# University of Padova

Department of Mathematics "Tullio Levi-Civita"

*Master Thesis in Cybersecurity*

# AGIR: Automatic Generation of Intelligence Reports

*Supervisor*
Prof. Mauro Conti
University of Padova

*Co-supervisors*
Dr. Francesco Marchiori
Dr. Nino Vincenzo Verde

*Master Candidate*
Filippo Perrina

*Student ID*
2057216

*Academic Year*
2022-2023

# Abstract

Natural Language Generation (NLG) tools are becoming increasingly important in today's fast-paced business environment. These tools can save organizations significant amounts of time and resources by automating the process of generating written content from structured data. NLG is widely employed in many fields producing computer-generated medical reports, weather forecasts or newspaper articles, however, little work has been done so far in the cybersecurity field. Nowadays, security analysts have to manually write reports starting from structured data such as STIX (Structured Threat Information eXpression) graphs and network logs, this task is very time-consuming.
In this thesis, carried out in collaboration with Leonardo S.p.A., we implement AGIR (Automatic Generation of Intelligence Reports), a NLG tool able to write intelligence reports starting from the JSON representation of STIX graphs. The purpose of AGIR is to assist analysts in the report writing process by providing them significant information and starting them off with a report that is as close as possible to an ideal final version of the report.
AGIR produces the final report in a two-stage pipeline. In the first step, it uses a template-based approach to build a baseline text that, in the second phase, is further refined through the use of ChatGPT APIs. The generated reports are then evaluated through the syntactic log-odds ratio (SLOR), a referenceless model-dependent metric for fluency evaluation, and a questionnaire-based human evaluation on three dimensions: correctness, fluency and utility. The generated reports overall reach good scores on all three levels, but there is room for improvement in the implementation of both steps. The first step introduces maintainability issues that can be circumvented by using a neural-based approach for the creation of the draft text. The second step can be improved by using a free and local deep learning model.

# Contents


Abstract iii

List of figures vii

List of tables ix

Listing of acronyms xi

1 Introduction 1

2 Background 3
- 2.1 Cyber Threat Intelligence . . . 3
  - 2.1.1 CTI types . . . 4
  - 2.1.2 Threat Intelligence Life Cycle . . . 4
  - 2.1.3 Pyramid of Pain . . . 5
  - 2.1.4 CTI reports . . . 7
- 2.2 Structured Threat Information eXpression . . . 8
  - 2.2.1 STIX Entities . . . 9
  - 2.2.2 STIX Relationship Objects . . . 10
  - 2.2.3 Example of a STIX Graph . . . 13

3 Related Works 17

4 Methodology 21
- 4.1 Input . . . 21
- 4.2 Content selection and text generation . . . 23
- 4.3 ChatGPT APIs . . . 25
- 4.4 Output . . . 25
- 4.5 Full Example . . . 27

5 Evaluation 35
- 5.1 Quantitative evaluation . . . 36
- 5.2 Qualitative evaluation . . . 37
  - 5.2.1 SLOR . . . 37
  - 5.2.2 Fluency, Correctness and Utility . . . 38

6 Conclusion 41

Appendix A Questionnaire results divided by report type 43

References 45

Acknowledgments 49

# Listing of figures


2.1 Threat Intelligence Lifecycle. . . . . 6
2.2 Pyramid of Pain. . . . . 7
2.3 Example of a simple STIX Graph. . . . . 13

3.1 Example of template text generation. . . . . 18

4.1 AGIR pipeline. . . . . 22
4.2 Example of a ChatGPT interaction. . . . . 26
4.3 Report of the graph in Figure 2.3. . . . . 31
4.4 Overview and stats section of a complete subject report. . . . . 32
4.5 Relationship and TTP section of a complete subject report. . . . . 33
4.6 IOC and useful resources of a complete subject report. . . . . 34

# Listing of tables

2.1 STIX entities and their description. . . . . 11
2.2 STIX relationships supported by AGIR. . . . . 12

5.1 AGIR accuracy results. . . . . 37
5.2 Average and standard deviation of SLOR scores. . . . . 39
5.3 Questionnaire results grouped by dimension. . . . . 40

A.1 Questionnaire results for overview report. . . . . 43
A.2 Questionnaire results for subject report. . . . . 44
A.3 Questionnaire results for timeline report. . . . . 44
A.4 Questionnaire results for vulnerability report. . . . . 44

# Listing of acronyms

**AGIR** . . . . . Automatic Generation of Intelligence Reports

**API** . . . . . Application Programming Interface

**APT** . . . . . Advanced Peristent Threats

**BLEU** . . . . . Bilingual Evaluation Understudy

**CTI** . . . . . Cyber Threat Intelligence

**CTIS** . . . . . Cyber Threat Intelligence System

**CVSS** . . . . . Common Vulnerability Scoring System

**IOC** . . . . . Indicator Of Compromise

**JSON** . . . . . JavaScript Object Notation

**LM** . . . . . Language Model

**METEOR** . . . . . Metric for Evaluation of Translation with Explicit ORdering

**NLG** . . . . . Natural Language Generation

**NLP** . . . . . Natural Language Processing

**OSINT** . . . . . Open Source Intelligence

**RNN** . . . . . Recurrent Neural Networks

**ROUGE** . . . . . Recall-Oriented Understudy for Gisting Evaluation

**SCO** . . . . . Stix Cyber-Observable Object

**SDO** . . . . . Stix Domain Object

**SLOR** . . . . . Syntactic log-odds ratio

**SOC** . . . . . Security Operation Center

**SRO** . . . . . Stix Relationship Object

**STIX** . . . . . Structured Threat Information eXpression

**TI** . . . . . Threat Intelligence

**TTP** . . . . . Tactics Techniques and Procedures

**WPSLOR** . . . . . Word-Piece Syntactic log-odds ratio

# 1 Introduction

In recent years, the world has witnessed an exponential increase in the number and sophistication of cyberattacks. Cyberattacks can range from simple phishing emails to complex attacks carried out by advanced persistent threats (APTs), which are highly sophisticated and stealthy cybercriminal groups that use a range of tactics, such as social engineering, malware and exploit kits to infiltrate and compromise targeted networks.

To defend against these threats, organizations need to have robust cybersecurity strategies that include the collection, analysis and dissemination of Cyber Threat Intelligence (CTI). CTI is a process that involves collecting and analyzing data from various sources, such as network logs, social media and dark web forums, to identify, understand and mitigate cyber threats. CTI provides organizations with actionable insights into the tactics, techniques, and procedures (TTPs) used by cybercriminals, as well as the indicators of compromise (IOCs) associated with their attacks. By leveraging CTI, organizations can improve their situational awareness, detect and respond to attacks more quickly reducing their overall risk exposure.

One important aspect of CTI is the production of security reports. Security reports are documents that provide detailed information about cyber threats, such as the TTPs used by cybercriminals, the vulnerabilities they exploit, and the IOCs associated with their attacks. Security reports are critical for sharing CTI within an organization and with external partners, such as law enforcement agencies and other cybersecurity organizations. However, writing a security report can be an incredibly time-consuming and resource-intensive task, requiring analysts to manually gather and analyze large amounts of data before summarizing their findings

in a clear and concise report.

To address this challenge, NLG techniques have been developed to automate the process of report writing. These techniques can help security analysts to save time and resources by automating the process of generating written content from structured data. However, despite the significant benefits of NLG, little work has been done so far in the cybersecurity field to develop and evaluate NLG tools for the generation of security reports.

The objective of this thesis is to address this gap in the literature by developing and evaluating AGIR, a NLG tool that can produce cybersecurity reports starting from the JSON representation of STIX graphs. This work has been done in collaboration with Leonardo S.p.A., an Italian multinational company active in different sectors. In particular, their cybersecurity division is specialized in CTI and through the construction of a Security Operation Center (SOC) and the launch of a new software called "Cyber Threat Intelligence System" (CTIS) they aim at protecting institutions, enterprises and citizens. AGIR constitutes a micro-service of the CTIS platform, through which clients will be able to automatically generate reports starting from STIX graphs contained in the CTIS knowledge base.

AGIR is able to generate cybersecurity reports using a two-stage pipeline. In the first step, it generates an automatic report using a template-based approach and then this report is refined using ChatGPT APIs to improve its fluency. At the moment, the tool is able to produce four different types of reports but the pipeline is easily extendable to other types of reports.

This thesis is organized as follows. In Chapter 2, we will introduce some key CTI concepts. In Chapter 3, we will talk about NLG and its state of the art in the cybersecurity field, then in Chapter 4, we will discuss the implementation of AGIR, going into the details of its pipeline. In Chapter 5 we will evaluate the model using both human evaluation and SLOR metric. Finally, Chapter 6 concludes this work.

# 2
# Background

In this chapter, we introduce Cyber Threat Intelligence. In particular, we will focus on how CTI information is exchanged through the use of STIX graphs, in order to fully understand the inputs of AGIR.

## 2.1 Cyber Threat Intelligence

Crowdstrike, an American security company, defines Threat Intelligence as: "Data that is collected, processed, and analyzed to understand a threat actor's motives, targets, and attack behaviours. Threat intelligence enables faster, more informed, data-backed security decisions and change in the behaviour from reactive to proactive in the fight against threat actors." [1]. From the first part of the definition, we can clearly understand that Threat Intelligence is a process that starts by collecting data from different sources and aims to understand the habits of attackers. Sources of threat intelligence data include open source intelligence, social media, device log files, internet traffic logs and data derived from the deep and dark web. From the second part of the definition, we can realize the importance of threat intelligence. Nowadays, threat intelligence has become a crucial part of companies' cyber security strategy since it allows them to be more proactive in their approach and determine which threats represent the greatest risks to a business. This puts companies on a more proactive front, actively trying to find their vulnerabilities and preventing hacks before they happen.

### 2.1.1 CTI types

CTI programs can provide different types of intelligence based on the targeted audience and what information it mainly focuses on. There are three types of CTI [2]:

- **Strategic Intelligence:** This less-technical, high-level threat intelligence provides an overview of the organization's threat landscape. The primary audience targeted by strategic threat intelligence is the non-technical audiences, like the company board of directors and the executive-level security professionals. Strategic intelligence helps high-level staff to understand the risks and vulnerabilities associated with the organization and the goals of threat actors and provide preventive mechanisms. Based on the intelligence, it enables executive staff to drive high-level organizational strategy.
- **Tactical intelligence:** This type of intelligence targets more technically proficient audiences and focuses on the immediate future. It reveals simple indicators of compromise (IoCs) such as: malicious domain names, URLs, IP addresses and unusual traffic. IT teams can identify certain threats and mitigate the organization's risks. Tactical intelligence is simple and automated, which can be consumed through techniques like data feeds and APIs. Since IoCs can easily be changed or obsolete quickly, tactical intelligence has a shorter lifespan than the other two types.
- **Operational intelligence:** Operational intelligence targets the cybersecurity professionals who are responsible for conducting daily operations in a SOC. It provides a more in-depth understanding of how attackers plan, execute and maintain cyberattacks and operations by understanding the attributes of adversaries like TTP used for cyberattacks. Operational intelligence helps improve threat monitoring, threat management and incident response tasks. Since TTPs cannot be changed easily, operational intelligence lasts longer than tactical intelligence. Still, there are challenges in accumulating operational Intelligence. For example, encrypted messaging apps like WhatsApp and Telegraph, used by attackers for communications, are not easy to access, and the language some threat groups use can be difficult to decipher.

### 2.1.2 Threat Intelligence Life Cycle

From the previous paragraphs we have understood that threat intelligence is a continuous process, and for this reason, is often referred to as a lifecycle (Figure 2.1). The threat intelligence lifecycle provides a framework for your security teams to plan and implement their protective tactics and strategies against malicious digital behaviours. There are six phases in the Cyber Threat Intelligence Cycle [3, 4]:

1. **Direction:** The direction phase of the lifecycle is when you set goals for the threat intelligence program. This involves understanding and articulating: the information assets and business processes that need to be protected, the potential impacts of losing those assets or interrupting those processes, the types of threat intelligence that the security organization requires to protect assets and respond to threats and the priorities about what to protect. Once high-level intelligence needs are determined, an organization can formulate questions that channel the need for information into discrete requirements.

2. **Collection:** During TI collection, the intelligence team is gathering information and context that fulfil the requirements laid out earlier. The intelligence is collected from sources such as social media, deep and dark web, network data and other open source intelligence (OSINT).

3. **Processing:** Processing is the transformation of collected information into a format usable by the organization. Almost all raw data collected needs to be processed in some manner, whether by humans or machines. Different collection methods often require different means of processing. For example, human reports may need to be correlated and ranked, deconflicted, and checked.

4. **Analysis:** Analysis is a human process that turns processed information into intelligence that can inform decisions. Depending on the circumstances, the decisions might involve whether to investigate a potential threat, what actions to take immediately to block an attack, how to strengthen security controls, or how much investment in additional security resources is justified. The form in which the information is presented is especially important. It is useless and wasteful to collect and process information and then deliver it in a form that can't be understood and used by the decision maker.

5. **Dissemination:** The threat Intelligence is now ready to be shared with the user, either through a report, feed, or automated platform. The security team will use the TI to build and act on priority plans for mitigation and proactive protection, focusing on alerts of the highest importance or impact to their organization. This is also the stage where automated remediation actions may occur – such as takedown requests, publishing of attack indicators, defense hardening, etc.

6. **Feedback:** Once intelligence has been sent to relevant business units and individuals, it is time to collect feedback from the organization to determine whether the intelligence analysis was timely, relevant, and actionable.

### 2.1.3 Pyramid of Pain

After identifying the threats, security professionals must implement countermeasures in order to avoid receiving successful exploits. When devising countermeasures and mitigations, vari-


**Figure 2.1:** Threat Intelligence Lifecycle.

ous conceptual models can be employed. However, the Pyramid of Pain (Figure 2.2) stands as the most commonly utilized. The Pyramid of Pain is a conceptual model for understanding cybersecurity threats that organizes IOCs into six different levels. Information security expert David J. Bianco was the first to formalize this idea in his article "The Pyramid of Pain" [5]. The six levels of IOCs in the Pyramid of Pain are organized in order of how "painful" they would be to the attacker if the victim discovered them and took action against them [6]. From the bottom to the top of the pyramid—from least painful to most painful—these IOCs are:

- **Hash values:** A hash value is a software or file "signature" that is the output of a complex cryptographic hash function such as SHA-1 and MD5. These hash functions practically guarantee that two different files will not have the same hash value.

- **IP addresses:** An Internet Protocol (IP) address is a set of numbers that uniquely identifies a computer or other device connected to the Internet.

- **Domain names:** A domain name is a string of text that uniquely identifies an Internet resource such as a website or server.

- **Network artifacts/host artifacts:** A network artifact is produced as the result of some network activity, while a host artifact is produced as the result of some activity on a host machine.

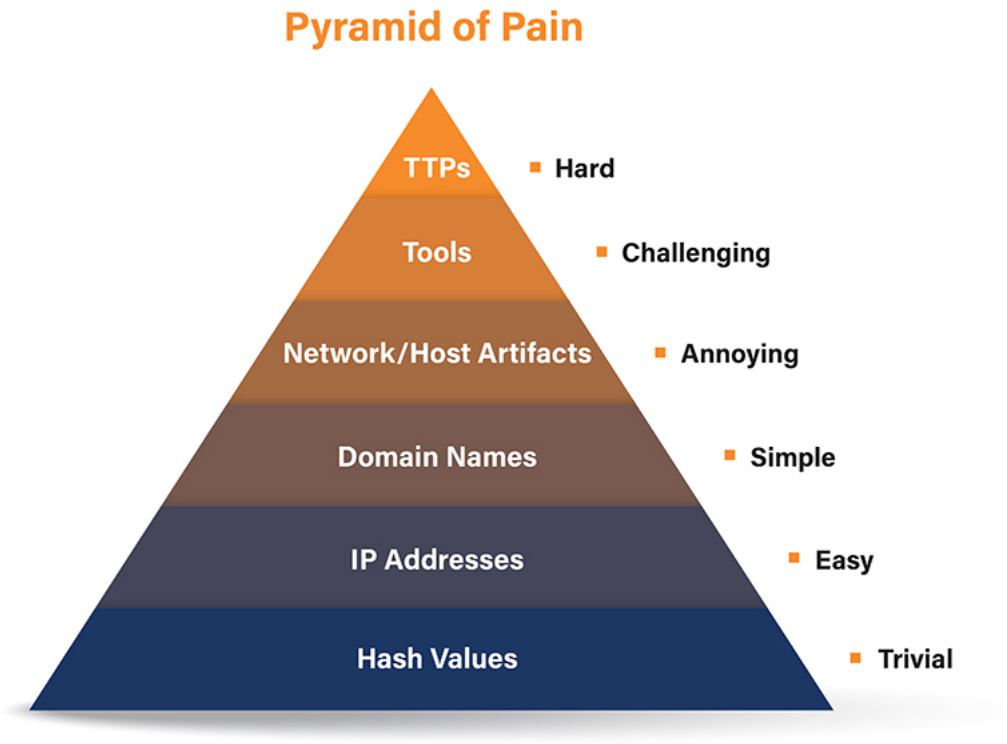


**Figure 2.2:** Pyramid of Pain.

- **Tools:** Attackers use various software tools and platforms to carry out attacks (such as backdoors or password crackers).
- **Tactics, techniques, and procedures (TTPs):** Attackers often have a modus operandi that identifies them—everything from the initial method of entry to the means of spreading throughout the network and exfiltrating data.

Using the Pyramid of Pain security professionals can prioritize the countermeasure to implement and maximize their impact.

### 2.1.4 CTI reports

One key result of the TI lifecycle is the production of documentation providing information on a certain subject or event. One of the main means of sharing this information is through the use of CTI reports. The process of writing a cybersecurity report involves these steps:

1. **Identify the purpose and scope of the report:** The first step in writing a cybersecurity report is to determine its purpose and scope. In fact, there are a lot of different types of

CTI reports, for example, we can write a report to summarize the results of a penetration testing exercise or to investigate a security incident. Defining the purpose and scope of the report will help to organize the report and choose the appropriate information.

2. **Gather information:** The next step is to gather information relevant to the purpose and scope of the report. This may involve collecting data from various sources, such as logs, system configurations, and network traffic.

3. **Analyze the data:** Once the necessary information is gathered, you'll need to analyze it to identify any security vulnerabilities or incidents. This may involve reviewing logs and system configurations to identify potential weaknesses or examining network traffic to identify signs of a breach.

4. **Organize the report:** The next step is the organization of the report, which is very dependent on its type. At the start, there is usually an outline of the key findings, such as the vulnerabilities or incidents identified during your analysis. Then, an overview of the methodology used to gather and analyze the data. Finally, recommendations for addressing any vulnerabilities or incidents identified.

5. **Write the report:** Once the report is organized the analyst has to manually write it, including in the report relevant screenshots, graphs and other visual aids to help illustrate findings and recommendations.

## 2.2 Structured Threat Information eXpression

Structured Threat Information Expression, better known as STIX, is a free and open source language for the representation and sharing of Cyber Threat Intelligence [7]. STIX enables organizations to share CTI with one another in a consistent and machine readable manner, allowing security communities to better understand what computer-based attacks they are most likely to see and to anticipate and/or respond to those attacks faster and more effectively. STIX is designed to improve many different capabilities, such as collaborative threat analysis, automated threat exchange, automated detection and response, and more. Nowadays, although other languages still exist and are used for CTI, it is considered by the majority of security people as the de-facto standard in this field. In addition to information sharing, STIX can also be utilized for cyber threat analysis. In fact, information inside a cybersecurity report can be easily represented by an equivalently descriptive STIX file. These STIX files from version 2.0 are in JSON format, currently, we are at version 2.1 of the STIX language. STIX objects extracted

from a report can be represented as a connected graph of nodes and edges, in which each node represents an entity and each edge represent a relationship between entities. Thus, a CTI report can be also represented by this graph of STIX objects, these are the inputs of AGIR. In the following subsections, STIX's main features are presented.

### 2.2.1 STIX Entities

STIX entities are the nodes of a STIX graph and they include precise information about the entity. They are of two types: STIX Domain Objects and STIX Cyber-Observable Objects.

- **STIX Domain Objects:** high-level intelligence objects that represent behaviours and constructs that threat analysts would typically create or work with while understanding the threat landscape. In STIX 2.1 there are 18 SDOs and they are characterized by the property and relationship information that are provided for them.
- **STIX Cyber-Observable Objects:** represent observed facts about a network or host that may be used and related to higher level intelligence to form a more complete understanding of the threat landscape.

Both categories have associated properties and relationships. Properties are attributes of an object, they specify certain information about the subject, for example, the type of an object. Each object has associated just a subset of all the possible properties and of that subset some properties are defined as required and others as optional.

Properties are divided into two categories: Common Properties which are shared by all objects and Specific Properties which are used just by a subset of objects. The most relevant Common Properties are:

- **type:** identifies the type of STIX Object.
- **id:** uniquely identifies this object.
- **created:** represents the time at which the object was originally created.
- **external_references:** specifies a list of external references which refers to non-STIX information. This property is used to provide one or more URLs, descriptions, or IDs to records in other systems.

Specific Properties tend to give more useful information because they are specific to a subset of objects, examples of such properties are:

- **name:** A name used to identify the entity.
- **description:** A description that provides more details and context about the entity, potentially including its purpose and its key characteristics.
- **aliases:** Alternative names used to identify the entity.
- **first and last seen:** The first and last time that the entity was first seen in action.
- **objective:** The entity's primary goal, objective, desired outcome, or intended effect.
- **entity_types:** The type of the entity being described (e.g. entity_types for a malware could include backdoor, dropper...).

AGIR supports 13 out of the 18 entity types, the remaining five (grouping, malware analysis, note, observed data and opinion) are not included. This limitation is due to the fact that the excluded entity types are not used in Leonardo's CTI platform, however, the tool remains extensible to any new entity type that may be added. A complete list of STIX entity types and their description is present in Table 2.1.

### 2.2.2 STIX Relationship Objects

STIX Relationship Objects (SROs) are the edges of a STIX Graph and connect STIX Domain Objects together, STIX Cyber-observable Objects together, and connect STIX Domain Objects and STIX Cyber-observable Objects together to form a more complete understanding of the threat landscape. STIX defines many relationship types to link together SDOs and SCOs. All these relationships can be seen in Table 2.2. Like Objects also SROs have some Common Properties that convey information on a specific relationship, some relevant properties are:

- **relationship_type:** The name used to identify the type of Relationship.
- **description:** A description that provides more details and context about the Relationship, potentially including its purpose and its key characteristics.
- **source_ref:** The id of the source object.
- **target_ref:** The id of the target object.

| Entity Type | Description |
|---|---|
| Attack Pattern | A type of TTP describing ways adversaries attempt to compromise targets. |
| Campaign | A grouping of adversarial behaviours that describes a set of malicious activities or attacks (sometimes called waves) that occur over a period of time against a specific set of targets. |
| Course of Action | An action taken either to prevent an attack or to respond to an attack that is in progress. |
| Grouping | Represent a set of data that, in time, given sufficient analysis, would mature to convey an incident or threat report. |
| Identity | Represents actual individuals, organizations, or groups as well as classes of individuals, organizations, systems or groups. |
| Indicator | Contains a pattern that can be used to detect suspicious or malicious cyber activity. |
| Infrastructure | Represents a type of TTP and describes any systems, software services and any associated physical or virtual resources intended to support some purpose. |
| Intrusion Set | A grouped set of adversarial behaviours and resources with common properties that is believed to be orchestrated by a single organization. |
| Location | A Location represents a geographic location. |
| Malware | A type of TTP that represents malicious code. |
| Malware Analysis | Captures the metadata and results of a particular static or dynamic analysis performed on a malware instance or family. |
| Note | Informative text to provide further context. |
| Observed Data | Conveys information about cyber security related entities such as files, systems, and networks using the STIX Cyber-observable Objects (SCOs). |
| Opinion | An assessment of the correctness of the information in a STIX Object produced by a different entity. |
| Report | Collections of threat intelligence focused on one or more topics. |
| Threat Actor | Actual individuals, groups, or organizations believed to be operating with malicious intent. |
| Tool | Legitimate software that can be used by threat actors to perform attacks. |
| Vulnerability | A weakness that can be exploited to negatively impact a system. |

**Table 2.1:** STIX entities and their description.

| Source | Type | Target |
|---|---|---|
| Attack Pattern | delivers | Malware |
| Attack Pattern | targets | Identity, Location, Vulnerability |
| Attack Pattern | uses | Malware, Tool |
| Campaign | attributed-to | Intrusion Set, Threat Actor |
| Campaign | compromises | Infrastructure |
| Campaign | originates-from | Location |
| Campaign | targets | Identity, Location, Vulnerability |
| Campaign | uses | Attack Pattern, Infrastructure, Malware Tool |
| Course of Action | investigates | Indicator |
| Course of Action | mitigates | Attack Pattern, Indicator, Malware, Tool Vulnerability |
| Identity | located-at | Location |
| Indicator | indicates | Attack Pattern, Campaign, Infrastructure, Malware, Threat Actor, Tool |
| Indicator | based-on | Observed Data |
| Infrastructure | communicates with | Infrastructure |
| Infrastructure | consists-of | Infrastructure, Observed Data |
| Infrastructure | controls | Infrastructure, Malware |
| Infrastructure | delivers | Malware |
| Infrastructure | has | Vulnerability |
| Infrastructure | hosts | Tool, Malware |
| Infrastructure | located-at | Location |
| Infrastructure | uses | Infrastructure |
| Intrusion Set | attributed-to | Threat Actor |
| Intrusion Set | compromises | Infrastructure |
| Intrusion Set | hosts | Infrastructure |
| Intrusion Set | owns | Infrastructure |
| Intrusion Set | originates-from | Location |
| Intrusion Set | uses | Attack Pattern, Infrastructure, Malware Tool |
| Intrusion Set | targets | Identity, Location, Vulnerability |
| Malware | authored-by | Threat Actor, Intrusion Set |
| Malware | beacons-to | Infrastructure |
| Malware | exfiltrate-to | Infrastructure |
| Malware | controls | Malware |
| Malware | downloads | Malware, Tool, File |
| Malware | drops | Malware, Tool, File |
| Malware | exploits | Vulnerability |
| Malware | originates-from | Location |
| Malware | targets | Identity, Infrastructure, Location Vulnerability |
| Malware | uses | Attack Pattern, Infrastructure Malware, Tool |
| Malware | variant-of | Malware |
| Threat Actor | attributed-to | Identity |
| Threat Actor | compromises | Infrastructure |
| Threat Actor | hosts | Infrastructure |
| Threat Actor | owns | Infrastructure |
| Threat Actor | impersonates | Identity |
| Threat Actor | located-at | Location |
| Threat Actor | targets | Identity, Location, Vulnerability |
| Threat Actor | uses | Attack Pattern, Infrastructure Malware, Tool |
| Tool | delivers | Malware |
| Tool | drops | Malware |
| Tool | has | Vulnerability |
| Tool | targets | Identity, Infrastructure, Location Vulnerability |

**Table 2.2:** STIX relationships supported by AGIR.

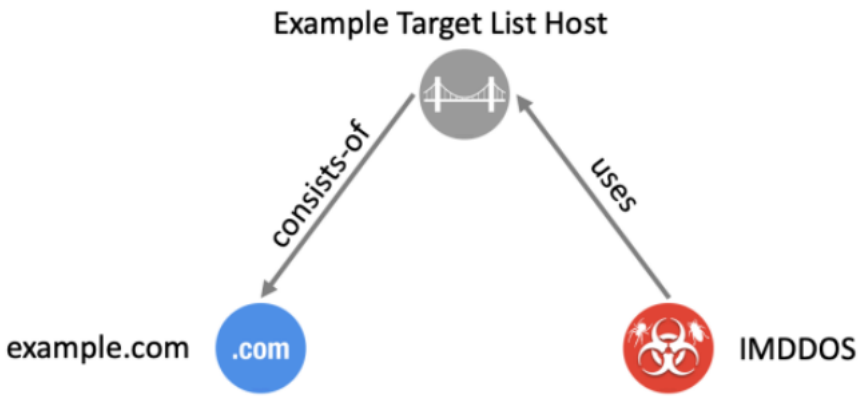


**Figure 2.3:** Example of a simple STIX Graph.

### 2.2.3 Example of a STIX Graph

In this section, we analyze a complete example of a STIX Graph in order to fully understand what is the input that both analysts and AGIR use when writing a report and how is possible to obtain useful information from this input.
In Figure 2.3, we can see a very simple example of a STIX Graph consisting of just three entities: 1 infrastructure, 1 malware and 1 domain name. Looking at the graph we can see just the name property of the entities but from the JSON representation of the graph, we can extract more insightful information. For every entity in the graph, we have a corresponding object in the JSON that contains the properties associated with the entity. For example, looking at the JSON we can understand that the malware IMDDOS is a bot malware. In the graph, we can also see two relationships: 1 uses and 1 consists-of. As for the entities, also every relationship has a related JSON object that contains its properties.

**Listing 2.1:** JSON representation of a STIX graph.

```
{
"type": "infrastructure",
"spec_version": "2.1",
"id": "infrastructure--d09c50cf-5bab-465e-9e2d-543912148b73",
"created": "2016-11-22T09:22:30.000Z",
"modified": "2016-11-22T09:22:30.000Z",
"name": "Example Target List Host",
"infrastructure_types": ["hosting-target-lists"]
```

```
},
{
"type": "relationship",
"spec_version": "2.1",
"id": "relationship--37ac0c8d-f86d-4e56-aee9-914343959a4c",
"created": "2016-11-23T08:17:27.000Z",
"modified": "2016-11-23T08:17:27.000Z",
"relationship_type": "uses",
"source_ref": "malware--3a41e552-999b-4ad3-bedc-332b6d9ff80c",
"target_ref": "infrastructure--d09c50cf-5bab-465e-9e2d-543912148b73"
},
{
"type": "malware",
"spec_version": "2.1",
"id": "malware--3a41e552-999b-4ad3-bedc-332b6d9ff80c",
"created": "2016-11-12T14:31:09.000Z",
"modified": "2016-11-12T14:31:09.000Z",
"is_family": true,
"malware_types": [
 "bot"
],
"name": "IMDDOS"
},
{
"type": "relationship",
"spec_version": "2.1",
"id": "relationship--81f12913-1372-4c96-85ec-E9034ac98aba",
"created": "2016-11-23T10:42:39.000Z",
"modified": "2016-11-23T10:42:39.000Z",
"relationship_type": "consists-of",
"source_ref": "infrastructure--d09c50cf-5bab-465e-9e2d-543912148b73",
"target_ref": "domain-name--3c10e93f-798e-5a26-a0c1-08156efab7f5"
},
{
```

```
43 "id": "domain-name--3c10e93f-798e-5a26-a0c1-08156efab7f5",
44 "type": "domain-name",
45 "value": "example.com"
46 }
```

Usually, CTI analysts start writing a report by looking at the STIX graph describing the subject of the report. The STIX graph can be either manually made by a human or automatically built starting from the information present in the knowledge base of the company writing the report. Leonardo is an example of this process, it has a software called Cyber Threat Intelligence System (CTIS) where employees can manage all the intelligence contained in the knowledge base under the CTIS. One feature of CTIS is that employees can visualize the information related to a specific entity/incident using a STIX Graph, this representation is used by analysts as a starting point for the writing process. AGIR constitutes one of the micro-services of the CTIS platform. Analysts when visualizing the graph can request to AGIR one of the four reports and the tool will produce the final report and return it in output inside the CTIS.

# 3
# Related Works

Natural Language Generation (NLG) is the subfield of Natural Language Processing (NLP) that focuses on building computer systems that can produce natural language output (e.g., documents, reports, summaries, etc.).
NLG technologies can be divided into two main categories: text-to-text and data-to-text. Text-to-text applications utilize existing texts as their input and generate a new, coherent text as output through automatic processes. Examples of such applications include machine translation, summarization, and paraphrasing. On the other hand, data-to-text systems convert non-linguistic structured data (e.g., a table or a graph) into natural language text, these technologies can be applied in tasks such as weather forecasting, patient reports or sports summaries.
The methods for the two categories are similar, but since our contribution falls in the second one we will proceed by focusing on data-to-text generation.

The mainstream methods to approach data-to-text generation are three: rule-based, template-based and neural-based.

- **rule-based:** These methods usually follow a three-staged pipeline architecture. The first module, the Document Planner, combines content selection and text structuring. Thus, it is concerned mainly with the choice of 'what to say' [8]. The resulting text plan, a structured representation of messages, is the input to the Microplanner, which typically combines sentence aggregation, lexicalisation and referring expression generation [9]. If

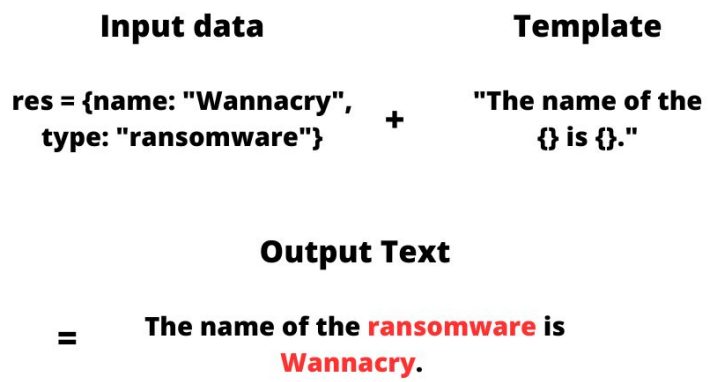


**Figure 3.1:** Example of template text generation.

text planning amounts to deciding what to say, sentence planning can be understood as deciding how to say it. All that remains then is to actually say it, i.e., generate the final sentences in a grammatically correct way, by applying syntactic and morphological rules. This task is performed by the Linguistic Realiser [10].

- **template-based:** Template-based approaches, unlike the previous ones, map their non-linguistic input directly (i.e., without intermediate representations) to the linguistic surface structure. In these systems, text generation is done via string manipulation; the user writes a program which includes string patterns that contain empty slots where other strings must be filled in. An example of such manipulation can be seen in Figure 3.1.

- **neural-based:** Neural-based approaches, are usually data-driven and do not need any form of manual feature engineering, managing to learn how to produce high quality text descriptions by themselves. These approaches usually form end-to-end models, meaning that none of the aforementioned stages is explicitly modelled; instead, neural models learn to directly generate the utterances from the input data.

The first two approaches were the leading paradigms in the early stages of NLG and their differences were a huge discussion topic [11]. Template-based systems were considered inferior to rule-based systems, despite being "Turing equivalent", because they don't embody generic linguistic insights [12]. This perception was later dispelled by the development of more complex template-based systems, such as D2D [13], which were capable of varying output based on context and performing complex syntactic operations such as aggregation. In general, we can say that template-based systems offer ease of development, higher control of the output and computational speed at the cost of fluency, maintainability and flexibility, whereas for rule-based systems the opposite holds.
With the advent of machine and deep learning the neural-based approach has become the leading paradigm in modern NLG. Neural-based systems are preferred due to their increased ability in generalization and output variance. These upsides, however, come at the cost of being completely unable to control the generation procedure, often ending up in having texts which do not accurately reflects the input information. Moreover, these systems need to be trained on large datasets, making this approach unfeasible in cases where such a set is not available.

In addition to these methods, there are also hybrid approaches that combine two or more methods in order to leverage the distinct advantages of each approach.
For example, Kale and Ratsogi [14] implemented automatic responses for virtual assistants with a two-stage pipeline. In the first step, they used a template-based approach to build a schematic baseline response. In the second step, they trained T5, a pre-trained language model, to rewrite the robotic response into coherent, natural-sounding text. Using this method they were able to combine the strengths of both approaches, obtaining a fluent and diverse response thanks to the neural approach, while having also control and ease of development offered by the template method.

NLG is used in many sectors including journalism, medicine, weather and sports, however, in cybersecurity there are almost no traces of NLG methods. To the best of our knowledge, there are only two examples of NLG systems used in the cybersecurity realm. In the first example, Das and Varma [15] built a system for advanced email masquerading in which the text of the email is automatically generated using Recurrent Neural Networks (RNN). In the second example, S. Polzunov and J. Abraham [16] implemented Narrator[1], a tool that is able to build intelligence reports starting from the JSON representation of STIX graphs. Using Narrator,

[1] https://polzunov.com/narrator/

an operator is able to see an interactive visualization of the graph and obtain four types of reports, produced using a rule-based approach. The reports can be edited and exported in PDF format.

AGIR builds on top of Narrator by leveraging different approaches that improves the overall performances. In the first step of the pipeline, we use a template-based approach inspired by the D2S system. In the second one, we use a technique similar to the one implemented by Kale and Ratsogi [14], using ChatGPT to rewrite a more human-like report.

# 4 Methodology

The multiple issues stated in the related works chapter, in particular, the lack of a large dataset and the need to have control over the output, lead to the selection of a template-based approach for the first two stages of the pipeline. The final pipeline (shown in Figure 4.1) is composed of three segments:

1. **AGIR_entity_type:** content selection from the input JSON.
2. **AGIR_report_type:** text generation through the use of templates.
3. **ChatGPT_API:** text paraphrase in order to obtain a more human-like text.

## 4.1 Input

Previously we have shown an example of a standard JSON representation of a STIX graph. Now we see the structure of a JSON input used in the Leonardo company which is the specific input to AGIR. From STIX graphs the CTIS will produce a JSON representation that has this structure:

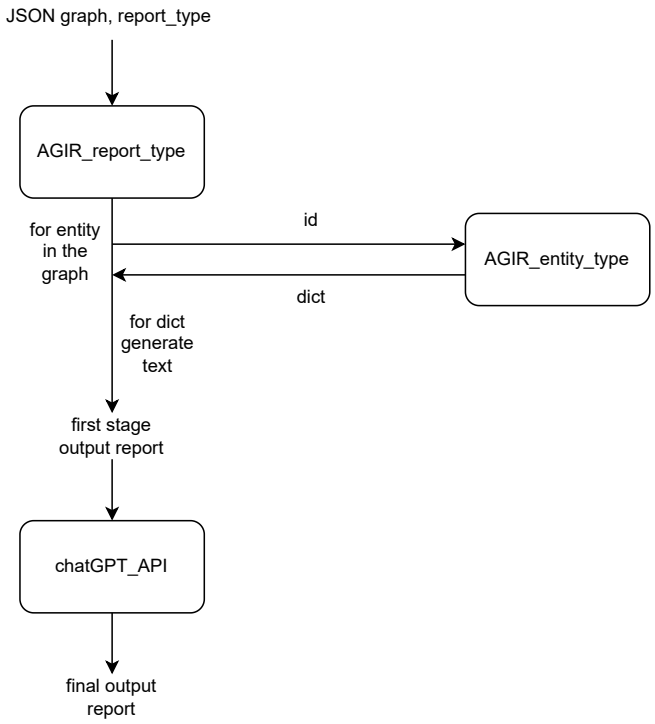


**Figure 4.1:** AGIR pipeline.

**Listing 4.1:** Skeleton of AGIR's input.

```
{
    "selectedEntity": { "properties of the subject entity" },
    "relationshipByID": { "relationships of the subject entity" },
    "idNodeRelation": { "relationships created real-time" },
    "edgeGraph": { "entities highlighted by the analyst and their
       relationships" },
    "graphItemsTrash": { "discarded/hidden entities" }
}
```

It's important to note that from this input we are able to obtain information about all the nodes and edges in any possible graph. In fact, every STIX graph in the CTIS is represented using a star structure with the subject being the centre of the star. The nodes and edges for this star are covered by the first two sections of the JSON: `selectedEntity` and `relationshipByID`. From the starting star, we have three possible ways to add nodes and edges to the graph.

1. **Expand a node:** The first one is to highlight an existing node and expand it. Performing this action all the entities and relationships related to the selected node will be retrieved from the knowledge base and added to the graph using a star structure having the highlighted node as the centre. The IDs for all these added edges and relationships are contained inside `edgeGraph`, using the id we can query the knowledge base to retrieve the information related to that entity or relationship.

2. **Create a relationship real-time:** Analysts can create a new relationship in real time between two entities when visualizing the STIX graph. The information for these relationships is contained in the `idNodeRelation` section.

3. **Hide an edge/node:** There is also the case where we have a starting graph having elements that we will like to exclude in the final report. Analysts can select a certain component of the graph and hide it. The IDs of such hidden elements are included in the `graphItemsTrash` section.

Associated with the graph is a second parameter called `report_type` in the input that specifies the report type we want in the output.

## 4.2 Content selection and text generation

In the above pipeline both `AGIR_report_type` and `AGIR_entity_type`, for ease of representation, are described as a single module, but they represent respectively 4 and 13 modules. For

each of the four report types we have a module that handles the text selection and structuring, based on the `report_type` parameter we choose among the 4 possible modules. Similar to this, for every supported entity type we have a corresponding module that retrieves the information for that entity and performs content selection.

In the first step of our system, we look at the report type parameter and select the corresponding `report_type` module. The selected module will receive as input the JSON representation of the graph presented before. Then, `report_type` selects from the JSON the entities to be included in the report based on predefined rules. Each `report_type` has different rules, so it will select different entities from the same input. For each selected entity, the id and type are extracted, and based on the entity type we call and pass the id to the corresponding `entity_type` module.

The role of each `entity_type` module is to retrieve the information associated with the entity from the knowledge base, filter only the relevant information and return the selected content to the `report_type` module. The steps to perform this task are the same for each `entity_type` module:

1. Initialization of a dictionary with six keys, each representing different categories of information for the final report. The six keys are: overview, relationships, stats, useful resources, IOC and TTP.
2. Querying the knowledge base through multiple API calls that, based on the entity id, return dictionaries containing information about the entity and its relationships.
3. Performing content selection on the obtained information. This filtering is based on rules defined during the implementation phase with Leonardo employees. The information is inserted into the initial 6-key dictionary based on its nature. For example, properties specific to the entity are inserted in the overview section while information about related IOC is inserted in the IOC section.
4. The dictionary containing the information is returned to the `report_type` module.

At this point, `report_type` will have a dictionary for every previously selected entity. The next step is to check for relationships created in real-time by the operator. In case there is one or more of these relationships, report_type will extract the id of the relationship from the graph and create a new dictionary containing the information about these created relationships. Now `report_type` has all the necessary information to write the first stage report. The report is

written in two phases. In the first one, the text is written, for every dictionary `report_type` loops over all the sections generating some text. In particular, for every piece of information in a section, it will select the corresponding template and insert the elements of that information in the placeholders. Once a section is finished it will pass to the following one iterating through all sections and dictionaries. At the end of this phase, we will have all the sentences of our report grouped by section and subject. The last thing to do is to insert these paragraphs in the predefined text structure of the report. For all four report types, a total of 85 templates were used.

## 4.3 ChatGPT APIs

The final step of the pipeline is the use of ChatGPT APIs in order to improve the fluency of some sections of the report. In particular, the overview and relationship texts are passed to the APIs of ChatGPT with a prompt asking for a more fluent version of the text while keeping the information in the text unchanged. The text returned by the deep learning model is inserted in the final report replacing the original text, an example of such an interaction can be seen in Figure 4.2. After this step, we have the final output report of AGIR. Two problems may arise in the use of ChatGPT for this paraphrasing task: the cost of such an approach and the lack of control over the text produced by the model. However, running an experiment on a sample of 36 reports the results were satisfying. The cost of the production of a single report is on average 0,0024 dollars, making it feasible for a large company such as Leonardo. As regards the lack of control, we will discuss the results in the following chapter.

## 4.4 Output

The output of AGIR is a docx document containing the intelligence report, built using the python-docx[1] library. There are four types of reports: overview, subject, timeline and vulnerability. They have different structures and different use cases, however, they all derive from the six-section layout mentioned before, which now we see in detail. As we said before for every selected entity we produce text divided into six sections:

- **Overview:** Contains specific information about the entity. The intent of this section is to introduce the entity and tell us its properties. Name, motivations and description are examples of information conveyed in this section.

[1] https://pypi.org/project/python-docx/

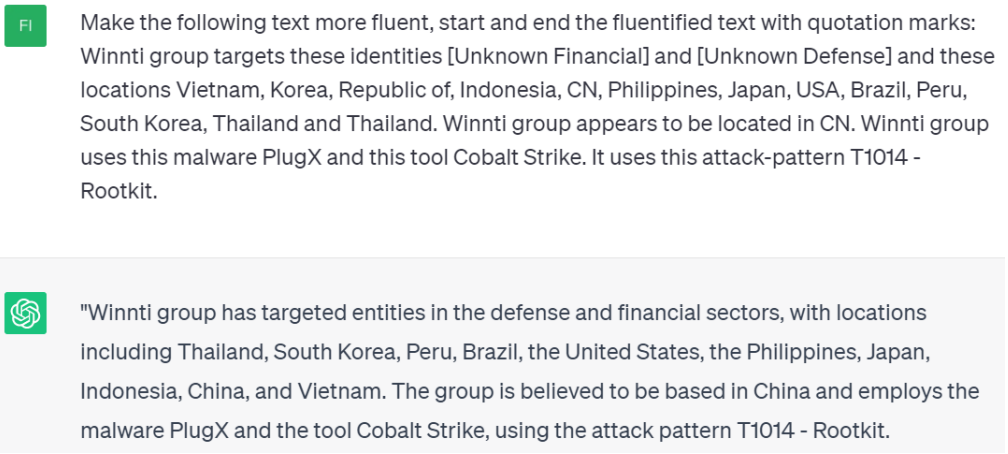


**Figure 4.2:** Example of a ChatGPT interaction.

- **Relationships:** Contains information about the relationships of the entity, such as the entities linked to it and the nature of those relationships. This section helps to locate the entity in the threat landscape.
- **Stats:** This section exploits the information contained in the knowledge base to provide more details about the nature of related entities. In particular, this section shows statistics about the nature of some entity types, namely: Attack Pattern, Malware, Intrusion Set, Infrastructure, Threat Actor and Tool. For example, looking at this section we can understand that an intrusion set is related to 8 malware, 4 of them being droppers and the other 4 being backdoors.
- **Useful Resources:** External links to retrieve more information about the entity are inserted here.
- **IOC Table:** The table consists of all the IOCs, with the specific type and value, related to the entity. The supported IOC types are sha-256, sha-1, md5, url, mac-address, ipv4-address, ipv6-address and domain-name.
- **Mitre Matrix:** The matrix contains the attack patterns related to the entity. For every attack pattern, the matrix shows the name, the tactic, the ATT&CK code and the description.

Let's now understand how each report type is structured and how the information from these sections is used to form each final report. As we know we have 4 report types:

1. **Overview report:** The purpose of this report is to provide a clear overview of the entire graph allowing the analyst to obtain a clear understanding of the situation depicted in

the graph. This report is useful if we want to provide generic information about an incident. For every selected entity all the texts from the six entities will be included. For example, if we have 4 selected entities the report will include 4 concatenated sub-reports all having the 6 sections described before.

2. **Subject report:** This report is useful when we already have in mind a clear subject and we want to focus the report on this entity. This report will contain all the information from the six sections of the subject. Moreover, will include texts for the relationship, IOC and mitre sections for all other selected entities. This choice is driven by the fact that these details help us to better understand the landscape where the subject is inserted.

3. **Timeline Report:** This report provides a timeline overview of all the entities related to a certain graph. It helps to view the chronological order of the events in the graph and to report them in order to understand their sequencing. For all the entities only the overview section is provided and every date property of the entities is reported in chronological order in the report.

4. **Vulnerability Report:** The last report is the most specific one, it is used when we want to report about specific vulnerabilities related to a certain entity. For example when we found a vulnerability in a well know tool and we want to report our findings to the community. Vulnerability reports contain the overview section for the subject entity and all the vulnerabilities related to that entity. Moreover, for each vulnerability it includes tables showing specific properties such as CVSS score, recommended mitigations and vulnerable configurations.

## 4.5 Full Example

In this section, we will see a complete example of generating a report using AGIR to better understand its functioning. Let's suppose we want a subject report on the graph shown in Figure 2.3. The input will consist of the JSON representation of the graph and the `report_type` parameter with a value of "subject".
Firstly, the input is sent to the CTIS endpoint where AGIR is working. AGIR checks the value of the `report_type` parameter and accordingly passes the JSON to the subject module. As mentioned earlier, the subject module selects the subject and all other entities highlighted by the analyst from the JSON. In our case, only the infrastructure entity is selected, and its id is added to a list. This list is then iterated, and for each id, we perform two steps. The first step is to retrieve the corresponding entity type from the id by querying the knowledge base. The

second step is to call the corresponding `entity_type` module based on the entity type. In our example, we take the first id from the list (i.e., the id of the "Example Target Host List") and query the knowledge base to discover that it is an infrastructure entity. Therefore, we pass the ID to the `get_report_infrastructure` module (i.e., the `entity_type` module for the infrastructure type).
At this point, the control temporarily switches to the `entity_type` module, which performs the following steps:

- Initialize the dictionary with 6 keys, where the initial value of each key is an empty list.
- Using the received id as a parameter, make a call to the knowledge base that returns a JSON with specific information about that entity. In our case, it queries the knowledge base using the infrastructure's id and receives back the JSON containing the information about that entity. Let's assume that the returned JSON is as follows:

```
{ “name”: “Example Target List Host”,
“type”: “infrastructure",
“created": “2016-11-22T09:22:30.000Z"}
```

- Check if the JSON contains significant properties and insert them into the overview section. The properties are extracted based on rules agreed upon with Leonardo's analysts. In our example, we extract only the first two entities because the "created" property is not considered significant. If there are links to articles/information pages about the entity in the JSON, they will be placed in the useful resources section.
- Query the knowledge base for the relationships of the entity. In our case, we will receive a JSON response of this type.

```
{ “relationship-list-source”: [{“type” : “consists-of”, “target”
: “example.com”, “target-type” : “domain-name”}],
“relationship-list-target": [{“type” : “uses”, “target” :
“IMDDOS”, “target-type” : “malware”}]}
```

- Checks if the JSON contains supported relationships (listed in Table 2.2) and, based on the entity that the subject is related to, place it in different sections of the dictionary. If the relationship is with entities of the type indicator, the information will be placed in the IOCs section. If the type is attack pattern, it will be placed in the TTPs section. In all other cases, the information is placed in the relationships category. In our case, the first relationship would be placed in the IOCs category, and the second relationship would be placed in the relationships section.

- Query the knowledge base for statistics related to the entity. In our case, we will receive a JSON response of this type.

```
                    { “attack-pattern”: [],
“intrusion-set": [{“name" : “IMDDOS", “type" : “bot"}],
                        “malware": [],
                    “infrastructure": [],
                     “threat-actor": [],
                         “tool": [] }
```

- Insert the statistics information into the stats section of the JSON.
- At this point, we have extracted all possible information about that entity from the knowledge base, and we return the compiled JSON to the subject module. Our returned JSON would look like this:

```
  { “overview”: [{“names”: {“name”: “Example Target List Host'',
                    “type”: “infrastructure”}}],
   “relationships": [{ “uses”: { “type”: “malware”, “objects”:
                            [“IMDDOS”]}}],
“stats": [{“type”: “malware”, “stats”: [“count”: 1, “type”: “bot”,
                       “names”:[“IMDDOS'']}],
                             “TTP": [],
    “IOC": [{“type”: “domain-name”, “value”: “example.com”}],
                    “useful resources": [] }
```

The IDs in the list have been exhausted, as only the infrastructure entity was selected from the input JSON. The last step of the content selection process is checking created relationships, looking at the `idNodeRelation` section. In our case, this section is empty so we can move on. At this point, we have all the necessary information about the graph to write the report. Stored in a list, we have all the dictionaries containing the information about the entities we want to describe (in our case, just one). To write the text, we proceed as follows:

1. For each dictionary, we initialize another dictionary containing four strings corresponding to the four text sections of the report (excluding IOC and TTP, which are tables and do not require generating text).
2. We iterate over the four subsections of the dictionary, and for each of them we loop through every piece of information (i.e., every dictionary containing information in one of the four sections) they have.

3. For each piece of information, there is a corresponding template. When processing this piece of information, we pass the values of the information and its name to a function that handles the insertion of these values into the template. For example, in our case, we start by processing the "names" piece of information. We first retrieve the "names" template that is "_ is _ _". Then, in the three placeholders, we insert "name", "an" and "type" respectively, resulting in the final text being "Example Target List Host is an infrastructure". In this case, we see an example of a referring expression problem, when generating the sentence we have to decide whether to insert "a" or "an" based on the next word. All these syntactic decisions are handled using the inflect[2] library.

4. Once the sentence has been completed, it is added to the respective text section (in the previous case, "overview"), and we move on to the next piece of information.

5. In the end, we will have a dictionary containing four text sections that correspond to the report text for the entity.

Before creating the final report we process each overview and relationship text using ChatGPT. The text for these sections is sent to ChatGPT APIs in order to improve its text quality. The output of this step will replace the original text in the dictionary.
The final step of the report is the actual document creation. As mentioned earlier, this step follows predefined rules and varies depending on the report. In our case, we want to include all sections from the subject entity and the relationship, IOC and TTP sections from the other selected entities.
To do this, we insert the overview and stats sections of the subject entity into the document first. Next, we iterate over all the dictionaries containing the text sections and insert each text from the relationships section labelled with the corresponding entity it refers to.
We repeat the same procedure for inserting the IOC and TTP tables. Finally, we include only the useful resources section from the subject entity. With that, we have completed the report generation process. Since we have only the subject in our final report we will have just the overview section, the relationship section, the stats section and the IOC section of the subject. TTP and useful resources sections are not inserted because we have no piece of information for these categories. Our final report can be seen in Figure 4.3. As we can see, it is very sparse, and this is because the final report heavily depends on the quantity and quality of information contained in the input JSON. In our case, the graph consisted of 3 entities and the JSON had only a few properties for each entity. By leveraging a broader knowledge base like Leonardo's,

[2] https://pypi.org/project/inflect/

# Report on Example Target List Host

## Overview

The Example Target List Host is an infrastructure specifically designed for hosting target lists.

## Stats

The infrastructure is related to this malware:
- 1 bot (IMDDOS)

## Relationships

### Example Target List Host

Example Target List Host is utilized by the IMDDOS malware.

## IOCs

| Source | Type | Value |
|---|---|---|
| Example Target List Host | domain-name | example.com |

**Figure 4.3:** Report of the graph in Figure 2.3.

the reports are able to convey more information and prove more useful, as seen in the example report shown in Figure 4.4, Figure 4.5 and Figure 4.6.

# Report on Whitefly

## Overview

Whitefly is an intrusion group, also known as Whitefly, whose main objective is to steal sensitive information through espionage. It was first observed on May 26, 2020, and its most recent activity was detected on October 12, 2021. This cyber espionage group has been in operation since 2017, targeting various organizations in Singapore from different sectors. Their primary focus is to obtain a significant amount of confidential data. Whitefly has been connected to an attack against SingHealth, Singapore's largest public health organization.

## Stats

The group is related to these malwares:
- 2 loaders (Vcrodat and Nibatad)
- 2 downloaders (Vcrodat and Nibatad)
- 1 backdoor (ShimRAT)
- 1 info stealer (ShimRAT)
- 1 exfiltration (ShimRAT)

The group is related to these tools:
- 1 tunneling (Termite)
- 1 exfiltration (Termite)
- 1 backdoor (Termite)
- 1 downloader (Termite)
- 1 credential stealer (Mimikatz)
- 1 keylogger (Mimikatz)

It is related to these attack-patterns:
- 1 command-and-control (T1105 - Ingress Tool Transfer)
- 1 credential-access (T1003.001 - LSASS Memory)
- 1 resource-development (T1588.002 - Tool)
- 1 privilege-escalation (T1068 - Exploitation for Privilege Escalation and T1574.001 - DLL Search Order Hijacking)
- 2 executions (T1059 - Command and Scripting Interpreter and T1204.002 - Malicious File)
- 3 defense-evasions (T1027 - Obfuscated Files or Information, T1036.005 - Match Legitimate Name or Location and T1574.001 - DLL Search Order Hijacking)

**Figure 4.4:** Overview and stats section of a complete subject report.

## Relationships

### Whitefly

Whitefly has set its sights on various identities, including those belonging to unknown media, defense and telecommunications entities, in addition to locations such as Myanmar, South Korea and Singapore. In order to carry out these attacks, the group employs malwares such as ShimRAT and Vcrodat, as well as tools like S0002 - Mimikatz and Termite. The attack pattern used by Whitefly is identified as T1204.002 - Malicious File.

### S0002 - Mimikatz

The tool operates within the INFR_MF infrastructure and serves the needs of the Operation Wocao campaign, as well as the intrusion-sets Turla, APT1 and APT28.

## Mitre Matrix

| Source | Name | Tactic | ATT&CK Code | Description |
|---|---|---|---|---|
| S0002 - Mimikatz | T1098 - Account Manipulation | persistence | | Adversaries may manipulate accounts to maintain access to victim systems. Account manipulation may consist of any action that preserves adversary access to a compromised account, such as modifying credentials or permission groups. These actions could also include account activity designed to subvert security policies, such as performing iterative password updates to bypass password duration policies and preserve the life of compromised credentials.<br><br>In order to create or manipulate accounts, the adversary must already have sufficient permissions on systems or the domain. However, account manipulation may also lead to privilege escalation where modifications grant access to additional roles and permissions. |

**Figure 4.5:** Relationship and TTP section of a complete subject report.

## IOCs

| Source | Type | Value |
|---|---|---|
| Whitefly | mac-addr | 00-08-74-4C-7F-1D |

## Useful Resources

Useful material to know better the set can be found at:
https://attack.mitre.org/groups/G0107 and
https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/whitefly-espionage-singapore.

**Figure 4.6:** IOC and useful resources of a complete subject report.

# 5 Evaluation

This chapter is dedicated to presenting and analysing the results obtained by AGIR. First, we analyze the accuracy obtained by AGIR and then we confront AGIR with two similar systems. We recall that our aim is to write a precise human-like report that supports analysts and reduces the time they spend in the report-writing process.

Assessing the quality of NLG model output is challenging, mainly because the majority of NLG tasks are open, in the sense that the target for a given input might not be unique. For this reason, there are different ways of evaluating NLG systems. In particular, we can distinguish between intrinsic and extrinsic evaluation methods [17]. In the case of NLG, an intrinsic evaluation measures the performance of a system without reference to other aspects of the setup, such as the system's effectiveness in relation to its users. In our example scenario, questions related to text quality and correctness of the report qualify as intrinsic, whereas the question of whether the system actually achieves its goal in supporting the report writing process is extrinsic.

Intrinsic evaluation in NLG is dominated by two methodologies, one relying on human judgements (and hence subjective), and the other on automatic metrics.

- **Human ratings:** Human judgements are typically elicited by exposing naive or expert subjects to system outputs and getting them to rate them on some criteria. Outputs are usually evaluated using Likert scales [18].

- **Metrics:** Automatic metrics take in input the text and provide a score applying some formulas. The majority of metrics (e.g. BLEU, METEOR and ROUGE) require a reference text to compute a score, however, in recent years, referenceless metrics (e.g. SLOR and WPSLOR) have been developed.

In contrast to intrinsic methods, extrinsic evaluations measure effectiveness in achieving a desired goal, this evaluation is dependent on the application domain and purpose of a system.

In the next sections, we show the results achieved by AGIR by using both intrinsic and extrinsic methods. In particular, AGIR was evaluated both quantitatively using precision, recall and F1-score, and qualitatively using SLOR and a human questionnaire.

## 5.1 Quantitative evaluation

We evaluate the accuracy of AGIR to understand if the use of ChatGPT has introduced phenomena of omission or hallucination in the final text. For this analysis, we do not compare AGIR with Narrator and first step AGIR, because these two models are entirely rule or template based and, unless there are implementation errors, they will not introduce accuracy issues. We use three different metrics for the evaluation of AGIR accuracy: precision, recall and F1-score. We define as True Positives (TP) information that are present both in the final report and in the starting JSON, False Positives (FP) information that are present in the report but not in the initial JSON and False Negatives (FN) information that are present in the input but are not found in the report. In this context, we describe information as a field of the input JSON that we expect to find in the final report. We define precision, recall and f1-score as follows.

$$Precision = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN},$$

$$F1 = 2\frac{Precision * Recall}{Precision + Recall}.$$

To evaluate the model, a sample of 12 reports (3 reports for each type) was used. For each JSON file, the information that was supposed to be included in the report was highlighted, and later checked if this information was present in the report. The entire process was manual.

| Precision | Recall | F1-Score |
|---|---|---|
| 1.000 | 0.993 | 0.996 |

**Table 5.1:** AGIR accuracy results.

The metrics scores are shown in Table 5.1. As we can see from the results, in no case the use of ChatGPT has introduced a new information in the report (i.e. hallucination) and in just a few cases the model has omitted information from the JSON.

## 5.2 Qualitative evaluation

In this section, we first introduce SLOR and then we compare our system with two similar systems: Narrator and first step AGIR (i.e., without ChatGPT intervention).

### 5.2.1 SLOR

In this section, we present SLOR [19], the metric used for the evaluation of the reports. The choice of this metric was driven by the fact that is the de-facto standard metric for the evaluation of text fluency in the case of referenceless evaluation, that is our case. SLOR assigns to a sentence S a score which consists of its log-probability under a given Language Model (LM), normalized by unigram log-probability and length:

$$SLOR(S) = \frac{1}{S}(\ln(p_M(S)) - \ln(p_u(S)),$$

where $p_M(S)$ is the probability assigned to the sentence under the LM, which can be expressed in the following product using Bayes rule:

$$p_M(S) = p(\langle t_1, t_2, \ldots, t_{|S|}\rangle) = p(t_1)\prod_{i=2}^{|S|} p(t_i|t_1, \ldots, t_{i-1})$$

and $p_u(S)$ is the unigram probability of the sentence S computed as follows:

$$p_u(S) = \prod_{t \in S} p(t).$$

The intuition behind subtracting unigram log-probabilities is that a token which is rare on its own (in contrast to being rare at a given position in the sentence) should not bring down

the sentence's rating. The normalization by sentence length is necessary in order to not prefer shorter sentences over equally fluent longer ones. Finally, note that the sentence log-probability normalized by sentence length corresponds to the negative cross-entropy of that sentence according to the language model employed during the evaluation. SLOR scores have no theoretical limit, however taking as reference a previous study by Kann, Rothe and Filippova [20], we see SLOR scores generally ranging in the interval [1,3] , the higher they are the better the fluency of that text. We calculate the probability of a sentence using the pre-trained XLNet language model [21].

### 5.2.2 Fluency, Correctness and Utility

In this section, we show the results obtained evaluating three models: Narrator, first step AGIR and final AGIR (i.e., AGIR with ChatGPT intervention). Narrator scores are useful to have a reference for the evaluation of AGIR since our evaluation is the first example of NLG evaluation on systems producing intelligence reports. The distinction between the two steps of AGIR can let us understand what is the impact of using a paraphrasing model (i.e., ChatGPT) to improve text quality. For the evaluation of the three models we have used two methods:

1. referenceless automatic fluency evaluation through SLOR;
2. questionnaire-based intrinsic and extrinsic human evaluation.

For both the evaluation methods we have taken 12 random JSON representations of STIX graphs and from this set, we have produced 12 reports for each model, 3 for each report type.

First, we analyze SLOR results. We recall that SLOR evaluates the fluency of a sentence. In order to analyze all three models we have extracted all the sentences from the overview and relationship sections of the 12 aforementioned reports. At the end of this process, we ended up having a set of 228 sentences for Narrator, 216 for first step AGIR and 178 for final AGIR. For each model, we evaluated each sentence using SLOR. The average and standard deviation of the obtained SLOR scores is reported in Table 5.2: as we can see, the average values are very close to each other for the first two systems, while for the third model, we have a significantly higher score. From this result we can derive that implementation solely relying on rule or templates struggles in producing human-like text, whereas systems using deep-learning models to generate or paraphrase text have better fluency results. Hence, from this analysis, we can conclude that the use of ChatGPT to improve the text quality was successful.

| Model | Avg SLOR | Stdev SLOR |
|---|---|---|
| Narrator | 2.13 | 0.90 |
| First Step AGIR | 2.16 | 1.07 |
| Final AGIR | 2.75 | 0.72 |

**Table 5.2:** Average and standard deviation of SLOR scores.

Let us now discuss the questionnaire-based human evaluation. This procedure involves rating the quality of a small sample of texts according to three dimensions:

- **Fluency**, whether the text is easy to read and understand;
- **Correctness**, whether the content of the text is true and derivable from the input data;
- **Utility**, whether the text helps the user to do a task. In our case if the text helps the user to write the final report faster.

The judges for this human questionnaire are 38 analysts from the Leonardo company. Fluency and correctness dimensions are evaluated using a Likert scale from 1 (not good) to 5 (very good). Utility, instead, is measured by asking judges how long they think it will take to write the final report starting from the output of the systems. The utility scores are compared with a baseline value (i.e., how long is, on average, the report writing process). The baseline score is 127,3 minutes (i.e., 2 hours and 7 minutes), obtained by asking to all analysts on average how long they take to write a report. In order to avoid results being influenced by a single questionnaire instance, we have created 3 of them with the same structure and divided the analysts into three groups of equal size. The questionnaires are composed of four sections, one for each report type, containing one report for each system for a total of 12 reports for each questionnaire. Each report is evaluated on the three dimensions cited before. In Table 5.3 we report the summary of the received feedback, grouped by dimension evaluation. As we can see, our system outperforms the baseline reference in every category. Regarding the intrinsic dimensions, AGIR has achieved more than sufficient values for both categories. Looking at the fluency values, we have confirmation that using a deep learning model for the paraphrasing part of the text significantly improves the fluency of the output, resulting in a 0.65 increase compared to First Step AGIR. We can see a more surprising result if we look at the correctness. For both instances of AGIR, we observe good results. In particular, for Final AGIR, we obtain a slightly better

| Model | Fluency | Correctness | Utility |
|---|---|---|---|
| Narrator | 2,98 | 3,00 | 97,5 |
| First Step AGIR | 3,48 | 3,77 | 79,6 |
| Final AGIR | 4,13 | 3,90 | 74,3 |

**Table 5.3:** Questionnaire results grouped by dimension.

value, which dispels previous doubts about the introduction of hallucination or omission phenomena caused by the use of the deep learning model. Regarding the usefulness of the model, analysts believe that AGIR is able to reduce the time required for report production by 42.6%, resulting in a total time reduction of 54 minutes. It should be noted that this result is only an estimate extracted from this questionnaire, and to obtain a more reliable result, it would be necessary to test it in the actual production of a report. Results divided by report type can be seen in Appendix A.

# 6 Conclusion

In this thesis, we discussed the intelligence report generation problem in the cybersecurity field. This task is particularly relevant in order to save time in the report-writing process. In order to address this problem, we proposed AGIR, a NLG system able to produce 4 different types of reports, composed of two blocks:

1. template-based approach to build a baseline report;
2. use of ChatGPT to improve the fluency of the report.

The first part of the pipeline does the majority of the work by creating an initial version of the report from the input JSON. In this phase, the tasks of content selection, text structuring, and sentence creation are performed. The second part focuses on improving the fluency of the report by leveraging ChatGPT API to perform paraphrasing.
The obtained results have been then analyzed through the means of intrinsic human evaluation, extrinsic human evaluation and the SLOR metric for fluency. The generated reports overall reached good levels of fluency and correctness. In particular, the use of ChatGPT as a paraphrasing tool has increased the fluency of the reports (as demonstrated by the questionnaire and SLOR scores). Furthermore, the use of the deep learning model has slightly increased the correctness value, which shows that it has not introduced phenomena of hallucination or omission. Regarding extrinsic evaluation, analysts have positively evaluated AGIR, believing that it can contribute to reducing report writing time by 42.6%.

For future work, we would like to verify the correctness of the obtained values for the extrinsic evaluation by having experts in the field test AGIR in the context of writing a real security report. Currently, AGIR uses ChatGPT, which can introduce cost and privacy issues. An interesting research direction would be the development of an equivalent deep learning model that can be used locally, avoiding the aforementioned problems. Another contribution could be the production of a large dataset containing the relevant STIX properties for a set of entities, on which a language model can be trained to generate the initial report. This would eliminate the maintainability issue introduced by the use of templates. Finally, it would be interesting to analyze the use of AGIR as an aggregator of different reports. Security companies, such as Leonardo, receives everyday dozens of security reports even on the same incident or threat actor. Then, they usually employ an information extraction tool such as STIXnet [22] or TIM [23] to extract the entities and relations from the report and populate the database. From the information, retrieved from different reports, contained in the database, AGIR could automatically generate a single report merging the information from all the various sources.

# A

# Questionnaire results divided by report type

| Model | Correctness | Fluency | Utility |
|---|---|---|---|
| Narrator | 2,92 | 2,77 | 109,23 |
| First Step AGIR | 3,85 | 3,46 | 86,92 |
| Final AGIR | 4,08 | 4,23 | 78,85 |

**Table A.1:** Questionnaire results for overview report.

| Model | Correctness | Fluency | Utility |
|---|---|---|---|
| Narrator | 2,85 | 2,92 | 91,15 |
| First Step AGIR | 3,92 | 3,54 | 68,08 |
| Final AGIR | 4 | 4,15 | 72,31 |

**Table A.2:** Questionnaire results for subject report.

| Model | Correctness | Fluency | Utility |
|---|---|---|---|
| Narrator | 3,38 | 3,15 | 111,53 |
| First Step AGIR | 3,46 | 3,38 | 114,61 |
| Final AGIR | 3,61 | 3,92 | 104,61 |

**Table A.3:** Questionnaire results for timeline report.

| Model | Correctness | Fluency | Utility |
|---|---|---|---|
| Narrator | 2,84 | 3,07 | 78,07 |
| First Step AGIR | 3,84 | 3,53 | 48,84 |
| Final AGIR | 3,92 | 4,23 | 41,53 |

**Table A.4:** Questionnaire results for vulnerability report.

# References


[1] K. Baker, "What is cyber threat intelligence?" 2023. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/threat-intelligence/

[2] S. Wickramasinghe, "Cti: The cyber threat intelligence guide," 2022. [Online]. Available: https://www.splunk.com/en_us/blog/learn/cyber-threat-intelligence-cti.html

[3] M. Kirschner, "Understanding the cyber threat intelligence cycle," 2021. [Online]. Available: https://www.zerofox.com/blog/cyber-threat-intelligence-cycle/

[4] "What the 6 phases of the threat intelligence lifecycle mean for your team," The recorded future team, 2020. [Online]. Available: https://www.recordedfuture.com/threat-intelligence-lifecycle-phases

[5] D. Blanco, "The pyramid of pain," 2013. [Online]. Available: https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

[6] D. Tidmarsh, "What is the pyramid of pain, and why is it important in threat detection?" 2022. [Online]. Available: https://www.eccouncil.org/cybersecurity-exchange/threat-intelligence/pyramid-pain-threat-detection/

[7] "Stix version 2.1," Oasis Open, 2021. [Online]. Available: https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html

[8] D. D. McDonald, "Issues in the choice of a source for natural language generation," *Comput. Linguist.*, vol. 19, no. 1, p. 191–197, mar 1993. [Online]. Available: https://aclanthology.org/J93-1009.pdf

[9] E. Reiter and R. Dale, *Building Natural Language Generation Systems*, ser. Studies in Natural Language Processing. Cambridge University Press, 2000.

[10] A. Gatt and E. Krahmer, "Survey of the state of the art in natural language generation: Core tasks, applications and evaluation," 2018. [Online]. Available: https://arxiv.org/abs/1703.09902

[11] E. Reiter, "Nlg vs. templates," 1995. [Online]. Available: https://arxiv.org/abs/cmp-lg/9504013

[12] K. van Deemter, E. Krahmer, and M. Theune, "Squibs and discussions: Real versus template-based natural language generation: A false opposition?" *Computational Linguistics*, vol. 31, no. 1, pp. 15–24, 2005. [Online]. Available: https://aclanthology.org/J05-1002

[13] M. Theune, E. Klabbers, J. R. De Pijper, E. Krahmer, and J. Odijk, "From data to speech: a general approach," *Natural Language Engineering*, vol. 7, no. 1, p. 47–86, 2001. [Online]. Available: https://www.cambridge.org/core/journals/natural-language-engineering/article/abs/from-data-to-speech-a-general-approach/0F7AE15E130F64C36E98826ABED68C0E

[14] M. Kale and A. Rastogi, "Template guided text generation for task-oriented dialogue," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, nov 2020, pp. 6505–6520. [Online]. Available: https://aclanthology.org/2020.emnlp-main.527

[15] A. Das and R. Verma, "Automated email generation for targeted attacks using natural language," 2019. [Online]. Available: https://arxiv.org/abs/1908.06893

[16] S. Polzunov and J. Abraham, "Narrator: Generating intelligence reports from structured data," 2020. [Online]. Available: https://www.eclecticiq.com/resources/narrator-generating-intelligence-reports-from-structured-data

[17] K. S. Jones and J. R. Galliers, *Machine Translation*, vol. 12, no. 4, pp. 375–379, 1997. [Online]. Available: http://www.jstor.org/stable/40008377

[18] E. Reiter, "How to do an nlg evaluation: Human ratings in artificial context," 2017. [Online]. Available: https://ehudreiter.com/2017/01/09/human-ratings-nlg-evaluation/

[19] A. Pauls and D. Klein, "Large-scale syntactic language modeling with treelets," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2012, pp. 959–968. [Online]. Available: https://aclanthology.org/P12-1101

[20] K. Kann, S. Rothe, and K. Filippova, "Sentence-level fluency evaluation: References help, but can be spared!" in *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, oct 2018, pp. 313–323. [Online]. Available: https://aclanthology.org/K18-1031

[21] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019. [Online]. Available: http://arxiv.org/abs/1906.08237

[22] F. Marchiori, "Stixnet: entity and relation extraction from unstructured cti reports," Master's thesis, University of Padova, 2022. [Online]. Available: https://hdl.handle.net/20.500.12608/33779

[23] Y. You, J. Jiang, Z. Jiang, P. Yang, B. Liu, H. Feng, X. Wang, and N. Li, "Tim: threat context-enhanced ttp intelligence mining on unstructured threat data," *Cybersecurity*, vol. 5m no.1, 2022. [Online]. Available: https://cybersecurity.springeropen.com/articles/10.1186/s42400-021-00106-5

# Acknowledgments

I would like to express my gratitude to everyone who has played a practical role in bringing this project to fruition. First, I am immensely grateful to Professor Mauro Conti for giving me the opportunity to work on this project and for providing guidance throughout these 10 months. I extend my sincere appreciation to Francesco Marchiori for consistently being available to offer invaluable advice. I would also like to thank the team at Leonardo: Nino, Andrea and Davide, for welcoming me into their company and providing support during my internship. Additionally, I am grateful to Sergey Polzunov and Professor Giorgio Satta for providing assistance when needed.

Moving on, I would like to thank those who have supported me and continue to support me daily in everything I do, contributing indirectly but significantly to this project. I begin by expressing my heartfelt gratitude to my family for always supporting me, without putting any pressure, and for the love they give me every single day. To my cherished friends, "I regaz", I am grateful for the countless smiles and carefree moments we share, I know that I can always count on you. Lastly, I reserve the most profound appreciation for Sara, the beacon of my life, who daily guides me with her light towards the harbour. Throughout the completion of this thesis, I have often been at open sea, and your light, as always, has led me to safety.