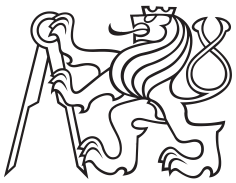
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

A Network Dataset of Normal, Malware, Attack and Background Traffic on a Real Network

Bc. Štepán Bendl

Supervisor: Ing. Sebastian Garcia, Ph.D.
Supervisor–specialist: Ing. Veronica Valeros
May 2023

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Bendl Štěpán** Personal ID number: **483816**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Cyber Security**

II. Master’s thesis details

Master’s thesis title in English:

A Network Dataset of Normal, Malware, Attack, and Background Traffic on a Real Network

Master’s thesis title in Czech:

Dataset normálního, škodlivého, útočného a ostatního síťového provozu v reálné síti

Guidelines:

- Perform a search of existing datasets in the area of network security. From your research, identify the elements you will focus on when creating a new dataset.
- Use a survey of the cybersecurity professional community to determine what the requirements are from potential users.
- Design the dataset in accordance with the principles of reproducibility and artifacts creation of the scientific community, the devices to be used, and the need for all types of network communication. Use the requirements you gathered from the community survey to guide your design.
- Implement the plan proposed during the design phase on a real network
- Process the collected data
- Create a dataset that is ready to use
- Within the dataset, create a description of the components used, the type of software, the type of attacks, and the intended use of the dataset.
- Publish the labeled dataset and the methodology

Bibliography / sources:

[1] M. Dehghan, B. Sadeghiyan, E. Khosravian, A. S. Moghaddam, and F. Nooshi, ‘ProAPT: Projection of APT Threats with Deep Reinforcement Learning’. arXiv, Sep. 15, 2022. Accessed: Oct. 10, 2022. [Online]. Available: http://arxiv.org/abs/2209.07215
[2] A. Alshaibi, M. Al-Ani, A. Al-Azzawi, A. Konev, and A. Shelupanov, ‘The Comparison of Cybersecurity Datasets’, Data, vol. 7, no. 2, p. 22, Jan. 2022, doi: 10.3390/data7020022.
[3] V. Valeros and S. Garcia, ‘Hornet 40: Network dataset of geographically placed honeypots’, Data in Brief, vol. 40, p. 107795, Feb. 2022, doi: 10.1016/j.dib.2022.107795.
[4] S. Myneni et al., ‘DAPT 2020 - Constructing a Benchmark Dataset for Advanced Persistent Threats’, in Deployable Machine Learning for Security Defense, vol. 1271, G. Wang, A. Ciptadi, and A. Ahmadzadeh, Eds. Cham: Springer International Publishing, 2020, pp. 138–163. doi: 10.1007/978-3-030-59621-7_8.
[5] H. Lawrence et al., ‘CUPID: A labeled dataset with Pentesting for evaluation of network intrusion detection’, Journal of Systems Architecture, vol. 129, p. 102621, Aug. 2022, doi: 10.1016/j.sysarc.2022.102621.

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

Name and workplace of master's thesis supervisor:

Ing. Sebastián García, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second master's thesis supervisor or consultant:

Ing. Veronica Valeros Department of Computer Science FEE

Date of master's thesis assignment: **10.02.2023** Deadline for master's thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Sebastián García, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

Acknowledgements

My sincerest thanks go to the Stratosphere laboratory team, who have supported me throughout the journey of completing this thesis. I am deeply grateful to Ing. Sebastian Garcia, Ph.D., and Ing. Veronica Valeros for their unwavering support and guidance not only during working on this thesis.

Declaration

I declare that I have prepared this thesis independently and that I have cited all sources and literature used.

In Rakovník, 15th May 2023

Štěpán Bendl

Abstract

With the increasing use of technology and the growing number of cyber-attacks, the need for robust and representative security datasets is crucial to learn how to create better tools to detect security attacks. While security datasets have been valuable in advancing cybersecurity research, most existing datasets are limited in scope and do not capture the full range of threats and vulnerabilities. Improved datasets that address these limitations would enable faster progress in cybersecurity research. Our approach involves the design of a new network security dataset through interviews with the community, designing a dataset that uses real-world network traffic data, and doing known security attacks to create a diverse and representative dataset.

The CTU-SME-11 dataset includes seven days of network traffic on eleven devices connected in an internal network. Those devices are of various operating systems, hardware, and intended use, which makes the dataset very heterogeneous. Apart from human-generated benign traffic, the dataset includes malware captures, attacks inside the network and from the internet, and attacks with data exfiltration. The biggest value of this dataset are ground-truth labels, which allow consumers to evaluate the performance of their models and algorithms accurately.

This thesis describes the whole creation process of a network dataset of normal, malware, attack, and background traffic on a real network. The CTU-SME-11 dataset contains in total around 160 GB of PCAP files and around 99,000,000 expert-labeled network flows. We hope that this dataset will serve as a foundation for future research in the field of network security datasets and will become a new benchmark dataset to be used by the cybersecurity community.

Keywords: network security, dataset, traffic capture, malware traffic, benign traffic

Supervisor: Ing. Sebastian Garcia, Ph.D.

Abstrakt

Vzhledem k rostoucímu využívání technologií a zvyšujícímu se počtu kybernetických útoků je nezbytné mít k dispozici robustní a reprezentativní bezpečnostní datasety. Tyto datasety jsou klíčové pro získání informací, které nám pomohou vytvořit lepší nástroje pro odhalování bezpečnostních hrozeb. Většina současných bezpečnostních datasetů však postrádá několik aspektů, kvůli kterým nejsou pro výzkumné účely zcela vhodné. Přístup zvolený v rámci této této práce zahrnuje návrh nového datasetu na základě sběru požadavků od komunity profesionálů v počítačové bezpečnosti. Na základě těchto požadavků je pak navržen dataset, který je následně vytvořen ze síťového provozu realné počítačové sítě.

Nově vytvořený dataset CTU-SME-11 obsahuje sedm dní síťového provozu na jedenácti zařízeních připojených ve vnitřní síti. Tato zařízení mají různé operační systémy, hardware a zamýšlené použití, což činí soubor dat velmi různorodým. Kromě člověkem generovaného neškodného provozu obsahuje datová sada chování malwaru, útoky uvnitř sítě a z internetu a zachycení provozu s exfiltrací dat. Nejhodnotnější částí datasetu je označení provozu, což umožňuje uživatelům jednoduše vyhodnotit efektivitu jejich modelů a algoritmů.

Tato práce popisuje celý proces vytváření sady síťových dat o běžném provozu, provozu se škodlivým softwarem, provozu s útoky a provozu na pozadí v reálné síti. CTU-SME-11 obsahuje celkem přibližně 160 GB souborů PCAP a přibližně 99 000 000 označených síťových toků. Doufáme, že tato datová sada poslouží jako základ pro budoucí výzkum v oblasti síťové bezpečnosti. Snahou je, aby se stala novou referenční datovou sadou pro komunitu zabývající se kybernetickou bezpečností.

Klíčová slova: zabezpečení počítačové sítě, dataset, zachycení síťového provozu, síťový provoz malwaru, neškodný síťový provoz

Contents

Figures

Tables

Chapter 1

Introduction

In today's highly interconnected world, cybersecurity has become a crucial concern for individuals, organizations, and governments. The increasing reliance on digital technologies and the internet has made us more vulnerable to cyber threats that can compromise our personal and sensitive data, disrupt critical infrastructure, and cause significant financial losses [1]. As such, protecting our digital systems and information from cyber adversaries is increasingly important [2].

Cyber threats are continuously evolving, and adversaries are becoming more sophisticated in their attacks, making it challenging to stay ahead of the threat landscape [1]. In recent years, we have witnessed an increase in the number of sophisticated cyber-attacks that have caused significant damage to organizations worldwide [3]. For example, the WannaCry ransomware attack in 2017 affected over 100,000 organizations in 150 countries, causing billions of dollars in damages [4]. Similarly, the SolarWinds supply chain attack in 2020 compromised several US government agencies and large corporations, highlighting the vulnerability of even the most sophisticated organizations to cyber threats [5].

Creating and improving network security datasets is an essential activity in the world of cybersecurity [6]. Datasets provide the data needed to train and test security systems such as intrusion detection and prevention systems, firewalls, and anti-virus software. They allow researchers and practitioners to understand the current security threats and develop new methods to detect and prevent them. Additionally, having diverse and representative datasets enables the creation of more robust and accurate models that can better detect and respond to real-world attacks. This helps to improve the overall security of networks and protect them from cyber-attacks.

Creating and improving network security datasets is a challenging task. Firstly, it is challenging to produce real attacks and obtain benign traffic due to legal, privacy, and ethical concerns. Furthermore, datasets need to be designed properly to be representative of the current threat landscape that wants to be studied, which is challenging when the threats that need to be detected are continuously changing. Additionally, the cybersecurity community has diverse needs, making it hard to create a dataset that caters to all their needs.

Our methodology to create the dataset started with reviewing previous existing datasets, then conducting interviews with cybersecurity experts to find their needs, then designing the new dataset, searching for recent threats, and finally executing the planned scenarios on a real network. Throughout the research process, the main concerns were the diversity of attacks, the authenticity of benign traffic, and the recency of malware samples, all of which are essential in creating a useful and effective cybersecurity dataset.

The contributions of this thesis are the following:

- CTU-SME-11 [7], a new novel labeled network traffic dataset of eleven devices over the course of seven days mimicking a small-medium enterprise setting. The dataset contains human-generated benign traffic, traffic from 14 real and recent malware samples, and diverse manual attack traffic. The devices incorporated in this dataset are of various types, such as mobile devices, virtual machines, IoT devices, and hardware PC.
- The CTU-SME-11 is published in Zenodo and available to the community.
- The work in this thesis was presented as a poster in the Poster Conference organized by CTU in May 2023.
- A working network setup, methodology and documentation that can be used to create similar datasets in the future.

This thesis presents the complete process of creating a network dataset that includes normal, malware, attack, and background traffic captured from a real network. The resulting CTU-SME-11 dataset consists of approximately 160 GB of PCAP files and around 99,000,000 expert-labeled network flows. Our aim is that this dataset will serve as a foundational resource for future research in the field of network security datasets, becoming a widely recognized benchmark dataset adopted by the cybersecurity community.

Chapter 2

Evaluation of existing network security datasets

This chapter aims to provide a summary of already existing network security datasets. We highlight the strengths and weaknesses of each dataset and discuss their relevance to modern network security challenges. The evaluation process involves comparing the existing datasets based on their size, diversity, and representativeness of real-world scenarios. The main goal of the evaluation is to identify the strengths and limitations of the existing datasets to determine the need for a new dataset.

Furthermore, we review and propose best practices and mistakes to avoid when creating datasets, which can help guide the design of new datasets that meet the needs of modern network security research.

2.1 KDD99

The KDD99 dataset [8] was compiled during the years 1998 and 1999. This dataset represented the first systematic approach to Intrusion detection system (IDS) data generation and was a precious and innovative resource at the time of release. KDD99 is labeled with five possible labels. One of the labels is "Normal". It represents the non-attack type of data. The other four labels are attack types: DOS (Denial of Service), R2L (Root to Local), Probe (Probing attacks), and U2R (User to Root). The KDD99 was captured in CSV format. Each record in the dataset represents a single network connection and contains various attributes such as the source and destination IP addresses, port numbers, and protocol. The dataset was created by collecting data from the network connections of a military simulation system. The simulation system was designed to mimic realistic network traffic and attack scenarios, so the data in the KDD99 dataset is not real network traffic captured from an actual network. The data are not separated into multiple files; one file with all records is supplied. Unfortunately, the KDD99 dataset has many shortcomings. The first one is that the data is quite unbalanced, which makes the classification results biased toward the majority class. Another problem is that it contains various duplicate and redundant records [6]. Due to the development in the field of computer networks, the KDD99 dataset

is outdated today. Despite this fact, the KDD99 dataset remains the most widespread dataset for testing intrusion detection systems at present [6].

2.2 NSL-KDD

The NSL-KDD [9] was proposed to overcome the shortcomings of the KDD99 dataset. The NSL-KDD dataset was created by deleting duplicate and redundant records from the KDD99 dataset 10 years after the creation of the original KDD99 dataset in 2009. Therefore, it contains less data than the original dataset. To avoid the classification bias problem, records of different classes were balanced in the NSL-KDD dataset. However, NSL-KDD did not add new data. Because of that, minority class records are still lacking, and their samples are still out-of-date [6].

2.3 ISCX-IDS 2012

The ISCX-IDS 2012 [11] dataset contains both normal and malicious network traffic activity for seven days. There are four attack scenarios in the entire dataset: infiltrating the network from inside, HTTP denial of service, distributed denial of service using an IRC botnet, and SSH brute force. Each attack scenario lasted a single day, and three days contained only benign traffic. The dataset is labeled for normal and malicious flows. The dataset only contains four malicious scenarios. No malware captures were included in the dataset [12], [11].

2.4 CTU-13

The CTU-13 dataset [13], published in 2013, comprises thirteen network traffic captures containing botnet traffic. The network consisted of virtualized computers running Windows XP on a Linux Debian host bridged into a university network. The released set of network traffic contains both the traffic from the virtualized computers and benign data from benign users in the university. The original version contains only NetFlows. The extended version includes PCAP with correct headers, but the payload was removed due to privacy reasons. The labeling strategy assigned three different labels: background, botnet, and normal. Most of the traffic was labeled as background [13].

2.5 UNSW-NB15

The UNSW-NB15 [10] dataset was created in 2015 using four tools, namely: IXIA Perfect-Storm, Tcpdump, Argus, and Bro-IDS. IXIA Perfect-Storm was used to create attack traffic, including DoS, reconnaissance, exploits, shellcode, and worms. Tcpdump, Argus and Bro-IDS were used for post-processing (i.e.

splitting the PCAP files or generating features). The classification of data was done between normal data and nine types of attacks. The authors configured three virtual servers to capture network traffic and extract features. The features were divided into five categories - basic (e.g., record total duration), flow (e.g., source IP address), content (e.g., source TCP sequence number), time (e.g., record start time), additional (e.g., No. of flows that has methods such as Get and Post in HTTP service), and label (e.g., attack category) [10]. The UNSW-NB15 dataset has been used in various modern studies. At present, the influence of UNSWNB15 is inferior to that of KDD99 [6]. We believe this dataset's main downside is generating malicious traffic through the IXIA Perfect-Storm tool and not using a real network.

2.6 CIC-IDS2017

The goal of the CIC-IDS2017 [14] dataset was to cover the most up-to-date common attacks. Another priority of the researchers creating this dataset was to generate realistic background traffic. This dataset also meets the 11 criteria of the evaluation framework for intrusion detection dataset, published by Gharib et al., 2016 [15]. An introduction to these criteria is in the next subsection.

The CIC-IDS2017 dataset has both benign behavior and also details of recent malware attacks: brute force FTP, brute force SSH, DoS, heartbleed, web attack, infiltration, botnet, and DDoS. The labels of this dataset are based on the timestamp, source and destination IPs, protocols, source ports, and destination ports. The entire network topology was configured to bring together this dataset which contains Modem, Firewall, Switches, Routers, and nodes with different operating systems. Those operating systems are Windows 10, Windows 8, Windows 7, Windows Vista, Windows NT, and Windows XP, Apple's macOS, iOS, and open-source operating systems such as Linux [6]. As a disadvantage for this dataset, we see a lack of human-generated benign traffic and small diversity of used devices (i.e. missing mobile and IoT devices).

2.7 CSE-CIC-IDS2018

The CSE-CIC-IDS2018 [16] dataset includes detailed information on attacks with abstract distribution models that can be applied to various network protocols with different topologies for computer systems. The dataset includes seven different attack scenarios, including, Heartbleed, brute force, DoS, web attack, infiltration attack, botnet attack, and DDoS attack. Compared to CIC-IDS2017, the CSE-CIC-IDS2018 dataset was prepared from a much more extensive network of simulated client targets, and attack machines [16]. The downside of this dataset is mainly the absence of human-generated benign traffic.

2.8 DAPT 2020

The creators of DAPT 2020 [17] recognized a gap in the field of Advanced Persistent Threats (APT) datasets. To address these concerns, they proposed the dataset DAPT 2020, which consists of attacks that are part of Advanced Persistent Threats. These attacks are hard to distinguish from expected traffic flows. The attackers use various techniques to gain access to the target's network, such as social engineering, phishing, and malware. Once they have access, they often use stealthy methods to maintain a presence on the network and evade detection. DAPT 2020 captures the various aspects of real-world APT attacks. These include attack behavior both at the public-to-private interface (attacks from the internet) and inside the network. The threat model used to create the APT dataset incorporates the four main phases of an APT attack - reconnaissance, foothold establishment, lateral movement, and data exfiltration [17]. The possible improvements we see in this dataset are the use of various device types and execution on a real network.

2.9 IoT-23

IoT-23 [18] is a dataset of network traffic from the Internet of Things (IoT) devices. It has 20 malware captures executed in IoT devices and three captures for benign IoT devices' traffic. The network traffic captured for the benign scenarios was obtained by capturing the network traffic of three different IoT devices: a Philips HUE smart LED lamp, an Amazon Echo home intelligent personal assistant, and a Somfy smart door lock. It is important to mention that all devices used in this dataset are real hardware. This allowed the creators to capture and analyze real network behavior. Both malicious and benign scenarios run in a controlled network environment with an unrestrained internet connection like any other real IoT device [18]. The biggest difference from other datasets compared as a part of this thesis is that the IoT-23 dataset contains traffic for attacks from the network to the Internet. As a drawback of this dataset, we consider using only IoT devices, which reduces the possibility of using this dataset for a real company/home network.

2.10 Hornet 40

Hornet 40 [19] is the first dataset designed to help understand how the geolocation of honeypots may impact the inflow of network attacks. The dataset contains forty days of raw flow data captured from eight cloud Linux passive honeypot servers. Each honeypot was located in a different city in the regions of North America, Asia, and Europe. No honeypot software was running on the Linux servers, but each IP address received connections from the Internet. All source IP addresses communicating with the honeypots are considered attacking IPs due to one of the definitions of honeypots: since a

honeypot is not an authorized production service, nobody should connect to it, and therefore all connections are considered attacks [19].

2.11 CUPID

The CUPID [20] dataset is a labeled dataset with penetration testing for evaluation of network intrusion detection. The objective of CUPID is to reflect both human interaction on a network and automated traffic for benign and malicious activities. Benign data were generated by scripting virtual user actions through PowerShell. Each workstation belongs to a specific mock user, and each mock user is assigned a fixed profile: administration, engineering, or business. Malicious data was generated by triggering tools available through the Kali Linux distribution. A preconfigured script was used on each attacking node using command line arguments for existing exploit tools. Additionally, human-generated malicious and benign traffic was gathered. The human-generated malicious traffic is a key feature of the CUPID dataset. The creators recruited ten ethical penetration testers of various skill levels to participate in the creation of CUPID. They captured the normal browsing traffic of these ten testers as they conducted the same activities as the scripted users. After that, one hour was given to the testers to attack the servers [20]. While this dataset is one of very few that contains human-generated benign traffic, this traffic is captured only for one hour, which we consider insufficient. Moreover, as this dataset focuses solely on human attackers, we find the improvement for this dataset in the execution of malware samples.

2.12 Comparison of existing datasets

Table 2.1 summarizes the examined datasets, providing a side-to-side view of the characteristics of the main datasets. It can be observed that most of the datasets include a reduced number of computers in the dataset. Not all datasets include attacks from the Internet, and very few of them include advanced attacking techniques such as lateral movement or APT-style attacks.

Dataset	Number of computers	Attacks from the Internet	Days captured	Attacks in, out & insider	Labels	Lateral movement	APT	Year
KDD99	N/A	No	63	In	Yes	No	No	1999
NSL-KDD	N/A	No	63	In	Yes	No	No	2009
ISCXIDS2012	N/A	No	7	In & insider	Yes	No	No	2012
CTU-13	13	Yes	12	In & out	Yes	No	No	2013
UNSW-NB15	5	No	2	In	Yes	No	No	2015
CIC-IDS 2017	12	No	5	In	Yes	No	No	2017
CSE-CIC-IDS2018	520	No	10	In & insider	Yes	No	No	2018
DAPT2020	2	No	5	In	Yes	Yes	Yes	2020
IoT-23	23	No	5	In & out	Yes	No	No	2020
Hornet 40	8	Yes	40	In	No	No	No	2021
CUPID	23	No	4	In	Yes	No	No	2022

Table 2.1: Comparison of evaluated existing network security datasets.

2.12.1 Dataset evaluation framework

The dataset evaluation framework created by Gharib et al., 2016 [15] is a set of rules and best practices for IDS dataset evaluation. The framework can also be used as a benchmarking method to measure the quality of newly created datasets (i.e., ensuring that the dataset meets the proposed criteria). The framework introduces 11 features that a *good* dataset should have according to their criteria. The features are:

1. **Complete network configuration** - It is necessary to have a realistic configuration in the testbed to capture the real effects of attacks.

2. **Complete traffic** - Based on the traffic generation techniques, it is possible to have realistic, pseudo-realistic, or synthetic traffic in a dataset.

3. **Labeled dataset** - If there are no correct labels, without a doubt, it is not possible to use a dataset, and the results of the analysis also are not valid and reliable.

4. **Complete interaction** - For the correct interpretation of the evaluation of the results, one of the vital features is the amount of available information for anomalous behavior. So, having all network interactions such as within or between internal LANs is one of the major requirements for a valuable dataset.

5. **Complete capture** - Even in a complete traffic dataset, it is essential to capture all traffic for researchers who want to evaluate their proposed detection systems. It seems some of the datasets are capturing traffic partially and removing part of the traffic which is Non-functional or not labeled, while it is very influential to have all traffic together to calculate the false positive percentage of an IDS system.

6. **Available protocols** - Provide the presence of all commonly available protocols, such as HTTP, HTTPS, FTP, SSH, and email protocols.

7. **Attack diversity** - Types of attacks are changing and updating daily. So, having the ability to test and analyze IDS and IPS systems by these new attacks and threat scenarios is one of the most important requirements that a dataset should support.

8. **Anonymity** - The privacy-compromising issues occur when both the IP and payload are available. Most of the datasets removed their payload entirely, which decreases the usefulness of the dataset, especially for some detection mechanisms such as deep packet inspection (DPI).

9. **Heterogeneity** - In the IDS domain, it is possible to have different sources for creating a dataset, such as network traffic, operating systems logs, or network equipment logs. A homogeneous dataset with one source type can be useful for analyzing a specific type of detection system.

10. **Feature set** - The main goal of providing a dataset is its usability for other researchers to test and analyze their proposed system. One of the main challenges is how to calculate and analyze the related features.

11. **Metadata** - Lack of proper documentation is one of the main issues in available datasets in this area. Most datasets do not have documentation, or even if they have, it is incomplete. Insufficient information about the network configuration, operating systems for the attacker and victim machines, attack scenarios, and other vital information can detract from the usability of a dataset for researchers [15].

2.13 Best practices for creating new IDS datasets

Complementing the 11 features that are mentioned in Section 2.12.1, this thesis summarizes a set of best practices and issues to avoid while creating new network security datasets. Based on our research, we propose a basic set of things to do and things to avoid.
Things we propose while creating a new network security dataset:

- Capture raw data in PCAP and other formats
- Capture the data as bidirectional flows
- Label data according to the different needs of the dataset consumers
- Preserve the privacy of the experiment participants
- Capture over longer time periods (hours to days, sometimes weeks)
- Capture real network traffic from as large an environment as possible and ideally multiple of these from different networks
- Use realistic configuration from the network and generated traffic point of view
- Capture all phases of the cyber kill chain [21]
- Have a realistic balance of the data, and an option to decide which balance may be used by the researcher

Things we suggest to avoid while creating a new network security dataset:

- Use old attacks
- Have lack of human-generated traffic
- Have duplicate and redundant records
- Include artifacts (such as researchers connecting to a machine) in the dataset

Chapter 3

Network security community needs

To create a new, rich, and useful network security dataset, it is paramount to evaluate what are the current needs of the network security community. To achieve this goal, multiple cybersecurity members around the world were interviewed to gather their needs and preferences on the topic.

The interviews were conducted online following a format of semi-structured interviews [22]. The main theme was about how an ideal network security dataset should look like. The interviews were recorded with the consent of the participants. The analysis of the interviews helped identify community needs that can be grouped into three core topics: design, output, and labels. Each of these core topics is discussed next.

3.1 Normal data

Generating normal data was the most crucial topic during the interviews with the security experts. All respondents stated that having realistic benign data is crucial for a good network security dataset. The ideal way of getting the normal data would probably be to infect a real network inside a company with malware and make a capture of that. This approach is unfortunately not possible, as no company would voluntarily get infected by malware and break their cyber infrastructure. But knowing this, the question of obtaining normal data then transforms to "How to be as close as possible to the ideal scenario?".

No specific suggestions were provided in this area. This is due to the complexity and difficulty of generating normal data, which is a large and relevant problem in the field still today.

3.2 Design

When it comes to design, four core suggestions were extracted from the interviews, mostly aimed at making the dataset as realistic as possible.

3.2.1 Timing of the attacks

The first suggestion was intended to make the dataset realistic. According to the experts, with this suggestion incorporated, the dataset should follow a timeline of a traditional cyber attack. The following steps should mimic such an attack:

1. Attackers infect multiple victims with malware
2. The malware sends a bit of malicious traffic
3. Next 2-3 days the malware does nothing malicious
4. After this period, the malware starts to be active again - exfiltrate data, do damage to the victim's machines

While in most of the existing datasets, the infected device starts sending malicious traffic almost immediately after being infected, it is often not the case in real-world attacks. The interesting part of this technique is the period after the pause.

3.2.2 Malware overlap

In the existing network security datasets, it is not common to have a device infected with multiple malware samples at the same time. On real-world machines, it is possible to have the device infected with multiple malware. As those scenarios are not well documented, the knowledge about the network traffic remains limited.

The goal of this suggestion is to provide a representative case of network traffic when two or more samples do overlap.

3.2.3 IP addresses

Multiple suggestions regarding IP addresses have been received during the interviews.

- Attacking public IP addresses should not be reused. If they are reused, it is too easy to detect. Use a different IP address for each attack.
- Exfiltrate data to IP addresses that share the address with a trustworthy service (such as Discord, Dropbox, Google Drive, etc.)

3.2.4 Continuous traffic

Some of the existing network security datasets captured the network capture separated by days. This implies that the malicious actions conducted by the same malware on day X and day X+1 are not captured in a single file. The connected data holds significant value and might play a crucial role in understanding the dataset within a broader context. Therefore it would be beneficial to provide network traffic data captured for a continuous period,

such as one week, rather than having captures from a bigger time frame and then putting them together.

3.3 Format of the Output

There were multiple recommendations on the data formats and the format of the data included in the dataset. Those recommendations are:

- Provide a detailed description of each attack in the README file (Proposed by 50% of interviewed researchers).
  - This description should include information about the malware, time of execution, number of packets sent, etc.
- Provide a summary of the intended usage of the dataset.
- Describe what the dataset contains (Proposed by 75% of interviewed researchers).
- Describe the balance of the dataset (Proposed by 50% of interviewed researchers).
- Describe the artifacts of how the data was created.
- Provide a detailed description of labels (proposed by 75% of interviewed researchers).
- Provide the captures in network flow format.
- Provide Suricata alerts.
- Ensure that the data is minimized to its smallest possible size without excluding any essential components.

3.4 Labels

All interviewed security experts mentioned the need for labels in datasets. Having a labeled dataset is crucial for future work with the dataset. When seeing a label, the researcher immediately gets information about the kind of data. Without having the label, the time spent trying to understand what is going on may dramatically increase. Also, when the dataset is used for supervised learning, a label is one of the required attributes.

3.4.1 Label types

While individuals may envision the dataset is labeled with either a "Malicious" or "Benign" label, it is often not enough. Identifying a packet or flow as malicious or benign does not provide complete information. When researchers are focused on a specific problem, they often need to know what kind of

malicious behavior they are dealing with exactly. Is it command and control traffic? Are we looking at network scanning, or is it a DoS attack?

The other side of the problem is benign traffic and its labels. We might be tempted to assign a "Benign" label to all the traffic from the IP addresses we consider benign. Although, the important question is, "How do we know that the traffic is not malicious?". The answer is simple. We can only assume that the traffic from devices we have not infected on purpose is benign.

Another thing to consider is that in the dataset creation environment, the not-infected computers reside in a network with infected computers. The infected computers might be port scanning the internal network, sending broadcast traffic to other computers in the network, and other activities, that we simply cannot consider as benign. This kind of traffic should be considered malicious traffic and assigned a "Malicious" label.

3.5 The capture duration

The needs for different capture sizes vary from use case to use case. Researchers specializing in honeypots prioritize datasets that offer maximum duration. Researchers specializing in analyzing network traffic in infected environments require a dataset of sufficient duration to capture all anomalies and different scenarios yet small enough so researchers can work with it comfortably. The suggestion received from two of the interviewees is to have around a week of traffic with multiple clients in the inspected network. A great dataset should also consider different real-life scenarios, such as capturing holidays, weekends, power-downs, moving the device between work and home networks, etc.

3.6 Variety of traffic

Datasets should contain a wide variety of traffic. The larger the variety, the wider use the dataset can have and the better the ML models using this dataset will be. The variety of traffic is connected with the wide variety of labels mentioned in the previous sections.

Variety should also be considered when choosing the devices present in the network topology used for capturing network traffic for the dataset. Ideally, the topology should be a representative example of a bigger network containing different device types, operation systems, and network elements.

The last concern regarding the variety of traffic was to incorporate multiple network protocols. The reasons are the same as in the previous paragraphs.

Chapter 4

Design of the dataset

This chapter introduces the design of the CTU-SME-11 dataset. It explains the reasons behind the decisions taken during the preparation phases. The goal of the CTU-SME-11 dataset is to mimic a network of a small-medium enterprise (SME), such as a startup developing web applications. The design decisions taken in the rest of this chapter are in light of this scenario.

Crafting a well-designed experiment is crucial at every stage of creating a new dataset. It not only defines the desired structure and format of the final dataset but also serves as a guiding compass for decision-making throughout the data capture and post-processing phases.

First, this chapter introduces network topology, which is used for the capture of data for CTU-SME-11. It continues with a technical solution for data capturing and a schedule of execution. The chapter ends with a labeling methodology to be used in the labeling phase.

4.1 Network topology

The network topology is one of the most important parts of design decisions. It defines the number of devices, the type of devices, how the devices are connected, and other additional supporting hardware (routers, switches, etc.). The full network topology is shown in Figure 4.1.

4.1.1 Devices

The devices were chosen to fit the dataset scenario, that of a small-medium enterprise. The mixture of devices includes desktop devices, mobile devices, and smart devices. Desktop devices are a mixture of real devices and virtual machines (VMs). The remaining devices were all real hardware devices. Additionally, there are hardware networking components such as routers and switches.

All popular operating systems were incorporated. On desktop operating systems are Windows, macOS, Linux, and Raspberry Pi OS. For Windows, the network contains a domain controller (DC) with three active users. This was chosen as it is very common for malware to take advantage of the interconnection between DC and the users (the attack is known as Pass the

hash[36]). These four Windows machines are hosted together with a Linux virtual machine (VM) on a Linux host using VirtualBox. Apart from these VMs, there is a physical machine representing macOS. These devices are connected via a network bridge to a switch, which sends packets to a remote packet capture device.

The virtual machines are not only selected to create a diverse network topology but also to replicate small-medium enterprise (SME) environments, where virtual machines could be a common part of network topology.

When it comes to mobile devices, the two main competitors are represented in our dataset as well. Android and iOS devices capture today the vast majority of the mobile devices market [33]; therefore, it is important to have them in the network.

To complement the startup SME environment, the setup includes smart devices in the network. Two of the most common smart devices are included, a Google Chromecast TV assistant and an Amazon Alexa Echo 1st generation.

Mobile devices, together with smart devices, are connected to a router via WiFi. This router is then connected to the central switch. The central switch is then connected to a router which gives Internet access to the devices. This router is also configured to redirect some external IP addresses and ports to some of the internal devices, effectively making Amazon Alexa, Raspberry Pi, and Chromecast accessible from the Internet. Therefore, we can also expect to see attacks from the Internet.

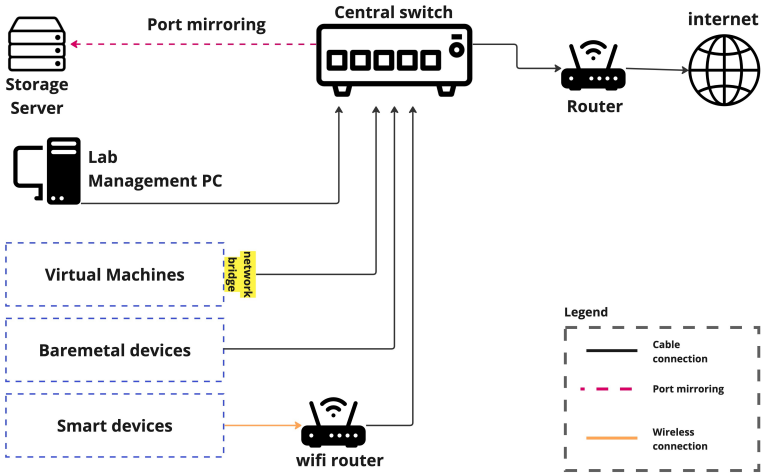


Figure 4.1: Network topology diagram of the network created for the CTU-SME-11 dataset. Desktop devices, both bare-metal, and VMs, are connected to the central switch. Smart devices, mobile and IoT, are connected to a WiFi router, which is connected to the central switch. The central switch provides access to the internet and simultaneously forwards the incoming traffic to the storage server. The lab management PC is used on demand to control the devices.

4.2 Traffic capture

As described in the network topology, the main traffic capture was done in a capture server, which had a SPAN port [1] from a central switch. All devices are connected to the switch, enabling the capture of all the traffic using port mirroring. The capturing device stored one separate PCAP capture per device in the network. Due to network bandwidth restrictions, it is impossible to avoid packet loss in high-traffic situations. Therefore it could have happened that a very small amount of packets were lost in high-traffic situations. However, we expect this to happen to less than a dozen packets per day. As a backup mechanism, two more extra packet captures were done. One in the same capturing device but for all the traffic, not separated by devices or days. And the second backup was done by VirtualBox on each virtualized device.

The main capture was done in PCAP format. The rationale behind this is that, primarily, researchers need PCAP files since it is the most comprehensive traffic file format, while at the same time, it is possible to use these files to derivate data such as network flows (Zeek flows, Argus flows, or others), Suricata alerts, and more. The advantage is that while running the malware and doing traffic capture, the focus is only on capturing the PCAP files, and there is no need to set up, configure, and monitor the generation of other files.

The capture files are divided into single days (from 00:00 UTC to 23:59 UTC). The main advantage of having the capture split by day is that the file size is smaller, and therefore there is a clearer possibility of describing what happened in the network (i.e., which flows are malicious and which are benign). While there are drawbacks to this solution, such as cross-cutting communications spawning multiple days, this is overall a better solution in terms of how the final users of the dataset are consuming the data.

4.3 Execution schedule plan

In this section, we discuss the execution schedule of the dataset creation process. Figure 4.2 represents the plan of the execution schedule visually. At a high level, the general requirements for the dataset were as follows:

- The experiment duration is designed for a full week, from Monday to Sunday.
- During the first and last day, no malicious activity is present on the devices.

[1] SPAN (Switched Port Analyzer) is a dedicated port on a switch that takes a mirrored copy of network traffic from within the switch to be sent to a destination. The destination is typically a monitoring device or other tools used for troubleshooting or traffic analysis [23].

- During the first and last day, all devices shall be adequately secured to prevent any malicious activity, except for potential attacks from the internet targeting devices with public IPs.
- Throughout the days that have malicious traffic, benign traffic must continue uninterrupted.

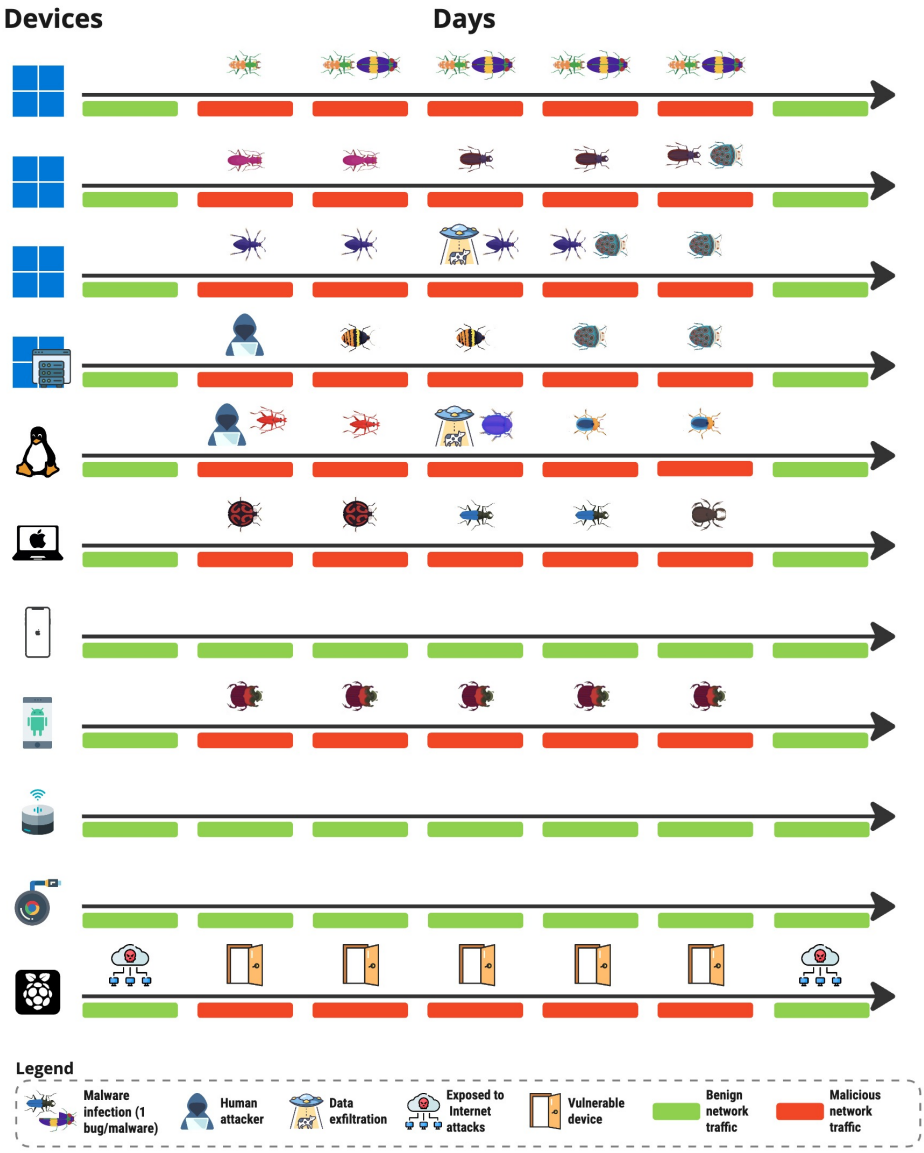


Figure 4.2: Schedule of the execution for the CTU-SME-11 capture.

4.3.1 Day 1

On the first day, the plan is to collect the human and computer-generated benign traffic. For these purposes, seven volunteer researchers were physically

assigned to the devices with the task of using the device for their usual work. As humans generate the traffic, it should be possible to see a decrease in traffic during lunch break and after working hours. No attacks or clicking on suspicious links were allowed. On the other hand, it is more than welcome to accept all of the cookies and visit various benign sites. Participants were asked not to log in to personal or work-related websites with their own real credentials to protect their privacy.

4.3.2 Day 2

On the second day of the experiment, the dataset creation process witnesses the onset of malicious traffic. However, the iOS mobile device, Alexa Echo 1st Gen, and Google Chromecast devices remain benign. Notably, the Alexa, Raspberry Pi, and Chromecast devices are susceptible to attacks from the internet due to their public IPs.

Furthermore, specific Windows virtual machines (VMs), namely VM1, VM2, and VM3, are deliberately infected with distinct malware variants that have been pre-collected and tested before the start of the experiment. Similarly, the Ubuntu server, MacOS device, and Android device are infected with distinct malware for data collection and testing purposes. The Windows Active Directory (AD) server is subject to attacks by a human attacker operating from the Ubuntu server, replicating an insider threat scenario.

4.3.3 Day 3

On the third day of the experiment, additional malware samples are executed on the devices. Specifically, the Windows AD server is no longer subject to attacks and instead is utilized for running another malware sample. Additionally, the Windows VM1 is infected with an additional malware variant while retaining the previously injected malware, resulting in a simultaneous infection with two distinct malware samples. Notably, the remaining devices continue to maintain the state from Day 2, as the intention is to sustain the malware execution for a prolonged duration (beyond a small number of hours).

4.3.4 Day 4

Day four is planned to be a day of malware rotation for some devices. Windows VM2 is reset back to a benign snapshot and infected with a new malware sample. The exact process is planned for the MacOS machine, yet this machine needs to be reverted to a benign state with a Time machine backup. The Windows VM3 and Ubuntu server becomes a part of a human-controlled data exfiltration attack while still running malware samples in the background. The rest of the devices are left unchanged.

4.3.5 Day 5

On day five, the plan is to infect all virtual machines and macOS machine with ransomware samples.

For Windows machines, the ransomware is executed on Windows VM2, where malware is still running from the previous day. The ransomware should spread to other Windows machines while taking advantage of the AD connection between the machines. The AD server is cleaned from the previously running malware before the ransomware infection.

The Ubuntu server is infected with specific Linux ransomware. For the macOS machine, macOS ransomware is selected and run on this machine. Both of these devices are be cleaned before the ransomware infection. The Android device keeps running the malware from day 2.

4.3.6 Day 6

On day six, all devices keep running the same malware(s) as on day five. There is not a lot of benign traffic in the background, as this day is Saturday, and we mimic the whole workweek throughout this experiment. There is a cleanup that happens by the end of the day to let malware samples run as long as possible while still keeping the last day only benign. The virtual machines are restored to a benign snapshot, and bare metal devices are either reset to factory settings (Android) or reverted back via TimeMachine (macOS). In the case of Raspberry Pi, the SD card with the operating system is exchanged and made safe again.

4.3.7 Day 7

The last day of the experiment is benign only. Not even human-generated traffic takes place, and only the machine-generated benign is allowed. As this day is planned for Sunday, we simulate a day where no one is working, and mostly background traffic is present in the captures.

4.4 Labeling

The labels for the dataset are assigned to the network flows that were generated by the Zeek traffic monitoring tool [24]. A tool called NetflowLabeler [25] is used to label network flows. Using this tool, we are also using a dedicated terminology and topology as described by [26]. A configuration of this tool is needed to assign the labels. Therefore all of the flows are labeled by a professional security researcher.

The NetflowLabeler tool enables the assignment of a main label, such as "Malicious" or "Benign" but also allows for the assignment of more detailed and informative labels. Consequently, the dataset consumers can select the specific labels that best align with their unique research or operational requirements. The labeling methodology is discussed deeply in chapter 9.

Chapter 5

Selection of malicious actions

The selection of malicious actions is an essential component of creating a network security dataset. Malicious traffic refers to network traffic generated by malware and other malicious actions, such as attacks or data exfiltration, performed by humans or other methods. For this experiment, we required malware with specific characteristics, which are described below:

- Malware should be recent: malware should have been created and seen in recent days or months.
- Malware should generate network traffic: as part of its behavior, we expect the malware to use the network.
- Malware should be detectable: the amount of network traffic should not be too small, making it still detectable by machine learning algorithms.
- Malware or malicious tool should be easy to set up: as many malicious actions need to be manually executed in the dataset, we needed to make sure that the time needed for the creators to set up and run malware/attacks/exfiltration is not too high to make the execution possible on schedule.

5.1 Malware

This section focuses on the selection of malware samples that are executed and captured during the dataset creation.

As already mentioned, we wanted to ensure that the malware is recent and can be detected through network traffic (i.e., it generates network traffic). At the same time, the aim was to create a dataset with a wide variety of malware. To fulfill this goal, each selected malware should be different.

The malware was selected from various sources, including semi-public sources like MalwareBazaar[1] and JoeSandbox[2], private sources of captured malware, and public crowdsourced reports on platforms like Twitter[3] by

[1] `https://bazaar.abuse.ch/`
[2] `https://www.joesandbox.com/`
[3] `https://twitter.com/`

searching for `#malware`. Additionally, multiple GitHub repositories and security-related blogs were searched to find the most common and recent malware samples [4] [5] [6] [7] [8].

The hashes of the collected malware were then verified through VirusTotal [27]. The verification included the following:

- Check for the first and last submission to the VirusTotal platform to verify that the malware sample is not too old.
- Check if there are identified hardcoded IP addresses in the static content of the malware indicating possible network behavior.
- Check if there is an analysis from any malware sandbox and if that can indicate network connections.

If the verification was successful, we proceeded with testing the malware in our infrastructure to make sure our devices and virtual machines were capable of running the sample. This testing is not part of the final dataset. During this testing, we executed the malware, verified the network traffic, and searched for potential malicious patterns. If this second phase was successful, the sample was ready for use during the dataset creation.

It was common that it was not possible to execute some files or that they did not show any network traffic. The lack of network traffic is the main reason behind not executing any malware in our dataset.

A total of 14 different malware executables were selected for the creation of the CTU-SME-11 dataset. The overview of the selected malware is shown in Table 5.1, along with the malware platform, probable name, type of malware, file type, and the malware executable SHA256.

ID	Platform	Name	MType	File Format	SHA256
1	Windows	Nanocore	Adware	EXE	f26c9aff73c041c5a506b8e8ede7bc50fe468d34b0cbf750940bb195eb068b0b
2	Windows	RedLineStealer	Infostealer	EXE	3c7883524728e43e640e1b53d4ce65d582cf7b7c42ca2baf3371ac752ddc216d
3	Windows	TrickBot	Banking trojan	EXE	1e90b6fc99a908420de123418deded8d8eadf2114ac43ee1ec366681b5358c17
4	Windows	RHADAMANTHYS	RAT + info stealer	EXE	e09f0bd79308d5a35381b2921ca5f0f609225120157de3d093554eb4b611839e
5	Windows	AgentTesla	Infostealer	VBS	9625958ccd5b7aa03607ca23df3d239ee90a64021a998ab4d8fe71f664155e6a
6	Windows	Remcos	RAT	EXE	08c829e7056b8e022539076acbc962dea072e6506184d4036b785cb0e4592371
7	Windows	Lockbit 2.0	Ransomware	EXE	74437ac6c9f630c52c7e230d57d38c4cbc3affb3bec9215f090a0e3dca8e9d78
8	Linux	PwnRig Miner	Cryptominer	SHELL	4495b55b4f0634def0c91b044b178f5404c8b18effd871d0a06634d15830fe0e
9	Linux	KinSing	Infostealer	ELF	be0fe6a1344b052d4f61cca910f7d26ad02d283f280014eeca0d1cc729e6822a
10	Linux	Lupper	Worm	ELF	2f09fd9d1d76f5fef5b8608326d25847824e6b3eef939727deb985ff2ac0ae07
11	MacOS	Nukesped	RAT	Mach-O	dced1acbbe11db2b9e7ae44a617f3c12d6613a8188f6a1ece0451e4cd4205156
12	MacOS	RealTimeSpy	Spyware	ZIP	11ce12eaa7740581c7ea8cfde45fd84c4255dbb201a5aafb4085bfcc3ea66aab
13	MacOS	Evilquest	Ransomware	PKG	9e8c30955ccb5797efaab676ffdf36fe08ce32d4aab4d18e1a9ed2be43d5db0f
14	Android	Octo	RAT	APK	144cb58badccd0d201ef6b9befef75f7a2860cf6b978b14d9acf9b77a6a3fdfb

Table 5.1: The 14 selected malware executables for Linux, Windows, MacOS and Android were selected to be used in the CTU-SME-11 dataset.

[4] `https://github.com/ytisf/theZoo`
[5] `https://github.com/Da2dalus/The-MALWARE-Repo`
[6] `https://github.com/Vichingo455/MalwareDatabase`
[7] `https://www.sentinelone.com/blog/`
[8] `https://www.malwarebytes.com/blog`

5.2 Attacks

The attacks present in this dataset were not pre-scripted but instead, they were performed by security professionals. The attacks correspond to the actions of an attacker that is already inside the network[9]. During the attack selection, we decided to use common attack frameworks. The reason is that it covers the most common attacks and does not need a special setup in terms of intentionally vulnerable service or security misconfigurations.

The attacks target the Active Directory Domain Controller, as that is one of the most valuable services of the whole network. When compromised, the Domain Controller should provide access to four out of eleven devices in the network.

5.3 Exfiltration

Data exfiltration is the process of stealing sensitive information from an organization by sending it to a system outside the organization. It is often carried out by insiders or attackers who have gained unauthorized access to the network.

When selecting a method for exfiltration, it is crucial to consider the potential risks associated with each option. In this particular case, the primary concern was to avoid using clear-text HTTP requests to ensure the exfiltration method remained covert and therefore more difficult to detect.

It may be tempting to exfiltrate data to a cloud storage service like Google Drive, Dropbox, or OneDrive, to mask the activity of a malicious attacker potentially. Although, there are several reasons why this approach is not used in the dataset. At first, it is hard to find malware that does so, as the cloud services block this communication in a matter of days. Secondly, this exfiltration is not very common, for the reason stated before.

Therefore, after careful consideration, the decision was made to use exfiltration by DNS, as this exfiltration is far more common. This method allows for the activity to be not encrypted. Ultimately, the chosen exfiltration method strikes a balance between remaining covert and maintaining visibility in network traffic, helping to ensure a comprehensive network security dataset.

The PyExfil [29] python library was used to perform the DNS exfiltration. Getting DNS responses to the exfiltration queries is necessary to make the exfiltration work. For that, we set up a DNS server in a Digital Ocean cloud virtual machine that was returning a generic answer to all queries.

The exfiltration was planned into four batches of two hours each. Each batch had a different setup in terms of the size of DNS packets and the delay

[9]These types of *insider attacks* can refer to two things. First, to a security threat or breach that originates from within an organization by individuals with authorized access to its resources. This can include employees, contractors, or other trusted individuals, with a variety of motivations, such as financial gain, revenge, or curiosity [28]. Second, it can be done by an external attacker that gained access to the local network by other means, such as phishing.

between them. This approach once again allows researchers to select the difficulty of detecting this traffic. The size of exfiltrated data ranged from approximately 14KB to 9MB.

Chapter 6

Laboratory infrastructure

A special network mimicking a small-medium sized enterprise was designed and created for network traffic collection and malware infections in the Stratosphere Laboratory at the Faculty of Electrical Engineering, Czech Technical University in Prague [30]. This network was then used for creating the CTU-SME-11 dataset. The devices connected to the laboratory network were selected to be very diverse in terms of operating systems and intended usage. All devices in the network had access to the Internet, three of those devices had a public IP address, and therefore they were also accessible from the Internet. At the same time, we needed to make sure that the malware samples were executed in a safe manner. To make the executions safe, the network was configured to be separated from the production network of the Stratosphere Laboratory.

The network setup contained 11 devices. All machines were connected to a central switch, which had a SPAN port to a capture server, where the traffic capture took place. A diagram describing the infrastructure and assigned IP addresses can be found in figure 6.1 and each device will be described in detail in the following (sub)sections.

The main router of this network was Ubiquity EdgeRouter 12 v1.10.7 with EdgeOSv2.0.9-hotfix.2 operating system. The internal IP address of the router was `192.168.1.1`. This router had three active interfaces - one was connected to the storage server, one to the internet, and the last to the central switch.

Other 21 devices that were not directly part of the experiment were also connected to the network, as the capture laboratory was not designated purely for creating the CTU-SME-11. It is possible to see communication with those devices in the dataset. One device was dedicated to management purposes. The IP addresses of the devices that are not part of the experiment are: `192.168.1.25`, `192.168.1.38`, `192.168.1.5`, `192.168.1.138`, `192.168.1.132`, `192.168.1.139`, `192.168.1.18`, `192.168.1.30`, `192.168.1.204`, `192.168.1.131`, `192.168.1.129`, `192.168.1.130`, `192.168.1.134`, `192.168.1.100`, `192.168.1.104`, `192.168.1.110`, `192.168.1.111`, `192.168.1.112`, `192.168.1.113`, `192.168.1.114`, `192.168.1.115`, and `192.168.1.116`.

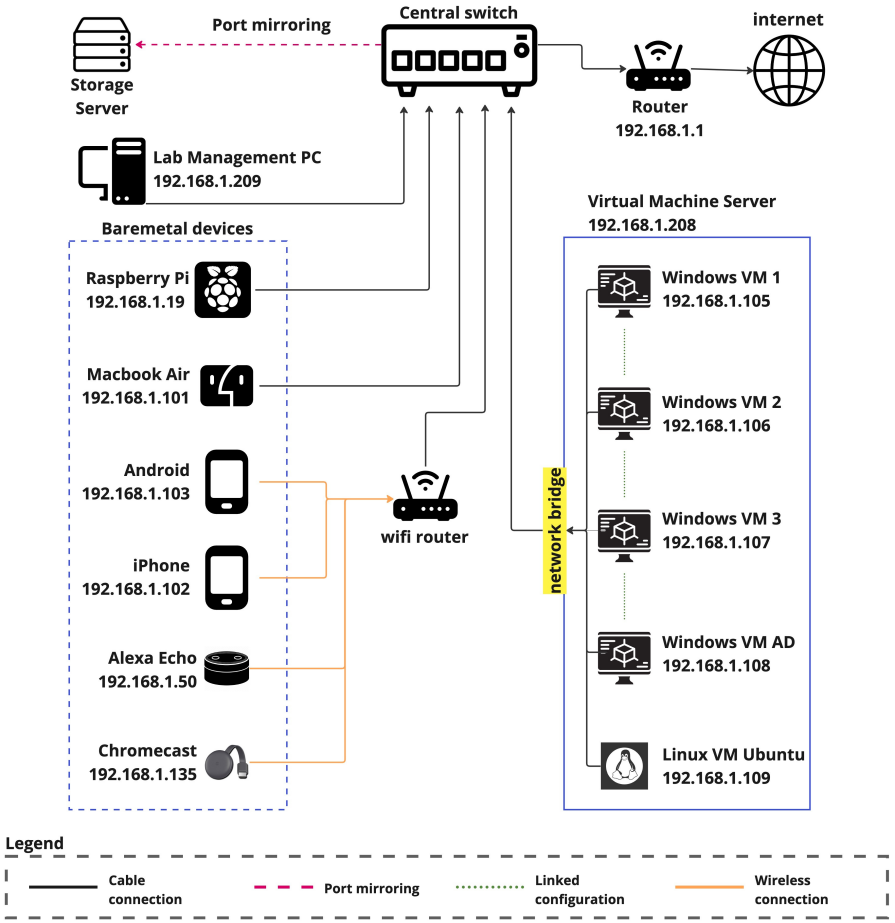


Figure 6.1: Network infrastructure diagram of the CTU-SME-11 dataset. Desktop machines, both virtual and bare-metal were connected directly to the central switch. Smart devices were connected to a WiFi router, which was then connected to the central switch. All devices are on the same IP range and can connect to each other.

6.1 Virtual machines

As nowadays virtual machines (VMs) are very common in all kinds of networks, we decided to make virtual 5 out of the 11 devices. The network contained three `Windows 7 Enterprise` virtual machines (later referred to as "Windows client machines"), one
`Windows Server 2012 R2`, and one `Ubuntu 22.04`. All of those machines had a 64-bit architecture.

For virtualization, we used VirtualBox software. The VMs were using a bridged interface. In this mode, the virtual machine application is 'bridged' into the physical network adapter of the host machine, and it receives its own IP address from the network's DHCP server. This enables the virtual machine to communicate with other devices on the physical network and access external resources just like any other physical machine on the network. Unfortunately, the network communication between virtual machines with bridged interfaces does not leave the host machine into the physical network. All virtual machines were run on the same host machine. IPv6 was disabled on all virtual machines to force the traffic to use IPv4, the only IP protocol version captured.

6.1.1 Windows virtual machines

All Windows virtual machines were assigned 6 GB of RAM and 2 CPU cores. In the case of Windows client machines, the disk capacity was 20 GB, whereas the Windows Server had 40 GB of disk space.

After installing Active Directory Domain Services (`ADDS`) and the Domain Name System (`DNS`), the Windows Server machine was promoted to a Domain Controller, allowing the Windows virtual clients to join its domain. The default gateway of the Domain Controller was pointed to the main router that takes care of the whole network. The only configured DNS server address for the applications in the Windows Server was the loopback address (`127.0.0.1`). The Windows Server license was activated as a part of a free one-year trial.

The three Windows virtual clients were connected to the Windows domain of the domain controller. Only one user was used for this purpose, and the user and credentials were shared with all Windows virtual clients. This decision was taken looking to increase the malware attack surface as much as possible. This user was granted "Domain Admin" privileges, which is against the best practices for securing active directory [31].

The only DNS server on the Windows virtual clients was set up to be the Windows Server Domain Controller. The same goes for the default gateway. To be able to join the domain, the clients needed an active NetBIOS TCP service and a Windows Internet Naming Service (`WINS`). Opposite to the Windows server, the Windows virtual clients did not have the Windows license active. To our best knowledge, this did not affect any behavior of the machines.

6.1.2 Ubuntu virtual machine

To make the dataset more diverse in terms of operating systems, we incorporated a Linux virtual machine. As the Linux distribution, we choose Ubuntu, since it is one of the most widely used distributions today [32]. More specifically, the version of Ubuntu OS used was 22.04. On this virtual machine, we prepared a few services beforehand. A MySQL database and an Apache web server were installed, but as we did not plan to use malware that would take advantage of vulnerabilities in this software, we did not intentionally misconfigure them. MySQL was installed with *mysql secure installation* method. Apart from this, an NTP service was installed on this machine.

6.1.3 Bridged network issues

As already mentioned, the virtual machines were set up to use the bridged network connection. This means that the virtual machines are connected directly to the physical network adapter and can have an IP address in the local network given by the DHCP. This configuration enables capturing network traffic in the storage server with the possibility of differentiating the traffic from each virtual machine.

Unfortunately, this configuration introduced an issue that we did not anticipate when designing the laboratory. The traffic *between* the virtual machines that are bridged is *not* sent to the storage server but is handled inside the host machine instead. Therefore the traffic between, for example, a Windows client and the Windows server was not captured in the storage server because it never went through it. This resulted in a loss of visibility in the main PCAP files created in the storage servers losing the traffic of the inner network of VirtualBox. Fortunately, there is a solution. In VirtualBox, it is possible to set a trace file. VirtualBox then captures the packets sent in between the machines in this file, so it is not completely lost. This gave us two PCAP files, which needed to be merged after the end of the creation phase.

6.2 Bare-metal devices

The designed laboratory was meant to contain virtual machines as well as bare-metal devices. We wanted to ensure we would stick with the design and incorporate six of these devices. That should once again increase the realness of the network and increase the variety of captured traffic. The devices are described in the following subsections.

6.2.1 Mobile devices

The network included two mobile devices with two different operating systems. Those systems were Android and iOS, the two most popular operating systems of the last decade [33]. As an Android representative, we connected a Nokia

TA1043 phone. As an iOS representative, an iPhone XS was used. Both devices have some common applications installed to generate some amount of network traffic without the need to manipulate them. Those applications were installed from Google Play for Android and App Store for iOS. Those apps included mostly social networks, applications for streaming music and videos, and games. Both of these devices were connected to the network via a Wi-Fi router that is also part of the experiments. On the Android device, the randomization of MAC addresses was disabled, as we manually assigned IP addresses in the network based on the MAC address.

Apart from the benign applications, the iPhone XS was also used to control most of the IoT devices in the laboratory. The controlled IoT devices included the Chromecast and Alexa that are included in our dataset, but also other devices that are not included in our dataset. We believed that this would generate a representative amount of benign network traffic.

6.2.2 MacBook Air laptop

As macOS is not easy to virtualize, we needed to devise a solution for getting traffic from this OS into the dataset. Fortunately, an old MacBook Air was available in the Stratosphere Laboratory, so we took advantage of it and used it. The full specification of the MacBook Air was *MacBook Air 13-inch, Early 2015*. Before the experiment, we took a Time Machine backup so that we would not need to reinstall from scratch with each cleanup. This mainly reduced the device's downtime when resetting it to a clean state. To prevent the device from going to sleep we used a tool called Caffeinate [1]. This device was connected via an ethernet cable to the network. Miro and Discord applications were installed on this device before the start of the experiments.

6.2.3 IoT devices

Regarding the IoT devices, we chose one smart assistant, Amazon Echo Dot 2nd Generation, a smart TV Chromecast, and one Raspberry Pi 3 Model B+ 2017 minicomputer.

Both smart assistants were connected to the network via the Wi-Fi router. The Chromecast device was also connected to an external monitor. No previous setup was done on these devices.

The Raspberry Pi was connected to the network via an Ethernet cable. This device was meant to be opened for attacks from the internet. At the start of the experiment, a strong password was assigned to this device. This password was interchanged for a weak one on the second day at 09:04 UTC. For resetting this device, we created three SD cards with similar configurations so it was easy to go back to a clean state. Raspberry Pi OS LITE 64-Bit was installed on this device.

[1] `https://github.com/domzilla/Caffeine`

6.3 Traffic capture

Capturing network traffic is the most important step in creating a network traffic dataset. In this work, we decided to capture the traffic in files with the PCAP format, as it is currently the best packet capture format [34]. To make and store the capture we used a tool called tcpdump [35], a command-line utility that lets you capture packets transmitted over a network interface and store them in a file for later analysis. It is a powerful tool that can capture a wide range of network traffic, including TCP, UDP, ICMP, and many other protocols.

When capturing network traffic, it is important to specify the type of traffic that you want to capture. In our case, we instructed tcpdump to capture IPv4 traffic and the protocols above it. This was a limitation of the infrastructure and devices. Unfortunately, this configuration means we did not capture ARP and IPv6 protocols.

The captures were done in a separate file per device per day. Each capture started at 00:00 UTC and ended at 23:59.59.999999 UTC, when the capture of the next day was automatically started.

The traffic was captured in a storage device assigned explicitly for capturing. This device received a copy of all the packets in the central switch using a port mirroring configuration. The storage device does not have an IP address assigned in the network, so other devices can not communicate with it, and therefore, it does not contaminate the network traffic.

Chapter 7

Experiments to create the dataset

This chapter provides a comprehensive overview of the actions taken during the experiment week, organized by device. Within each section, it details the actual actions, progress, and outcomes of the experiment. The current actions and outcomes are slightly different from the Execution schedule plan in section 4.3, which outlines the intended course of the experiment. The experiment ran for seven days in February 2023. The start of the experiment was on 20/02/2023. The end of the experiment was on 26/02/2023.

A visual representation of the dataset can be seen in Figure 7.1.

7.1 Setup

Before the start of the creation of the dataset, it was necessary to prepare multiple things. A few weeks before the experiment started, it was necessary to create many accounts in multiple online applications and services. Different accounts were created for Google, Facebook, Microsoft, Github, iCloud, Spotify, and Discord. Other accounts were created using logging in through one of the identity providers, such as Google or Microsoft. These accounts were, in general, shared between the devices if necessary.

Regarding the Active Directory, two users were created. One user with administrator privileges and one without them. The administrator was used for logging in to the Windows server machine, while the non-administrator user was used for logging in to the Windows client machines.

On mobile devices, multiple applications from AppStore and Google Play were downloaded and initialized so that they would generate benign traffic.

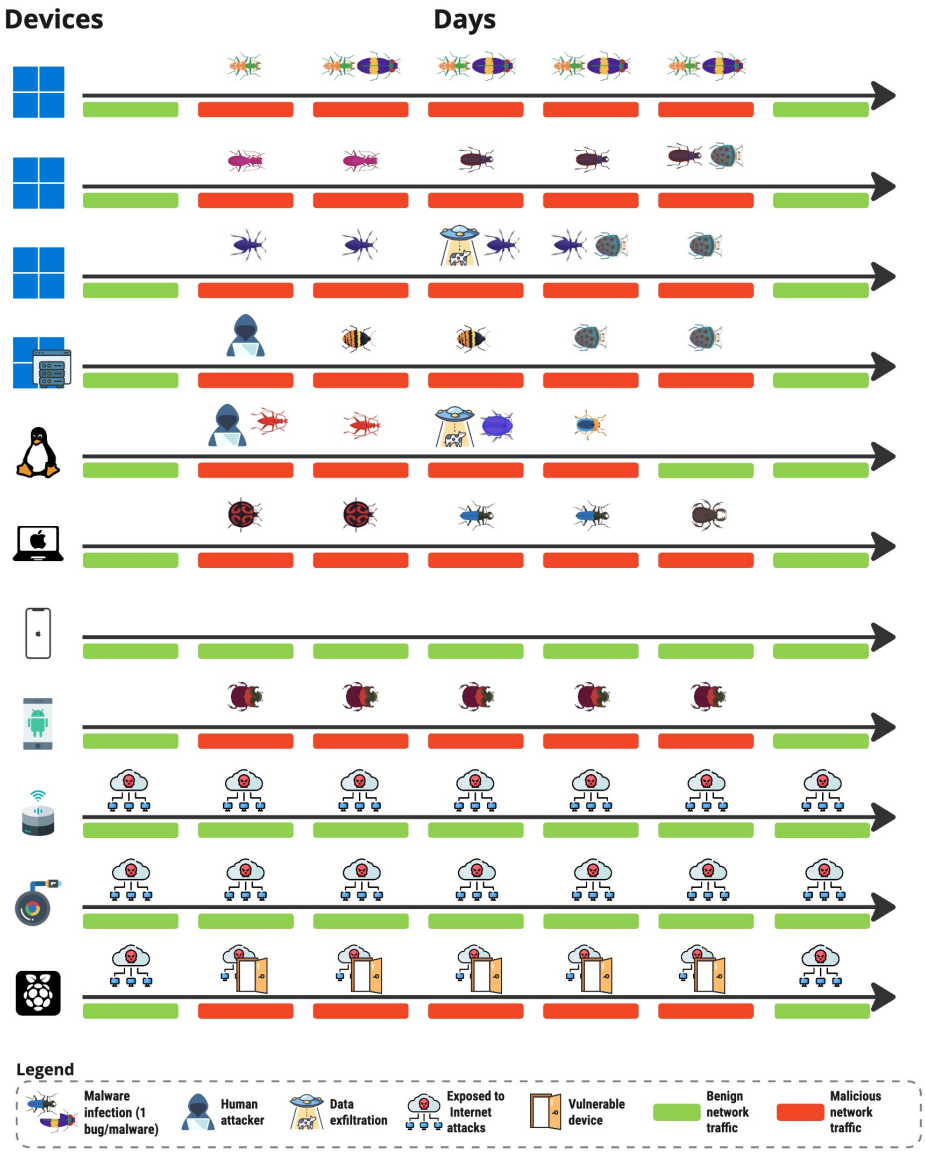


Figure 7.1: Diagram of what type of traffic was executed for each device each day in the CTU-SME-11 dataset The devices are from top to bottom three Windows clients, a Window server, a Linux Ubuntu, a macOS, an IoS, an Android, an Amazon Alexa, a Chromecast, and a Raspberry Pi. The colors and icons every day are explained in the legend.

7.2 Windows Client VM 1

Day 1 of the CTU-SME-11 capture week was dedicated, in every device, to capture human-generated benign traffic. The Windows Client VM 1 was operated by a user over the whole workday. The guidelines for the user operating the device included browsing social networks, watching videos, downloading images, seeing the news, sending emails, trying internet speed, joining video calls, and reading research papers. During the day, there was also one reboot of the device to increase the virtual machine's resources. At 16:30 CET, the simulation of a workday ended. Tabs were left open in the device.

Day 2 Started at 7:36 CET with human-generated benign traffic. After a few minutes of generation, the device was infected with the first malware sample called NanoCore adware (f26c9aff73c041c5a506b8e8ede7bc50fe468d34b0cbf750940bb195eb068b0b). The malware was executed at 7:42:35 CET. During the rest of the day, benign traffic occasionally was generated by a human operator.

On **Day 3**, it was found that VirtualBox capture was not enabled for this device (but the central capture in the storage device was done correctly). Therefore the device was turned off at 9:14 CET to enable the PCAP capture in Virtualbox. The NanoCore adware, which was already infected on this device before rebooting, started again without the need for manual execution. During the day, benign traffic was generated by a human operator. The benign activities mostly include the activities described on day 1. In the afternoon, Windows Client VM 1 was infected with another malware sample, making it infected with two simultaneous samples. The RHADAMANTHYS info stealer (e09f0bd79308d5a35381b2921ca5f0f609225120157de3d093554eb4b611839e) was executed at 16:06:19 CET.

On **Day 4** and **Day 5**, we continued with the occasional human-generated benign traffic. Furthermore, the malicious traffic provided by the two running malware samples continued as well.

In general, **day 6** was the day when all devices were taken back to their clean state. On day 6, the Windows Client VM 1 device was reset to a benign snapshot (which was captured on day two at 7:33 CET). The rollback was done at 13:42 CET. After this action, the device was left without any interaction, so all its traffic was benign.

During **Day 7**, the whole traffic of the device was benign and without human interaction.

7.3 Windows Client VM 2

During **day 1**, the Windows Client VM 1 was operated by a user over the whole workday. The guidelines for the user operating the device included browsing social networks, watching videos, downloading images, seeing the

news, sending emails, trying internet speed, joining video calls, and reading research papers. During the day, there was also one reboot of the device to increase the virtual machine's resources. At 16:30 CET, the simulation of a workday ended. Tabs were left open in the device.

On **Day 2**, the human-generated benign traffic started at 7:47 CET. After a few minutes of traffic generation, the device got infected with its first malware sample, called Redlinestealer (3c7883524728e43e640e1b53d4ce65d582cf7b7c42ca2baf3371ac752ddc216d). The malware was executed at 7:50:00 CET. During the rest of the day, together with the malware, some benign traffic was occasionally generated by a human user.

On **Day 3**, it was found that the VirtualBox PCAP capture was not enabled for this device. Therefore the device was turned off at 9:14 CET to enable the capture. After starting the VM again, the Redlinestealer malware was still running without the need for manual execution. During the day, benign traffic was generated by a human operator. Redlinestealer was still running on this device.

On **Day 4**, the Windows Client VM 2 was rolled back to a benign snapshot at 9:37:40 CET. After the device booted up again, a user operator generated a bit of benign traffic. After this, the device was infected with the malware AgentTesla (9625958ccd5b7aa03607ca23df3d239ee90a64021a998ab4d8fe71f664155e6a) at 9:53:30 CET. For the rest of the day, together with the malware, a human user was browsing the internet occasionally.

On **day 5** at 8:50:00 CET, the device was infected with the Lockbit 2.0 ransomware (74437ac6c9f630c52c7e230d57d38c4cbc3affb3bec9215f090a0e3dca8e9d78). The malware encrypted files in this device, and after, it also accessed remote files in the AD Windows Server using the SMB protocol and also encrypted those. The ransomware did not manage to spread using shared domain credentials, but it was able to encrypt files that were shared across multiple devices in the domain. To confirm this hypothesis, during the day, some extra folders were shared using SMB in the Windows AD server, and the remote encryption was confirmed. The Windows Client VM1 and Windows Client VM3 were left with no extra folders shared.

On **day 6** at 14:28 CET, the Windows Client VM2 was rolled back to a benign snapshot taken on day two at 7:33 CET. After this action, the device was left without any human interaction.

On **day 7**, there was no user interaction with the Windows Client VM 2 device, and only the default Windows traffic and open applications traffic was generated.

7.4 Windows Client VM 3

During **day 1**, the Windows Client VM 3 was operated by a user over the whole workday. The guidelines for the user operating the device included

browsing social networks, watching videos, downloading images, seeing the news, sending emails, trying internet speed, joining video calls, and reading research papers. During the day, there was also one reboot of the device to increase the virtual machine's resources. At 16:30 CET, the simulation of a workday ended. Tabs were left open in the device.

On **day 2**, the device was occasionally operated by a human user. At 12:41:19 CET, the device was infected with the malware called RemcosRAT (08c829e7056b8e022539076acbc962dea072e6506184d4036b785cb0e4592371). During the rest of the day, the malware was left running.

On **day 3**, there was no notable user action performed. Most of the traffic was generated by the malware and by the operating system and other computers connecting to it.

Day 4 on Windows Client VM 3 was dedicated to data exfiltration attacks. The exfiltration was done using the PyExfil python library [29]. This library can do DNS exfiltration. DNS exfiltration was chosen for this scenario because it is one of the most common and easy-to-do exfiltration attack. The exfiltration was performed to a server under our control in the Digital Ocean cloud, where a real DNS server was set up. The exfiltration was performed in 4 degrees of intensity, where each degree lasted 2 hours:

- 14:40:10 CET Exfiltrate over DNS with delay between packets = 0.1s and payload size = 10B
- 16:59:00 CET Exfiltrate over DNS with delay between packets = 0.1s and payload size = 128B
- 19:10:00 CET Exfiltrate over DNS with delay between packets = 5s and payload size = 10B
- 21:25:00 CET Exfiltrate over DNS with delay between packets = 5s and payload size = 128B

For DNS exfiltration, 10 bytes is considered small, and 128 bytes is considered quite high. The maximum length of a domain name is 256 characters.

On **Day 5**, Lockbit 2.0 ransomware (74437ac6c9f630c52c7e230d57d38c4cbc3affb3bec9215f090a0e3dca8e9d78) was executed on Windows VM2, which could have taken advantage of the shared domain account. To our best knowledge, this ransomware did not use this vulnerability. Therefore we believe the Windows VM3 device remained benign until the rest of the experiment.

On **day 6** similarly to Windows Client VM2 and Windows Client VM3, the device was rolled back at 14:28 CET to a benign snapshot (taken on day two at 7:33 CET).

There was no interaction with the device on **day 7**.

7.5 Windows Server VM AD

Day 1 matches the activities of day 1 of the Windows VM 1. Additionally, there was some minor setup done in the server management console on this

device.

On **day 2** the Windows Server VM AD was attacked from the Ubuntu server. The attack included trying to log into the AD and resources enumeration of shared folders via the Metasploit attack framework.

On **day 3**, the Windows VM AD was infected with Trickbot malware (1e90b6fc99a908420de123418deded8d8eadf2114ac43ee1ec366681b5358c17) at 16:15:51 CET. Before this malware execution, the device was occasionally operated by a human user performing routine actions.

Day 4 brought no notable action on Windows Server VM AD. The Trickbot malware, which infected this device on day 3, was still running untouched, and the device was used for benign purposes by a human operator in the background.

On **Day 5**, the virtual machine was rolled back to a snapshot taken after day 1. After it was cleaned, it got infected with the ransomware called Lockbit 2.0 which originated from Windows VM2. The malware did not manage to get full access to the VM. Therefore, only folders shared with the Windows Client VM 2 were encrypted.

The Windows Server VM AD was rolled back to a benign state once more on **day 6** at 14:28 CET. After this rollback, no further interaction was done with the machine by a human operator.

There was no interaction with the device on **day 7**.

7.6 Ubuntu VM

On **day 1**, the Ubuntu VM was operated by a human user during the whole workday. The activities on this device included the installation of packages (net-tools, vim, htop, vscode), browsing the internet, trying the internet speed, downloading and reading research papers, installation of Miniconda and basic python packages, installation of Docker and pulling a small Docker image, and watching YouTube videos.

From 10:45 CET to 10:49 CET the Ubuntu VM was rebooted to increase the assigned resources so it would be easier to operate.

On **day 2**, the Ubuntu VM was infected with PWNrig crypto-miner (4495b55b4f0634def0c91b044b178f5404c8b18effd871d0a06634d15830fe0e) at 7:58:44 CET. However, as the malware took most of the VMs resources and made the machine unusable, the device was reset back at 10:35 CET on day 2 to a benign snapshot taken after on day 2 at 7:33 CET. For the rest of the day, the device was used by human operator to attack the Ubuntu VM. The attack started at 14:06:45 CET and ended at 15:25:20 CET.

On **day 3**, at 16:26:03 CET the PWNrig crypto-miner was run again, but now using the *nice* command with a priority of 20 (`sudo nice -n 20 pwnrig.elf`). Additionally, benign traffic continued as before, with human users accessing the VM and using it for minor operations.

On **day 4**, 5he Ubuntu VM was reset back to a benign snapshot at 9:07:39 CET. After it was cleaned up, new malware was run on this device. At 9:19:34 CET, the device was infected with Kinsing malware

(be0fe6a1344b052d4f61cca910f7d26ad02d283f280014eeca0d1cc729e6822a). This malware was scanning a very large number of IP addresses. Therefore, it was terminated on the same day at 16:42 CET by resetting the VM back to a benign snapshot.

The original plan for **day 5** was to infect the Ubuntu VM with Linux ransomware. Unfortunately, we were unable to find Linux ransomware that would send network traffic from the device. Therefore, we decided to infect the device with the Lupper worm (2f09fd9d1d76f5fef5b8608326d25847824e6b3eef939727deb985ff2ac0ae07) at 18:46:24 CET. As this malware was scanning IP addresses on the internet, we stopped the infection at 21:30 CET on the same day.

Day 6 was a cleaning day for all the machines, including Ubuntu VM. It was rolled back to a benign snapshot at 13:42 CET. After this rollback, no further interaction was done with the machine from user operators.

On **day 7** there was no further interaction from user operators with the device, however, all the benign and automatic traffic from the OS remained.

7.7 MacOS laptop

Day 1 was designated to generate human-generated benign traffic. On the MacOS device, this included browsing social networks, watching videos, downloading images or PDFs, seeing the news, sending emails using online web email services, uploading files to Dropbox and Google Drive, checking internet speed, reading and downloading research papers, and backing up data to iCloud.

On **day 2**, the human-generated benign traffic started at 9:51 CET. The device was then infected with Nukesped RAT (dced1acbbe11db2b9e7ae44a617f3c12d6613a8188f6a1ece0451e4cd4205156) at 10:05:13 CET. For the rest of the day, the device was used for minor benign activities by several human users. This malware sample was attributed to a North Korean cyber actor, probably named Lazarus Group or APT38 [1].

Day 3 did not bring any new malware infection. The Nukepsed malware was still running on the MacOS laptop, and the device was occasionally used by a user to browse the internet. It is worth mentioning that on this day, the Brew package manager and XCode developer tools were installed on the device.

On **day 4**, the device was reset back to the benign state on using Time machine backup. The process started at 8:23:22 CET and was finished at approximately 8:41 CET. After a few minutes of human-generated benign traffic, the device was infected with a new malware sample. RealTimeSpy spyware (11ce12eaa7740581c7ea8cfde45fd84c4255dbb201a5aafb4085bfcc3ea66aab) was executed at 8:58:35 CET.

For **day 5**, a ransomware was planned. The laptop was once again reset to

[1] `https://objective-see.org/blog/blog_0x6E.html`

a benign state at 15:05 CET. After the reset finished, Evilquest ransomware (9e8c30955ccb5797efaab676ffdf36fe08ce32d4aab4d18e1a9ed2be43d5db0f) was executed on this device at 15:57:15 CET.

The device was reset back to a benign state using a Time machine backup on **day 6**. The process started at 13:30 CET. After it finished, no further actions were done with this machine.

There was no user interaction with the device on **day 7**.

7.8 iOS phone

From day 1 to day 5, the iOS device was used for benign activities by a user. The activities included IoT devices control, browsing social media, installing applications from App Store, playing music and videos, and playing games. There was no malicious traffic on this device.

There was no interaction with the device on **day 6** and **day 7**.

7.9 Android phone

From day 1 to day 5, the Android device was used for benign activities by a user for browsing social media, installing applications from App Store, playing music and videos, and playing games. Note that the Android phone was already powered on and working when the first day was started, and therefore the traffic shows already benign ongoing connections from this device.

On **day 2**, the device was infected with the Octo RAT malware (144cb58badccd0d201ef6b9befef75f7a2860cf6b978b14d9acf9b77a6a3fdfb) at 10:11:01 CET.

During the rest of day 3, and whole day 4 and day 5 the Android device only had the Octo RAT malware running and the default activities of the OS. The infection was stopped by factory resetting the device on **day 6** at 13:14 CET. After factory resetting the device, the randomization of the MAC address needs to be turned off again. This was done at 13:39 CET.

There was no user interaction with the device on **Day 7**.

7.10 Raspberry Pi

On **day 1**, the first interaction with the Raspberry Pi device was our own administrative connection at 16:15 to manage it using SSH from the IP address 192.168.1.209. At 16.17 we installed the Kodi media center [2] on the Raspberry Pi. Apart from these actions, no interaction with this device was made on day 1. This device was exposed to the internet by having its own public IP address in the main router. This IP address received attacks from the internet during the whole of day 1. However, since the SSH password of the admin user was very strong on day 1 no attack was successful to log

[2] `https://kodi.tv/`

in. We verified this using the capability of Zeek to identify successful SSH logins and our own research verification. However, all the attempts from the internet were considered and labeled as malicious attacks.

At 10:04 on **day 2**, the credentials on the Raspberry for the admin user were changed to `admin`, making it vulnerable to attacks from the internet. This setting remained unchanged until day 6. The first successful attack and login to SSH from the internet happened on day 2 at 11:04:14 CET.

During day 3, day 4 and day 5, the device was kept with an easy-to-guess weak password on the admin account and reachable from the internet. Therefore it received many attacks, which were labeled.

On **day 6**, the device was restored to a benign state by swapping the SD card in the device. Therefore the device was made resistant to internet attacks.

There was no user interaction with the device on **Day 7**.

7.11 Alexa Echo

Similarly to the Raspberry Pi and Google Chromecast, the Alexa Echo assistant was assigned its own public IP address in the main router. Therefore, it was visible and attacked from the internet. To our best knowledge, the Alexa assistant has not been exploited.

During **day 1, day 2, day 3, day 4, day 5, and day 6** the Alexa Echo assistant was occasionally used by a user using their voice to deliver the news, play games, stream music, or answer questions of the researchers.

There was no user interaction with the device on **Day 7**.

7.12 Chromecast

Similarly to the Raspberry Pi and Alexa Echo, the Google Chromecast device was assigned a public IP address in the main router. Therefore, it was visible and attacked from the internet. To our best knowledge, the Chromecast device has not been exploited.

During **day 1, day 2, day 3, day 4, day 5 and day 6** the Google Chromecast was used for streaming videos from Youtube on the local monitor.

There was no user interaction with the device on **Day 7**.

Chapter 8

Dataset processing

This chapter describes the processing of the raw data in PCAP format into other formats. Also, it introduces an issue where traffic between virtual machines did not go through the capture server.

8.1 VirtualBox captures

Virtual machines were running on the same host in a Virtualbox application. As we learned later, the inter-VM communication did not leave the host machine. Therefore, the packets going from one VM to another were not captured in the central storage server. Because of that, we started capturing the network traffic within Virtualbox using the VBoxManage *trace file* feature. This allowed us to capture the inter-VM communication as well. The PCAP capture file from the central storage then had to be merged with the Virtualbox capture in order to have all the traffic inside one PCAP file.

The PCAP file in the storage server and the PCAP file in the Virtualbox had two different *views* of what the same device did. So merging them only improves them, since the traffic no traffic is duplicated.

This merge was done only for Windows Client VM 2, Windows Client VM 3, and Windows Server VM AD on days 5 and 6. The reason is those were the only days where inter VM traffic was generated. On these days, ransomware was running, taking advantage of the interconnection of those machines with the Windows Server VM AD controller. The ransomware had no traffic to the internet. Therefore, it was decided to incorporate the Virtualbox PCAP file into the dataset in order to see the malicious behavior. Only packets inside the local network were taken into account (packets to the internet were not taken from the Virtualbox PCAP since those were present in the central storage PCAP file).

Merging the PCAPs included shifting the Virtualbox PCAP into the correct date and time, as the Virtualbox captures started at the start of the UNIX epoch (1.1.1970). Therefore it is possible to have slight (milliseconds to seconds) inconsistencies in the case of the time and date for the virtual machine PCAPs.

8.1.1 Steps to merge VirtualBox PCAPs

To merge the two PCAP files we did the following steps:

1. The captures from VirtualBox were modified to keep only the packets from the local network.

```
tcpdump -s0 -n -r *
   original_PCAP_from_virtualbox*.pcap net
   192.168.0.0/16 or net 10.0.0.0/8 or net
   172.16.0.0/12 -w *local_net_only*.pcap
```

2. Edit the time of PCAP with Editcap, as VirtualBox PCAP starts at the start of the epoch (1.1.1970)

```
editcap -t *seconds_to_add_since_first_packet
   * *local_net_only*.pcap *
   local_and_time_moved*.pcap
```

3. Keep only packets that are within the boundaries of a day that is currently being processed

```
editcap -A *epoch_timestamp_of_day_start* -B
   *epoch_timestamp_of_day_end* *
   local_and_time_moved*.pcap *
   final_virtualbox*.pcap
```

4. Merge processed PCAP with a PCAP from the capture server

```
mergecap -w output.pcap *from_observer*.pcap
   *final_virtualbox*.pcap
```

8.2 Generated files

The dataset is organized in a hierarchical structure, with data split by the device on the highest layer and further divided by day on the lower layer. Specifically, for each device included in the dataset, there is one PCAP file for each day of data collection.

The PCAP files contain raw network traffic data captured from the respective devices over the course of a day. For each of these PCAP files, additional files have been generated using Zeek network analysis framework [24].

The Zeek logs provide metadata and context for the network traffic, such as IP addresses, ports, protocols, and flow information. These logs can offer insights into network communication patterns and behavior.

Additionally, we labeled the Zeek logs files one by one as described in Section 9. The label consists of two parts, a high-level label and a detailed label. The labels can be found in the `*.log.labeled` files inside the `/zeek` directory. The labeling was done with the help of NetflowLabeler tool [25].

Because of that, the configuration file for this tool is also provided in the `/artifacts` folder.

Chapter 9

Labeling

A crucial part of any network security dataset are labels and how they are assigned. Therefore, we pay special attention to the labeling procedure.

For labeling purposes, a tool called "NetflowLabeler" [25] was used. The tool is used to assign the labels but the process of deciding which labels to assign was taken from a work on network traffic labeling methodology ([26]). This methodology was used to decide the labels for this dataset. In summary, the dataset is labeled on a network flow level, including the extended flows that Zeek creates for other protocols, such as in the *http.log* file. For brevity, we refer to all these as *flows*.

Each network flow is assigned two labels, a main label, and a detailed label. The main label can be of three types: *Malicious*, *Unknown*, or *Benign*. *Malicious* was assigned to all the confirmed attacks either from a device or to a device. It is about the intention of the action. *Benign* was assigned to all the flows that were verified coming from the operating systems or were done by the human user controlling the device. *Unknown* was assigned to those flows that were not confirmed to come from benign actions, but they were confirmed not to be attacks. This includes some network traffic between devices, such as the switch.

The detailed label on each flow should then describe more about what is the flow about so researchers have the opportunity to fine-tune their detections. This information can include which of the source and destination IP addresses are malicious, what technique was being used, or what application protocol was used.

The NetflowLabeler tool uses a configuration file to assign labels to the flows. This configuration file holds all the information needed to label each flow. We published the configuration file used on each day on each device together with each labeled group of files.

To make the labeling more precise, we identified beforehand the IP addresses of well-known and confirmed benign organizations and services, such as Google, Facebook, Twitter, etc. All traffic to or from these IP addresses was labeled *Benign* in the main label, and they have a detailed label with the name of the service.

To confirm the correct identification of malicious traffic, we ran the same malware samples before the creation of the dataset and we identified the

IP addresses that the malware is connecting to. This helped us later in the labeling phase, where we just searched for those malicious IPs (while we also tried to find new ones) and assigned a proper label to them. The traffic that is labeled malicious is either an attack from the internet, a network connection of some malware sample, an exfiltration of data scripted by the creators, or a human attack performed by the creators.

9.1 Public IPs

In terms of labeling, there are two kinds of devices in this dataset: devices that have a private IP address but also have a public IP address, and devices that only have a private IP address. While putting the labels, it was critical to know what kind of device we were dealing with.

Anyone on the internet (including potential attackers) can see the devices with a public IP address. Therefore we consider all the connections initiated by a public IP address malicious since the devices can be considered as a type of honeypot (a non-official production system that no one should connect to).

In case the connection is initiated from the device to the internet, and the device is not considered compromised at the time of the connection, we label this connection as `Benign, From_benign-To_benign`. If the organization owner of the public IP is known, the label has in format `Benign, From_benign-To_benign-*Name_of_organization*`. An organization name is, for example, 'Dropbox'.

9.2 Private IPs

The situation of the devices only with private IP addresses is slightly different.

Given a device that we are labeling (called here the studied device), all the connections from it are labeled according to the knowledge if it was attacking or not at that time. If another device in the network connects to the studied device then the label also depends if we know if the other device was attacking or not.

In the case the studied device connects to the internet, we rely on our list of well-known internet services to label them as benign. If case the remote IP address on the Internet does not belong to a well-known service we check if it was generated by the human user. If it was not, then we check if it was generated by any malware running in the studied device. This process is tedious but important to guarantee correct labels.

9.3 Labeling example

This section outlines the labeling procedure employed during the labeling phase by providing an example. The procedure involved three steps:

1. Determining Execution Time: The execution time of the attack or malware was sought by referring to the documentation and the previously executed malware instance. Once obtained, the corresponding timestamp was used to locate the first packet of the malware within the PCAP file.

2. Identifying Malicious IP Address: The first found packet revealed the IP address(es) to which the malware was connecting. To verify this, the IP address was cross-checked with VirusTotal's and Shodan's databases. If a malicious match was found, the IP address was recorded in the labels.conf file. In cases where multiple suspicious IP addresses were detected, each address was subjected to verification through VirusTotal and Shodan before being added to the labels.config file.

3. Labeling conn.log: The conn.log file is the main Zeek file with real network flows. It is labeled using the NetflowLabeler tool. This step involved marking proper flows as malicious. Any recurring patterns, such as connections to similar ports but different IP addresses or unfamiliar IP addresses in the conn.log.labeled file were thoroughly examined. Suspected IP addresses were verified once again. If confirmed as malicious, they were included in the labels.conf file, and the entire conn.log file was relabeled. This iterative process continued until no further suspicious IP addresses were identified.

4. Labeling of the rest of Zeek files: From the labeled conn.log file we transfer the labels to the other corresponding lines in the rest of the Zeek log files. Each of these files, such as *http.log* was produced by a flow in the *conn.log* file. For this, we used the *zeek-files-labeler.py* tool in the NetflowLabeler tool. The transfer of labels is done using the unique flow id provided by Zeek.

By following this general approach, the labeling procedure ensured comprehensive coverage of potentially malicious network connections.

Chapter 10

CTU-SME-11 overview

10.1 Addressing the problems of current datasets

At the start of our research, we stated that it is possible to create a new dataset that would address the current problems of existing datasets. Those issues included the realness of the experiment, recency, (im)balance of the data, length of the experiment, or lack of information about the creation and usage of the dataset.

To enhance the authenticity of the experiment, we conducted it on a real computer network and incorporated both human-generated benign traffic and human-initiated attacks. We believe that this approach significantly increased the realism of the study. At the same time, we are aware that it is very unlikely that a computer network with eleven devices will be infected with 14 malicious programs and attacked by a human attacker from inside the network. Yet we find this a trade-off between having an acceptable amount of content and the realness of the experiment.

To ensure the experiment's recency and relevance, we not only used the most recent network protocols, such as QUICK, DoT, and DoH, but we also made a significant effort to collect recent malware samples to execute during the experiment. This approach helped to ensure that the experiment reflects the current threat landscape.

Regarding the balance of the CTU-SME-11 dataset, we were once again aiming for realness and a sufficient amount of malicious and benign network flows. The traditional approaches in cybersecurity research on threat detection typically require a balanced (50/50) amount of malicious and benign traffic. This is to guarantee that the machine learning models that can not deal with unbalanced datasets can be trained accordingly. However, this requirement is highly unrealistic. In a real scenario, there are two types of situations. In the first realistic situation, the vast majority of flows are benign, with none or very few malicious flows (such as a RAT, crypto-miner, or ransomware). In the second realistic situation, most of the flows are malicious, with very few benign flows. This is the case with port scanners (e.g., Nmap), malware distribution (e.g., Mirai), and DoS attacks. Therefore, requiring a completely balanced dataset or a mostly benign dataset is unrealistic.

Our dataset is prepared to have all three situations and is designed to be

consumed as a whole or in parts. Therefore, researchers can pick up to view the dataset per device or per day for multiple devices.

The whole dataset has 1,644% benign flows, 0,021% unknown flows, and 98,335% of malicious flows. But this unbalance is on purpose because it includes the very large scan done by malware on the Linux Ubuntu device on days 4 and 5. Consumed like this, researchers can train on the first day since it is all benign and then should detect the very large attack on days 4 and 5 in one device. Then they should continue to detect the much lower malicious signals for the rest of the days.

If the fourth day of the Linux Ubuntu client is not considered, then the dataset has 73,916% benign flows, 0,940% unknown flows, and 25,145% malicious flows.

The total length of the experiment is seven days, capturing both working days and one weekend. This length brings a sufficient time frame for the malware to perform all actions that it is capable of, and at the same time, the dataset is not too big to be comfortably downloaded and used.

Lastly, this thesis should serve as a very detailed description of the dataset, including all the possibly needed information.

10.2 Structure of the dataset

The dataset is organized in a hierarchical structure, with data split by device on the highest layer and further divided by day on the lower layer. Specifically, for each device included in the dataset, there is one PCAP file for each day of data collection.

The PCAP file for the day is stored in the *raw* directory.

From these PCAP files, Zeek logs were generated and labeled. The labeled Zeek files are stored in the *zeek* directory. Notice that if a Zeek log does not need labels (e.g., stats.log), then it is not labeled, and its extension remains '.log'.

The configuration file used to label the Zeek logs of the day is provided in the `artifacts/` folder for that day.

For each day, a README.txt (and its interpretation README.html) is stored in the main directory of the day.

The general structure of the dataset is illustrated below for day 2023-02-20 of the Honeypot-Minicomputer-RaspberryPi-Gen3-20 device:

```
|- Honeypot-Minicomputer-RaspberryPi-Gen3-20
|    |- 2023-02-20
|    |   |- artifacts
|    |   |   |- labels.config
|    |   |- raw
|    |   |   |- 2023-02-20-00-00-03-192.168.1.19.pcap
|    |   |- README.html
|    |   |- README.md
|    |   |- zeek
```

```
|    |   |   |- capture_loss.log
|    |   |   |- conn.log.labeled
|    |   |   |- dhcp.log.labeled
|    |   |   |- dns.log.labeled
|    |   |   |- dpd.log.labeled
|    |   |   |- files.log.labeled
|    |   |   |- http.log.labeled
|    |   |   |- loaded_scripts.log
|    |   |   |- notice.log.labeled
|    |   |   |- ntp.log.labeled
|    |   |   |- packet_filter.log
|    |   |   |- radius.log.labeled
|    |   |   |- reporter.log
|    |   |   |- sip.log.labeled
|    |   |   |- ssh.log.labeled
|    |   |   |- ssl.log.labeled
|    |   |   |- stats.log
|    |   |   |- tunnel.log.labeled
|    |   |   |- weird.log.labeled
```

10.3 Dataset in numbers

The CTU-SME-11 dataset was captured for seven days using 11 main devices, making it a total of 77 PCAP captures. In total, 59 of the 77 captures contain malicious packets. The final size of the PCAP files is on average 163 GB. The total duration of all PCAP files in hours is approximately **1681**h (slightly over 70 days). The average duration of a PCAP is approximately 21h50min in one device per day.

In terms of network flows, the dataset contains 77 conn.log.labeled files. All of them have a size of **18 GB**. The total number of records in all those flow conn.log.labeled files is **99,993,509**. Out of this number, 1,608,273 flows are labeled as *benign*, 98,319,252 as *malicious*, and 65,984 as *unknown*.

Figure 10.1 shows the distribution of network flows across all days and all devices. Considering that the number of malicious flows is extremely high, we used a logarithmic scale on the Y-axis.

If we do not consider the attacks on the Linux device on days 4 and 5, the number of network flows is 2,223,630. With 1,608,273 benign, 549,373 malicious, and 65,984 unknown.

10.3.1 Linux VM

The total number of flows for Linux VM is **97,830,092**. The size of the network flow files is approximately 17,42 GB. Figure 10.2 shows the distribution of outgoing and incoming bytes for every day of the Linux VM.

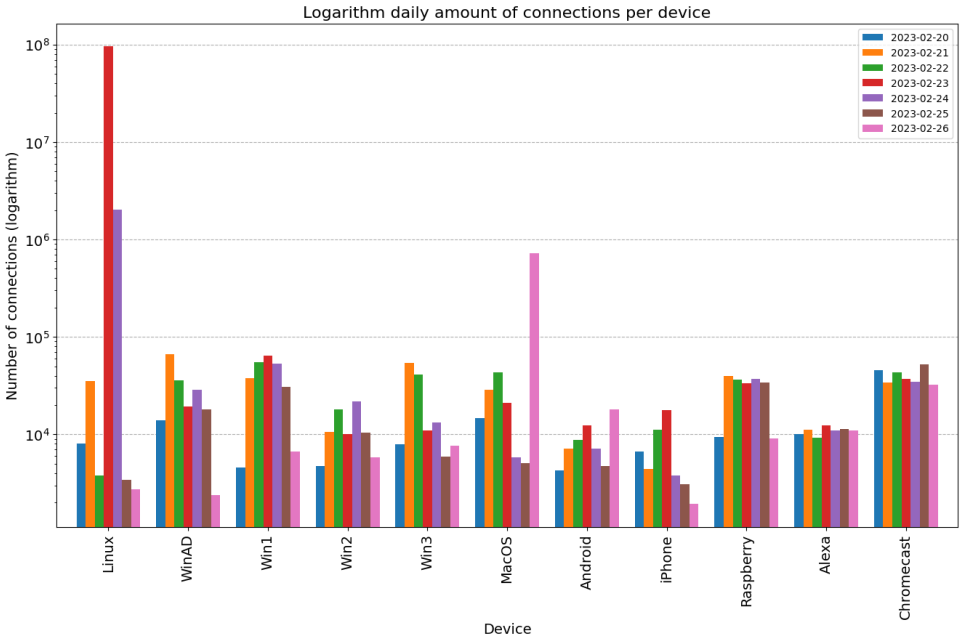


Figure 10.1: Histograms of the number of network flows for all devices on all days. The Y-axis is in logarithmic scale. Notice that the Linux device on days 4 and 5 has a large number of flows due to a scan done by the malware.

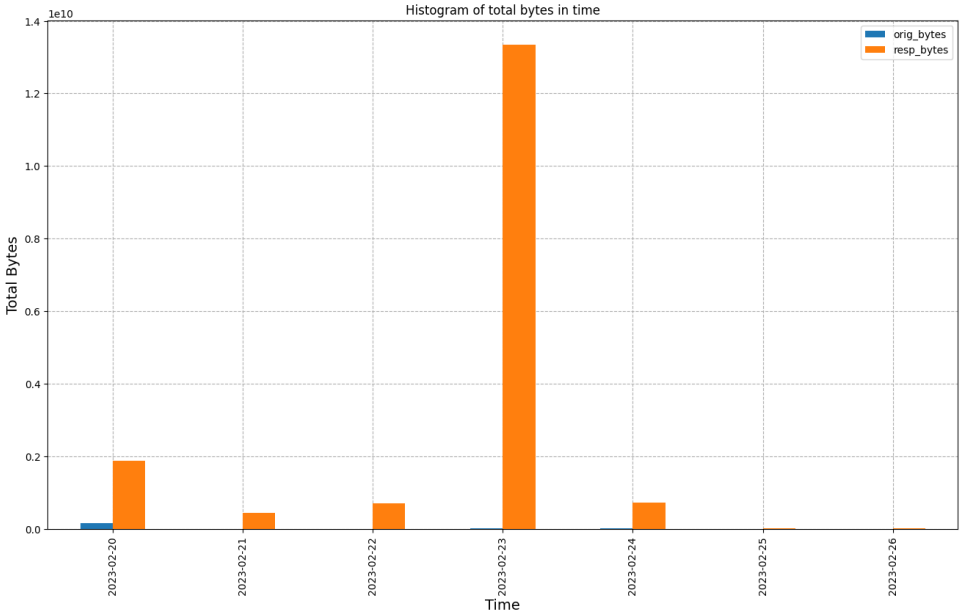


Figure 10.2: Histogram of the sum of incoming (resp_bytes in orange) and outgoing bytes (orig_bytes in blue) across all days for the Linux VM.

Figure 10.3 shows the histogram of the top 10 destination ports contacted by the Linux VM for each day. It can be clearly seen that port 80 was the port used for the scan.

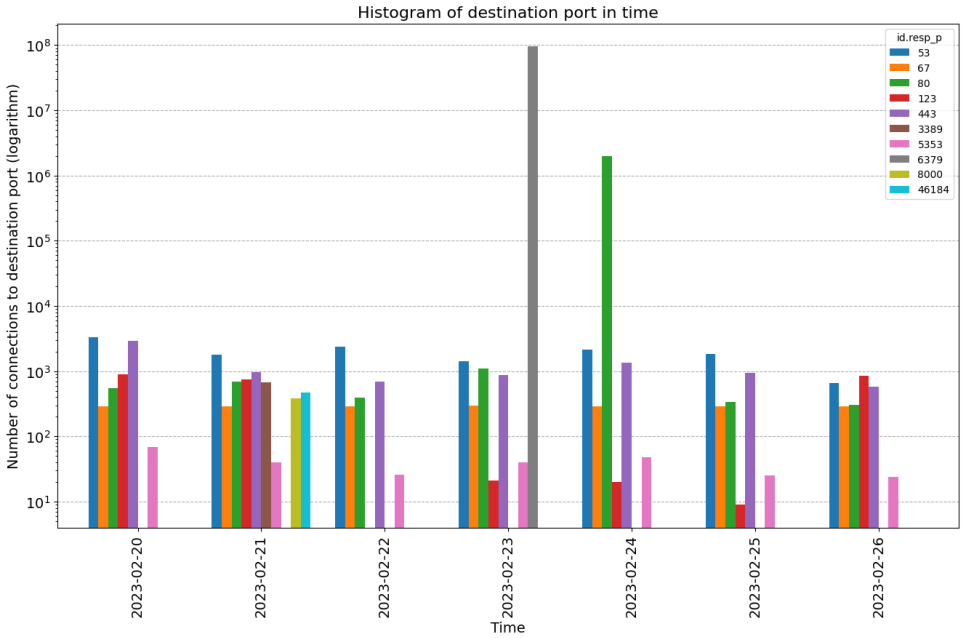


Figure 10.3: Histogram of top 10 destination ports used by the Linux VM for each day of the experiment in logarithmic scale.

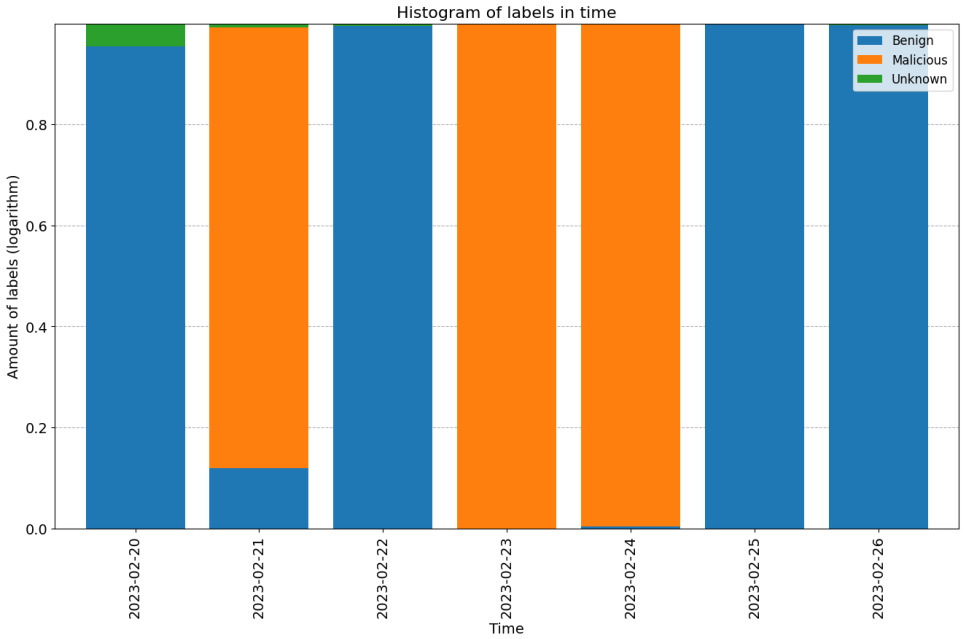


Figure 10.4: Stacked percentage plot of the number of labels for each day for the Linux VM.

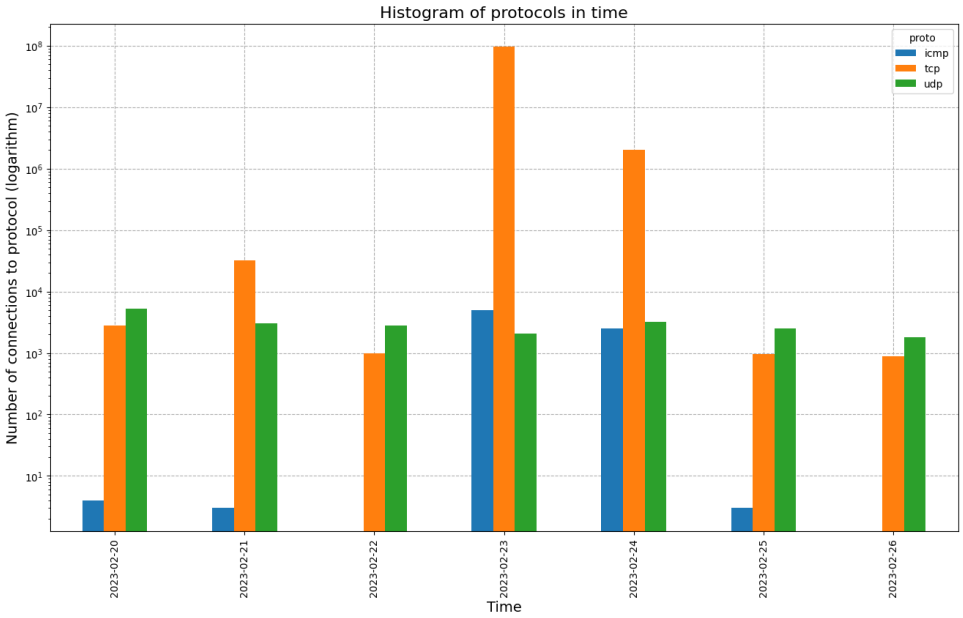


Figure 10.5: Histogram of protocols used by Linux VM across all days in logarithm scale.

10.3.2 Windows VM 1

The total number of flows for Windows VM 1 is **252,308**. The size of the network flow files is approximately 32,44 MB. As figure 10.7 shows, the most frequently used port during the weekdays is port 53, which refers to DNS service. Figure 10.6 shows the decrease in traffic during the weekend.

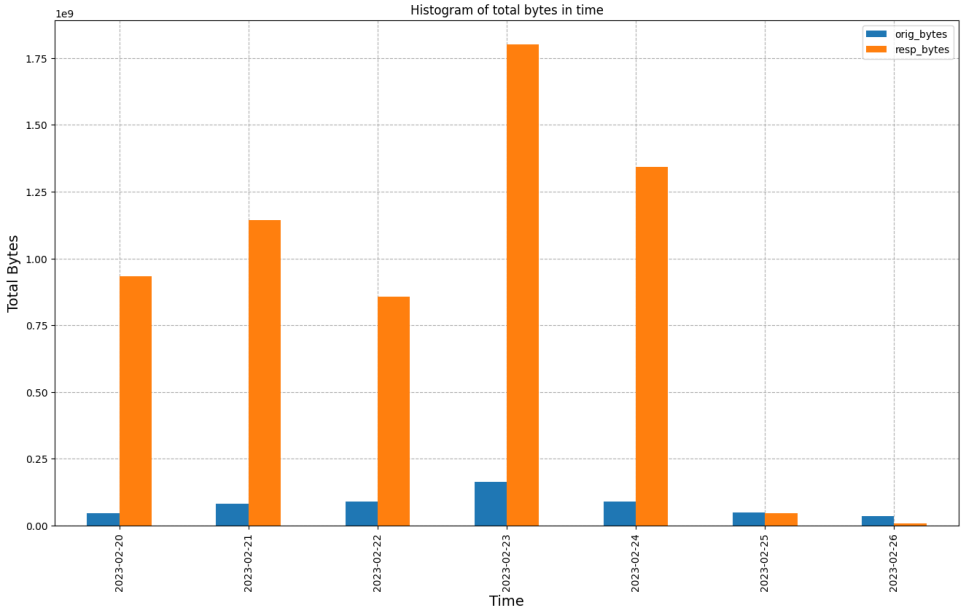


Figure 10.6: Histogram of the sum of incoming and outgoing bytes across all days of the experiment on Windows VM 1.

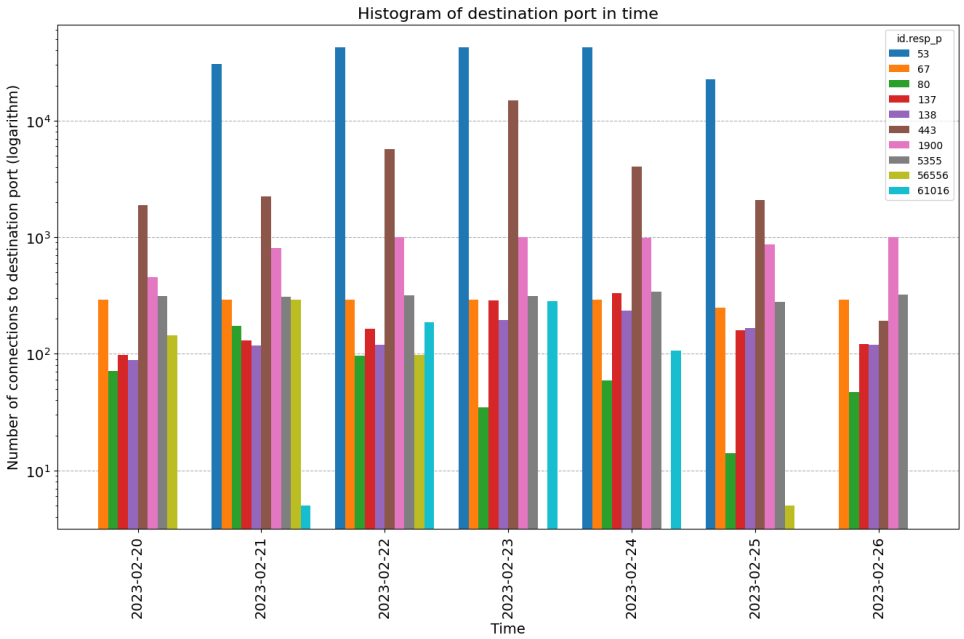


Figure 10.7: Histogram of top 10 destination ports used by the Windows VM 1 for each day of the experiment in logarithmic scale.

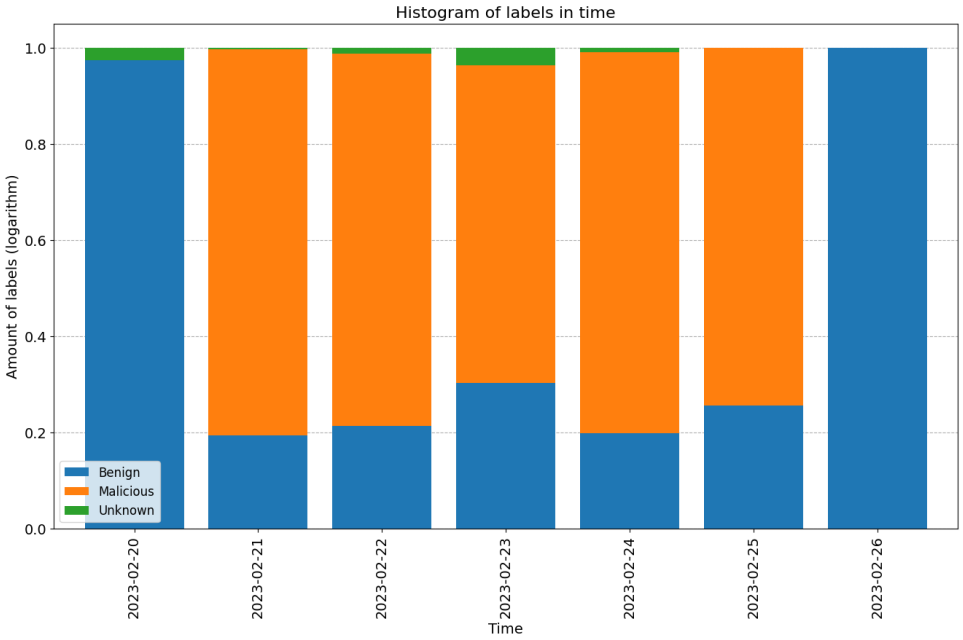


Figure 10.8: Stacked percentage plot of the number of labels for each day for the Windows VM 1

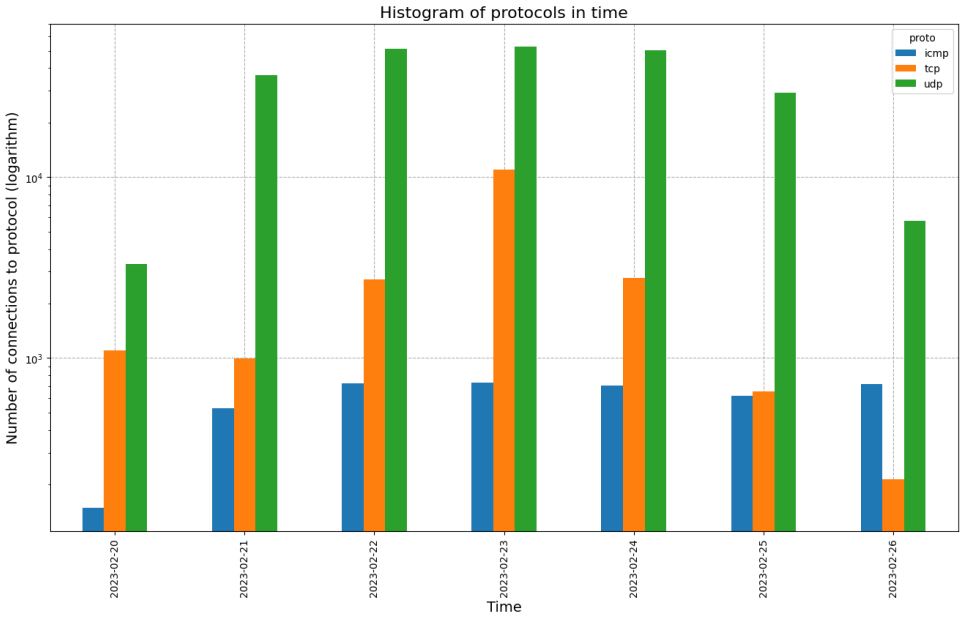


Figure 10.9: Histogram of protocols used by Windows VM 1 across all days in logarithm scale.

10.3.3 Windows VM 2

The total number of flows for Windows VM 2 is **81,678**. The size of the network flow files is approximately 13,2 MB. Figures 10.10 and 10.11 show an

increase in content consumption on day 5 of the experiment, which is caused by leaving a youtube video playing on this machine.

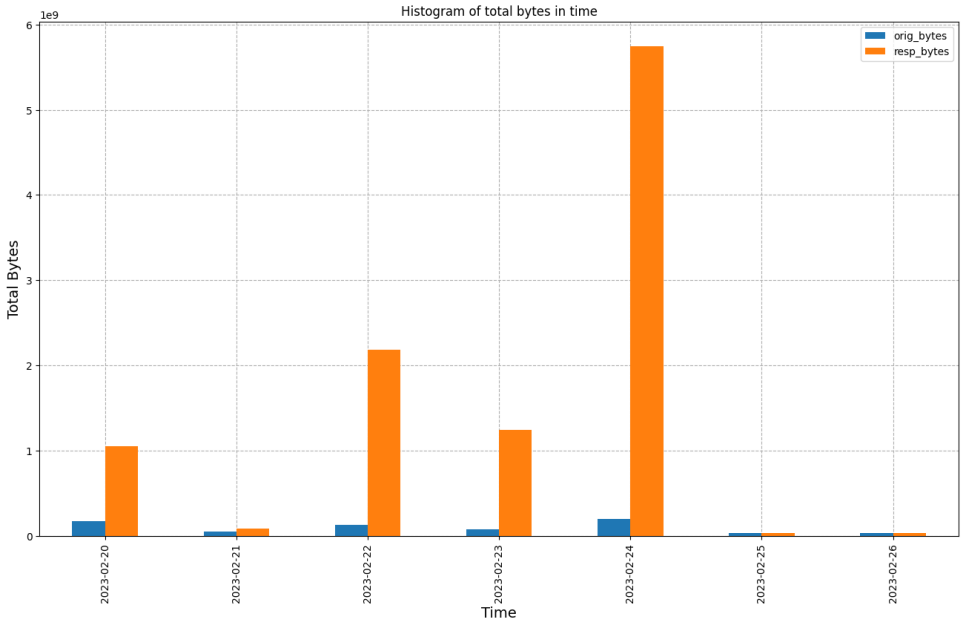


Figure 10.10: Histogram of the sum of incoming and outgoing bytes across all days of the experiment on Windows VM 2.

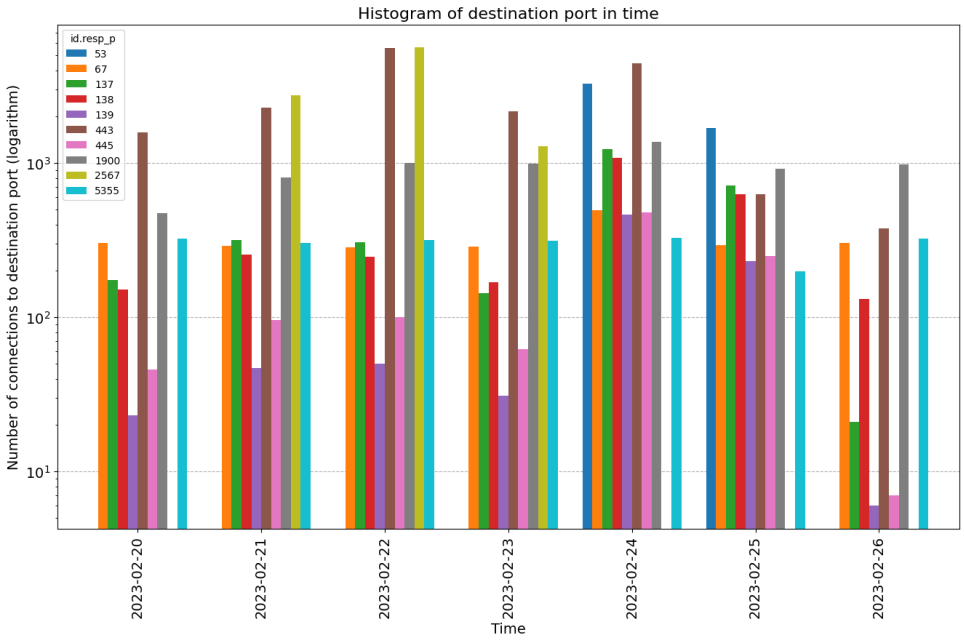


Figure 10.11: Histogram of top 10 destination ports used by the Windows VM 2 for each day of the experiment in logarithmic scale.

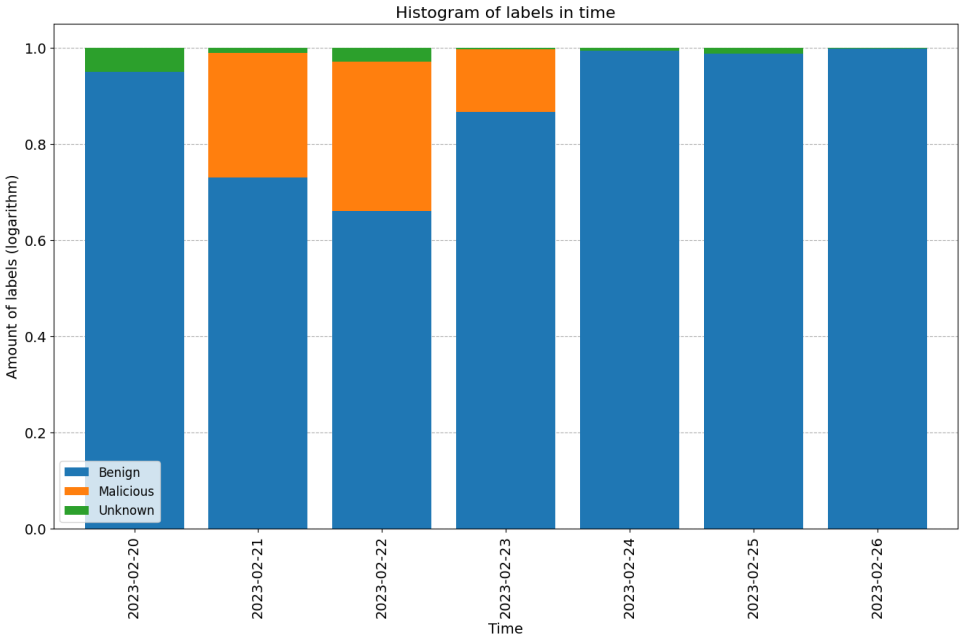


Figure 10.12: Stacked percentage plot of the number of labels for each day for the Windows VM 2.

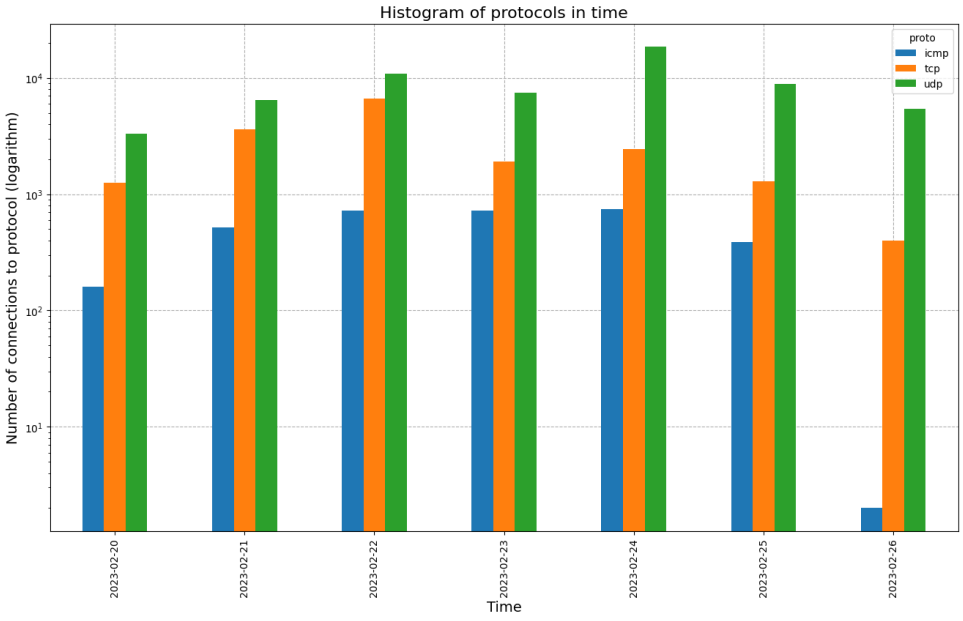


Figure 10.13: Histogram of protocols used by Windows VM 2 across all days in logarithm scale.

10.3.4 Windows VM 3

The total number of flows for Windows VM 3 is **141,072**. The size of the network flow files is approximately 24,88 MB. In days two and three,

figures 10.15, 10.15, 10.17 and show data exfiltration to TCP port 5888 by RemcosRAT malware.

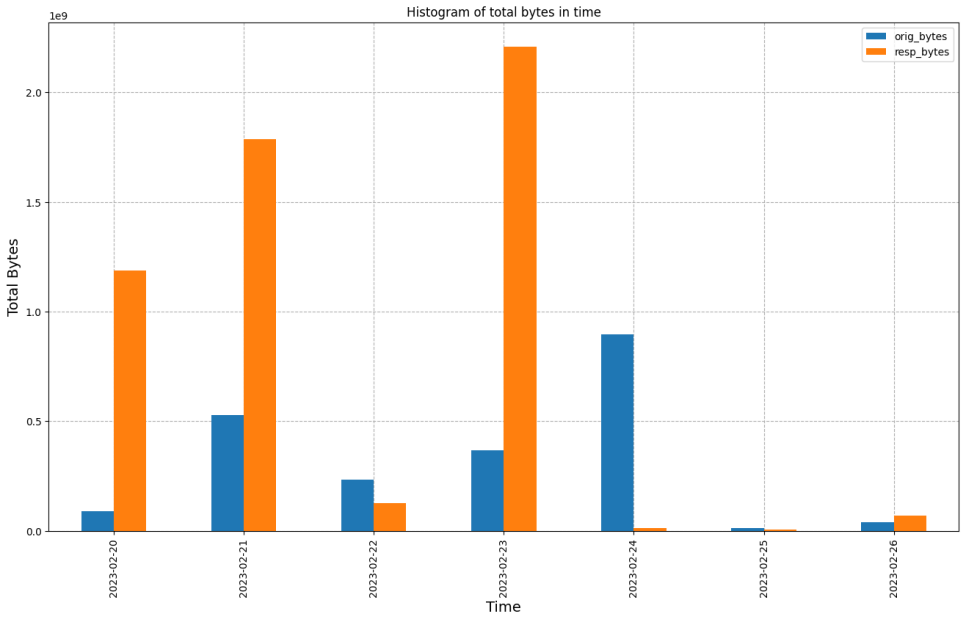


Figure 10.14: Histogram of the sum of incoming and outgoing bytes across all days of the experiment on Windows VM 3.

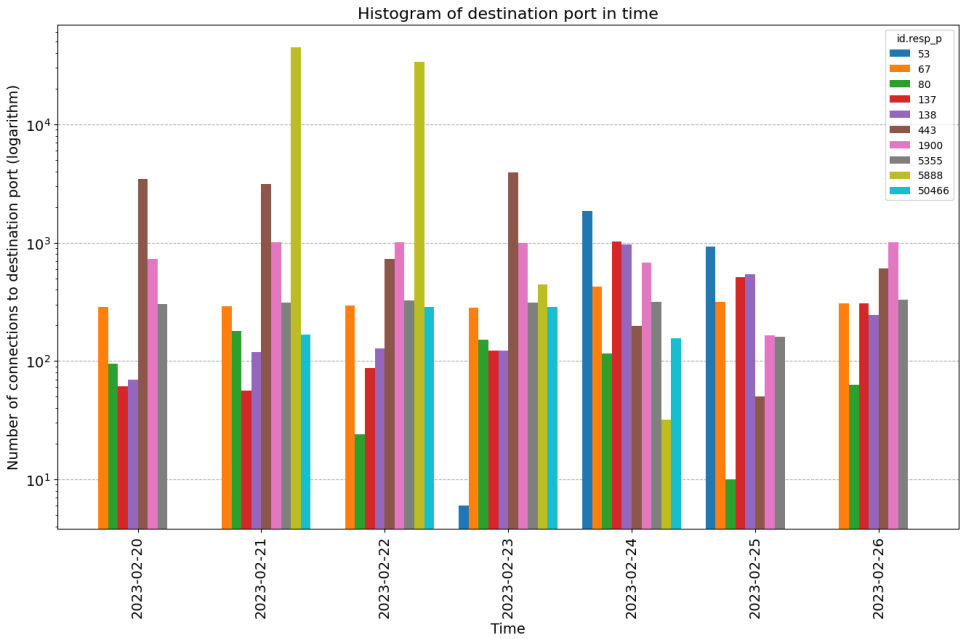


Figure 10.15: Histogram of top 10 destination ports used by the Windows VM 3 for each day of the experiment in logarithmic scale.

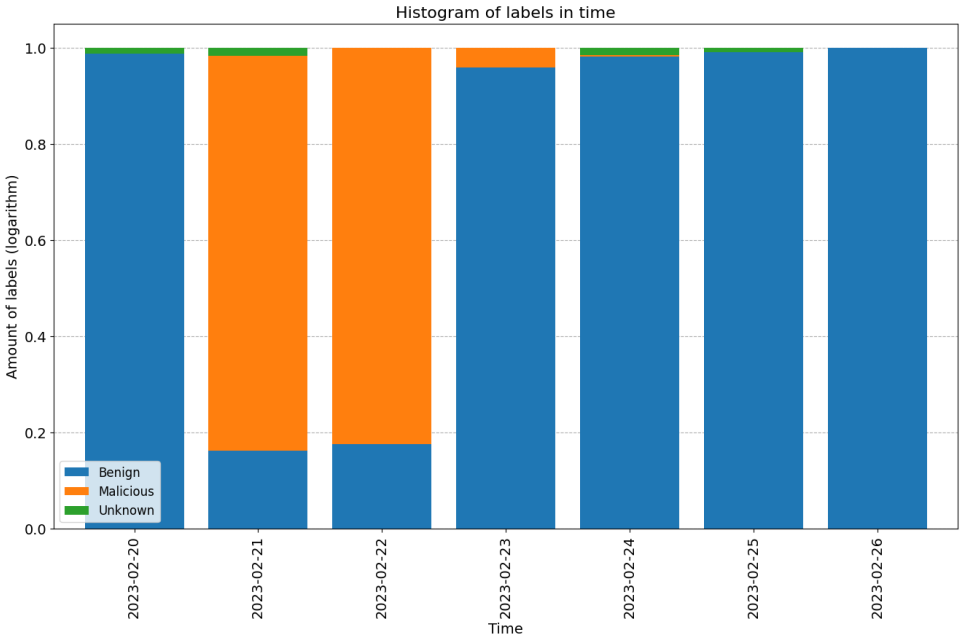


Figure 10.16: Stacked percentage plot of the number of labels for each day for the Windows VM 3.

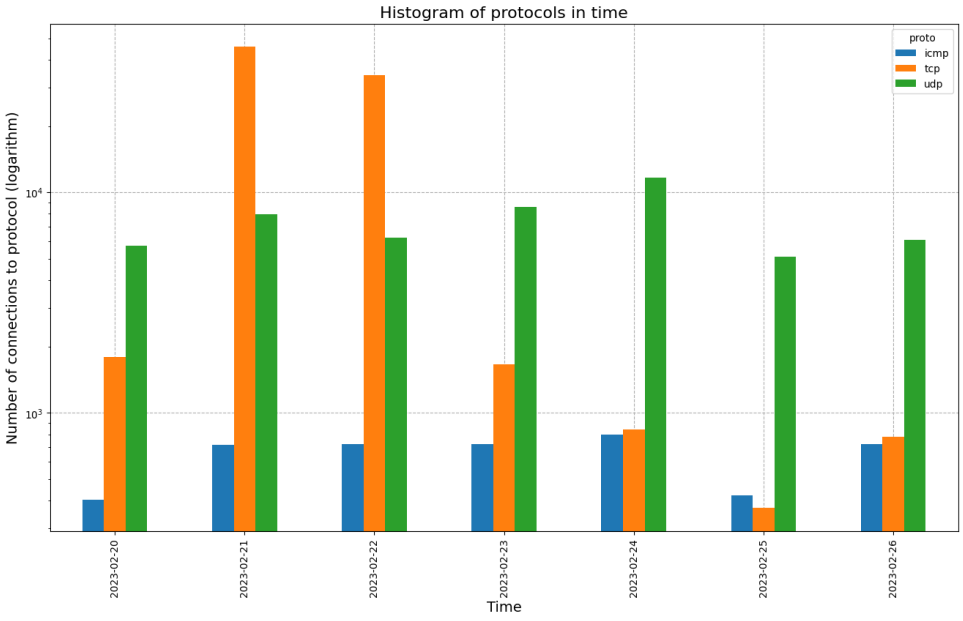


Figure 10.17: Histogram of protocols used by Windows VM 3 across all days in logarithm scale.

10.3.5 Windows VM AD

The total number of flows for Windows VM AD is **184,540**. The size of the network flow files is approximately 27,37 MB.

Figure 10.18 displays the Trickbot malware infection on day three. From figure 10.19, it is apparent that the Windows AD server served as the primary DNS server for Windows client machines.

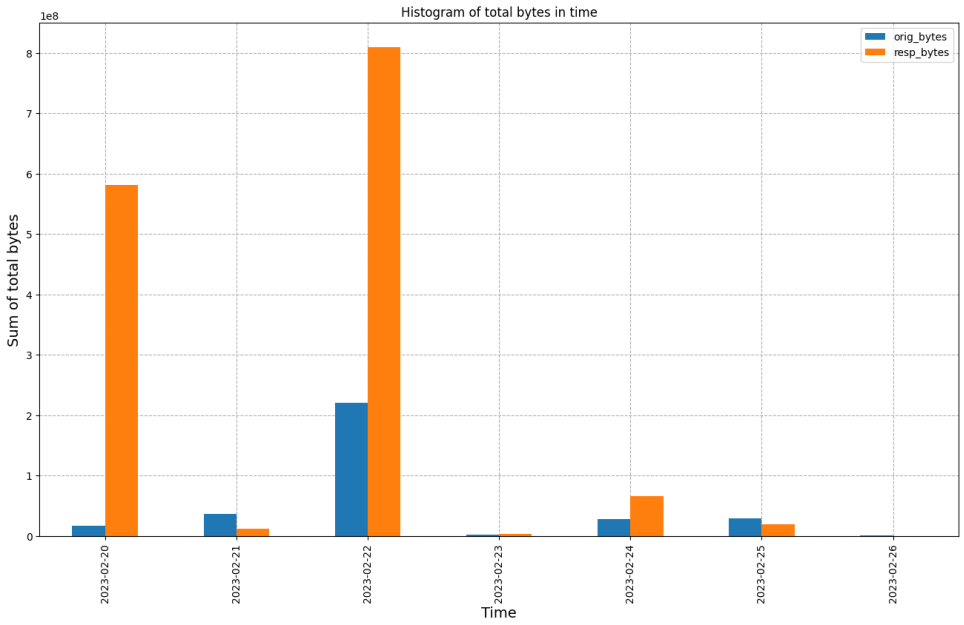


Figure 10.18: Histogram of the sum of incoming and outgoing bytes across all days of the experiment on Windows VM AD.

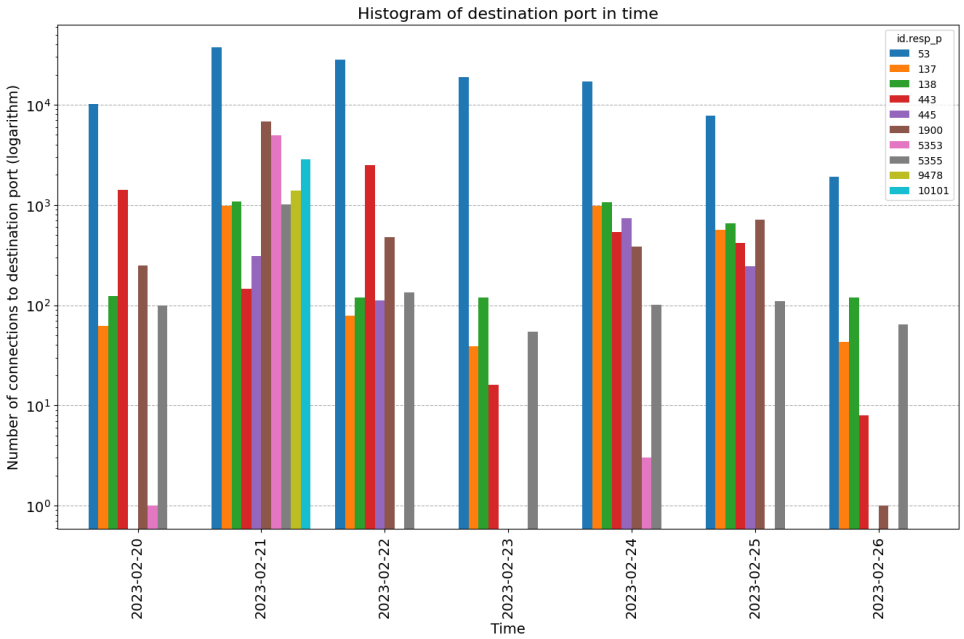


Figure 10.19: Histogram of top 10 destination ports used by the Windows VM AD for each day of the experiment in logarithmic scale.

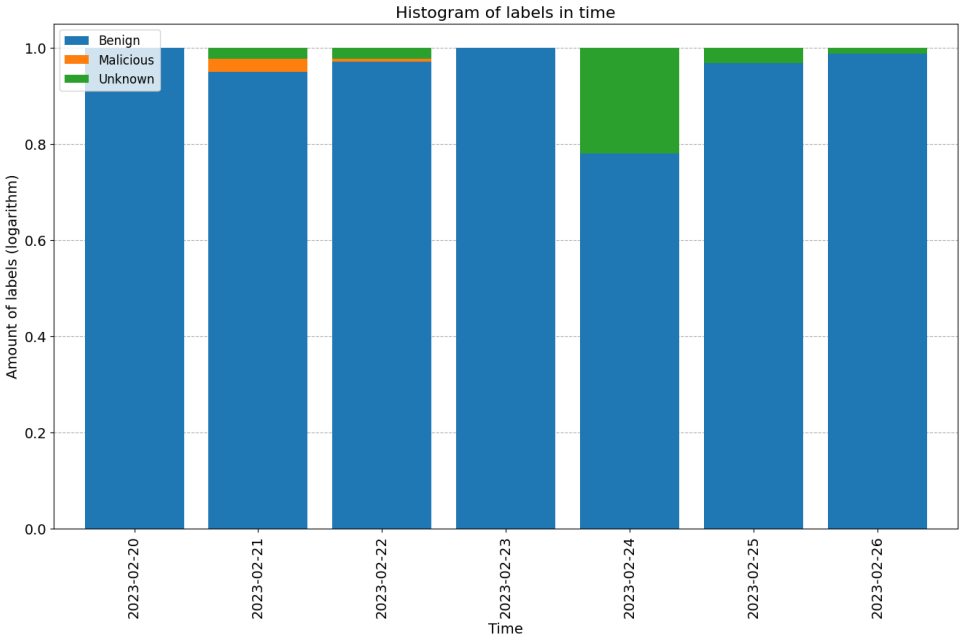


Figure 10.20: Stacked percentage plot of the number of labels for each day for the Windows VM AD.

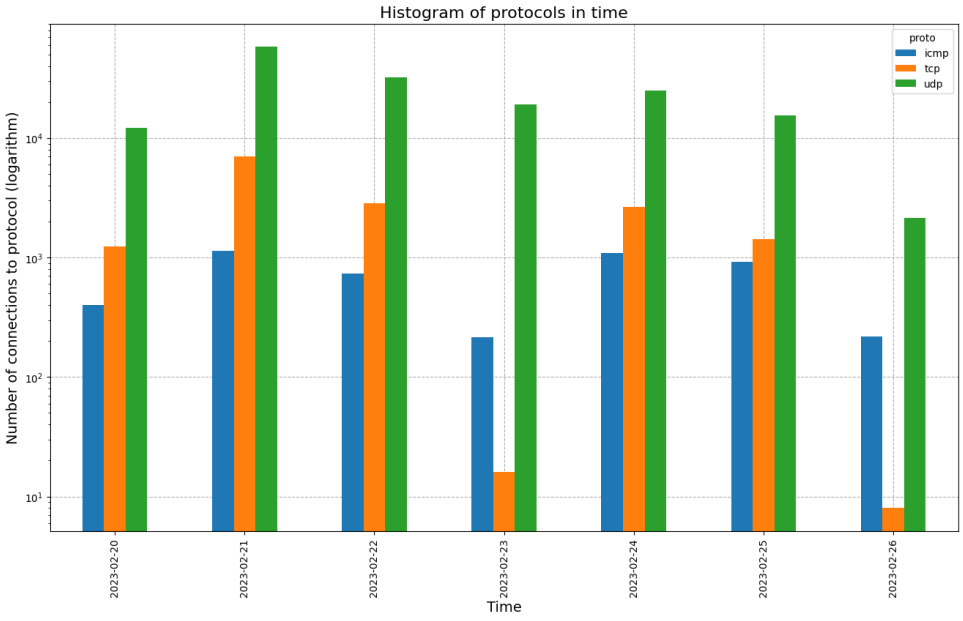


Figure 10.21: Histogram of protocols used by Windows VM AD across all days in logarithm scale.

10.3.6 MacOS laptop

The total number of flows for MacOS laptop is **836,782**. The size of the network flow files is approximately 131,77 MB.

Compared to other devices, figure 10.22 shows a higher amount of bytes. That is more likely caused by the fact that this device was bare-metal and more user-friendly to use.

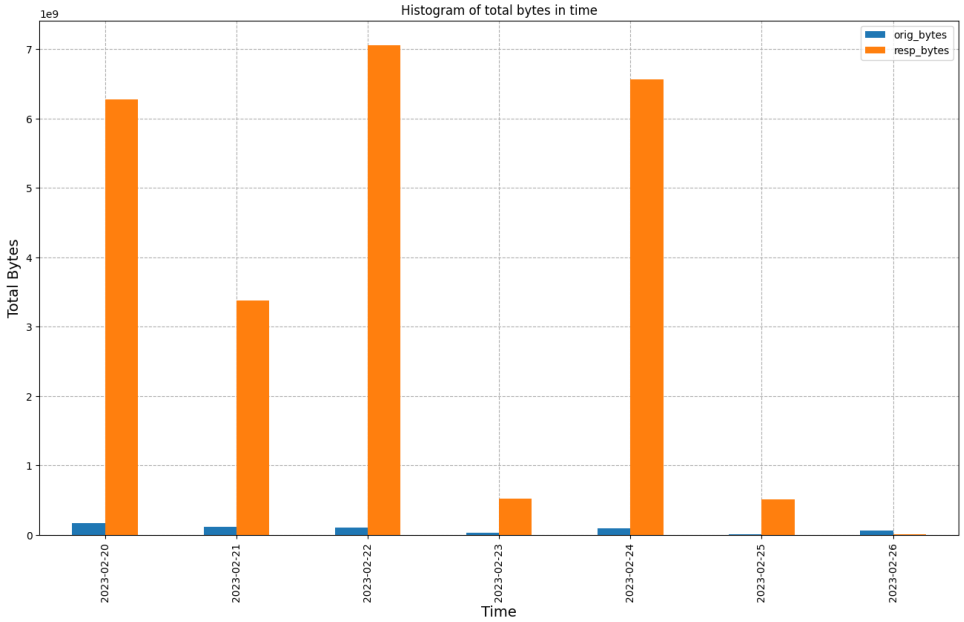


Figure 10.22: Histogram of the sum of incoming and outgoing bytes across all days of the experiment on MacOS laptop.

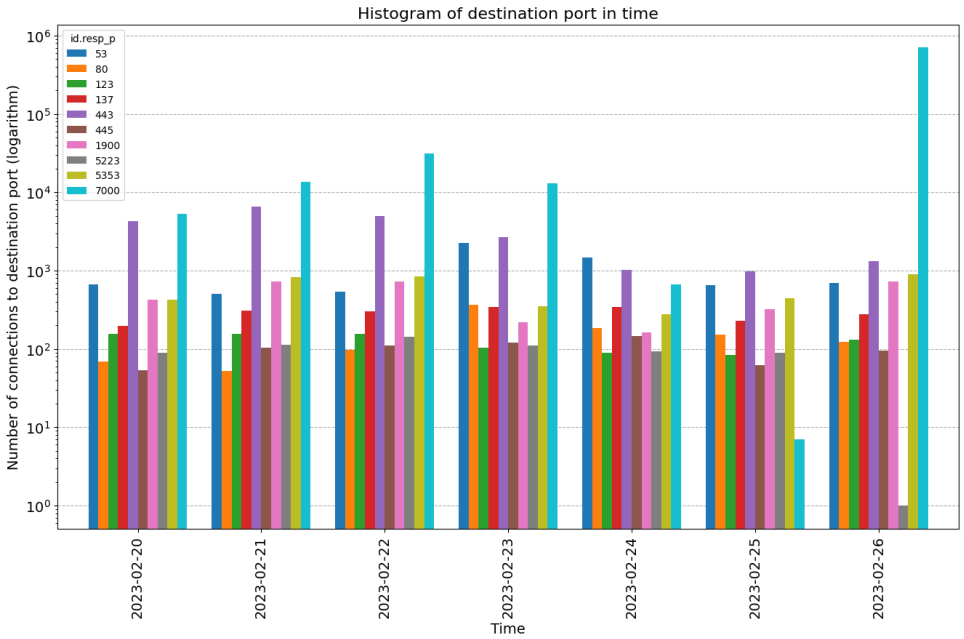


Figure 10.23: Histogram of top 10 destination ports used by the MacOS laptop for each day of the experiment in logarithmic scale.

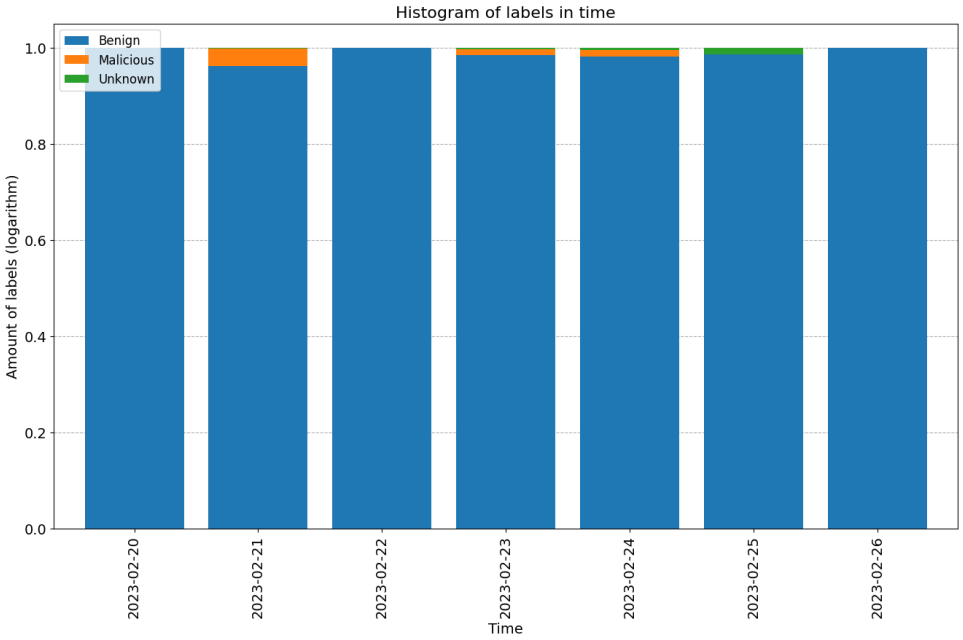


Figure 10.24: Stacked percentage plot of the number of labels for each day for the MacOS laptop.

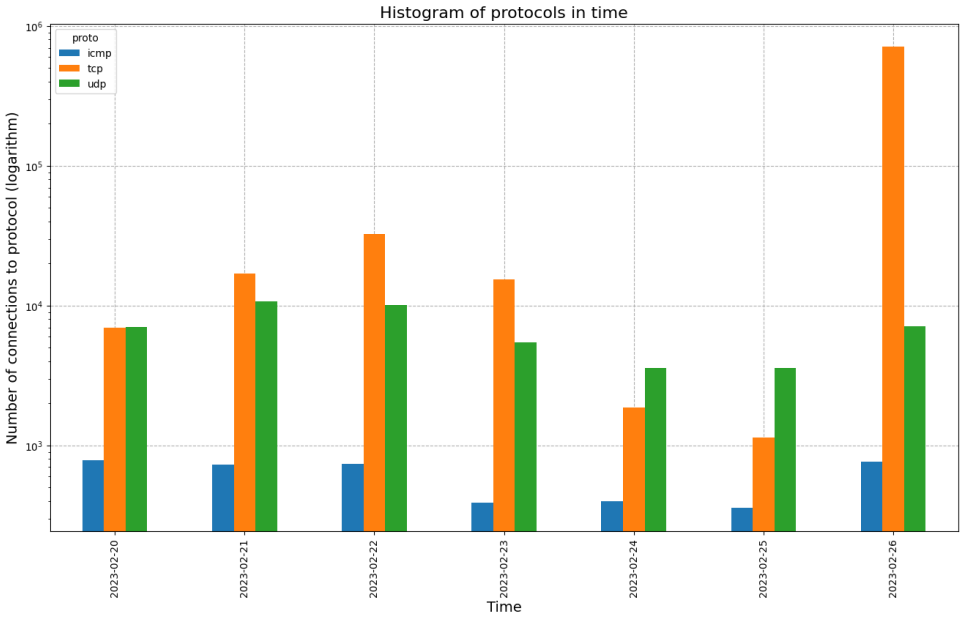


Figure 10.25: Histogram of protocols used by MacOS laptop across all days in logarithm scale.

10.3.7 iPhone

The total number of flows for iPhone is **48,671**. The size of the network flow files is approximately 7,71 MB.

In figure 10.27, you can see that most of the traffic across all days went to port 443 (HTTPS), as this device was left uninfected across the whole experiment and mostly used by researchers to produce benign traffic.

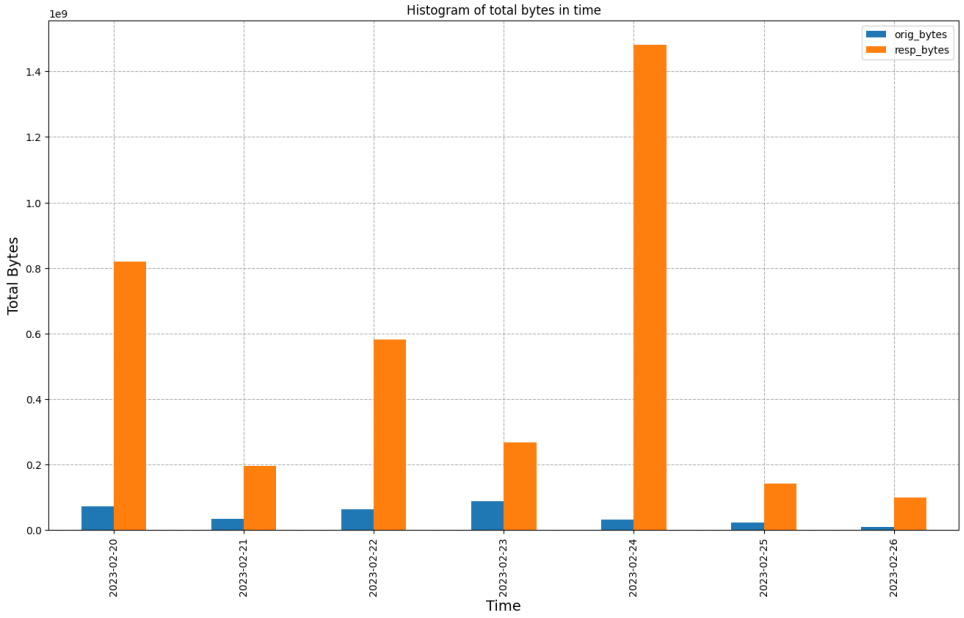


Figure 10.26: Histogram of the sum of incoming and outgoing bytes across all days of the experiment on iPhone.

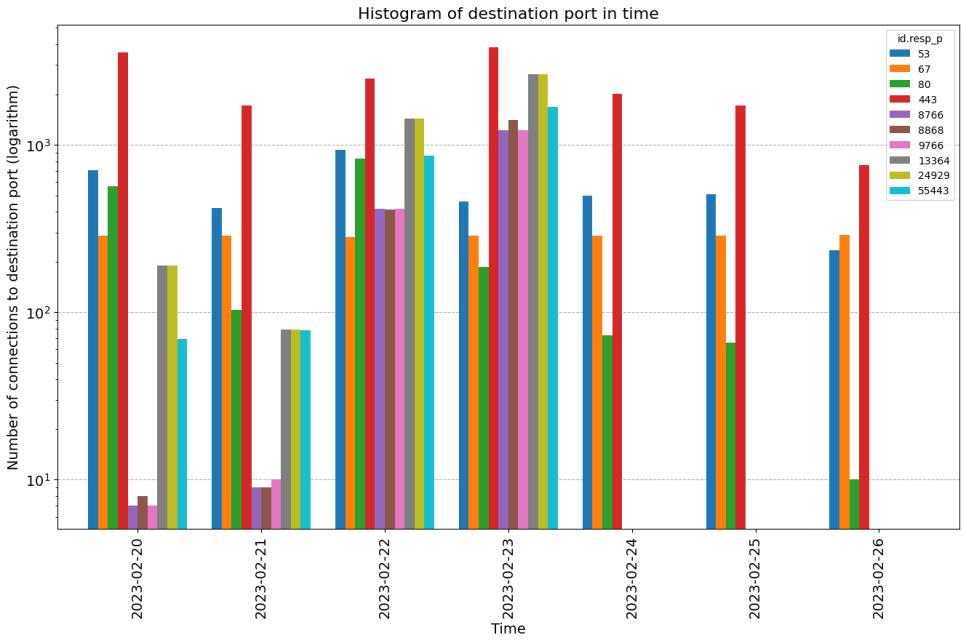


Figure 10.27: Histogram of top 10 destination ports used by the iPhone for each day of the experiment in logarithmic scale.

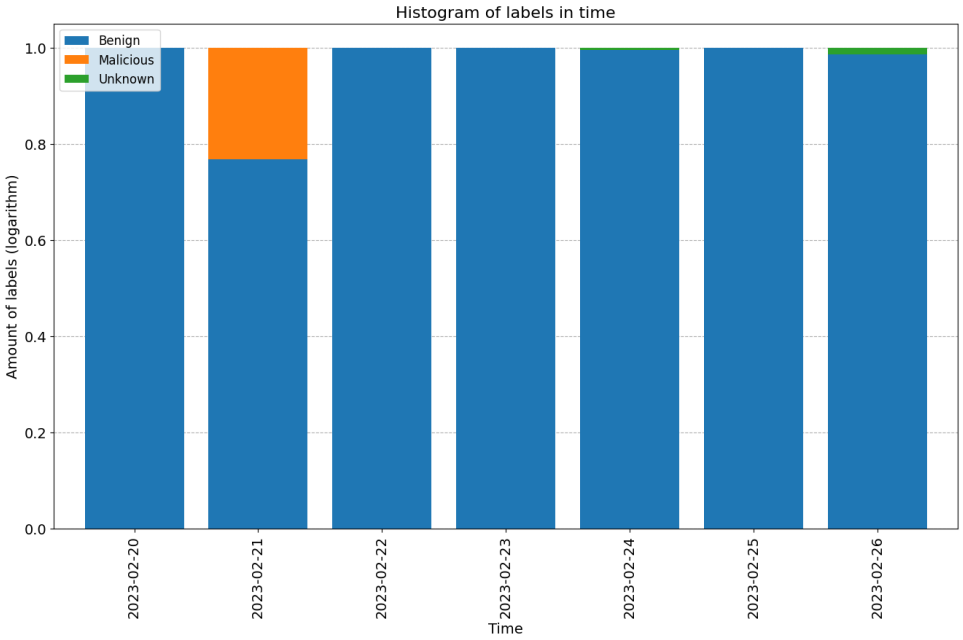


Figure 10.28: Stacked percentage plot of the number of labels for each day for the iPhone device.

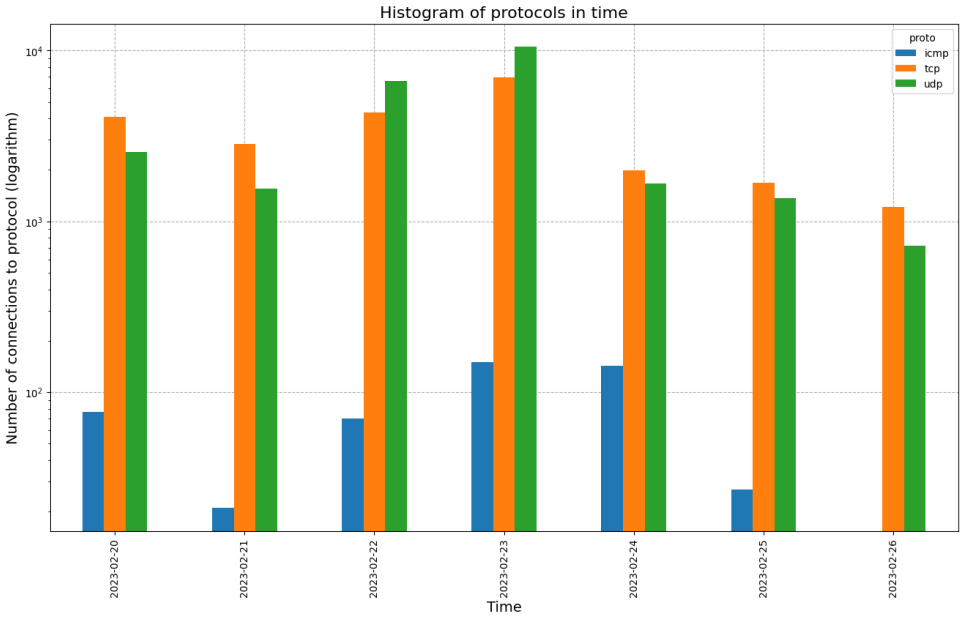


Figure 10.29: Histogram of protocols used by iPhone across all days in logarithm scale.

10.3.8 Android phone

The total number of flows for Android phone is **62,384**. The size of the network flow files is approximately 10,14 MB.

Similarly to iPhone, figure 10.31 shows that most connections were heading to port 443 (HTTPS).

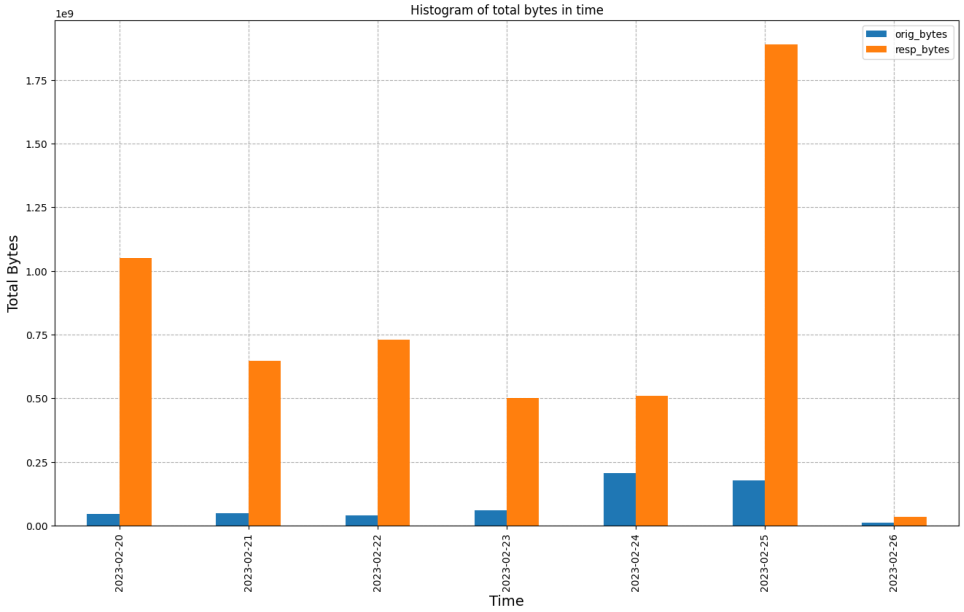


Figure 10.30: Histogram of the sum of incoming and outgoing bytes across all days of the experiment on Android phone.

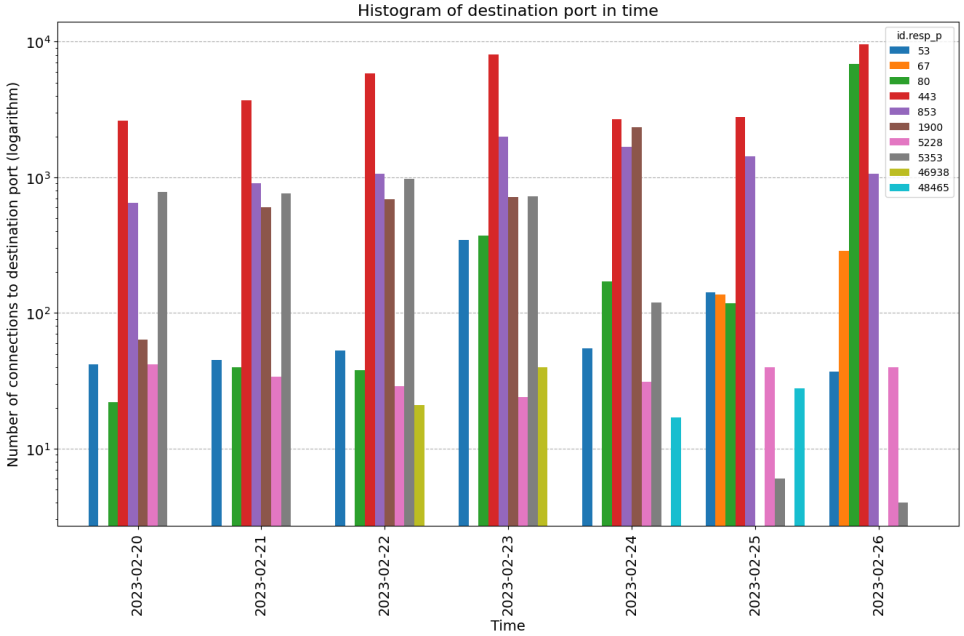


Figure 10.31: Histogram of top 10 destination ports used by the Android phone for each day of the experiment in logarithmic scale.

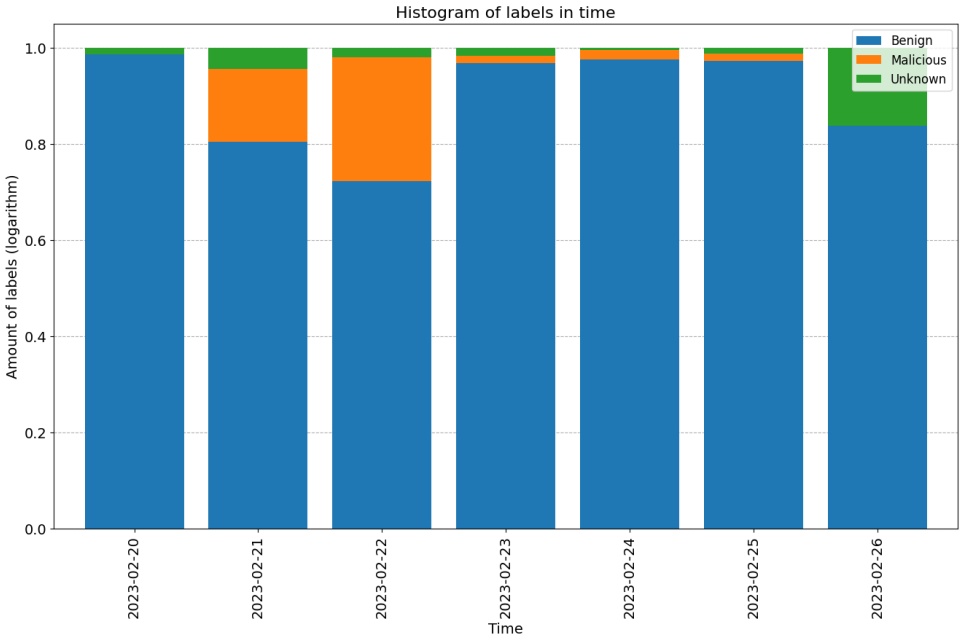


Figure 10.32: Stacked percentage plot of the number of labels for each day for the Android phone.

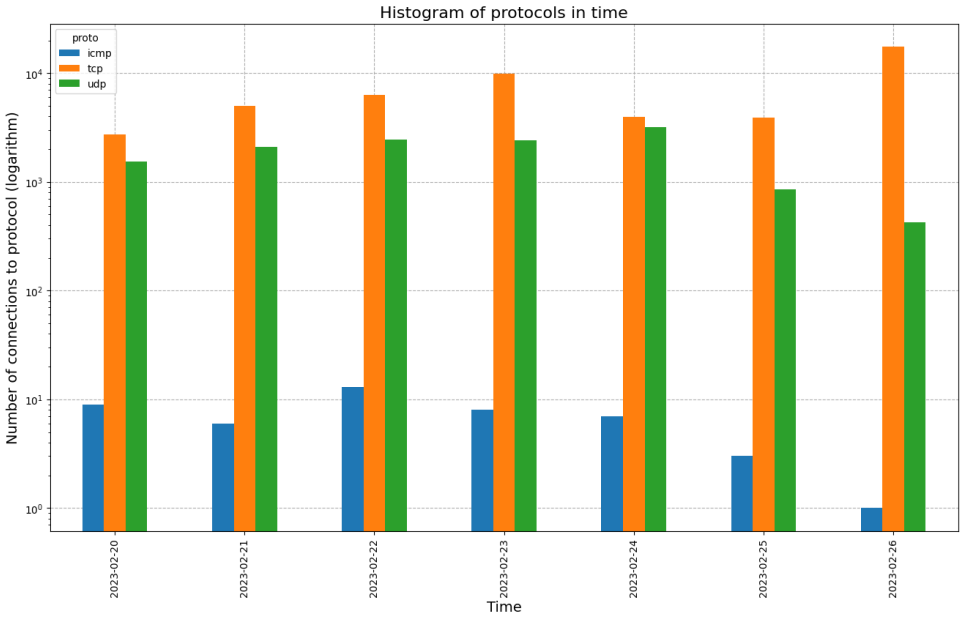


Figure 10.33: Histogram of protocols used by Android phone across all days in logarithm scale.

10.3.9 Alexa assistant

The total number of flows for Alexa assistant is **76,073**. The size of the network flow files is approximately 11,55 MB.

On day four, figure 10.34 shows a spike in bytes sent and received, caused by leaving the Spotify application playing through Alexa assistant for most of the working hours. This is supported by figure 10.36, where day four has more benign traffic than the other six days.

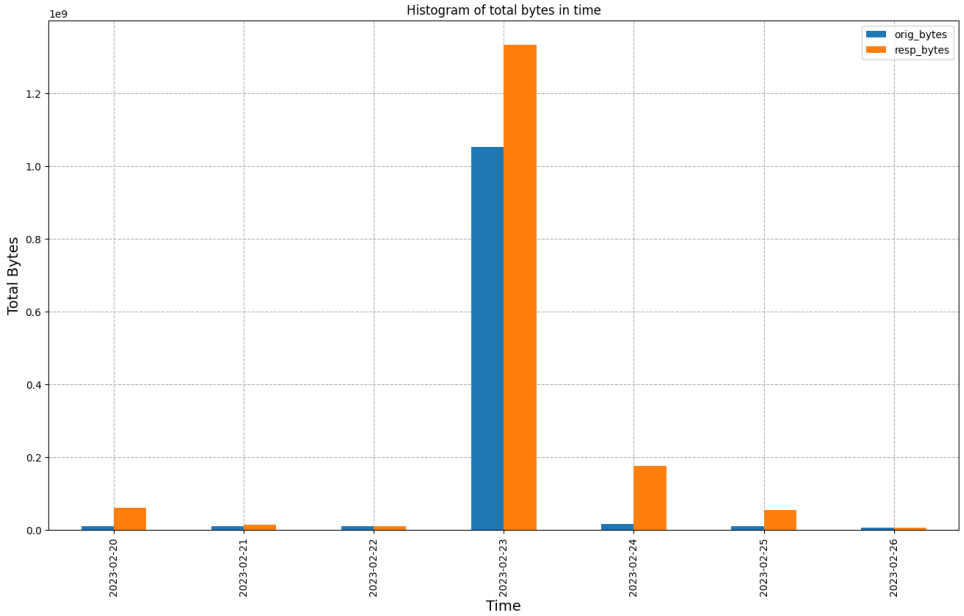


Figure 10.34: Histogram of the sum of incoming and outgoing bytes across all days of the experiment on Alexa assistant.

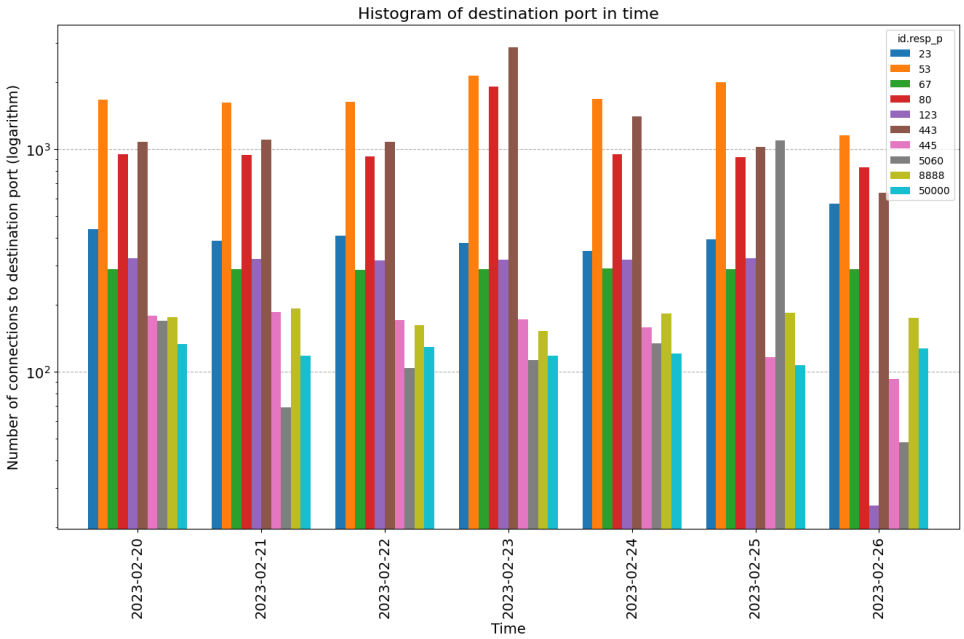


Figure 10.35: Histogram of top 10 destination ports used by the Alexa assistant for each day of the experiment in logarithmic scale.

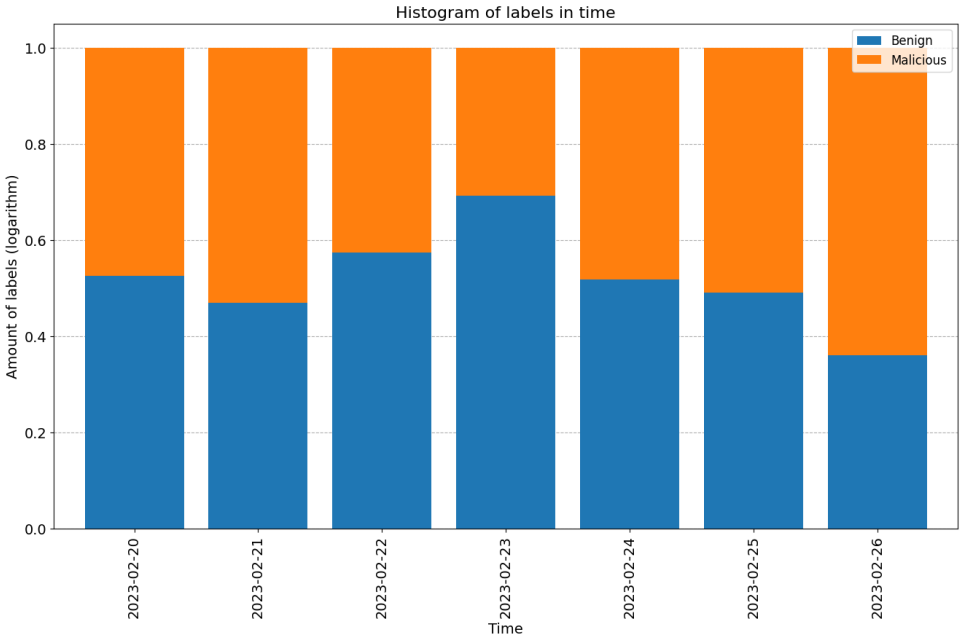


Figure 10.36: Stacked percentage plot of the number of labels for each day for the Alexa assistant.

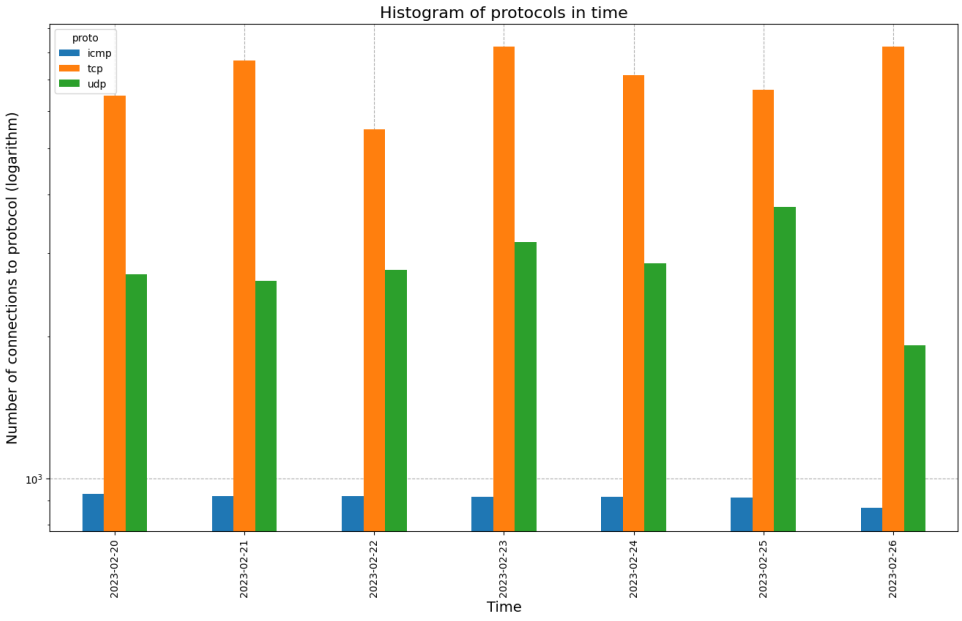


Figure 10.37: Histogram of protocols used by Alexa assistant across all days in logarithm scale.

10.3.10 Chromecast TV assistant

The total number of flows for Chromecast TV assistant is **280,643**. The size of the network flow files is approximately 43,49 MB.

As the Chromecast TV assistant was left uninfected throughout the experiment, figure 10.39 shows that the top ten destination ports are still the same on all but the last day. Moreover, there is only one port addition on the last day, and the other nine ports are the same as in the previous days.

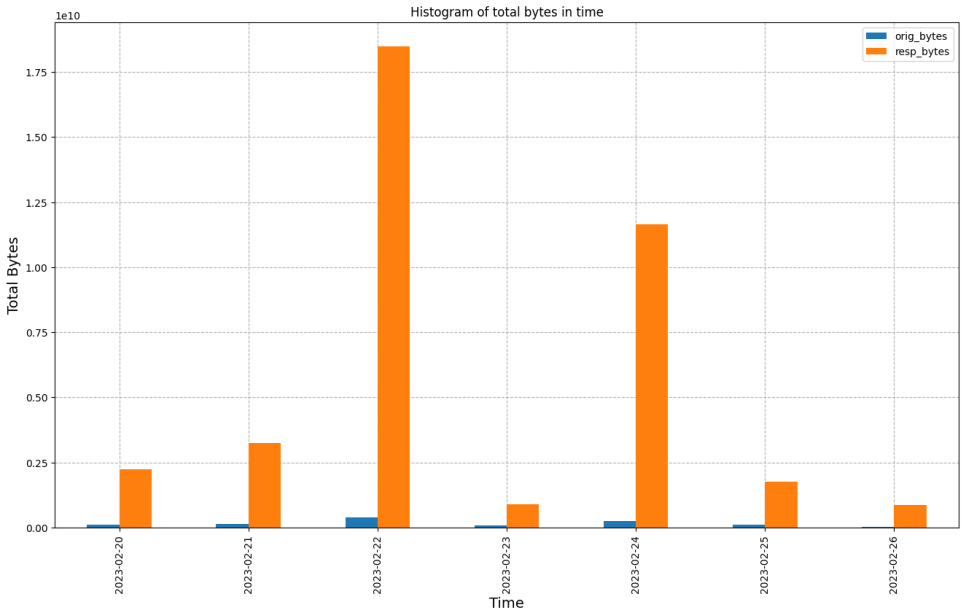


Figure 10.38: Histogram of the sum of incoming and outgoing bytes across all days of the experiment on Chromecast.

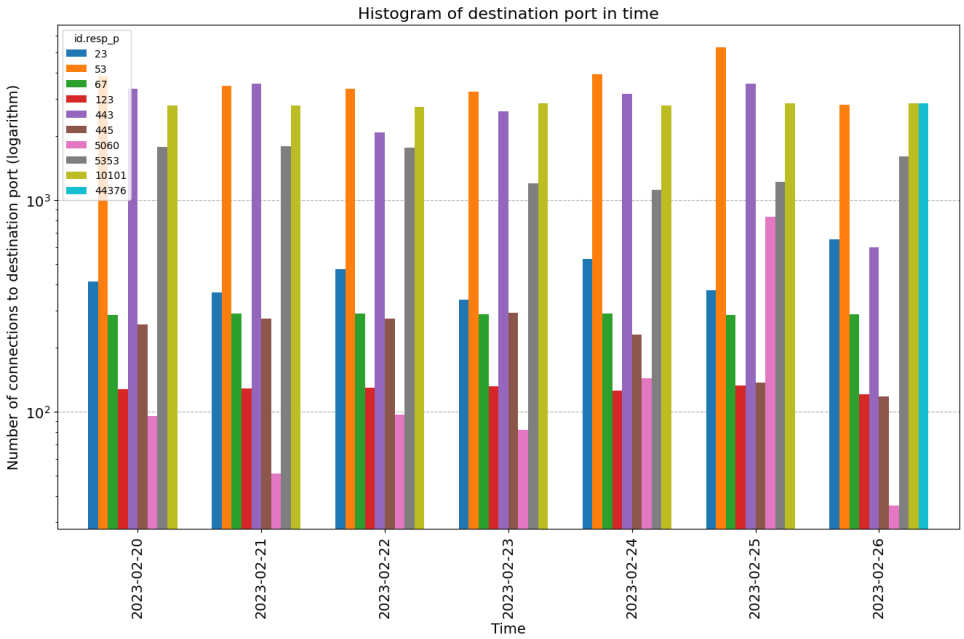


Figure 10.39: Histogram of top 10 destination ports used by the Chromecast for each day of the experiment in logarithmic scale.

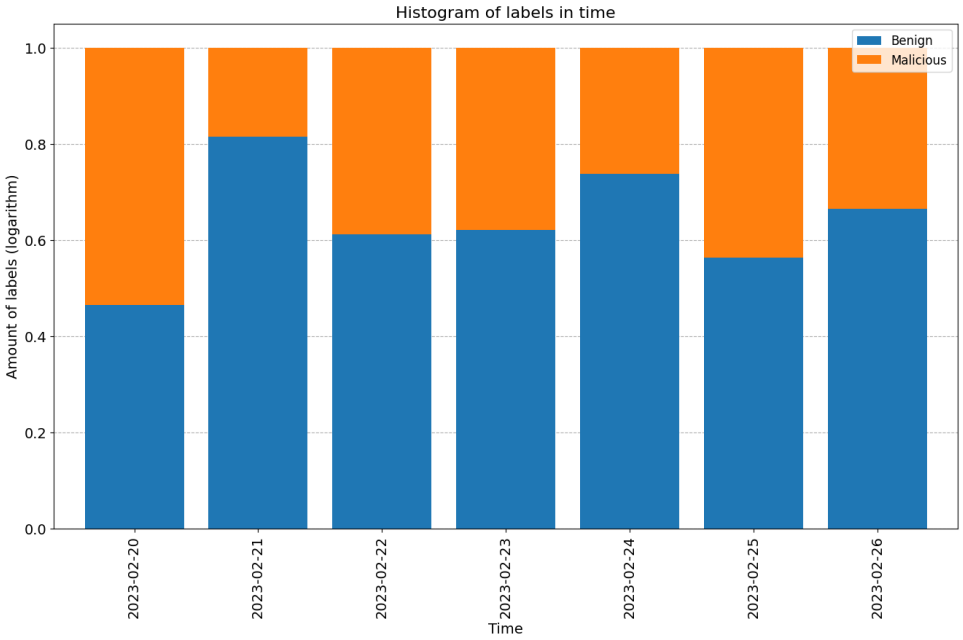


Figure 10.40: Stacked percentage plot of the number of labels for each day for the Chromecast assistant.

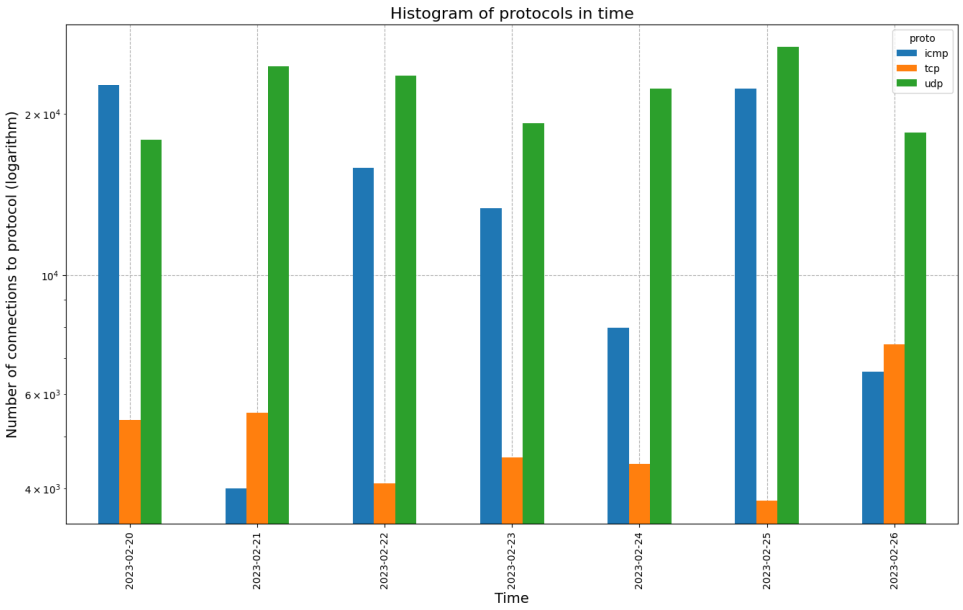


Figure 10.41: Histogram of protocols used by Chromecast across all days in logarithm scale.

10.3.11 Raspberry Pi

The total number of flows for MacOS laptop is **199,266**. The size of the network flow files is approximatelly 31,90 MB.

Figure 10.42 demonstrates the possibility of exploiting this device from the internet on days two to six, as there is a very small amount of traffic on the first and the last day.

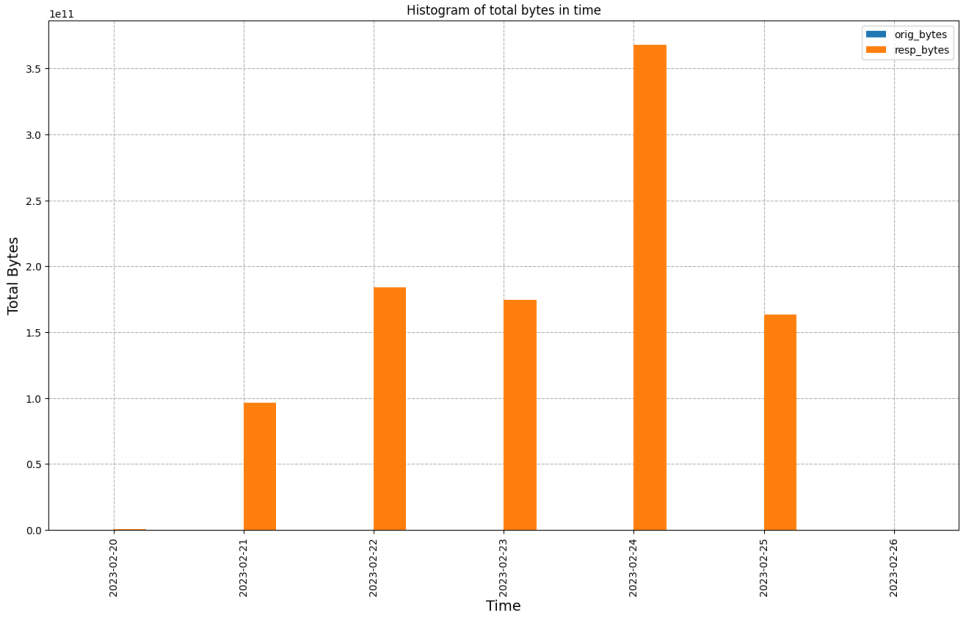


Figure 10.42: Histogram of the sum of incoming and outgoing bytes across all days of the experiment on Raspberry Pi.

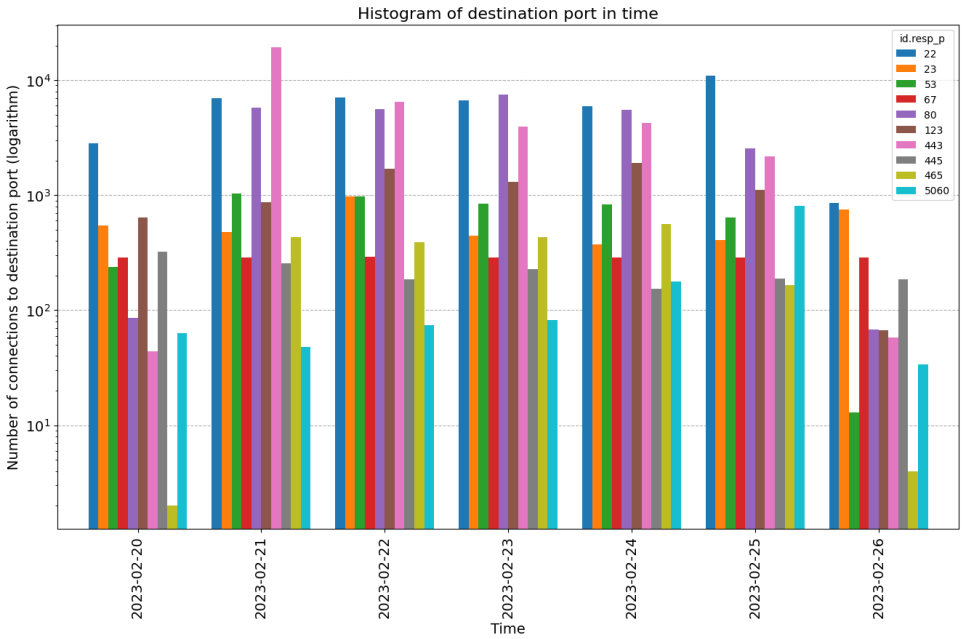


Figure 10.43: Histogram of top 10 destination ports used by the Raspberry Pi for each day of the experiment in logarithmic scale.

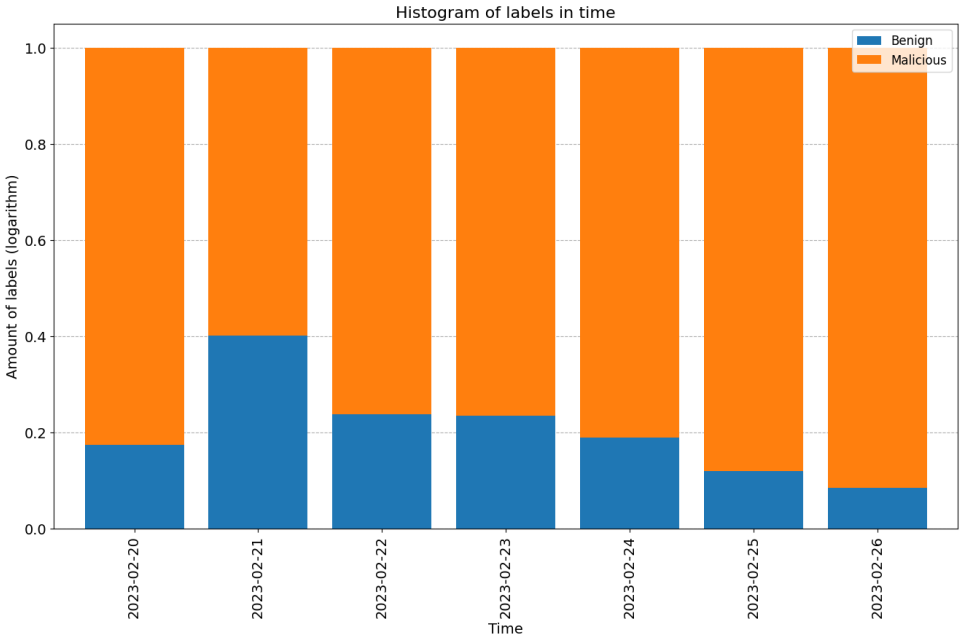


Figure 10.44: Stacked percentage plot of the number of labels for each day for the Raspberry Pi.

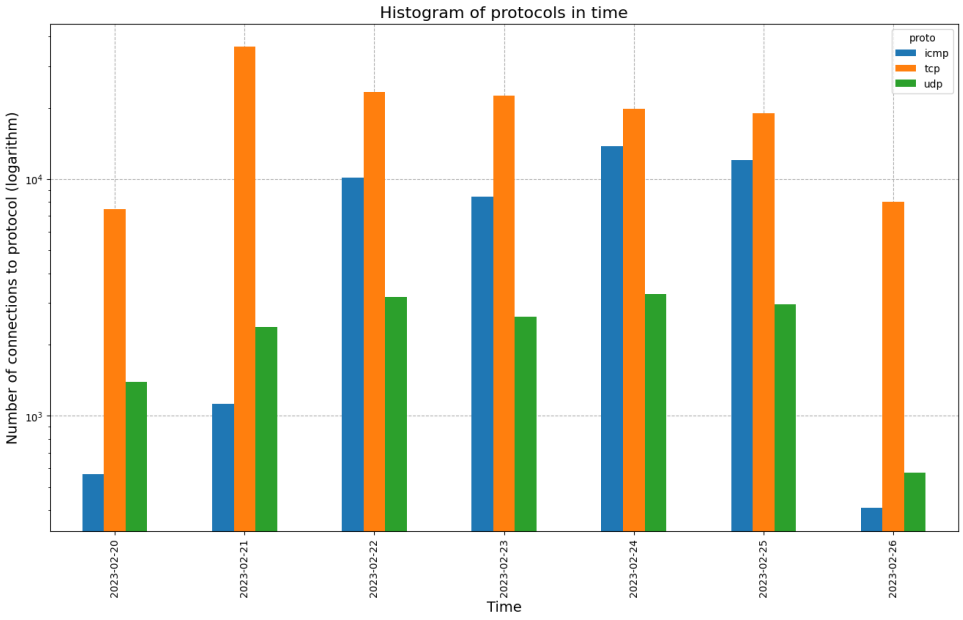


Figure 10.45: Histogram of protocols used by Raspberry Pi across all days in logarithm scale.

Chapter 11

Intended usage

This chapter introduces three possible use cases for the CTU-SME-11 dataset, together with the data selection that we recommend for each use case.

11.1 Anomaly detection

The first selected use case is for testing anomaly detection algorithms. All anomaly detection algorithms need a baseline of benign traffic. This is the sole purpose of the first day of the dataset, where for all devices the whole traffic is benign. This traffic can be used to learn the benign ('normal') patterns.

There are multiple suitable parts of our dataset to test your anomaly detection algorithms for detecting malicious traffic in a large portion of benign traffic. Specifically, we can recommend taking traffic of MacOS from day two (2023-02-21), where the ratio of **benign:malicious is 28372:30**.

A similar proportion of labels for Windows machines can be found in Windows VM 2 on day 5 (2023-02-24). Here the ratio of benign:malicious is even more significant: **21545:2**. This is suited for algorithms that want to detect anomalies that are 'too small' instead of 'too large',

If the anomaly detection algorithm is based on volume and needs to detect what is 'too large', then day 4 and day 5 of the Linux VM should be a validation that the algorithm works, since the volume of malicious traffic on those days is much larger than the benign.

11.2 Clustering

Another technique that the CTU-SME-11 dataset can be used for is clustering. Examples of clustering are to cluster malware families based on their behavior. As our dataset has traffic from 14 malware samples and many attack tools, it is possible to try to cluster them.

For clustering different malware families in the same device, we recommend using Windows Client VM 2 day 3 and day 4. For clustering different malware families in different types of devices we recommend using Windows Client VM 2 day 3 and Linux Ubuntu day 3.

The complete list of malware samples together with their type (malware family) can be seen in Table 5.1. In total, we included six malware families: adware, info stealer, remote access trojan, ransomware, crypto miner, and spyware.

11.3 Classification

The last technique that we would like to introduce as a possible usage for the CTU-SME-11 is machine learning classification (or detection for the case of two labels). For classification, most researchers would expect an approximate 1:1 balance in the labels in order for their algorithms to work. Such a balance of labels can be found in the traffic of Amazon Alexa assistant on day 6, where the benign:malicious balance 5545:5775. However, note that the benign traffic is generated by the Alexa device and the malicious traffic is coming from the internet to the Alexa device, making it easy to detect.

A more challenging classification in a balanced scenario can be done by using Windows Client VM 3 day 3 together with Windows Client VM day 4. In total these two days have 34199 malicious flows and 17665 benign flows, making a ratio of 2:1 with a larger amount of malicious.

In case a non-IoT device is required, the selection will need to be modified across multiple days and filtered. For example, we can recommend taking traffic from Windows VM 2 from day 2, and benign-only traffic from the same device from day 4. Then, the balance of those samples will be 26783:30499 (benign:malicious).

Chapter 12

Limitations

12.1 VirtualBox captures

As already described in section 8.1, the capture server was missing packets from the internal communication between virtual machines (when a VM talked to another VM). Therefore, we needed to capture this traffic using VBoxManage the *trace* capability that creates a PCAP file in the VirtualBox host. Those two files then needed to be merged. This merge was done only for Windows VM 2, Windows VM 3, and Windows VM AD on days 5 and 6. Therefore on other days, there is no internal traffic from the communication between the virtual machines. For more details, visit section 8.1.

12.2 Rerunning experiments

Another issue we had is that for administration issues, some PCAP files created by VirtualBox with the *trace* capability were lost. Fortunately, the days lost were benign days and no malware traffic was lost. However, to keep the dataset consistent and so it would be possible to find traffic between the VM, some days needed to be redone approximately one week after the finishing of the dataset creation. Redoing of these experiments was done only with three virtual machines: Windows VM 2, Windows VM 3, and Windows VM AD. The rerun was done only for day five and day six. This benign traffic was then merged with the traffic for this VM captured by the central storage. Note that the communication from the VMs to the internet was not lost and the original was kept. The rerun was only for the internal VM-to-VM communication.

12.3 Splitting by days

As one PCAP file is generated for each day, it is very likely that a network flow starts one day and finishes the next. Therefore, Zeek will create one flow in the first day (marking that the end was not seen) and a different flow in the second day (marking that the beginning was not seen). However, this is a reality for every flow collector in the industry. Splitting the days like this

makes the dataset comfortable to use, not forcing the users to download large amounts of data.

12.4 Data loss for Windows client machines

Due to unknown reasons, the main capture in the storage server was interrupted on all three Windows client machines on day seven (we suspect memory issues). Due to this data loss, we were forced to capture day seven again. As capturing only the missing VMs would cause discrepancies in the data (such as missing interactions with other devices), and given that all the devices were still running uninterrupted in the laboratory, we decided to capture day "seven" again for all the devices some days after the real day seven. This new capture was done on 30.4.2023. The time of those PCAPs was moved with the tool *editcap* to match the last day of the dataset creation week (i.e. 26.2.2023).

12.5 Amount of malware samples

The dataset has 14 malware samples executed and captured. While we are aware that there are datasets with many more samples, this dataset aims for realness and heterogeneity in terms of platforms and traffic types instead of only executing a large number of malware samples. Therefore, while some might consider the number of malware samples a limitation, we rather consider it a tradeoff between keeping the creation of the dataset realistic and possible in terms of resources.

Chapter 13

Lessons learned

The creation of a network security dataset is a challenging task that requires careful planning, collaboration, and attention to detail. This chapter presents the lessons learned from our experience in creating a network security dataset for this thesis. We discuss the importance of defining research objectives, planning data collection, and documentation. We believe these best practices can help in other research projects involving data collection.

13.1 Defining research objectives

Before starting the creation of the network security dataset, it is important to define the research objectives precisely. This helps to identify the data that needs to be collected and the format in which it needs to be stored. The research objectives should guide the selection of relevant malware samples, benign traffic to be generated, and attacks to be performed. The research objective also helps to narrow or widen the scope of the work to be done, keeping the dataset within boundaries in terms of size and resources needed.

In case of this thesis, the research objective was clear from the start, which helped with designing how will the final dataset look like.

13.2 Data collection

The process of data collection can be time-consuming and resource-intensive. Planning the data collection process carefully is essential to ensure that all relevant data is captured. The sources of data should be identified in advance, and the tools and techniques to collect the data should be selected based on the research objectives. It is very important to test that all the data that should be collected are being captured, stored, and backed up. With the increasing requirements on the dataset size, number of devices used, and time length, we advise making the collection as automatized as possible. During the capture phase, many things need to be done or monitored. While not having the capture and backup of the data done automatically, some of the data will likely be lost, making it impossible to complete the experiment successfully.

Even though we have most of the data collection automated, we encountered two issues in terms of data collection. The first was not capturing VirtualBox PCAPs at the start of the experiment, while the second was losing the PCAPs, which were not stored and backed up automatically due to a human mistake while copying the files to a backup directory. Fortunately, since there were three capturing mechanisms being used simultaneously, the data could be recovered and merged.

13.3 Documentation

It is important to document the whole process, from planning the experiment through the capture phase and labeling. Any transformations or manipulations performed on the dataset should be documented as well. Proper documentation helps to ensure the reproducibility of the research and enables other researchers to use the dataset for their own work. It also greatly improves the ability to explain what is happening in the data in case of any inconsistencies or errors.

An important part of doing this thesis was the frequent work to produce intermediate results as part of the documentation. Working on such small side projects helped to have very clear ideas about the work very early on and therefore greatly facilitated the writing of this thesis. Two such efforts were made. First, a poster was created and presented at the POSTER 2023 conference [37] that can be seen in Apendix A. Second, a draft of a paper about this dataset was prepared for the Journal Data In Brief, to be sent.

Chapter 14

Conclusion

Generating new network security datasets is important for the field of network security. Cyber attacks are becoming more frequent and complex, and we need diverse and comprehensive datasets to develop and evaluate new security solutions. By capturing the latest attack techniques and malware samples, a new dataset can help researchers better understand the current threat landscape and develop more effective network security tools. High-quality network security datasets can also contribute to developing and testing machine learning algorithms, which have the potential to improve the detection and prevention of network attacks significantly. Creating a new network security dataset is a crucial step towards enhancing the security of our digital infrastructure and safeguarding against cyber threats.

Our methodology began with a thorough review of existing datasets, followed by conducting interviews with cybersecurity experts. Based on these insights, we designed a new dataset and searched for recent threats to ensure their relevance. After those preparations, we executed the planned scenarios on a real network. Finally, we took the effort to label the data on a network flow level, making the dataset ready to use for machine learning algorithms.

The dataset was created on a real computer network, capturing eleven devices of various types over the course of seven consecutive days. In total, the dataset contains approximately 160 GB of network traffic in the form of `PCAP` files and 20 GB in the form of Zeek logs. The total amount NetFlows is `99,993,509`, with `1,608,273` (1,608%) of them being labeled as "Benign", `98,319,252` (98,326%) labeled as "Malicious", and `65,984` (0,066%) unrecognized and labeled as "Unknown".

The main contribution of this thesis is a new labeled network security dataset, called **CTU-SME-11**. This dataset consists of network traffic generated by eleven distinct devices over a seven-day period, including benign traffic generated by humans, traffic from 14 real and recent malware samples, and attack traffic. The devices included in the dataset are of different types, ranging from mobile devices and virtual machines to IoT devices and hardware PCs. By incorporating this diverse range of devices, the dataset provides a comprehensive view of the network traffic of a small-medium enterprise and enables researchers to develop more robust security solutions to protect against various types of cyber attacks.

The dataset aims to fill in the gaps in recency and diversity of datasets in the network security field. Moreover, the documentation for this dataset is very extensive, which makes it easier for consumers to use this dataset.

Bibliography

[1] Cisco Umbrella. (2021). Cybersecurity Threat Trends Report. Available from: https://umbrella.cisco.com/info/cybersecurity-threat-trends-report [Accessed 13-May-2023]

[2] Yuchong, L., Qinghui L., A comprehensive review study of cyber-attacks and cyber security. Emerging trends and recent developments, Energy Reports, 2021. Available from: https://www.sciencedirect.com/science/article/pii/S2352484721007289

[3] Griffiths, C. The Latest 2023 Cyber Crime Statistics. AAG IT Services, 2023. Available from: https://aag-it.com/the-latest-cyber-crime-statistics/ [Accessed 12-May-2023]

[4] Wannacry ransomware attack summary. Data Protection Report [online]. 2017. Available from: https://www.dataprotectionreport.com/2017/05/wannacry-ransomware-attack-summary/ [Accessed 30-Apr-2023]

[5] SolarWinds attack explained: And why it was so hard to detect. CSO [online]. 2020 . Available from: https://www.csoonline.com/article/3601508/solarwinds-supply-chain-attack-explained-why-organizations-were-not-prepared.html [Accessed 30-Apr-2023]

[6] Rahul, Y., Pathak, P., and Saraswat, S. Comparative Study of Datasets used in Cyber Security Intrusion Detection. International Journal of Scientific Research in Computer Science, Engineering and Information Technology [online]. 302-312. ISSN 2456-3307. Available from: https://doi.org/10.32628/CSEIT2063103 [Accessed 22-Oct-2022]

[7] Bendl, Štěpán, Valeros, Veronica, & Garcia, Sebastian. (2023). CTU-SME-11: a labeled dataset with real benign and malicious network traffic mimicking a small medium-size enterprise environment (1.0.0) [Data set]. Zenodo. Available from: https://doi.org/10.5281/zenodo.7958259

[8] KDD Cup 1999 Data. (n.d.). Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Accessed 03-May-2023]

[9] Moustafa, N. and Ross, D. (2015). The NSL-KDD dataset and selected papers from the workshop on the learning to detect and diagnose network

intrusions. Journal of Machine Learning Research, 16(1). Available from: https://www.unb.ca/cic/datasets/nsl.html [Accessed 22-Oct-2022]

[10] Moustafa, N. and Slay, J. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 2015, pp. 1-6, https://doi.org/10.1109/MilCIS.2015.7348942. [Accessed 22-Oct-2022]

[11] Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A., Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Computers & Security, May 2012, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2011.12.012 [Accessed 22-Oct-2022]

[12] Khan, M., KARIM, M., Yangwoo, K. A Scalable and Hybrid Intrusion Detection System Based on the Convolutional-LSTM Network. Symmetry [online]. 2019, 11(4) [cit. 2022-10-22]. ISSN 2073-8994. Available from: https://doi.org/10.3390/sym11040583 [Accessed 22-Oct-2022]

[13] Garcia, S., Grill, M., Stiborek, J., Zunino, A. An empirical comparison of botnet detection methods. Computers & Security [online]. 2014, 45, 100-123. ISSN 01674048. Available from: https://doi.org/10.1016/j.cose.2014.05.011 [Accessed 22-Oct-2022]

[14] Sharafaldin, I., Lashkari, A., Ghorbani, A., “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018 [Accessed 23-Oct-2022]

[15] Gharbib, A., Sharafaldin, I., LASHKARI, A., Ghorbani, A. An Evaluation Framework for Intrusion Detection Dataset. In: 2016 International Conference on Information Science and Security (ICISS) [online]. IEEE, 2016, 2016, s. 1-6. ISBN 978-1-5090-5493-0. Available from: https://doi.org/10.1109/ICISSEC.2016.7885840 [Accessed 23-Oct-2022]

[16] Leevy, J.L., Khoshgoftaar, T.M. A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. J Big Data 7, 104 (2020). Available from: https://doi.org/10.1186/s40537-020-00382-x [Accessed 24-Oct-2022]

[17] Myneni, S., Chowdhary, A., Sabur, A., Sengupta, S., Agrawal, G.m Huang, D., Kang, M. (2020). DAPT 2020 -Constructing a Benchmark Dataset for Advanced Persistent Threats. [Accessed 24-Oct-2022]

[18] Garcia, S., Parmisano, A., Erquiaga, M.J. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.4743746 [Accessed 23-Oct-2022]

[19] Valeros, V., Garcia, S. Hornet 40: Network dataset of geographically placed honeypots. Data in Brief [online]. 2022, 40. ISSN 23523409. Available from: https://doi.org/10.1016/j.dib.2022.107795 [Accessed 23-Oct-2022]

[20] Lawrence, H., EZEOBI, U., TAUIL, O., NOSAL, J., REDWOOD, O., ZHUANG, Y, BLOOM, G. CUPID: A labeled dataset with Pentesting for evaluation of network intrusion detection. Journal of Systems Architecture [online]. 2022, 129 [cit. 2022-10-22]. ISSN 13837621. Available from: https://doi.org/10.1016/j.sysarc.2022.102621 [Accessed 23-Oct-2022]

[21] Dholakiya, P. What Is the Cyber Kill Chain and How It Can Protect Against Attacks. computer.org. Available from: https://www.computer.org/publications/tech-news/trends/what-is-the-cyber-kill-chain-and-how-it-can-protect-against-attacks [Accessed 13-May-2023]

[22] Semi-structured interview. Wikipedia. Available from: https://en.wikipedia.org/wiki/Semi-structured_interview [Accessed 11-May-2023]

[23] Network TAP versus a SPAN port. Niagara networks. Available from: https://www.niagaranetworks.com/solutions/tap-versus-span [Accessed 18-May-2023]

[24] Sharma, A., Kreibich, C., Bannat, F., Amann, J., Lehigh, K., et al. Zeek: An open source network security monitoring tool. Available from: https://zeek.org/ [Accessed 12-Mar-2023]

[25] Garcia, S., Valeros, V. NetflowLabeler: A configurable rule-based labeling tool for network flow files, Available from: https://github.com/stratosphereips/netflowlabeler [Accessed 12-Mar-2023]

[26] Garcia, S., Valeros, V. Towards a better labeling process for network security datasets, Available from: https://doi.org/10.48550/arXiv.2305.01337 [Accessed 21-May-2023]

[27] VirusTotal (2023). VirusTotal - Free Online Virus, Malware and URL Scanner. Available from: https://www.virustotal.com/ [Accessed 11-May-2023]

[28] What is insider threat. Microfocus. Available from: https://www.microfocus.com/en-us/what-is/insider-threat [Accessed 18-May-2023]

[29] ytisf (2023). PyExfil. GitHub repository. Available from: https://github.com/ytisf/PyExfil

[30] Garcia, S., Valeros, V., and others, *Stratosphere Research Laboratory, Cybersecurity group of the Artificial Intelligence Centre*, Faculty of Electrical Engineering at the Czech Technical University. Available from https://www.stratosphereips.org/ [Accessed 12-Mar-2023]

[31] Best practices for securing active directory. Microsoft Learn. (n.d.). Available from: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/best-practices-for-securing-active-directory [Accessed 12-Mar-2023]

[32] Sivanathan, K. 10 Most Popular Linux Distros for 2022. PackageCloud.io. Available from: https://blog.packagecloud.io/10-most-popular-linux-distros-for-2022/ [Accessed 12-Mar-2023]

[33] Detailed market size and share trends empower companies selling mobile phones to get ahead of market changes and compete more effectively. Available from: https://www.idc.com/promo/smartphone-market-share/os [Accessed 11-May-2023]

[34] Uhlár, J., Visual Analysis of Network Packet Capture Files (2020). Masaryk University, Faculty of Informatics. Available from: https://is.muni.cz/th/ejjrg/dp-uhlar.pdf [Accessed 11-May-2023]

[35] The Tcpdump Group. tcpdump: the network traffic analyzer and packet capture tool (2023). Available from: https://github.com/the-tcpdump-group/tcpdump [Accessed 11-May-2023]

[36] Asher, A. How Does Pass-The-Hash Work? risk3sixty 2022. Available from https://risk3sixty.com/2022/10/05/how-does-pass-the-hash-work/ [Accessed 11-May-2023]

[37] Štěpán Bendl. (2023). 27th International Student Conference on Electrical Engineering. Poster 2023. 18. https://poster.fel.cvut.cz/poster2023/ [Accessed 5-May-2023]

Appendix A

Poster for Poster2023 conference

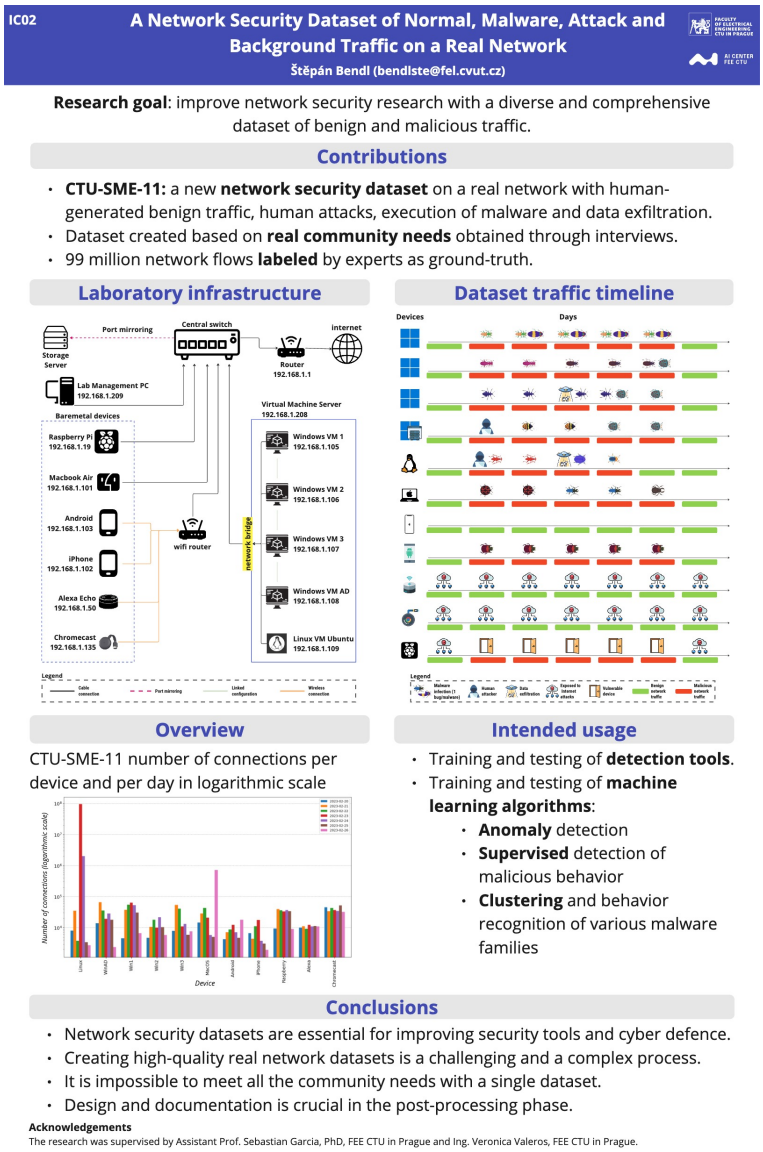


Figure A.1: Poster accepted and presented in the conference Poster2023.