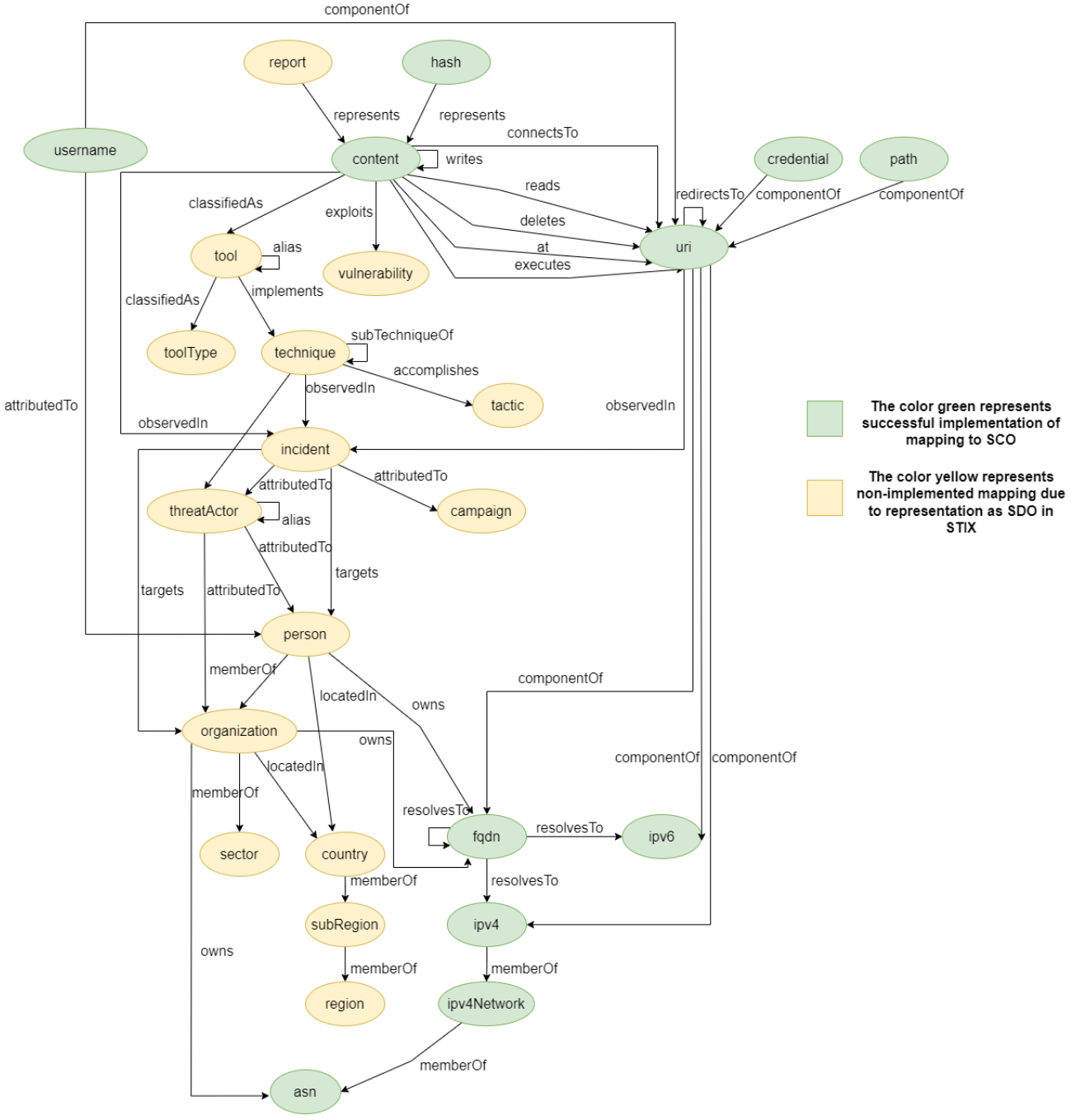

Figure 7.2: Color-Coded ACT Data Model (without mentions).

Chapter 8

Discussion

This chapter will expound upon the results from Chapter 7 and an extensive exposition of outcomes obtained from the newly created ACT connector. In addition, it gives an in-depth description of different edge cases and error handling when utilizing the connector and delves into the limitations concerning the connector.

The ACT connector, as for now, only enables users to view a set of observed objects, represented in Table 7.2. However, the results returned from ACT do not fully depict a threat landscape. ACT is useful to depict the full representation of a threat landscape. In contrast, it stores information notably on how an end-user could view, amongst others, threat actors and which organization it is seen related to. However, STIX-Shifter has not yet supported STIX 2.1 and SDOs are necessary. As written in their latest documentation of 2.1: "Higher Level Intelligence Objects that represent behaviors and constructs that threat analysts would typically create or work with while understanding the threat landscape." [63]. This was further explored in Chapter 11. In conclusion, this means that, for now, it is impossible to recreate a search in the ACT platform with the connector created in this thesis.

8.1 Error Handling

As delineated in Chapter 7, the created ACT connector facilitates the querying for patterns specified in Table 7.1, and the creation of a bundle from the corresponding results in Table 7.2. However, there are some instances where the connector may return a divergent data set. The first discovery pertains to models where a user searches for a pattern absent in ACT. For instance, an empty object will be returned if an end-user queries the IPV4 address "65.45.32.12". The rationale behind this outcome is attributed to the approach employed by ACT for storing and returning objects. When coding the connector, a challenge occurred when searching for several facts for an object, as it was uncertain that the ACT platform stores data about the specific presence of an object. For instance, given an IPv4 address, some values may contain the fact *componentOf* pointing to an URI. These scenarios will then return a specific UUID "0000d000-000a-0000-0000-000000000001". If this is the case, the program would halt and not produce the bundle, even though other objects and facts are present. Therefore, placeholder values were added for such facts to retain the already obtained data. Nonetheless, in the case of querying for an object, not in ACT, this will result in an empty bundle and not an error message. A similar case was tested against the MySQL connector to verify that any other connector performed this. This gave a similar result: an empty object was depicted when an object not in the MySQL database was queried.

The second case was related to searching for a pattern not mapped to data sources in ACT. This will result in the following error message:

```
2023-04-26 11:40:44,143 stix_shifter_utils.stix_translation.
   stix_translation_error_mapper ERROR    received exception
   => DataMappingException: Unable to map the following STIX
   objects and properties: ['mac-addr:value'] to data source
   fields
```

This exception conveys that the STIX object and property could not be mapped to the data source field in ACT. However, no action was taken in response to this exception, given that the values under consideration do not exist in ACT,

and searching for them will invariably yield no data.

8.2 Limitations

```
{
    "queries": [
        [
            "c.fact_search(fact_type=\"resolvesTo\", object_value
               =\"evil.org\", limit=1)",
            "c.fact_search(fact_type=\"componentOf\", object_value
               =\"evil.org\", limit=1)"
        ]
    ]
}
```

Listing 15: ACT query presenting the restricted limit of one.

Like any study exploring new fields, this thesis is subjected to limitations. The newly presented ACT connector is currently limited to only retrieving a single object rather than multiple. When creating a new query for the ACT API, the limit parameter is for now restricted to only one, as depicted in the Listing 15. This limitation is because most SCOs do not conform to the data structure list to display its values and must be of the data type, string, or integer.

An additional limitation identified in the ACT platform pertains to the absence of the properties first_observed, last_observed, and number_observed. These values are a part of the bundle as depicted in Listing 16. These fields are properties of the SDO observed-data and represent, respectively, when the SCOs was first observed, last observed, and the number of times each object present in a bundle was observed. To incorporate these values, they have to be mapped in the to_stix_map. However, as these values are currently unavailable in ACT, this mapping must be implemented in the future. Given that these values, as for now, are not mapped. STIX-Shifter will write the current time instead of the actual data for these fields.

8.2.1 Constraint Regarding FQDN

The successful mapping presented in Chapter 7 did not result in any data loss. This implies that, in the conversion of an ACT object to its corresponding SCO or custom object, there was no loss of data that was intended to be converged. However, an issue was encountered when converting FQDN.

The object FQDN in ACT stores information not just on a full domain name but in a few scenarios. It also depicts the IP address. This is a result of the ACT API not validating a new FQDN, meaning any value can be added to this object. An IPv4 address is not a domain name and given that this information is represented with its fact *resolvesTo* to an IPv4, it would be repetitive information in the analysis. A regex expression validating an FQDN must be added to the ACT platform to address this issue. An example of this could be the following regular expression:

```
r'^[a-zA-Z0-9]([a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(\.[a-zA-Z]{2,})+\$
    '<
```

8.2.2 Constraints Within STIX 2.0

In developing the STIX-Shifter connector for the ACT platform, the current study accommodated certain limitations originating from the 2.0 standard of STIX. STIX-Shifter supports both the 2.0, as well as the 2.1. However, for both these standards, it currently only supports SCO and not relationships as well as SDO. Explicitly, it is not possible to get proper 2.1 objects. By this, it means that the 2.1 objects are deprecated ones when the validator script converts them. According to the official documentation of STIX, 2.1 [28] in section 1.6.5, the cyber observable container is deprecated, and the implementers encourage the use of SRO instead. The model within the context of deprecated cyber observable containers is constrained to be unable to reference other objects. Referencing objects such as SDOs and other SCOs are not yet supported. In other words, a STIX 2.1 object does not give any real value yet. By this, it is impossible to see observable objects in a contextualized matter.

An example is when presenting a report, as it must be seen as related to other information, such as incidents, tools, or other relevant factors. This contextual information is crucial for understanding a report's significance and relevance of a report and for insight from the CTI, and for creating a desired SRO. Therefore, only depicting the report with its name and value from ACT would alone not provide sufficient information and context to understand the entire threat landscape of the report.

Chapter 9

Validation

STIX-Shifter includes a validator script described in Section 5.4, taking a new bundle of the observed object(s) created by a specific connector and validating the output. Each observed object mapped in this thesis is verified with the validator script. Notably, as outlined in Figure 6.1, the validation of the observed objects will not contain the identity object, as this represents an actual individual, organization, or group. The validator script will thereby only verify that the observed-data objects contain valid data and are in the correct format.

9.1 Validating a New Bundle in ACT

To validate bundles created by the new ACT connector, it was decided to test 20 different queries for each pattern against the execute function. This test aimed to ensure that the new connector accurately translates ACT data to a STIX bundle and that the translated data is usable and valid. It is worth mentioning that only existing data in the ACT platform was queried. Meaning that any results returning empty objects were omitted and not part of the final result depicted in Figure 9.1. This was excluded because this test was performed to see if a created bundle contains valid information, conforming to the technical specifications defined by STIX.

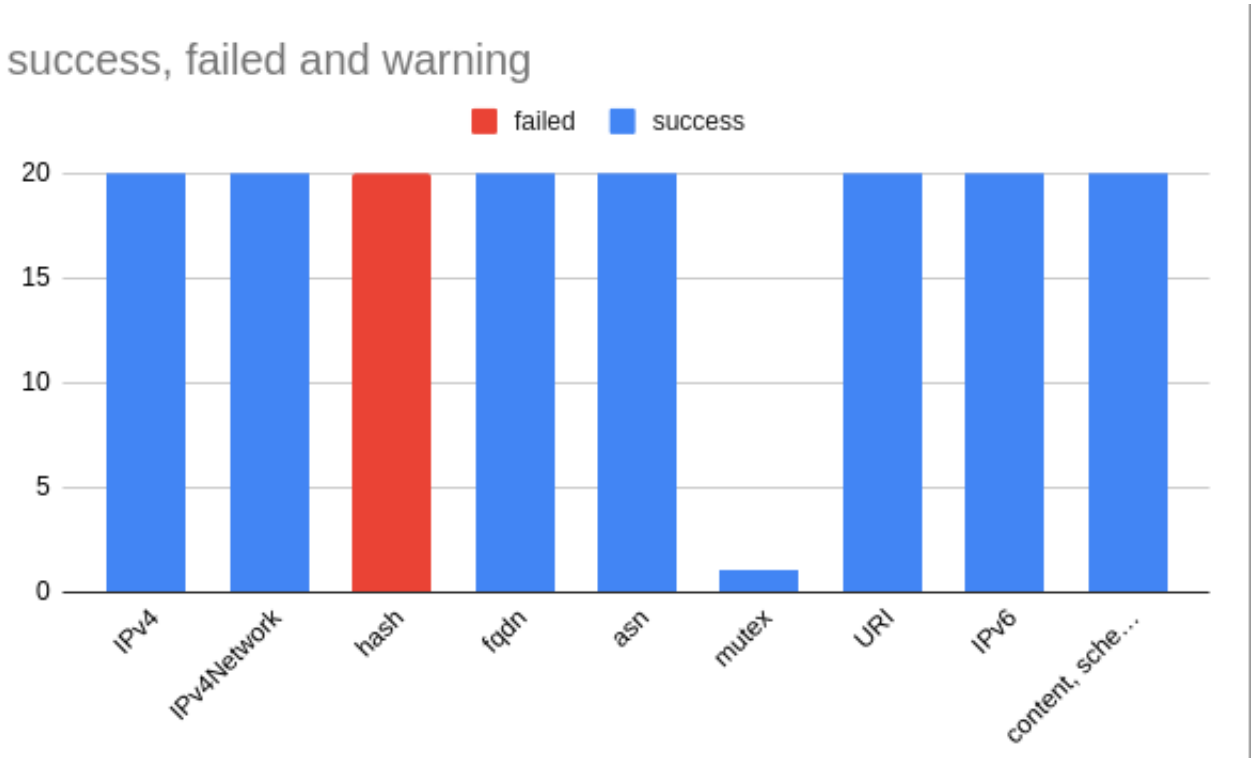

Figure 9.1: Succsessful, failed and warnings when utilizing ACT connector in STIX-Shifter

Figure 9.1 presents if validation of a bundle created by the ACT connector is successful, fails, or if it contains any warnings. The Figure is only based on searching for a single pattern, meaning operators are not accounted for in this test. It is considered successful if the retrieved bundle runs through the validator script without errors. The process necessitated certain assumptions regarding successfully displaying information in STIX-Shifter. Notably, given the absence of examples in the ACT platform for attributes such as username, userAgent, credentials, and operating system paths, a presumptive approach was adopted to envision the storage of these values in future iterations. Displaying these attributes involved utilizing mock values for the attributes above, as querying the ACT API for such objects was not feasible. The primary objective of Chapter 7 was to ensure accurate mapping of the objects, thereby enhancing the overall efficacy of the mapping process. However, the validation of these values cannot be entirely ascertained as it must be evaluated in the

context of values present in the existing platform to determine if the bundle is valid or fails. In the context of Mutex, at this stage, it represented only a single entry in the ACT database, resulting in a solitary entry within the graph.

Inadequate Hash Type Information

When employing the validation script, the segment for verifying cryptographic hash functions demonstrates that it has failed, indicated by the error rate in Table 9.1. As documented in Table 9.1, a random assortment of hashes is stored by ACT. To ensure accurate validation with the validation script, it is crucial to identify what hash type is in use. Despite ACT storing the hashes correctly, it does not provide information on what hash type it is, restraining the validation process. Each entry in Table 9.1 has an associated hyperlink that shows the corresponding hash in the ACT platform. These hashes have been identified outside of ACT using an online hash identifier.

SHA-256	ff0297066b2a1218f587f0e4ee5dfda4c11e09ad931d94c69166dd4e0ba3b9f3
MD5	230f5db4a9f3b09b85d8b66171f4b73e
SHA-1	5767653494d05b3f3f38f1662a63335d09ae6489
NTLM	1536:2fejjwf62hwhoov/tocwmxr8gp+8ubfnhzfb:2yxb/toigo/gb

Table 9.1: Different hash algorithms stored by ACT.

The insufficient provision of information regarding hash algorithms within the ACT platform presents a notable limitation to the utilization of hashes when mapped to STIX from ACT. Due to this limitation, the complete functional integration of hashes remains unattended. To overcome this challenge and facilitate the visualization of a hash within a bundle, the solution lies in incorporating the hash algorithm as an additional field within the ACT platform. This supplementary value should be stored as metadata associated with the object hash. This additional information will, in the transfer of data from ACT to STIX, serves to rectify the errors in Table 9.1.

9.1.1 Validating the Operator Functionality

To assess the functionality of the new ACT connector, queries with additional operators were tested to see how well it handled larger and more complex patterns. This assessment aimed to verify if there were any limitations to how much information could be obtained from the ACT platform.

To assess the complexity of a pattern translated to an ACT query, the assessment validated additional patterns against the new connector in STIX-Shifter. The idea was to evaluate if creating multiple queries for ACT was possible, which did not affect the connector's ability to handle larger data sets and did not degrade performance. Considering that not several similar patterns are executed, such as searching for similar SCOs in one query. Larger patterns did not affect the new connector's performance or ability to create a larger bundle of observed objects. An example of queries with the additional parameters ipv4, domain name, ASN, and hash gives the bundle depicted in Listing 16:

```
"type": "bundle",
"id": "bundle--eee41d56-18ba-4611-ae75-1e07d2c96c9c",
"objects": [
    {
        "type": "identity",
        "id": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
        "name": "ACT",
        "identity_class": "observed-data"
    },
    {
        "id": "observed-data--54a0cf8c-7c3e-4569-8741-b6bcb7d5ccbb",
        "type": "observed-data",
        "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
        "created": "2023-04-21T07:25:38.395Z",
        "modified": "2023-04-21T07:25:38.395Z",
        "objects": {
            "0": {
                "type": "file",
                "hashes": "230f5db4a9f3b09b85d8b66171f4b73e"
            },
            "1": {
                "type": "autonomous-system",
                "number": 11955,
                "name": "AS11955"
            },
            "2": {
                "type": "url",
                "value": "http://example.com/admin/index.php?"
            },
            "3": {
                "type": "ipv4-addr",
                "value": "192.124.249.137"
            },
            "4": {
                "ipv4_cidr": "192.124.249.0/24",
                "belongs_to_refs": "1"
            }
        },
        "first_observed": "2023-04-21T07:25:38.395Z",
        "last_observed": "2023-04-21T07:25:38.395Z",
        "number_observed": 1
    }
],
"spec_version": "2.0"
```


Listing 16: Full representation of a new bundle created with the ACT connector.

Chapter 10

Conclusion

This study has explored the workings of STIX and ACT, delving into their Data Model, structure, and compatibility. In addition, this project has investigated how feasible STIX-Shifter is for ACT when sharing CTI, as well as the information loss during the conversion, if any.

Research Question: How feasible is the development of a new STIX-Shifter module that allows the ACT platform to share its knowledge base comprising CTI?

Concerning the research question, this thesis has demonstrated one possible approach for creating a new connector in STIX-Shifter. This will enable ACT to share its knowledge base comprising CTI in a standardized format, specifically in the form of a bundle of an observed object. A new connector was achieved by establishing a detailed mapping between STIX patterns and specific data source values in the ACT platform and returning the results, translating them into a bundle of observed objects.

Subquestion 1 - To what extent is the sharing of CTI possible from ACT to STIX?

The mapping process facilitated the possibility of sharing CTI from ACT to STIX. Utilizing STIX-Shifter for processing observable data within the

context of sharing of CTI presents some limitations, particularly a broad understanding of the cyber threat landscape. The absence of contextualization between distinct observed data hinders the development of precise knowledge needed for extractable and actionable intelligence. It is possible to achieve contextualization for SCO when SDO and SRO are supported in STIX-Shifter.

Subquestion 2 - To what extent will there be a loss of data in the convergence of results returned from the ACT platform?

The newly created ACT connector in STIX-Shifter converts data from ACT to STIX as mapped cyber observable objects, which yields a one-to-one representation. In contrast, not when it comes to all objects in ACT, meaning the lack of representation of SDOs. The final product of a comprehensive connector ideally includes conversion of SDOs covering all data stored in the target platform. In order to reach these goals and address the limitations, future work explored the methods for contextualizing observable data and improving the reliability of information sharing.

Chapter 11

Future Work

11.1 Missing Functionality IPv6 and ASN

A challenge was encountered while mapping IPv6, as it was not feasible to map an existing entry to an ASN. To share additional information for an IPv6, a new graph node representing the IPv6 network must be created and connected to the corresponding ASN. This can be accomplished by utilizing the existing fact relationship between an IPv4 and an IPv4 network. Specifically, adding a new entry in the fact-types.json file is necessary, as outlined in the ACT documentation [64]. The new entry is structurally identical to the pre-existing fact relationship for IPv4 and can be created by generating a list of objects representing the ASN and IPv6 network.

```
{
    "destinationObjectType": "ipv6Network",
    "sourceObjectType": "ipv6"
},
{
    "destinationObjectType": "asn",
    "sourceObjectType": [
        "ipv4Network",
        "ipv6Network"],
},
```

Listing 17: IPv6 fact entry proposal.

Listing 17 shows how a new fact entry is added to the existing model in ACT. Both IPv6 and ASN already exist as an entry in ACT. The missing link added here is a fact type depicting how an IPv6 network can be a member of an ASN. This will now change the existing Data Model to the new Figure 11.1.

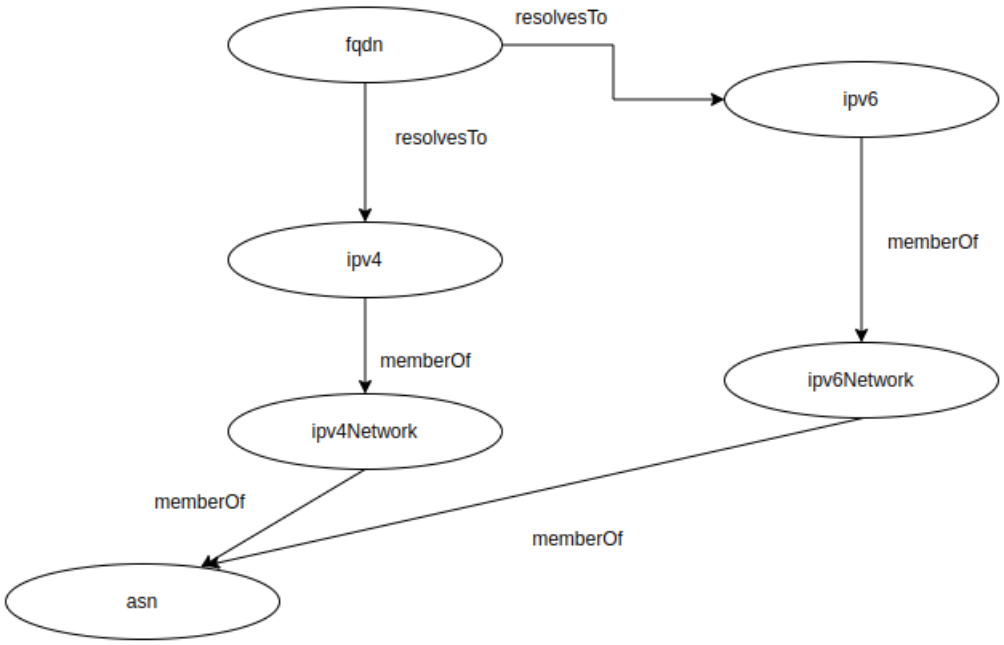

Figure 11.1: IPv6 Data Model for ASN.

11.2 Mapping of SDOs

This project focuses on the mapping of SCO stored and represented by STIX-Shifter, making it inconsistent with incorporating values such as ThreatActor and Techniques, which are maintained as SDOs within the framework. However, future iterations of the connector should aim to include SDOs.

11.2.1 STIX-Shifter Limitations

A prerequisite for the effective deployment of STIX-Shifter involves forming relationships between SDOs within the connector. Presently, functionality is only supported for SCOs [6], and SDO remains unsupported, making it unsensible for inclusion in the existing ACT connector.

11.2.2 ACT Limitations

Furthermore, for most of the SDOs in STIX, the required properties have increased and especially "created" and "modified." presently, ACT lacks complete support for metafacts. While it can fetch the metafact "observationTime" using a UUID, the current implementation restricts its use to return a single value. Nonetheless, once this feature is supported, it should be feasible to establish mappings as illustrated in Listing 18.

```
"uuid":
[
    {
        "key": "uuid.value",
        "type": "uuid.observationTime"
    }
],
```

Listing 18: Example of how ACT can provide the following values "created" and "modified" to be able to map STIX SDOs.

However, preparing the foundation for this capability will enable a smooth integration once support is extended in the following versions. This was documented in Appendices 12.2.

Chapter 12

Appendices

.1 One-legged facts

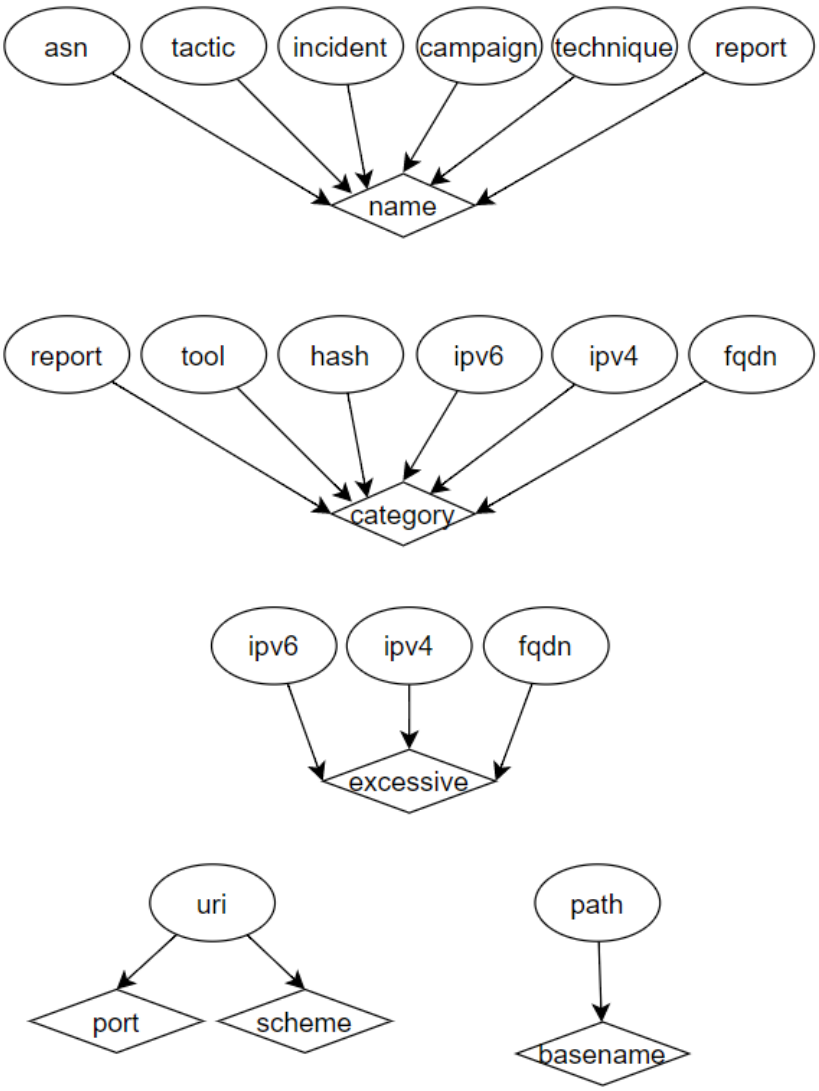

Figure 1: Representation of all one-legged facts within ACT

.2 SDO Mapping

As previously mentioned in the thesis, the integration of SDOs is not supported by STIX-Shifter [29]. Nevertheless, a selection of representative SDOs with the potential of mapping between ACT and STIX has been identified and included in this section to facilitate further development of this connector. Once it is supported in STIX-Shifter, it should be possible to add this functionality, and this supplement should contribute to accelerating the process. Moreover, this section stands out from the mapping results in Chapter 7; since it is not completed and tested, it should not be deemed functional and serves only as a possible approach. The current assortment of SDOs identified Identity, Campaign, Report, Tool, Vulnerability, ThreatActor, Incident, and Location, the rationale behind this selection is that ACT possesses representative values. The representative values are extracted from the Data Model of ACT represented in Figure 7.2, which are the remaining yellow nodes and matched with existing SDOs the STIX 2.1 documentation located under Chapter 4 on page 55 [35]. The mapping proposal is visualized as figures and code snippets different from Chapter 7, whereas this is focused on singular SDOs to enhance understanding.

.2.1 Identity

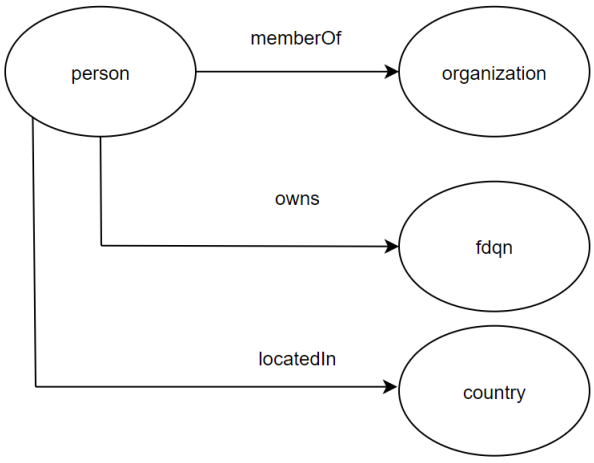

Figure 2: Visualization of person and its represented facts - organization, fqdn, country

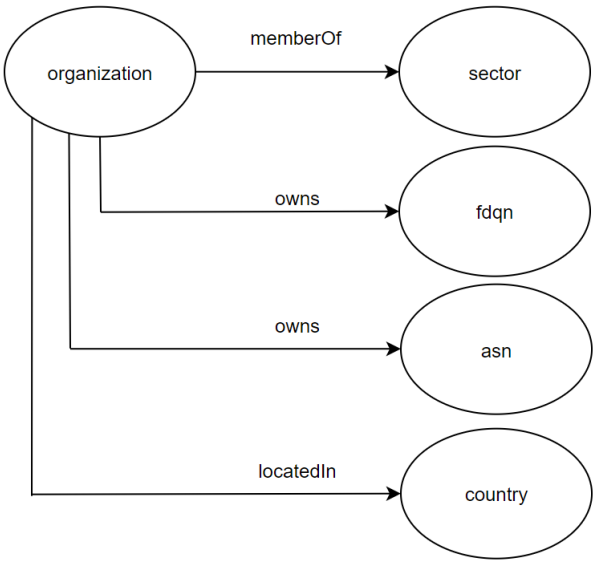

Figure 3: Visualization of organization and its represented facts - sector, fqdn, asn, and country

From STIX to ACT

A *person* and *organization* are mapped from STIX to ACT by the Identity SDO in STIX. From this SDO, the common property: name is utilized to create a data source entry in ACT for a person or organization value. Below is a potential configuration for this mapping:

```
"Identity":
{
    "fields":{
        "name": ["person", "organization"]
    }
},
```

Listing 19: Mapping of Identity SDO to ACT data source fields

From ACT to STIX

A *person* in ACT consists of outgoing edges as depicted in Figure 2. These edge nodes represent the facts *memberOf*, *owns*, and *locatedIn*. The fact *memberOf* represents how a *person* is a member of an organization. The fact *owns* shows what FQDN a person can possess. Lastly, the fact *locatedIn* demonstrates what *country* this *person* is located in.

An *organization* in ACT consists of outgoing edges as depicted in Figure 3. These edge nodes represent the facts *memberOf*, *owns*, and *locatedIn*. The fact *memberOf* represents how an organization is a member of a country. The fact *owns* shows what FQDN an organization can possess. The second *owns* represents what ASN an organization can possess. Lastly, the fact *locatedIn* demonstrates what country this organization is located in.

```
    "person":
    [
        {
            "key": "identity.name",
            "object": "organization_name"
        }
    ],
    "organization":
    [
        {
            "key": "identity.name",
            "object": "person_name"
        }
    ],
```

Listing 20: Mapping of person and organization from ACT to STIX

In Listing 20, the proposal mapping of a person and organization from ACT to STIX is represented. This model displays how an *organization* and *person* object

in ACT is mapped to its respective properties in the Identity SDO, its remaining required properties will be executed according to Listing 18.

.2.2 Campaign

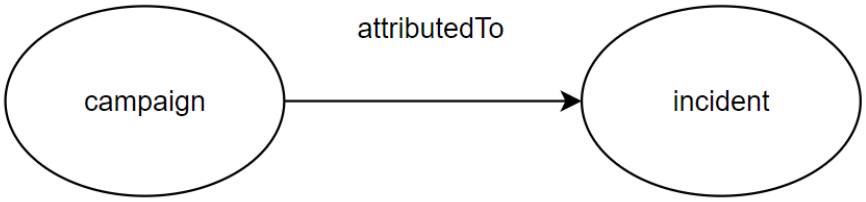

Figure 4: Visualization of campaign and its represented fact - incident

From STIX to ACT

A *campaign* is mapped from STIX to ACT by the Campaign SDO in STIX. From this SDO, the common property: name is utilized to create a data source entry in ACT for a campaign value. Below is a potential configuration for this mapping:

```
"campaign":
{
    "fields":{
        "name": ["campaign"]
    }
},
```

Listing 21: Mapping of Campaign SDO to ACT data source fields

From ACT to STIX

A *campaign* in ACT consists of one outgoing edge as depicted in Figure 4. This edge represents the fact *attributedTo*. The fact represents how a campaign is

attributed to an incident.

```
"campaign":
[
    {
        "key": "campaign.name",
        "object": "campaign_name"
    }
],
```

Listing 22: Mapping of campaign from ACT to STIX

In Listing 22, the proposal mapping of a campaign from ACT to STIX is represented. This model displays how a *campaign* object in ACT is mapped to its respective properties in the Tool SDO, and its remaining required properties will be executed according to Listing 18.

.2.3 Report

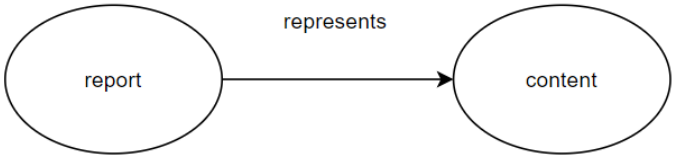

Figure 5: Visualization of report and its represented fact - content

From STIX to ACT

A *report* is mapped from STIX to ACT by the Report SDO in STIX. From this SDO, the common property: name is utilized to create a data source entry in ACT for a tool value. Below is a potential configuration for this mapping:

```
"Report":
{
    "fields":{
        "name": ["report"]
    }
},
```

Listing 23: Mapping of Report SDO to ACT data source fields

From ACT to STIX

A *report* in ACT consists of one outgoing edge as depicted in Figure 5. This edge represents the fact *represents*. The fact represents how it withholds *content*.

```
"report":
[
    {
        "key": "report.name",
        "object": "report_name"
    }
],
```

Listing 24: Mapping of report from ACT to STIX

In Listing 24, the proposal mapping of a report from ACT to STIX is represented. This model displays how a *report* object in ACT is mapped to its respective properties in the Report SDO, its remaining required properties will be executed according to Listing 18.

.2.4 Tool

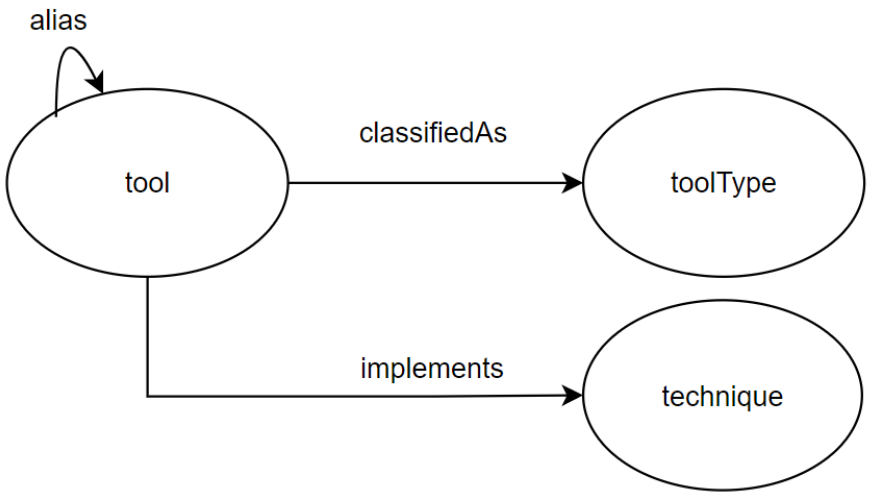

Figure 6: Visualization of tool and its represented facts - alias, toolType, and technique

From STIX to ACT

A *tool* is mapped from STIX to ACT by the Tool SDO in STIX. From this SDO, the common property: name is utilized to create a data source entry in ACT for a tool value. Below is a potential configuration for this mapping:

```
"Tool":
{
    "fields":{
        "name": ["tool"]
    }
},
```

Listing 25: Mapping of Tool SDO to ACT data source fields

From ACT to STIX

A *tool* in ACT consists of two outgoing edges as depicted in Figure 6. These edges represent the fact *classifiedAs* and *implements*. The first fact represents how a tool is classified into a type of tool, and the second fact represents how a tool implements a technique.

```
"tool":
[
    {
        "key": "tool.name",
        "object": "tool_name"
    }
],
```

Listing 26: Mapping of tool from ACT to STIX

In Listing 26, the proposal mapping of a tool from ACT to STIX is represented. This model displays how a *tool* object in ACT is mapped to its respective properties in the Tool SDO, and its remaining required properties will be executed according to Listing 18.

.2.5 Vulnerability

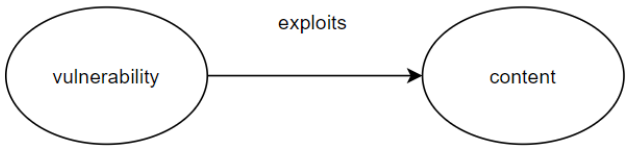

Figure 7: Visualization of Vulnerability and its represented fact - content

From STIX to ACT

A *vulnerability* is mapped from STIX to ACT by the Vulnerability SDO in STIX. From this SDO, the common property: name is utilized to create a data source entry in ACT for a vulnerability value. Below is a potential configuration for this mapping:

```
"Vulnerability":
{
    "fields":{
        "name": ["vulnerability"]
    }
},
```

Listing 27: Mapping of Vulnerability SDO to ACT data source fields

From ACT to STIX

A *vulnerability* in ACT consists of one outgoing edge as depicted in Figure 7. These edges represent the fact *exploits*. The fact represents how a vulnerability exploits a type of content.

```
"vulnerability":
[
    {
        "key": "vulnerability.name",
        "object": "vulnerability_name"
    }
],
```

Listing 28: Mapping of vulnerability from ACT to STIX

In Listing 28, the proposal mapping of a vulnerability from ACT to STIX is represented. This model displays how a vulnerability object in ACT is mapped to its respective properties in the Vulnerability SDO, and its remaining required properties will be executed according to Listing 18.

.2.6 ThreatActor

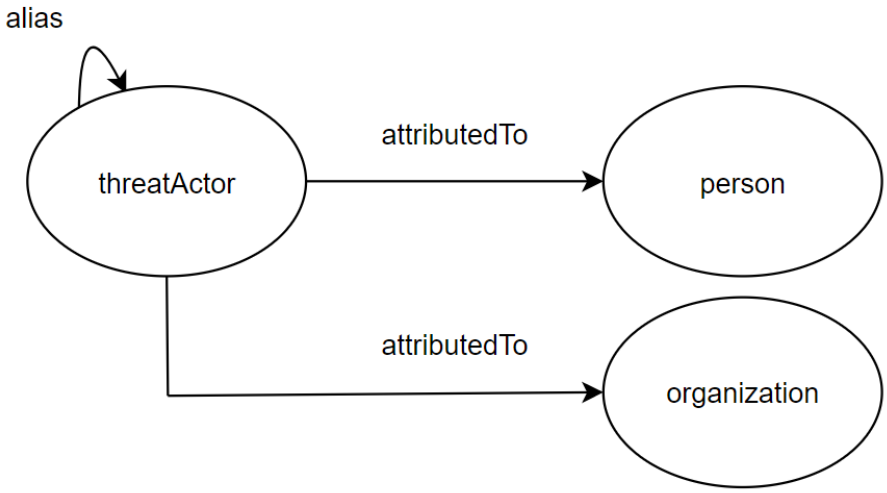

Figure 8: Visualization of ThreatActor and its represented facts - alias, person, and organization

From STIX to ACT

A *threatActor* is mapped from STIX to ACT by the Threat Actor SDO in STIX. From this SDO, the common property: name is utilized to create a data source entry in ACT for a threatActor value. Below is a potential configuration for this mapping:

```
"threat-actor":
{
    "fields":{
        "name": ["threatActor"]
    }
},
```

Listing 29: Mapping of Threat Actor SDO to ACT data source fields

From ACT to STIX

A *threatActor* in ACT consists of two outgoing edges as depicted in Figure 29. These edges represent the fact *attributedTo*. The first fact represents how a threatActor is attributed to a person, and the second fact represents how a threatActor is attributed to an organization.

```
"threatActor":
[
    {
        "key": "threat-actor.name",
        "object": "threat_actor"
    }
],
```

Listing 30: Mapping of threatActor from ACT to STIX

In Listing 34, the proposal mapping of a *threatActor* from ACT to STIX is represented. This model displays how a *threatActor* object in ACT is mapped to its respective properties in the Threat Actor SDO, and its remaining required properties will be executed according to Listing 18.

.2.7 Incident

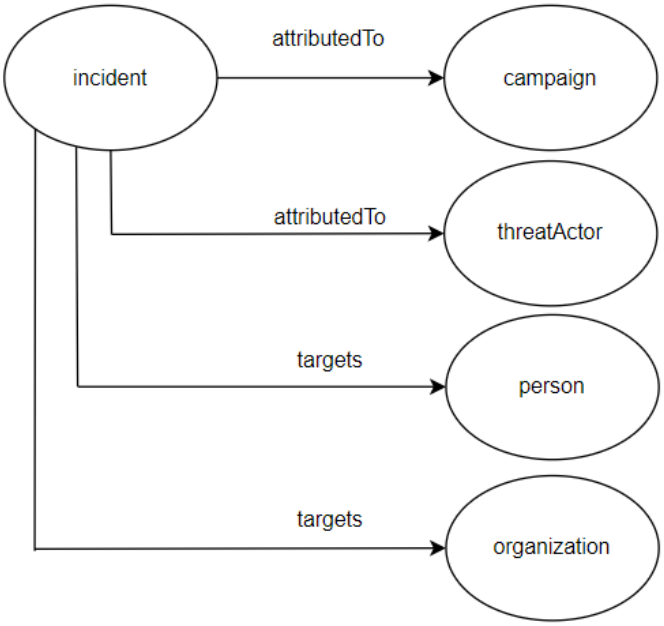

Figure 9: Visualization of Incident and its represented facts - campaign, threatActor, person and organization

From STIX to ACT

An *incident* is mapped from STIX to ACT by the Incident SDO in STIX. From this SDO, the common property: name is utilized to create a data source entry in ACT for an incident value. Below is a potential configuration for this mapping:

```
"Incident":
{
    "fields":{
        "name": ["incident"]
    }
},
```

Listing 31: Mapping of Incident SDO to ACT data source fields

From ACT to STIX

An incident in ACT consists of four outgoing edges as depicted in Figure 9. These edges represent the facts *attributedTo* and *targets*. The first fact represents how an incident is attributed to a campaign, and the second fact represents how an incident is attributed to a *threatActor*. The third fact represents how an incident is targeting a specific person and the fourth is an organization.

```
"incident":
[
    {
        "key": "incident.name",
        "object": "incident"
    }
],
```

Listing 32: Mapping of incident from ACT to STIX

In Listing 32, the proposal mapping of *incident* from ACT to STIX is represented. This model displays how an *incident* object in ACT is mapped to its respective

properties in the Incident SDO, and its remaining required properties will be executed according to Listing 18.

.2.8 Location

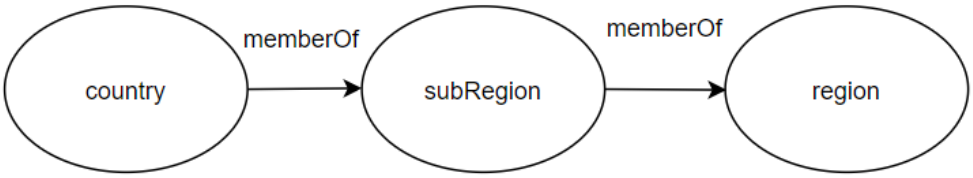

Figure 10: Visualization of Location and its represented facts - country, subRegion, and region

From STIX to ACT

A *country* is mapped from STIX to ACT by the Location SDO in STIX. From this SDO, the common property: name is utilized to create a data source entry in ACT for a *country* value. Further on *region* is utilized to create a data source entry for a *subRegion* value. Below is a potential configuration for this mapping:

```
"Location":
{
    "fields":{
        "country": ["country"]
    }
},
"Location":
{
    "fields":{
        "region": ["region"]
    }
}
```

Listing 33: Mapping of Location SDO to ACT data source fields

From ACT to STIX

A *country* in ACT consists of one outgoing edge to *subRegion*, and another one from *subRegion* to *region* as depicted in Figure 10. These edges represent the fact *memberOf*. The first fact represents how a country is a member of a subregion, and the second fact represents how a subregion is attributed to a region.

```
    "country":
    [
        {
            "key": "location.country",
            "object": "location_country"
        }
    ],
    "region":
    [
        {
            "key": "location.region",
            "object": "location_region"
        }
    ],
```

Listing 34: Mapping of region and country from ACT to STIX

In Listing 34, the proposal mapping of country and region from ACT to STIX is represented. This model displays how a country and region object in ACT is mapped to its respective properties in the location SDO, and its remaining required properties will be executed according to Listing 18. To include these objects, it is necessary to introduce a custom object for subRegion from ACT to prevent information loss. This value can be mapped as depicted in Listings 35 and 36.

```
"x-act": {
    "subregion": ["subRegion"]
}
```

Listing 35: Mapping of custom object subRegion from STIX to ACT

```
    "subRegion":
    [
        {
            "key": "x-act.subRegion",
            "object": "location_subRegion"
        }
    ],
```

Listing 36: Mapping of custom object subRegion from ACT to STIX

Bibliography

[1] Stuti Gupta. *Setting Up STIX Shifter*. Accessed: 2023-05-14. 2022. URL: https://community.ibm.com/community/user/security/blogs/stuti-gupta/2022/03/09/setting-up-stix-shifter.

[2] Robert M. Lee and Rebekah Brown. *For578.1: Cyber Threat Intelligence and requirements*. URL: https://www.sans.org/cyber-security-courses/cyber-threat-intelligence/.

[3] Jim Boehm et al. 'Cybersecurity trends: Looking over the horizon'. In: (March 10, 2022). URL: https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/cybersecurity/cybersecurity-trends-looking-over-the-horizon.

[4] Kurt Baker. *Open Source Intelligence (OSINT)*. Accessed on: March 29, 2023. February 28, 2023. URL: https://www.crowdstrike.com/cybersecurity-101/osint-open-source-intelligence/.

[5] Cloudflare. *What is STIX and TAXII?* https://www.cloudflare.com/learning/security/what-is-stix-and-taxii/. Accessed on: March 29, 2023.

[6] opencybersecurityalliance. *Stix-shifter*. Accessed on: June 06, 2022. URL: https://github.com/opencybersecurityalliance/stix-shifter/blob/develop/OVERVIEW.md.

[7] Niels Bjørn Andersen. *Why you should not choose the best security product*. Accessed: 2023-03-17. 2017. URL: https://www.ibm.com/blogs/nordic-msp/not-choose-best-security-product/.

[8] IBM Corporation. *IBM Study: Security Response Planning on the Rise, But Containing Attacks Remains an Issue*. Accessed: 2023-03-17. June 2020. URL: https://newsroom.ibm.com/2020-06-30-IBM-Study-Security-Response-Planning-on-the-Rise-But-Containing-Attacks-Remains-an-Issue.

[9] Michael Collier, Miriam Fernandez and Harith Alani. 'Identifying emergent vulnerability trends using Cyber Security Vulnerability and Breach data'. In: *Journal of Cybersecurity* 4.1 (Dec. 2018), tyy008. DOI: 10.1093/cybsec/tyy008. URL: https://academic.oup.com/cybersecurity/article/4/1/tyy008/5245383.

[10] Katherine Cort, George Bonheyo and mark Tardiff. 'Applying the scientific method to cybersecurity research'. In: *International Journal of Information Security* (2016). URL: https://www.researchgate.net/publication/308191945_Applying_the_scientific_method_to_cybersecurity_research.

[11] Gavin Wright and Tréa Lavery. *Scientific Method*. https://www.techtarget.com/whatis/definition/scientific-method#:~:text=The%20scientific%20method%20is%20the,and%20finally%20analyzing%20the%20results.. Accessed on: February 20, 2023.

[12] Indeed Editorial Team. *Requirements Analysis: Definition and Process*. https://www.indeed.com/career-advice/career-development/requirements-analysis#:~:text=Requirements%20analysis%20allows%20software%20engineers,to%20analyze%20its%20requirements%20properly.. Accessed on Februrary 20, 2023.

[13] Shona McCombes. *How to Write a Literature Review | Guide, Examples, & Templates*. https://www.scribbr.com/methodology/literature-review/. January 2, 2023.

[14] S. Bell. *Experimental Design*. 2009. URL: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/experimental-design.

[15] IBM Developer. *Build a connector for IBM Cloud Pak for Security with STIX Shifter*. https://developer.ibm.com/tutorials/build-a-connector-for-ibm-cloud-pak-for-security-with-stix-shifter/. Accessed on: March 30, 2023. June 25, 2020.

[16] Heggem. 'Finding new links in ACT data provided by Mnemonic using Graph Neural Networks'. MA thesis. 2021. URL: https://www.duo.uio.no/bitstream/handle/10852/91263/1/New_links_in_ACT_using_GNN_mkheggem.pdf.

[17] Mari Grønberg. 'Automatic Text Classification using Machine Learning'. MA thesis. University of Oslo, 2019. URL: https://www.duo.uio.no/bitstream/handle/10852/69063/groenberg_mari_thesis.pdf?sequence=1&isAllowed=y.

[18] Siri Bromander et al. 'Investigating Sharing of Cyber Threat Intelligence and Proposing A New Data Model for Enabling Automation in Knowledge Representation and Exchange'. In: (October 2021). URL: https://dl.acm.org/doi/pdf/10.1145/3458027.

[19] opencybersecurityalliance. *Available Connectors*. Accessed on: 2023-13-05. URL: https://github.com/opencybersecurityalliance/stix-shifter/blob/develop/CONNECTORS.md.

[20] Open Cybersecurity Alliance. *What is STIX Shifter?* 2021. URL: https://github.com/opencybersecurityalliance/stix-shifter/blob/develop/OVERVIEW.md#what-is-stix-shifter.

[21] Mnemonic AS. *ACT Examples*. https://act.mnemonic.no/examples/. Accessed June 06, 2023.

[22] Open Cybersecurity Alliance. *STIX Shifter Connector Coding Lab*. https://github.com/opencybersecurityalliance/stix-shifter/blob/develop/lab/connector_coding_lab.md. Accessed January 06, 2023.

[23] Rob McMillan. *Threat Intelligence Definition*. Gartner. 16 May 2013. URL: https://www.gartner.com/en/documents/2487216.

[24] United States Department of Defense. *Joint Publication 2-0: Joint Intelligence*. http://www.dtic.mil/doctrine/new_pubs/jp2_0.pdf. Joint Chiefs of Staff, Oct. 2013.

[25] CrowdStrike. *What Is Security Automation? Types, Benefits & 5 Best Practices*. https://www.crowdstrike.com/cybersecurity-101/security-automation/. Accessed on April 4, 2023. March 1, 2023.

[26] OASIS Cyber Threat Intelligence Technical Committee. *Introduction to TAXII*. Accessed: 2023-05-10. n.d. URL: https://oasis-open.github.io/cti-documentation/taxii/intro.

[27] Madelyn Bacon. *STIX (Structured Threat Information eXpression)*. 2015. URL: https://www.techtarget.com/searchsecurity/definition/STIX-Structured-Threat-Information-eXpression#:~:text=STIX%5C%20(Structured%5C%20Threat%5C%20Information%5C%20eXpression)%5C%20is%5C%20a%5C%20standardized%5C%20XML%5C%20programming,core%5C%20use%5C%20cases%5C%20for%5C%20STIX. (visited on 03/03/2022).

[28] *STIX™ Version 2.1*. OASIS Cyber Threat Intelligence Technical Committee, 20 March 2020. URL: https://docs.oasis-open.org/cti/stix/v2.1/cs01/stix-v2.1-cs01.pdf.

[29] OASIS Cyber Threat Intelligence Technical Committee. *STIX 2 Python API Documentation*. Accessed on: April 12, 2023. OASIS Open. URL: https://docs.oasis-open.org/cti/stix/v2.1/csprd01/stix-v2.1-csprd01.html.

[30] OASIS. *STIX Version 2.0: Part 4 - Cyber Observable Objects*. 19 July 2017. URL: https://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part4-cyber-observable-objects.html.

[31] S. Bradner. *rfc2119*. https://www.rfc-editor.org/rfc/pdfrfc/rfc2119.txt.pdf. 1997.

[32] Mandiant. *Pinpointing Targets: Exploiting Web Analytics to Ensnare Victims*. Accessed on: February 23, 2023. URL: https://www.mandiant.com/resources/pinpointing-targets-exploiting-web-analytics-to-ensnare-victims.

[33] PwC UK. *Operation Cloud Hopper*. Accessed on: February 23, 2023. URL: https://www.pwc.co.uk/issues/cyber-security-services/insights/operation-cloud-hopper.html.

[34] OASIS Cyber Threat Intelligence Technical Committee. *Visualized SDO Relationships*. URL: https://oasis-open.github.io/cti-documentation/examples/visualized-sdo-relationships.html.

[35] *STIX 2.1*. OASIS Cyber Threat Intelligence Technical Committee, 2021. URL: https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.pdf.

[36] OASIS Cyber Threat Intelligence (CTI) Technical Committee. *STIX™ Version 2.0. Part 5: STIX Patterning*. OASIS, 2017. URL: https://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part5-stix-patterning.html.

[37] OASIS. *STIX Version 2.0 Part 5: STIX Patterning*. 19 July 2017. URL: http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part5-stix-patterning.html.

[38] The MITRE Corporation. *STIX 2.0 Patterning*. Accessed on: 28 Jan, 2023. URL: https://stix2.readthedocs.io/en/latest/guide/patterns.html.

[39] opencybersecurityalliance. Accessed on: 23 August 2022. URL: https://github.com/opencybersecurityalliance/stix-shifter/blob/develop/OVERVIEW.md#architecture-context.

[40] Mnemonic. *Semi-Automated Cyber Threat Intelligence - ACT Platform*. 2022. URL: https://github.com/mnemonic-no/act-platform (visited on 09/05/2022).

[41] Siri Bromander. 'Understanding Cyber Threat Intelligence - Towards Automation'. PhD thesis. University of Oslo, 2021. URL: https://www.duo.uio.no/bitstream/handle/10852/84713/PhD-Bromander-2021.pdf?sequence=1&isAllowed=y.

[42] mnemonic-no. *Object Fact Model*. Accessed on: January 02, 2022. URL: https://github.com/mnemonic-no/act-platform/wiki/Object-Fact-Model.

[43] frbor. *act-api-python*. https://github.com/mnemonic-no/act-api-python/blob/master/act/api/fact.py#L97. 2021.

[44] mnemonic AS. *ACT API Documentation*. n.d. URL: https://act.mnemonic.no/swagger/#/development.

[45] *Mnemonic API*. Accessed on: 25 Jan, 2023. URL: https://act.mnemonic.no/swagger/#/.

[46] National Institute of Standards and Technology. 'Framework for Improving Critical Infrastructure Cybersecurity'. In: *Appendix B: Glossary* (April 16, 2018). Accessed: 2023-03-16, p. 47. URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf.

[47] Patrick Lambe. *Organising knowledge: Taxonomies, knowledge and organizational effectiveness*. Oxford: Chandos Publishing, 2007.

[48] Erik Sørli and Benjamin Jørgensen. *STIX Shifter ACT Connector*. https://github.com/cyentific-rni/act-stix-shifter.

[49] Stuti Gupta. *Setting up STIX-Shifter*. IBM Community. 2022. URL: https://community.ibm.com/community/user/security/blogs/stuti-gupta/2022/03/09/setting-up-stix-shifter.

[50] Open Cybersecurity Alliance. *STIX Shifter: STIX Translation and Transmission*. https://github.com/opencybersecurityalliance/stix-shifter. Accessed on March 25, 2023.

[51] W3Resource. *SQL Comparison Operators*. https://www.w3resource.com/sql/comparison-operators/sql-comparison-operators.php. Accessed on March 28, 2023.

[52] *Anti Malware Testfile*. https://www.eicar.org/download-anti-malware-testfile/. EICAR, 2006.

[53] Fredrik Borg. *act-api-python/helpers.py*. Accessed on March 24, 2023. URL: https://github.com/frbor/act-api-python/blob/master/act/api/helpers.py.

[54] Python Software Foundation. *urllib.parse — Parse URLs into components*. Accessed on: March 24, 2023. Python Software Foundation. URL: https://docs.python.org/3/library/urllib.parse.html.

[55] Internet Engineering Task Force. *RFC 3986: Uniform Resource Identifier (URI): Generic Syntax*. 2005. URL: https://www.rfc-editor.org/rfc/rfc3986.

[56] Internet Engineering Task Force (IETF). *DOMAIN NAMES - CONCEPTS AND FACILITIES*. Internet Request for Comments (RFC). 1987. URL: https://www.rfc-editor.org/rfc/rfc1034.

[57] Internet Engineering Task Force (IETF). *Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework*. Internet Request for Comments (RFC). 2010. URL: https://www.rfc-editor.org/rfc/rfc5890.

[58] *STIX-Shifter: Bundle Validator*. https://github.com/opencybersecurityalliance/stix-shifter/tree/develop/bundle_validator. Accessed on: January 04, 2023.

[59] OASIS Open. *CTI STIX Validator*. https://github.com/oasis-open/cti-stix-validator. Accessed on April 1, 2023.

[60] OASIS Cyber Threat Intelligence (CTI) Technical Committee. *Structured Threat Information Expression (STIX) Version 2.0*. 19 July 2017. URL: http://docs.oasis-open.org/cti/stix/v2.0/cs01/part2-stix-objects/stix-v2.0-cs01-part2-stix-objects.html#_Toc496714310.

[61] OASIS Cyber Threat Intelligence Technical Committee. *STIX2 Python Library Documentation*. https://stix2.readthedocs.io/en/latest/index.html. Accessed on March 1, 2023. 2021.

[62] *STIX-Shifter MySQL Populate Script Data*. https://github.com/opencybersecurityalliance/stix-shifter/blob/develop/stix_shifter/scripts/mysql_populate_script/data.csv. Accessed: April 14, 2023.

[63] OASIS. *STIX2.1)*. 2022. URL: https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.pdf (visited on 04/04/2022).

[64] Fredrik Borg. *Fact Types*. Accessed on: 17 November, 2022. URL: https://github.com/mnemonic-no/act-types/blob/master/act/types/etc/fact-types.json.