UCLouvain

ULB UNIVERSITÉ LIBRE DE BRUXELLES

UNIVERSITÉ DE NAMUR

RMA Royal Military Academy

HE2B ESI

HELB Ilya Prigogine

Generating realistic background traffic

Using GHOSTS framework

Adrien Defer

Master thesis submitted under the supervision of
Prof. Wim Mees

the co-supervision of
Prof. Georgi Nikolov

in order to be awarded the Degree of
Master in Cybersecurity
System Design and Analysis

Academic year
2022 – 2023

ABSTRACT

Realistic traffic generation is a major issue nowadays. It allows to simulate an infinite number of possible computer attack scenarios in order to train cybersecurity professionals in a realistic situation. However, this generation faces two major difficulties. Firstly, the simulation of the internet, whose size and continuously changes (to name these two points) are not factors that make the job any easier. Secondly, the modelling of human behaviour to generate realistic traffic remains to date impossible to achieve perfectly. This dissertation will first investigate the existing technologies in the field of realistic and non-realistic traffic simulation. A traffic generation model will then be presented and implemented. This model will draw parameters from different network traces to build different files that can be used with the GHOSTS framework to simulate realistic traffic. Finally, several results will be presented followed by a conclusion that will highlight the different important elements to remember as well as the difficulties encountered throughout the implementation this thesis.

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor, Professor Wim Mees, and my co-supervisor, Professor Georgi Nikolov. For their continuous support and valuable advices provided throughout this work.

Secondly, I would like to thank my family and friends for their support, especially Matteo Snellings and Arnaud Demeure who were often there to help me in any way they could.

Finally, I would like to thank the institutions that organise the Master in Cybersecurity for allowing free access to a lot of scientific resources that have been indispensable for the research related to this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

THE advent of the Internet is certainly one of the greatest revolutions of the end of the 20th century. Today, it has become an essential pillar in the information society we live in, connecting billions of people together and allowing the use of a multitude of different applications and protocols that are used in almost every industry. From the year of its invention by the ARPA (Advanced Research Projects Agency) in 1981 until today, the number of connected devices has never stopped growing, from 213 in 1981 [1] to over 4.9 billion [2] in 2021 (≈62% of the world's population). This number keeps increasing with a forecast of 5.6 billion in 2025 [3].

Unfortunately, the development and the massive use of the Internet has led to the emergence of a new kind of threat, commonly known as the cyber threat. Given the growing number of users, the attack space also increases drastically from year to year. According to statista figures, the number of cyber attacks in 2019 exceeded 31.000 and the number data breaches with confirmed data loss rose to almost 4.000 that year [4] with an average cost of $4.35 million for each data breach [5]. In 2022 alone, the total cost of cybercrime has been estimated at $8.4 trillion and forecasts show a steady increase in these costs with an estimated cost of over $20 trillion by 2026 [6].

In addition to the financial cost, cyber attacks also damage the image and reputation of companies and slow down their growth and operation. A recent case is the attack on the Vivalia hospitals in Belgium. After realising that a virus had been introduced in the system for some time, all consultations for the following day were cancelled and the institution was operating without IT and therefore in slow motion. It took between four and six months to get back to normal [7]. The Vivalia case is far from being the only one, according to several sources, the healthcare sector is among the top five industries

targeted by cyber attacks [8].

In the light of these large and growing statistics, it was time to find effective long-term solutions to reduce the cyber threat. Training and preparation appear to be effective solutions, but obviously a single computer is not enough to simulate attack scenarios sometimes requiring several machines and several network devices. It is in this perspective of creating a real "virtual battlefield" that the Cyber Range was born. The indicators below are a visualisation of some of the statistics found in [9], these demonstrate the importance of training and preparation of security teams against the growing cyber threat.

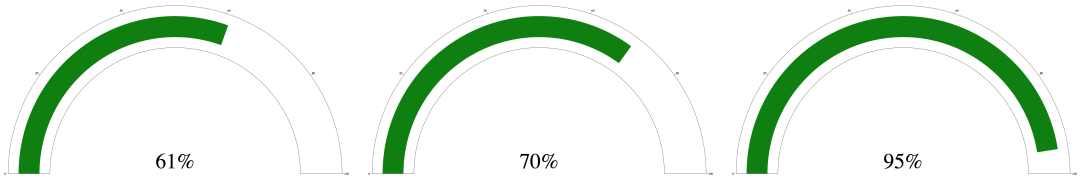


Figure 1.1: Statistics on (1) % of companies thinking their cybersecurity applicants **aren't qualified** (2) % of cybersecurity professionals claiming their organization is impacted by the cybersecurity **skills shortage** (3) % of cyber breaches caused by **human error**

1.1 Introduction to Cyber Ranges

Cyber ranges were first introduced to the military in response to the increase and variety of cyber threats. Over time, they have firstly become more and more sophisticated with the arrival of new technologies and secondly, become more and more popular as they develop in the *military*, *academic*, *commercial* or *private* sectors. Today there are many of them, some are even classified as confidential as they can be used as a military training area by some states or institutions. The Australian government survey [10] investigates the term "Cyber Range" and highlights some of them according to the area in which they are used.

There are many different but very similar definitions. Nevertheless, the one stated by the NIST (National Institute of Standards and Technology) seems to be the most complete.

Definition

Cyber ranges are interactive, simulated representations of an organization's local network, system, tools, and applications that are connected to a simulated Internet level environment. They provide a safe, legal environment to gain hands-on cyber skills and a secure environment for product development and security posture testing. [...] [11]

As stated in the NIST definition, CRs have been designed to be used in 3 main areas which are *training*, *research and development* and *testing*. However, the most popular and widespread activity in Cyber Ranges is training [10] (although they can support several activities, it is rare to see a CR set up only for use in one area).

We can define four types of CR which are *simulation*, *overlay*, *emulation* or *hybrid*. Each of these types has its own advantages and disadvantages and are designed for different use cases. The information below summarises what can be found on the subject in the articles [12, 10], the website [13] and the book [14].

Simulation-based Cyber Ranges

The objective is to replicate real world components (IT infrastructure) using virtualisation. The models created are high-level abstractions and will therefore not behave exactly like their real-world counterparts but will be very close. Software such as VMWare or its free equivalent Virtualbox, allow these types of Cyber Ranges to be set up relatively quickly. These virtualisation software packages offer tools that make simulation easier, such as snapshots and the possibility to create and customise virtual networks.

- ✓ Easy and fast to deploy
- ✓ Easy to add virtual components
- ✓ Highly scalable → Many VMs in a single device
- ✗ Lack of fidelity → High level abstraction of the real world devices
- ✗ Performance may be impacted depending on the host device

Emulation-based Cyber Ranges

The objective is to reproduce the behaviour of an IT infrastructure identically by knowing all the variables in advance (in contrast to the simulation which will extrapolate all the unknown variables). In order to achieve optimal realism, the emulated infrastructure is mirrored in an existing physical network. Hence, many elements can be added to extend the environment, such as DNS and/or log services. As with simulation, virtualisation is widely used in this type of CR.

- ✓ High realism and fidelity
- ✓ The same CR can be used for many different purposes and experimentation
- ✗ Can be very expensive for small organisations
- ✗ Need to set up and maintain dedicated devices and network resources
- ✗ Require software to configure the CR for each experiment

Overlay-based Cyber Ranges

The objective is to replicate an IT infrastructure as much as possible by being located on top of it. The only difference is that the Cyber Range does not receive real-time traffic. It is often used to add functionality to an existing service (e.g. the inclusion of the internet on top of the telephone service).

- ✓ High fidelity → True copy of the real infrastructure
- ✓ Cost advantage → No need for laboratories or dedicated testing facilities
- ✗ No formal testing → No repeatable tests and no control on the underlaying infrastructure
- ✗ Tests can cause disruptions to the underlaying infrastructure

Hybrid-based Cyber Ranges

As the name suggests, the objective is to take the best in each of the above categories.

- ✓ Flexibility to choose what you want and how to use it
- ✗ Difficult to integrate all the elements of the previous categories together
- ✗ Hard to deploy and maintain

1.2 The Royal Military Academy Cyber Range

The RMA Cyber Range (part of the RMA Cyber Defense Lab) is a simulation-based open source CR. This is the infrastructure that will be used in this thesis. All the code is available on GitLab and a rich documentation is available on the website. It has been designed for research and training in the academic and military sectors [15]. It consists of 3 essential elements which are a remote desktop gateway, an orchestrator and a hypervisor. The figure below comes from the survey [16] and represents the architecture of the Cyber Range. For ease of use, all CR VMs are accessible from a web interface.

Information

An *orchestrator* allows the automatic configuration, management and coordination of computer systems, applications and devices [17]. In this case it is the main component and serves several tasks.

- Provision the VMs (deploy images, customize the hardware and OS of each VM, install and configure additional software → IDS, traffic generator, vulnerable web server,...).
- Configure the different virtual networks.

- Create the required user account in the remote desktop gateway → the trainee can access their VM.

Information

A *hypervisor* is a software that allows you to create and run virtual machines while isolating the OSs and resources on the host computer [18]. In this case, the hypervisor is responsible for running the VMs and networks.

Information

A *Remote desktop gateway* is a software that allows a user to connect to internal network resources from outside a corporate firewall [19]. In this case, it is used to allow users to access the Cyber Range and use VMs.

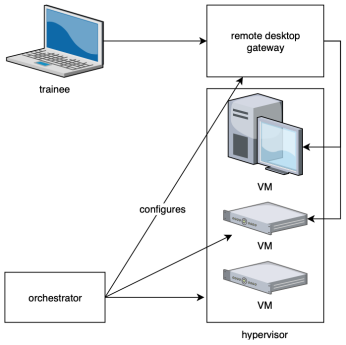


Figure 1.2: The RMA Cyber Range architecture

This CR was set up to improve Cyber Defense Situation Awareness (more information in [20]) which is one part of Endsley's decision making model (see figure 1.4). The scenarios used in this CR will therefore improve the *perception* - L_1 (ability to spot relevant elements in cyberspace), the *comprehension* - L_2 (ability to understand what is happening based on the elements identified) and finally the *projection* - L_3 (ability to predict what the elements identified will have as an impact) of the participants.

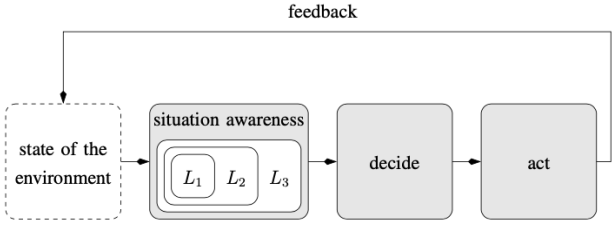


Figure 1.3: Endsley's decision making model

With good situational awareness training, they are able to make good *decisions* and *act* in the right way.

1.3 Introduction to traffic generator

The *perfect* cyber range must simulate different environments, all of which should be as close to reality as possible. In this way, during a training session, trainees are immersed directly in the heart of the situation and develop reflexes, methods and new reasoning to react quickly to any types of events during a potential real attack.

Limitation

If a CR has **no background traffic**. In this case the attacks will be directly detected because only the packets generated by the Red Team or by other people charged with the responsibility of being the "attackers" will circulate in the virtual network of the CR.

Limitation

If a CR has **always the same** (with a few variations) **background traffic**. In this case the exercises may be useful for the first few times but soon the trainees will start to understand the logic behind and the different patterns that the non-creative and non-random traffic generator generates. From that moment on, the training will be useless because the trainees will ignore the background traffic they know and see the suspicious packets directly (we come closer to the case where there is no traffic generator).

A good traffic generator (in the context of exercises in CRs) should generate background traffic similar to random internet traffic as a result of every day web activities. A tool like this is called a *higher fidelity traffic simulator* and it differs from a traffic generator in several ways.

Definition

Traffic generators simulate many devices on a network and the communication they would produce [...]. There are also higher fidelity traffic generators that, instead of simulating traffic, emulate user behaviour to generate real traffic from real applications. Emulated users are useful both for applied experiments and fundamental experiments to provide background noise for attackers and defenders. [21]

The main difference is that a *traffic generator* will simulate traffic whose content does not matter, the purpose being for example to test the maximum bandwidth of a link. In contrast, a *higher fidelity traffic generator* will emulate users in order to generate traffic as similar as possible to that generated by real users on the internet. The goal is for example to be used in a CR to train to spot attacks among traffic similar to the real world.

1.3.1 The Internet simulation problem

Although this tool is indispensable for the realism of the scenarios used in the cyber ranges, it remains a huge challenge today. [22] and [23] describe three reasons for this. The points below are mostly taken from these articles.

- Internet is a *immense moving target* → IP (Internet Protocol) architecture allows vastly different networks administered by vastly different policies to seamlessly interoperate. The more the internet interconnects small sub-networks, the more difficult it is to know its global behaviour.
- Internet is *huge* (↗ *heterogeneity*) → The more devices are connected, the more difficult it is to model all the different behaviours and protocols that can be found.
- The human *behaviour is evolving* → The functioning of the Internet depends directly on the behaviour of its users. It is constantly changing according to elements that cannot be emulated (e.g. the mood or desire of users) → difficult to emulate users behaviour on the internet over a long period of time.

Fortunately, in the case of traffic generation in a Cyber Range, not all the problems of simulating the Internet need to be addressed. Indeed, the problem is not to simulate the whole internet, but only a small part of it, which consists of the incoming and outgoing web traffic from a certain predefined infrastructure. Therefore, the first two problems do not arise. Nevertheless, there remains the difficulty posed by the last problem which is that human behaviour evolves very quickly and constantly. However, there are tools and methods to get around this problem and still generate human-like traffic.

1.4 Objective of the Master Thesis

The main objective of this thesis is to develop a web traffic generation model, also called a "traffic generator". This one will consist of two parts that work together to generate traffic in the most realistic way possible. The following figure shows the complete architecture of the model.

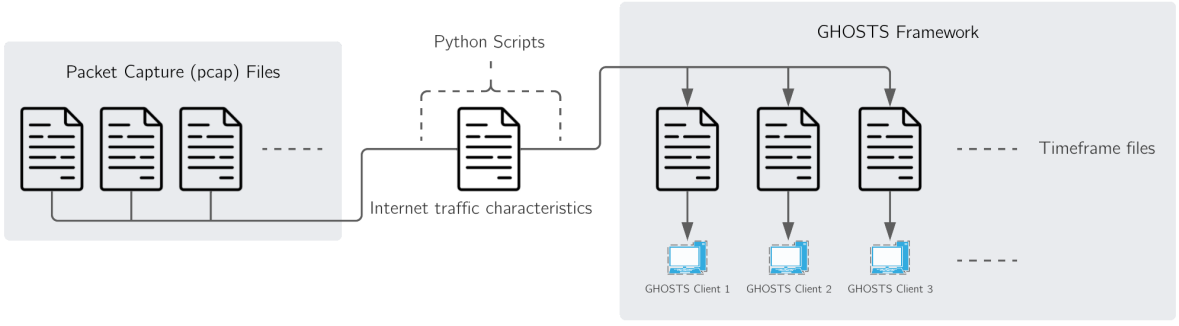


Figure 1.4: Internet traffic generation model architecture

The first part of the development called "Information extraction" consists in creating a python program that will extract in a JSON file (*Web traffic characteristics* file) the

main characteristics of one or several *packet capture* files. These characteristics will be discussed in chapter 2.

The second part of the development called "Information spreading" will also consist of building a python program with the difference that this one will distribute the information contained in the *Web traffic characteristics* file in one or more *timeframe* files. These are part of the GHOSTS framework (see section 2.3)

The goal of this work is therefore twofold. Firstly, a script must be designed to automate the extraction of relevant data on user behaviour on the Internet in pcap files. And secondly, `timeline.json` files usable in the GHOSTS framework must be generated automatically according to the extracted relevant information.

CHAPTER

2

LITERATURE REVIEW

BEFORE starting the design of the model, it is important to highlight some theoretical points on the subject that can be very useful. The aim of this chapter is to carry out a research study in order to firstly understand what has already been done in this field in the past. And secondly, to discover and study different parameters and methods that will be useful when designing the model in chapter 3 of this thesis.

Three important parts will be developed in this chapter. The first part will highlight different traffic generation technologies. The second section of this chapter will discuss the important characteristics of web traffic, which will be important when designing the first part of the model (Information extraction). Finally, the last section will present and detail the GHOSTS Framework which is the main component of the second part of the model (Information spreading).

2.1 Traffic generator technologies

There are many different traffic generators and technologies used to generate internet traffic, it's a developed area with different solutions offered by different companies or by different academic institutions. With a simple internet search you can already find sites, for example [24, 25, 26, 27], making rankings or lists of different traffic generators. The IEEE search engine can also serve as a proof of popularity for this technology with 1549 different articles published between 2000 and 2022 dealing with traffic generators.

As introduced in the first chapter, these traffics generators can be classified into two categories (see section 1.3). Firstly, *traffic generators* and secondly, *high fidelity traffic generators*. The difference between these two categories is not strictly marked in the

literature. However, this technology is called "traffic generator" to recall its primary function which is to generate traffic (realistic or not).

The following sub-sections will explore the technology in its entirety (without distinguishing between the two categories of traffic generators mentioned in the previous paragraph). As stated in the first paragraph, this is a fairly broad area. For convenience, the different categories of realism established in a study made by Vincent H. Berk, Ian Gregorio-de Souza and John P. Murphy [28] will be used. Existing examples of technology will then be cited and explained in each type of realism. Finally, a summary table will be presented, which will contain all the examples mentioned as well as their category of realism in addition to whether or not they are usable in a cyber range.

Since traffic generators are tools acting in the networking area, the OSI Model is very useful to visualise their operating layers. In addition, There is a logical link between the layer of application of the traffic generators in the OSI model [29] and their classification in the 3 different categories of realism. For these reasons, this section will begin with a quick review of this model.

2.1.1 The OSI model

The OSI (Open System Interconnection) model allows the separation and understanding of the communication processes between two devices in a network. Each layer supports the one above and provides services to the one below. The following figure illustrates the communication between two computers through the model layers. Finally, the higher up the layers of the model you go, the "closer" the protocols used will be to the users.

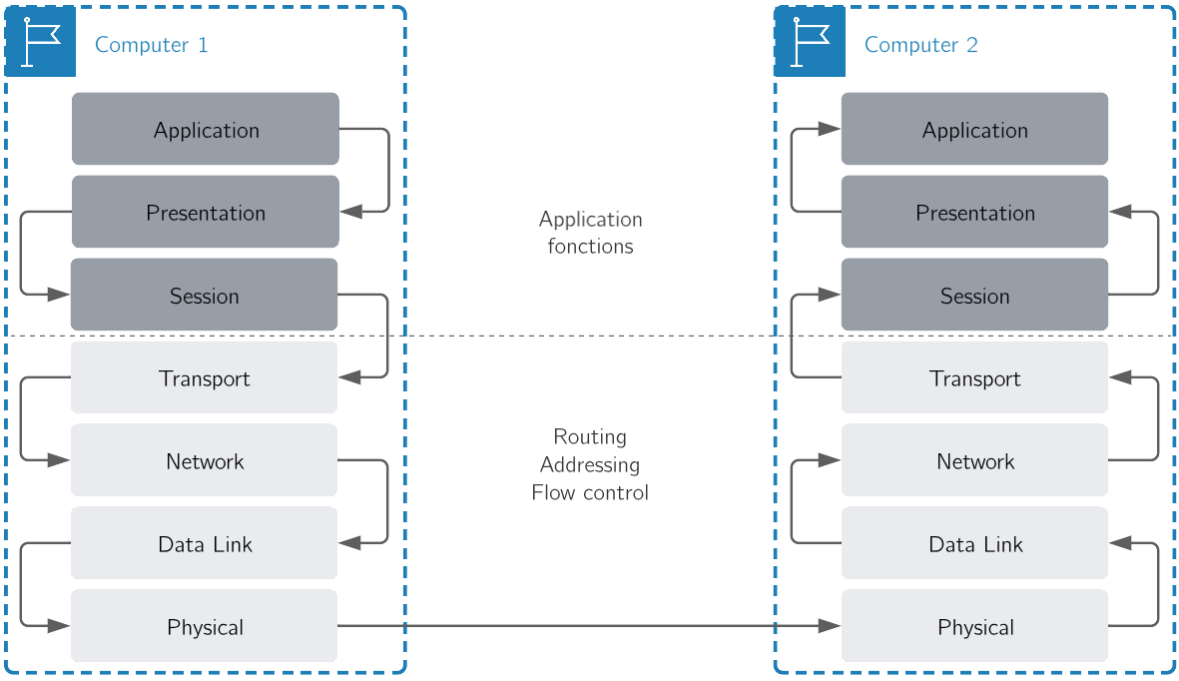


Figure 2.1: Communication between two computer through the OSI model

To facilitate the presentation of the different categories of realism, the model can already be divided into two main parts [30]. The first part (in dark grey on the figure) consists of the three highest layers and are responsible for the execution of specific tasks for each application. Some of them, for example, will use data encryption or will require a special format to process the data. The second part (in light grey on the figure) consists of the four lower layers and performs actions completely independent of the applications running above it. These actions are for example end-to-end data transport, routing, etc. The higher you go in the layers of the model, the more difficult it is to generate traffic that corresponds to that layer.

2.1.2 Statistical realism

The traffic generator types in this category are designed for the purpose of *infrastructure load testing*. They operate mainly on the first six layers of the OSI model and will therefore generate traffic consisting of TCP packets (or other protocols lower in the model). Only the headers are important for the use of this type of realism, which is why in many cases the payload of the generated packets is randomly determined or contains only the same bits.

Typical detailed examples of uses are (but are not limited to) :

- DOS training and experimentation : to artificially generate denial of service attacks, it is not necessary to have a level of realism in the traffic generated. The goal being to saturate a service in order to make it inaccessible to legitimate users, only the generation rate is important.
- Routers & switches behaviour : these network devices operate at layer 2 for switches and layer 3 (or 4 for advanced models) for routers. they are not designed to read the information contained in the higher layers. Testing them with packets generated above the transport layer would be useless.
- Bandwidth, delay and loss ratio testing : for similar reasons to the previous point, in addition to the fact that such characteristics do not depend on the realism of the packets, but only on the statistical measurements made during the generation and reception of the packets.

In consideration of the definition and the use made of the technologies belonging to this category of realism, they can be assigned to the class of **traffic generators**.

Existing technologies

A good first example is Ostinato [31]. This paid traffic generator can simulate synthetic traffic or can be used as a replay engine. In the first case, it can simulate customisable traffic from layer 1 to layer 4. It is equipped with a intuitive graphical interface that makes it easy to use. The uses of Ostinato are numerous but remain in the lower part of the OSI model. These can be found on the website but some examples are L2/L3

forwarding, load balancing or even storm control.

Other examples similar to Ostinato are described in several articles. First, the study [32] presents three other technologies which are PackETH [33], Iperf [34] and D-ITG [35]. These other examples do not operate higher than the transport layer (the fourth layer of the OSI model) and are, like Ostinato, used primarily for the purpose of measuring network performance. Other characteristics of each of these technologies are detailed in the first article cited at the beginning of this paragraph.

A second article, [36], presents a classification of different traffic generators according to 5 different categories. These have been established according to *the metrics used in their validation perspective*. For this class of realism, we can classify several traffic generators from three different categories of the study.

- Firstly, the *replay engines* category, we find technologies such as TCPReplay [37] and TCPivo [38]. These replicate packets previously captured by a packet collection engine (like tcpdump). We can't really assign them an OSI layer of reference because it depends on the content of the replicated files. Nevertheless, these two specific examples are used to test the performance of the network (the lower layers of the OSI model). Although they are similar, the article [38] points out that most of the performance issues not addressed in TCPivo are not addressed in TCPReplay.

- The second category is the *maximum throughput generators*. Here we find technologies such as Ostinato and Iperf described above, but also BRUTE [39], BRUNO [40] and KUTE [41]. BRUTE and BRUNO are very similar traffic generators because BRUNO (BRUte on Network prOcessor) is a modified version of BRUTE (Browny and RobUst Traffic Engine). The latter has limited performance due to the limitations of the computer architecture. BRUNO was developed to counter this limitation. Firstly, by using a classic PC architecture running BRUTE to handle the GUI and other high-level generation parameters. Secondly, by using a network processors (in this case the INTEL IXP2400 [42]) to generate the traffic in order to avoid problems that could occur on a standard architecture such as the maximum number of flows that can be generated simultaneously. KUTE (Kernel-based UDP Traffic Engine) is a kernel-based technology which will have a focus on two important aspects which are packet throughput and inter-packet time accuracy (both in terms of sending and reception of packets).

- Finally, the third category containing the *model based generators* describes the mathematical models that can be implemented in order to generate traffic at the flow level. Well-known examples are typically the Poisson based model [43] or the model based on the concept of the packet train [44]. These models are outside the scope of this thesis because they are often the subject of complicated mathematics not related to the objectives of this document.

As can be seen, there are many examples of traffic generators existing in this category of realism. This can be explained by the fact that the higher the realism, the more

difficult it is to get corresponding traffic. And the statistical realism category is the lowest of the three.

State of the technologies in this category

Most of the technologies mentioned in this subsection are stateless. They have no use in retaining the state of the traffic as they are used to produce it as quickly and accurately as possible without dealing about creating certain patterns or features that would require to remember the state of the traffic.

2.1.3 Content realism

The traffic generator technologies in this category are designed for *offensive operation training*. They will therefore be used for data exfiltration exercises, privilege escalation or any other type of exercise where the value of the packet information is not important. This category will already reach a high level of realism because traffic generators produce packets at the application layer of the OSI model (the highest layer). The access to this layer will allow to create realistic traffic at a given time but which does not evolve.

The biggest drawback of this level of realism is that, even if the contents of the packets are realistic, they do not reflect human behaviour. Hence, we quickly come to the second limitation on traffic generators stated in section 1.3 of the introduction chapter. This one puts forward the fact that the generated traffic must not be always the same otherwise the exercises become repetitive and less efficient.

Typical detailed examples of uses are (but are not limited to) :

- Data exfiltration exercises : traffic generation for this kind of training does not need to evolve over time and reproduce human behavior. The goal of the exercise being to get data out of a system, a simulation with a predefined scenario can be set up in order to generate for example fake emails, files, as well as internet requests to add realism to the exfiltration and make the task of detecting the exfiltration more complicated.

- Privilege escalation experiments : for this type of exercises, it is necessary to have working services in the Cyber Range. For example, users sending HTTP requests with working links or mails via SMTP with real addresses, ... The goal being for example to intercept some packets in order to recover useful data to continue the exercise.

In consideration of the definition and the use made of the technologies belonging to this category of realism, they can be assigned to the class of **high fidelity traffic generators**.

Existing technologies

A first existing technology is a model explained in [45]. This was developed by Chitra Javali and Girish Revadigar for the purpose of generating web traffic for use in exercises in a Cyber Range. This is the only work found to be the closest to the objectives of this thesis. In this study, the authors assumed that *generally a user browsing session consists of a sequence of webpages visited in a specified interval of time*. They also defined two different types of session which are *goal-oriented* and *general browsing*. Then, based on these assumptions, the researchers classified different user web sessions from a dataset with 1 million entries. The classification was based on a threshold (how many times the user has visited a page on the same subject in the same session) to define whether the session had a goal-oriented or general browsing purpose (in order to keep only the goal-oriented ones for the model construction). Finally, a model is built on the basis of a Markov chain (stateful traffic generator) representing the probabilities of arriving on a certain page according to the previous one visited.

This model meets the requirements of content realism because the traffic generated will be the same as that of a user visiting these different websites. However, to be classified in the higher category of realism, the model should incorporate events corresponding to human behaviour. For example, visiting certain pages at certain times or automatically refreshing popular web pages (in this case the dataset used to train the model is static and does not evolve unlike the websites visited which change often and quickly).

Another example is SWING [46]. It is a traffic generator based on learning certain characteristics from network traces and then generating traffic at the lowest level. The data retrieved by SWING can be classified into 4 categories, which are firstly user data. That is, connection times, activity distribution, etc. Secondly, session information, e.g. all data related to the access to a certain web page by a user. Next comes the category containing data about connections within a session, typically the size of packets or the number of request/response pairs. Finally, the last category groups together information related to the network, which is therefore the lowest level. They are, for example, the link loss rates or the latency. The higher the data (users or sessions), the more difficult it is to obtain and interpret them to recreate realistic behaviour. Once all this data has been collected, SWING will generate traffic packet by packet by considering the best possible behaviour of the previously analysed traces.

State of the technologies in this category

Most of the technologies mentioned in this subsection are stateful. In order to generate high-level traffic, these technologies must necessarily remember the state of the connections as well as the state of the sessions previously initialised. For example, generating a TCP session requires retaining the state of the connection as it is a stateful protocol.

2.1.4 Behavioral realism

The very few existing traffic generators in this category are used for the purpose of *intelligent analysis training*. This is the hardest level of realism to achieve because it is about emulating human beings to reproduce their behavior on the Internet. This category is in fact an extension of the previous one (content realism). In fact, the technologies will also generate packets at the application layer but will do it more intelligently by imitating human behavior in an autonomous way.

Typical detailed examples of uses are (but are not limited to) :

- Insider threat detection : this type of exercise requires the traffic generator to simulate different human behaviors including malicious ones. The generated traffic must be realistic and intelligently put on the network to make the task difficult and interesting for the trainee(s).
- Simulation of everyday attacks : with the possibility to emulate human behaviors on the internet, it is possible to create any situation of the daily life starting from a few parameters (time of the day, day of the week, environment, ...). The technology thus allows to create or recreate attack scenarios in order to train or prepare teams for all types of situations.

In consideration of the definition and the use made of the technologies belonging to this category of realism, they can be assigned to the class of **high fidelity traffic generators**.

Existing technologies

There is very little technology that can generate this level of realism. However, after much researches, two tools have been found. One is called TRex and is free but difficult to use and the other called LARIAT has a restricted use but is much easier to use.

TRex [47] is a free tool developed by Cisco for generating traffic at all layers of the OSI model. It does not have a graphical interface and requires good programming skills to be used effectively. The traffic is completely customisable and can therefore generate human traffic if used in this way.

On the other hand, LARIAT [48, 49] (Lincoln Adaptable Real-time Information Assurance Testbed) is a technology developed by the MIT Lincoln Laboratory to generate realistic background user traffic and real network attacks, verify attack success or failure, score ID system performance, and provide a graphical user interface for control and monitoring. In the case of traffic generation, the tool offers a graphical interface and the choice of a profile that is pre-configured with information gathered from traffic generated at a US military base. The generation is based on the probability that a user has to perform a specific action at a given point in time.

2.2 Main characteristics of the Internet traffic

The beginning of the information extraction phase in the design of the realistic traffic generation model requires analysing and selecting the most appropriate features in the pcap files. This data will allow the modelling of the previously analysed traffic at several levels. That is, from the physical level to the application level. The use of some of these data coupled with the GHOSTS framework will allow to emulate one or more users (see section 2.3.2).

The extraction of the characteristics will be done in two steps. In the first step, all the data that are relevant to a certain level of realism (as seen in the previous section) and that can be more precisely associated with a certain layer of the OSI model will be itemised. In a second step, all the data that fall under the global traffic analysis (that do not fit into a specific realism category or layer of the OSI model) will be detailed as well. Finally, a summary table will highlight all the characteristics mentioned as well as their belonging (or not) to a category of realism and to a layer of the OSI model.

2.2.1 Statistical realism characteristics

The characteristics belonging to this category of realism come from the first 6 layers of the OSI model. In short, this data will allow the correct modelling of packet data exchanges without considering their content. Each of the first 6 layers is listed below along with their specific characteristic(s).

Physical layer

This first layer of the model supports the transmission of bits between the different network devices. The main characteristic of this layer is the **bit rate**. It is interesting to obtain several variations of this data, typically its general average and its evolution in a given period.

Data link layer

This second layer of the model supports the transmission of frames (each consisting of multiple bits) between two network devices physically connected together. The main feature of this layer is the **MAC address**, which is unique to each network card and serves to identify a certain machine. Unlike IP addresses, the MAC address never changes and is therefore a good way to identify a particular user regardless of the IP address he uses.

Network layer

This third layer of the model supports the routing of packets (each consisting of multiple frames) between network devices. The first two characteristics of this layer are the **source IP address** and the **destination IP address**. The source address makes it possible to know more about the users, in particular if they are in the same subnet. The

destination address, on the other hand, makes it possible to have information about the activities of the user because after resolving this address, we can obtain a domain that can be linked to a website or something more precise than an IP address.

Other interesting characteristics to note are those related to packet statistics. These include the **packet rate**, **packet size**, **inter packet time** and the total **number of packets exchanged** (counting those sent and received). Like the bit rate, it is useful to get several variations of this data such as the average value and the evolution over a certain period of time.

Transport layer

This fourth layer of the model allows an en-to-end connection between two devices. This is also where error handling is managed. The two important elements to extract from this layer are the **source port** and the **destination port** of each packet. The source port allows a distinction to be made between the different services that a user uses. Indeed, each service is assigned a new port that is not already assigned. All packets with the same source port can therefore be grouped together because they are linked to the same service. On the other hand, the destination port will allow to know the nature of the protocols used higher in the application layer (each protocol having one or more defined ports).

Another important characteristic to extract is the **round trip time** to reveal delay in a network. The evolution of this data over a period of time can reveal when the network or certain servers are saturated.

Session layer

This fifth layer of the model provides a communication channel between two devices for a defined service. This channel is called a session and is defined by a unique unused port number on the user side. The main elements of a session are the **start time**, the **end time** and the **total duration**.

Presentation layer

This sixth layer of the model allows the data to be formatted so that it can be correctly read by the application layer. It is also where the potential for data encryption is handled. No relevant characteristics specific to this layer can be extracted.

2.2.2 Content realism characteristics

The characteristics belonging to this category of realism come from the seventh layer of the OSI model. In short, this data will allow the content of the packet to be correctly modelled, regardless of how and when they are sent. The seventh layer is listed below along with its specific characteristic(s).

Application layer

This protocol is used directly by the software. The protocols located at this level allow the use of different services by the user. If the data is not encrypted, it is interesting to extract the content of the exchanged e-mails, the URLs of the visited websites or the content of the exchanged files. In short, all the data contained in the payloads of the packets is interesting. Nevertheless, it is obvious that the extraction of this data is a problem because it is an breach of the user's privacy. In addition, most of this data is encrypted and therefore unreadable.

However, there are still some interesting high-level data to be collected such as the **times when the user is active**, a **list of activities** done on the Internet or the **time between each activity**.

2.2.3 Global traffic characteristics

Among the characteristics that cannot be linked to one or another layer of the OSI model are the **number of different domains contacted** and the **number of sessions initiated** by each user [28].

2.2.4 Behavioral realism characteristics

The characteristics belonging to this category of realism does not correspond to a particular layer of the OSI model. The aim is to analyse the behaviour of a user at an even higher level. This is a subjective task because a user's behaviour can be interpreted in many different ways.

A lot of research has already been done and is still being done on the modelling of humans on the internet. There are many issues at stake, such as targeted advertising, effectiveness evaluation [50] or even health psychology [51]. However, it remains a great challenge to accurately model user behaviour. An excellent article by G.I. Webb, M.J. Pazzani, M.J. and D. Billsus [52] details 4 main reasons why the problem persists.

- The first reason is **the need for large data sets** : to get an accurate emulation of a user, you have to start by learning as much as possible about their online activities. This means collecting as much data as possible, as presented in the statistical realism characteristics section, in different time intervals. The more data collected, the more realistic the simulation will be. It is important to have data in all possible time intervals in order to know the activities throughout the year, throughout each day. The only way to get this much data is to monitor individuals 24/7 and this is unfortunately very difficult, if not impossible to achieve on a large scale. As is well stated in the article lastly cited : *"the learning algorithm does not build a model with acceptable accuracy until it has seen a relatively large number of examples"*.

- The second reason is **the need for labeled data** : it is very difficult from the data collected to know the impression of a user when he is on a web page or when

he is doing any other activity on the internet. This data is however useful to make predictions (if a user did not like a content or an activity, it is unlikely that he will do it again) and therefore build an accurate model.

- The third reason is the **concept drift** : this issue is related to the one stated in the section 1.3.1, i.e. human behaviour changes over time. Algorithms must therefore be able to adapt to many situations quickly. This ability for an algorithm is called the *drift concept*.

- The fourth reason is the **computational complexity** which can be very high when the model becomes complex. However, a human emulation model is by definition complex because there are many factors to take into account.

As the subject of emulation is very complex and the aim of this thesis is to take a first step into the field (in order to generate realistic traffic), complex theories (such as graph theory) or difficult to extract features to build mathematical models to emulate users will not be discussed. Instead, some features whose extraction has already been discussed in the previous two sections will be used to make interpretations about the behaviour of a user at a given time. These interpretations will allow to build a simple, but accurate model of a user at a given time.

Detection of a user's internet use

The first things to define are the times of day, week or year when a person will use the internet. This information can be obtained by observing the evolution of the **packet rate** or the **bit rate** over time. A high packet rate or bit rate at a given time with a certain device as source or destination means that the person is using the internet for various activities at that time.

However, a computer generates traffic at all times to perform various background tasks. If we only look at the packet rate, the distinction between the actual use of the Internet by a user and the various tasks performed by the device that also require the Internet will not be made. To counter this problem, an analysis of **ports**, **protocols** and **IP addresses** can be done to link each packet to a known application and thus identify the activity. For example, we know that the Microsoft Teams application uses UDP ports 3478 and 3481 for *live content* and TCP ports 80 and 443 for other content (list of the TCP port descriptions on this website). To distinguish Microsoft Teams from other applications using TCP ports 80 and 443, the destination IP address will be resolved to find a domain name that is or strongly resembles *microsoft.com* (the domain name can also be found in the TLS "Client Hello" packet, if this protocol is used).

If a match is made between all ports and addresses in order to discover the resulting activity, then it will be much easier to classify the activity as being voluntarily performed by a user or involuntarily performed and therefore considered as background noise.

In the case of the web (ports 80 and 443), which is more than 70% of the total traffic on internet [53], the goal will be to find the packets corresponding to the "Main Object"

[53]. That is to say the first file sent when doing a search on the internet. This is the same file that will contain the potential other servers to be contacted to fully load all the elements of a web page. The isolation of this main object makes it possible to know precisely when a user has made a certain search.

Isolation of the periods of use

Isolating the different periods of use will make it possible to know more precisely than the packet rate or the bit rate when a user has used the Internet. These periods of use are obtained firstly by keeping track of each start of activity. Then, if the difference between the start of an activity and the start of another activity is smaller than a certain threshold, then these two activities are considered to be part of the same period of use. Otherwise, if the difference between two activity starts is greater than a certain threshold, then the end of the period is marked by the end of the first activity, and the start of the next period by the start of the second activity.

Isolation & identification of activities

After determining more precisely the moments of Internet use, the packets corresponding to the same activity will be gathered in a session. Characteristics such as **start time**, **end time**, **total duration**, **domain name**, **IP address**, **port(s) used** and **statistics on the bits and packets exchanged** are collected to learn more about each activity. Isolation of an activity is done by gathering all packets to or from the same IP address and with the same port.

Timeline & Markov chain construction

The last step in data extraction is the construction of a timeline and a Markov chain. These elements will respectively allow to trace the course of all the activities of a user during the whole period of time analysed and to be able to predict which activity a user will carry out according to the previous one he has carried out. It is not uncommon to find Markov chains in traffic generators, notably in LARIAT and SWING which were described earlier in this thesis.

Grouping of activities

So that the traffic generation is not too similar to the analysed traffic but still represents the user's behaviour. The different activities will be assigned to a category that will represent the overall objective of a user at that moment. This idea is inspired by the "goal-oriented" sessions from [45]. For example, a group called "Social Networks" could typically contain :

- `www.facebook.com`
- `www.instragram.com`

- www.tiktok.com
- ...

If we notice that a user goes every Tuesday morning between 7.30 and 8 am on www.facebook.com or www.instragram.com, it will not be out of context to generate traffic to www.twitter.com during this period, when emulating this user (because twitter.com could also belong to the group "social networks"). Coupled with the Markov chain. These techniques will allow to generate traffic that is statistically realistic compared to what the user is used to do and in a realistic order also in view of the previous probability analysis in the Markov chain.

2.2.5 Summary table

The figure below is a summary table of all the characteristics to be extracted. These are classified according to their realism category and their level in the OSI model. It should be noted that the behavioral category of realism does not really have its own characteristics but will rather use the characteristics of the other categories in order to learn precisely the human activity.

Realism category	OSI layer	Caracteristics
Statistical realism	Physical layer	Bit rate
	Data link layer	MAC address
	Network layer	Source IP address
		Destination IP address
		Packet rate
		Packet size
		Inter packet time
		Number of packet exchanged
	Transport layer	Source port
		Destination port
		Round trip time
	Présentation layer	/
Content realism	Application layer	/
/	Global	Number of different domains contacted
		Number of sessions initiated
Behavioral realism	/	Detection of a user's internet use
		Isolation of the periods of use
		Isolation & identification of activities
		Timeline & Markov chain construction
		Grouping of activities

Figure 2.2: Summary table of all main characteristics of the internet traffic

2.3 The GHOSTS Framework

General HOSTS or GHOSTS framework is a tool *to create a high level of realism in cyber exercises by establishing and building behaviorally accurate, autonomous non-player characters (NPCs)* [54]. The framework perfectly addresses the problem posed in section 1.3, which is that generated background traffic must be similar to random internet traffic as a result of every day web activities. Indeed, [55] outlines the fact that the result of using GHOSTS is *a training experience that looks just as real as what cyber teams might see during normal operations.*

The framework allows you to create one or more NPCs (non player characters). These will emulate human behaviour by following a precise list of instructions contained in a script called TIMELINE.JSON (there are other configuration files for more advanced uses of the framework). The possibilities of activities are relatively numerous and can be personalised according to several parameters. The table 2.3 below comes from [54] and shows the different actions that NPCs can perform.

One of the great advantages of this framework is that it takes care of all the first two types of realism (statistical and content realism). In other words, having told it at a very high level what actions should be performed and in which way, GHOSTS will take care of the low-level traffic generation. And this, as if it were a human controlling the machine.

Capabilities	User actions	Execution methods
Web browsing	Browse	Random
	Enter text	Specific
	Click link or button	Looping
Terminal commands	Execute cmd commands	Random
	Execute PowerShell commands	Specific
		Looping
Inter-NPC communications	Email creation and management	Specific
		Looping
Office document management	Common file format for word processor	Random
	Spreadsheet	Specific
	Presentation documents	Looping

Figure 2.3: Table from [54] showing the different possible actions of the NPCs and their execution mode

All these customizable capabilities are previously entered into a json file which will then be read and the resulting actions will be executed by the NPCs. The website [56] contains a tutorial as well as a non-exhaustive list of some basic parameters that can be used in the `timeline.json` file. Here are some of them.

- Capability :
  - ❖ HandlerType → What type of action will be performed

- ❖ Initial → The first web page that will be opened when the browser is launched (Web Browsing)
- ❖ UtcTimeOn / UtcTimeOff → The time interval during which the browser will be launched

- Execution Method :
  - ❖ Loop → Loop through the events
  - ❖ Specific → Execute a specific event
  - ❖ Random → Execute the events randomly
- User actions :
  - ❖ TimeLineEvents → Groups together all the different events. Each of them has different parameters
    - ∗ Command : What will be executed
    - ∗ CommandArgs : The potential arguments that must be passed to the command
    - ∗ DelayAfter / DelayBeofre : The time delay for Before and After the command

Here are some examples of `timeline.json` files using the parameters mentioned and explained above.

```
{
    "TimeLineHandlers": [
        {
            "HandlerType": "BrowserFirefox",
            "Initial": "about:blank",
            "UtcTimeOn": "12:00:00",
            "UtcTimeOff": "12:30:00",
            "Random": "Loop",
            "TimeLineEvents": [
                {
                    "Command": "browse",
                    "CommandArgs": [
                        "https://www.google.com"
                    ],
                    "DelayAfter": 30000,
                    "DelayBefore": 0
                },
                {
                    "Command": "browse",
                    "CommandArgs": [
                        "https://www.facebook.com"
```

```
                    ],
                    "DelayAfter": 30000,
                    "DelayBefore": 0
                }
            ]
        }
    ]
}
```

Following this script, the NPC(s) will, between 12:00:00 (UTC) and 12:30:00 (UTC), first open a blank page with Firefox and then directly open Google.com. After that, during the half hour of execution, it will open either Google.com and Facebook.com every 30 seconds.

```
{
    "TimeLineHandlers": [
        {
            "HandlerType": "BrowserFirefox",
            "Initial": "about:blank",
            "UtcTimeOn": "12:00:00",
            "UtcTimeOff": "12:30:00",
            "Loop": "True",
            "TimeLineEvents": [
                {
                    "Command": "browse",
                    "CommandArgs": [
                        "https://www.google.com"
                    ],
                    "DelayAfter": 30000,
                    "DelayBefore": 0
                }
            ]
        },
        {
            "HandlerType": "BrowserFirefox",
            "Initial": "",
            "UtcTimeOn": "20:00:00",
            "UtcTimeOff": "22:30:00",
            "Loop": "True",
            "TimeLineEvents": [
                {
                    "Command": "browse",
                    "CommandArgs": [
                        "www.facebook.com"
                    ],
```

```
                        "DelayAfter": 1800000,
                        "DelayBefore": 0
                    }
                ]
            ]
        ]
}
```

In this slightly more complex example, the NPC(s) will perform the same actions as the example above between 12:00:00 (UTC) and 12:30:00 (UTC). In addition, it will, between 20:00:00 (UTC) and 22:30:00 (UTC), he will carry out another period of internet use by going to the Facebook site every 30 minutes.

These two relatively basic examples demonstrate the huge range of possible scenarios that can be recreated with the GHOSTS framework. In this case, the timeline files are always static and must be configured in advance. More advanced uses of the framework allow the use of a server that will dynamically change the timeline files of the NPCs to further increase realism and avoid having to "hardcode" all behaviours in advance. The server will also take into account the notion of **knowledge horizon** so that the behaviours of the NPCs remain logical.

The knowledge horizon

This concept will ensure that in a large-scale exercise, each NPC knows what they are supposed to know and no more. This concept will ensure that in a large-scale exercise, each NPC knows what they are supposed to know and no more. The central server that will manage this notion will therefore retain what each NPC knows as the exercise progresses to not have behavioral contradictions. A typical example would be that an NPC deletes a file before even creating it.

This notion of knowledge horizon is important when a lot of NPCs are working together. Especially when exercises require several teams of NPCs (blue, red, ...).

The use of GHOSTS in this thesis will be limited to the internet search functionality, since the "browse" method corresponds perfectly to what a user does when visiting websites.

CHAPTER

3

METHODS

THE first step in the methodology is to obtain as much relatively recent data as possible about the activities of one or more users on the Internet. However, this kind of information is very often not shared by those who hold it. For both privacy and exclusivity reasons. In order to have quality content to present in this thesis, a network trace dated 21 April 2023 between 8h00 and 23h00 will be used as an illustration. This information was retrieved using the tcpdump command and is derived from the use of the internet to carry out various activities during the preparation of this thesis. The model that will be built on the basis of these data will therefore be able to generate traffic similar to the one generated by *a student working on his thesis on 21 April 2023 between 8am and 11pm.*

This chapter on methods includes a first large part that will describe precisely the python script responsible for the **information extraction** phase. This one aims at extracting the different characteristics highlighted at the end of the previous chapter. The second part will focus on the design of a script responsible for the **information spreading** phase. This will take care of the transposition of characteristics on human behaviour extracted and processed thanks to the first script in one or several `timeline.json` files. These can subsequently be injected into the GHOSTS framework to generate realistic traffic similar to that analysed earlier.

For practical reasons, the traffic was captured in 16 different files, each file contains approx. 1 hour of traffic. All the code is available on the Github repository and the 16 pcaps files are also available in the Cylab cloud.

3.1 Information extraction phase

This first section on the information extraction phase will itself be divided into four sub-sections. Each of them will reveal a little more about the user's internet usage and preferences. These four parts will be respectively the *detection of Internet use*, the *isolation of the periods of use*, the *identification of activities* and finally the *timeline & Markov chain construction*.

Finally, a last sub-section will follow to explain the extraction of more general characteristics related to the whole network trace and not specific to the user as it is the case in the first four sub-sections.

3.1.1 Detection of Internet use

As detailed in this section 2.2.4 in the previous chapter, there are different characteristics to be extracted and interpreted in order to find out when the user is using the internet and to be able to classify correctly when the user is actually doing activities, and when the computer is using the internet to perform other tasks autonomously.

At a very high level, a first useful visualisation is the user's packet rate. This parameter makes it possible to distinguish the moments of strong solicitation of the network and thus inevitably the use of the internet. The figure below represents this packet rate from 8h00 to 23h00.

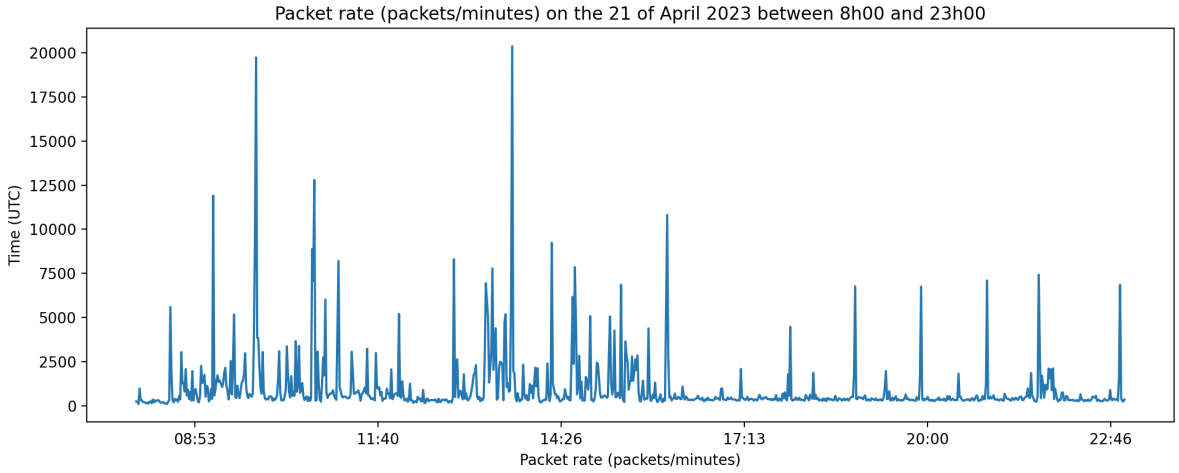
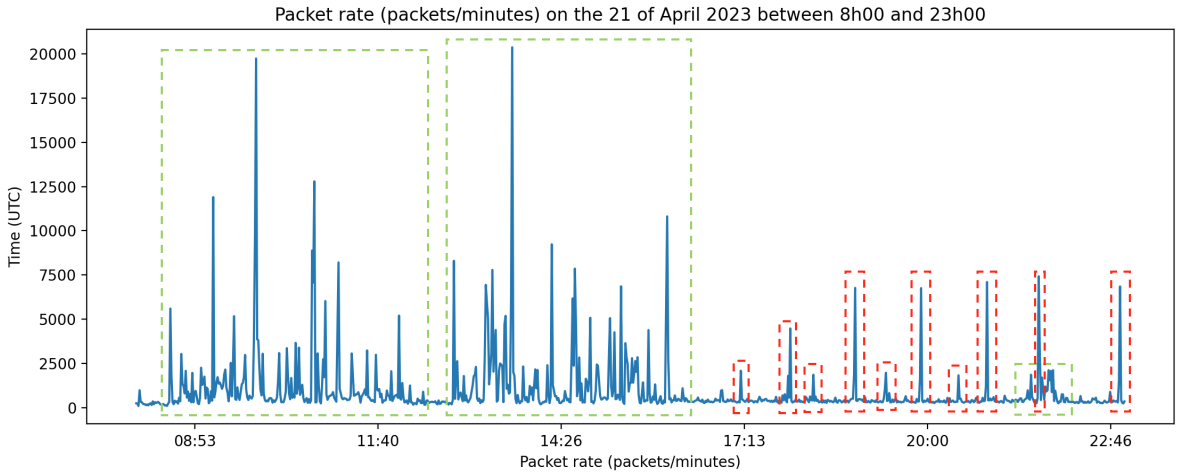


Figure 3.1: Packet rate in packets/minutes on the 21 April 2023

After a quick analysis, we can isolate three different areas with a very high packet rate. These most likely correspond to the time when the internet was used. The figure below shows the evolution of the packet rate, highlighting these areas with a high packet rate.



Packet rate in packets/minutes on the 21 April 2023

Since the trace takes the traffic of a student writing his thesis, we can make the link between the zones with a high packet rate (in green) and the study periods. The first two red areas probably correspond respectively to the morning study period from ±8h30 to ±12h00 and the afternoon study period from ±13h00 to ±16h45. The last small zone is probably related to the use of the internet for entertainment between ±21h30 and ±22h00.

some areas have been framed in red on the graph. These are times when there is a very large packet rate in a very short time. There is little chance that this is due to human activity as it does not resemble at all the previous green areas which correspond to user activity. We can consider that these red areas are background noise which can be an update or data that is sent to Apple's server (the capture was made on a MacBook Pro).

However, this background noise is represented by packet rate spikes which must also be found elsewhere in the graph and therefore pollute the areas where the user is using the internet. In order to precisely identify the periods of voluntary internet use, another feature will be extracted and interpreted. This is the main object rate.

Thesis limitation

Before continuing further it is important to clarify that this thesis is limited to user activity at the level of internet searches and therefore at the level of the HTTPS protocol on port 443. All other activities will not be covered in this thesis.

The main object [53] is the first HTML document that arrives at the user after the user has made a web request. It is always preceded by a TLS exchange (in the case of HTTPS) in order to make all the content of the communication secret. The main object should not be confused with the in-line objects which are all the objects containing the elements of a web page and which are contacted because they are mentioned in the main object document.

Possible improvement

An important enhancement to the model would be the addition of functionality to identify known applications based on the ports used by the packets. In order to not limit to web traffic as in this case.

In order to easily identify the main object, the focus will be on the TLS session that precedes it. Indeed, the first packet of the TLS exchange (the "Client Hello") contains information on the contacted IP address, including the domain name. This makes it much easier to access the domain name and avoids the need to resolve the IP address (a process that often leads to an exception). The graph below shows the number of websites contacted and therefore the number of hand objects obtained throughout the day by the user.

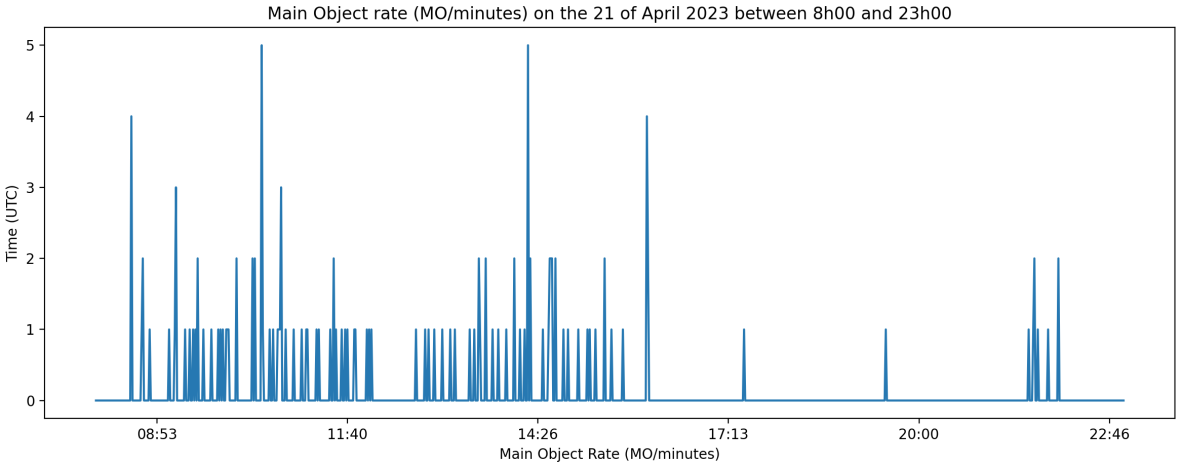


Figure 3.2: Main Object rate (MO/minutes)on the 21 of April

With this other figure, the zones defined in the packet rate analysis step can be confirmed and a precise start and end timing can be assigned to each zone. This timing is **very valuable information** for the simulation of the behaviour as it will allow to know the precise time intervals where the traffic should be generated.

3.1.2 Isolation of the periods of use

Now that the internet usage zones are confirmed, A more precise zoom in can be done to identify more rigorously small zones of use within these large periods. A good way to define the zones is by setting a threshold. This is the number of minutes that have passed without a new web site being searched by a user. After the last domain searched, if the threshold is exceeded, it is considered that the user is no longer using the internet or has left the device and the zone is therefore over.

The value of the threshold must be defined intelligently. Indeed, if it is too small, the zones may also be quite short and therefore very targeted on each domain search. This situation can be compared to overfitting in machine learning. That is to say that

the zones will resemble the analysed network trace too well and the simu- lation may therefore lack realism by resembling it too closely. Conversely, if the threshold value is too large, then the areas will also be very large. This risks encompassing too many moments of the day, including those where the user is not using the internet or is absent (moments that are not relevant for the future simulation). This phenomenon is similar to underfitting in machine learning, which roughly means that the model is too general and does not resemble the data provided. The figures below highlights these zones for different thresholds.

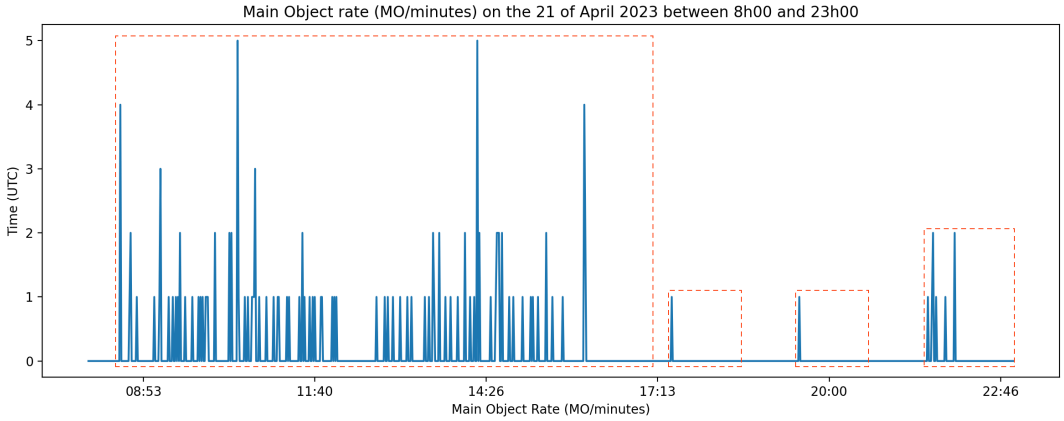


Figure 3.3: Zone detection with a threshold of 60 minutes

This first example shows a typical example of underfitting. The threshold is much too high and the areas are too large and do not accurately represent the times when the user is using the internet. Alternatively, the threshold value in this second example seems to fit the user's activity perfectly. No more relatively large spaces are visible in each zone and each zone contains a more or less significant number of connections to web pages.

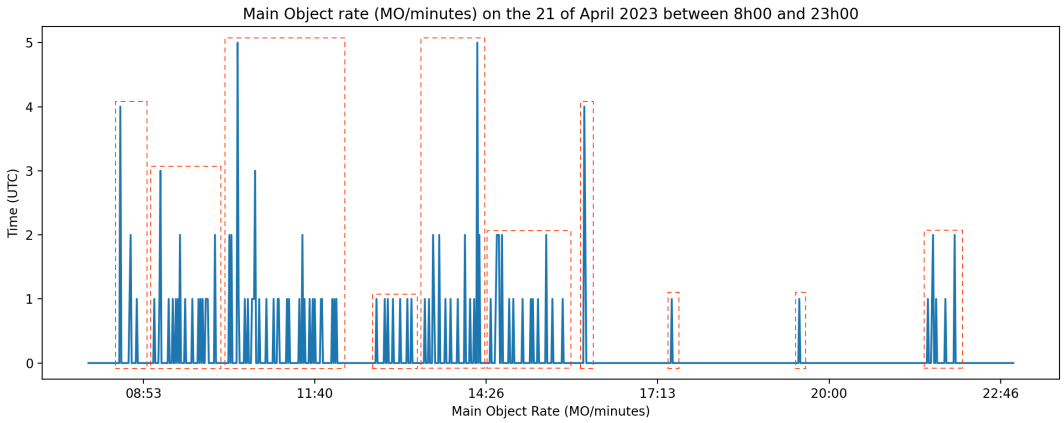


Figure 3.4: Zone detection with a threshold of 10 minutes

Once the zoning is more rigorously done, the intensity of Internet use in each zone can

be measured. Indeed, by counting the number of domains contacted in each zone, we can obtain the proportion of websites contacted in each zone compared to the total number of websites contacted, which is 129. Here is a pie chart visually representing the proportion of the area contacted for each zone. It shows the start and end times with the number of domains contacted and the proportion per zone.

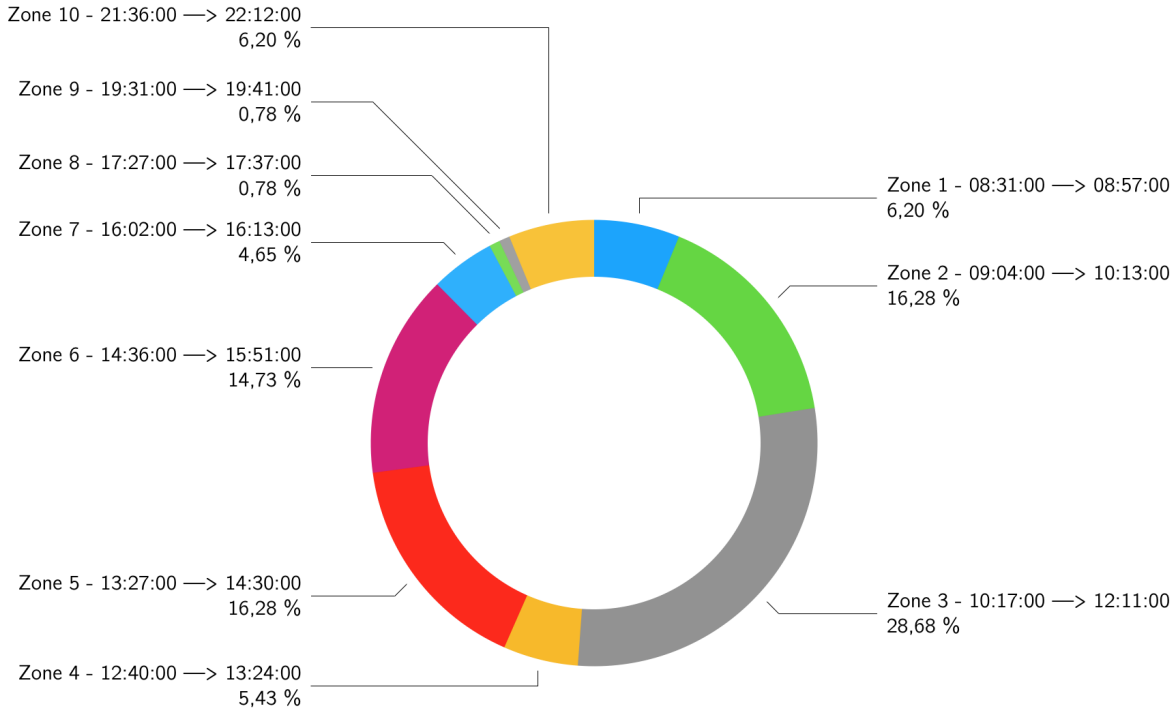


Figure 3.5: Proportion of domains contacted for each zone

Summary of the information collected in these two first sub-sections

This sub-section has allowed to get some important information about the user behaviour behind the network trace data. Indeed, the first visualisation allowed to roughly distinguish the areas of internet use. Then, with the help of other characteristics, the isolation in more precise zones could be done and we end up with a final result quite conclusive which dictates us the key moments in which it is necessary to generate traffic to simulate the human behaviour of this particular user on April 21st 2023.

3.1.3 Identification of activities

The next step in the analysis of human behaviour is to know even more about the user's activity. To do this, a zoom is carried out in each of the previously defined zones to find out how much time the user has spent on each website represented by the domain name he or she has used. This step is crucial because it will allow us to know more

about the preferences that the user had at that moment and therefore to be able to put forward similar preferences when generating traffic. The first part of this section will deal with the isolation of each activity and the second part will merge different activities together if necessary.

Distinction of activities

This is one of the most difficult characteristics to extract. You have to isolate each of the web sessions of each of the users. However, as explained very well in the article [53], an internet request does not only consist of a *"Main Object"*, but also of other *"in-line Objects"* which are located at different IP addresses from the location of the Main Object. In the case of the HTTP protocol, it is possible to easily find out which other domains have been contacted from a Main Object because the traffic is not encrypted and one can therefore read the payload data of the Main Object and discover the addresses. Conversely, this is very complicated to obtain in the case of HTTPS traffic because the communications are encrypted and it is therefore necessary to first decrypt and then read the contents of the payload. This operation is technical and takes time to set up. This is why, for the purpose of this thesis, another method will be adopted.

As in the previous sub-section, each user activity on the Internet will have a defined start from the first TLS *"Client Hello"* packet to the domain name of the activity. Then the IP address corresponding to that domain and the port associated to that session will be recorded and all packets going to and from that domain with the specific port will be retrieved for counting. The time of the last packet going to or coming from this same IP address will be considered as the end of the user's session on this website and therefore the total time spent on this activity will be calculated. Here is an overview of all the data that are collected for each activities.

```
# activity_dictionary = {
#   "81.243.1.152": {           The IP address
#       "62717": [              The port defining the session
#           "www.pathe.be",     The domain name
#           1682058672,         Time of start (seconds since epoch)
#           1682058675,         Time of end (seconds since epoch)
#           3,                  Total time (seconds)
#           12,                 Number of packet sent
#           15,                 Number of packet received
#           1551,               Number of bytes sent
#           6361,               Number of bytes received
#           7912,               Number of bytes exchanged
#       ]
#       "62718": [
#           "www.pathe.be",
#       ]
        ...
#   },
#   "2a00:1450:400e:80c::2003": {
#       "62721": [
#           ...
```

```
#         ]
#     },
#     ...: {
#         ...: [
#             ...
#         ]
#     }
# }
```

Merging activities

This technique works but has a flaw. Indeed, it can happen that browsers initiate several TLS sessions for a single HTTPS request. With the procedure explained above, several different activities could be created while the user may only have actually performed one. This problem can be solved by using an element that is already used in zone splitting, the threshold. If an IP address is contacted, an activity starts and is assigned to a port. Then, if after a short period of time, the same IP address is contacted but on a different port, then the packet should be considered as part of the same first activity and it is not correct to consider it as a new activity. This problem is the reason why many of the sessions can have a very short period of time (like in the example above) which certainly does not correspond to the actual time spent by a user on that activity. In order to solve the problem, all sessions whose difference between their start time is smaller than a certain threshold are merged together.

If two activities are to merge, then the start time of the first activity and the end time of the second will be kept, so the total duration will be recalculated to match again. All other characteristics are added together.

Possible improvement

An important enhancement to the model would be the addition of a functionality to link all *In-line Objects* to their corresponding *Main Object* and thus have a much more accurate idea of the duration of the different activities and the amount of data exchanged.

Summary of the information collected in this sub-section

This sub-section provided a lot of detail on the different activities that the user carried out during the period of time analyzed. It is now possible to know approximately how long the user(s) stayed on each of the visited web pages. In addition, other interesting information has been extracted from the sessions such as typically the start time, the end time, the number of packets received and sent and the number of bytes received and sent as well.

3.1.4 Timeline & Markov chain construction

The last step to complete in order to get a fairly accurate set of information about a user's behaviour is to track all these activities throughout the day. From the complete list of activities explained in the previous sub-section, each starting time of each activity will be isolated in order to build a timeline over the whole period analysed.

With the help of this timeline, a Markov chain will be constructed in order to predict in a statistically correct way the next activity that the user will perform according to the one he/she performed last. In order to build a Markov chain, we need to set up a graph where each node represents an activity. Then, each time the user changes activity, a link must be created between the activity he has just finished and the next activity he has started. In this way, the traffic corresponding to the activities that will be generated will be probabilistically correct and consistent with the user's habits. Here is an overview of the Markov chain that has been constructed with the analysed network trace.

```
# "Markov chain": {
#     "www.pathe.be": {
#         "www.pathe.be": 33.33333333333333,
#         "www.imperva.com": 33.33333333333333,
#         "www.like2sport.com": 33.33333333333333
#     },
#     "www.google.be": {
#         "www.overleaf.com": 100.0
#     },
#     "www.overleaf.com": {
#         "www.overleaf.com": 33.33333333333333,
#         "www.deepl.com": 12.5
#         ...
#     },
#     ...: {
#         ...
#     }
# }
```

Information about user behaviour from this Markov chain can typically be that `www.pathe.be`, `www.google.be` and `www.lagomframework.com` are all equally likely to be accessed after the user has visited `www.pathe.be`.

Here is a more global view of the complete Markov chain.

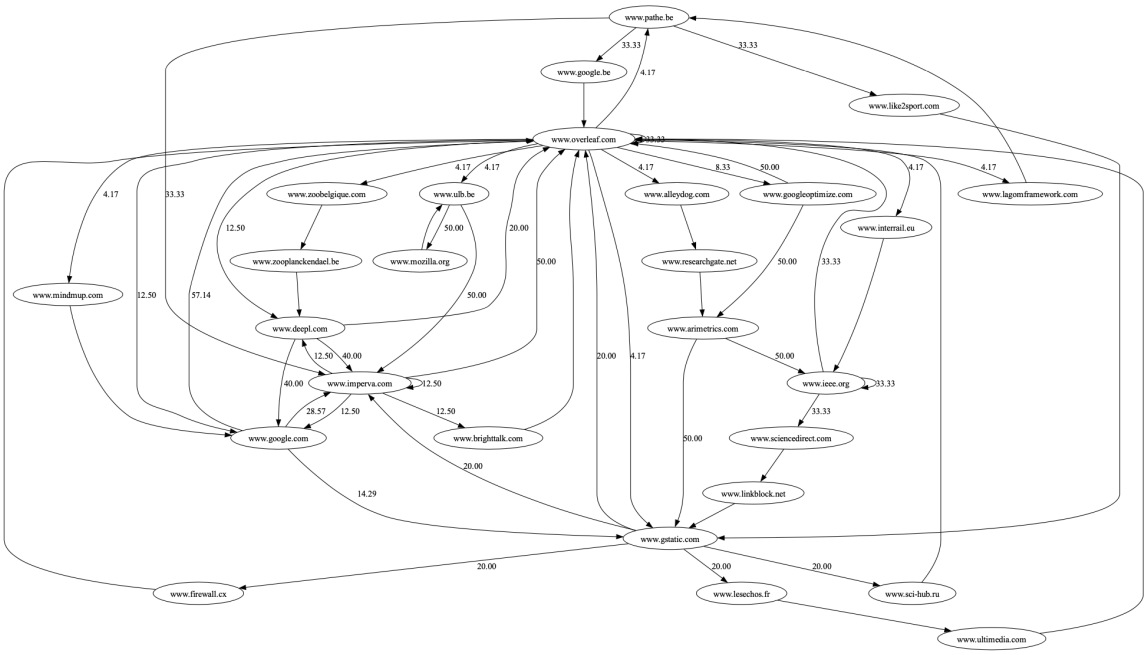


Figure 3.6: Markov chain of the web researches performed by a user on the 21 of April

3.1.5 Global statistics

In addition to creating a personalised file for each user with all his/her characteristics, the script allows to save more general data on the network trace. These are listed below and are saved in the `global-statistics.json` file.

- The global number of bits (ingoing, outgoing, total exchanged).
- The global number of packets (ingoing, outgoing, total exchanged).
- The global bit rate (ingoing, outgoing, total)
- The global packet rate (ingoing, outgoing, total)
- The global inter-packet-time rate (ingoing, outgoing, total)
- The global protocol rate (ingoing, outgoing, total)

These different data allow the exchange of packets to be viewed at a lower level. This makes it possible for example, to identify burst periods thanks to the inter-packet time ratio, the periods of use of the various protocols or the periods when a lot of data has been exchanged. Here is an overview of how this data is saved in the file.

```
# {
#     "Bits statistics": {
#         "Ingoing": 1877114832,
#         "Outgoing": 363642880,
#         "Total": 2240757712
#     },
#     "Packets statistics": {
#         "Ingoing": 266177,
#         "Outgoing": 226297,
#         "Total": 492474
#     },
#     "Rate statistics": {
#         "1682056800": {
#             "Bits_rate": {
#                 "Ingoing": 0,
#                 "Outgoing": 0,
#                 "Total": 0
#             },
#             "Packets_rate": {
#                 "Ingoing": 0,
#                 "Outgoing": 0,
#                 "Total": 0
#             },
#             "Inter_time_rate": {
#                 "Ingoing": 0.0,
#                 "Outgoing": 0.0,
#                 "Total": 0.0
#             },
#             "Protocols_rate": {
#                 "Ingoing": {},
#                 "Outgoing": {},
#                 "Total": {}
#             }
#         },
#         ...
#         "1682056810": {
#             "Bits_rate": {
#                 "Ingoing": 88584,
#                 "Outgoing": 23920,
#                 "Total": 112504
#             },
#             "Packets_rate": {
#                 "Ingoing": 26,
#                 "Outgoing": 25,
#                 "Total": 51
#             },
#             "Inter_time_rate": {
#                 "Ingoing": 0.005162744217429349,
#                 "Outgoing": 0.001239248539405935,
#                 "Total": 0.0
#             },
#             "Protocols_rate": {
#                 "Ingoing": {
#                     "TCP": 13,
```

```
#                         "TLS": 13
#                     },
#                     "Outgoing": {
#                         "TCP": 14,
#                         "TLS": 11
#                     },
#                     "Total": {
#                         "TCP": 27,
#                         "TLS": 24
#                     }
#                 }
#             },
#             ...
#         }
# }
```

The data for the ratios are classified by second. In the overview above we see that no packets were sent or received during the second 1682056800 (since the epoch), which is precisely 08:00:00 on the day the network trace was taken. On the other hand, at second 1682056810 from the epoch, which corresponds to 08:00:10 on that day, packets were exchanged and the various characteristics were updated to give an average of the ratio information of the packets that were exchanged during that second plus information on the protocols.

3.2 Information spreading phase

This second section on the information spreading phase will itself be divided into three sub-sections. Each of them will propagate more and more precise information about the user's behaviour. These three parts will be respectively the *grouping of activities*, the *periods of use spreading* and finally the *activities spreading*.

3.2.1 Grouping of activities

As explained in the literature review, one way to keep traffic similar while having different activities is to group them by theme. These themes will represent the overall goal of the user when performing an activity. By using this grouping with the Markov chain discussed earlier, it is possible to reconstruct the whole period under analysis in a chronologically correct way while keeping a good balance between similarity and creativity. Too much similar traffic would make the model look like a replay engine while too much creative traffic would be irrelevant to the previously analysed user behaviour.

For simplicity, the groups have been created in advance in a `groups.json` file. Ideally, these different groups should be filled with websites that the user usually visits. This way, no activity happens by chance. Unfortunately the amount of starting data is too small to fill all groups efficiently. The groups were therefore arbitrarily filled with a few popular websites for each category.
Here is an overview of the contents.

```
# {
#     "Recherches": ["www.google.be", "www.mozilla.org", "www.opera.com", ...],
#     "Cinemas": ["www.pathe.be", "www.kinepolis.be", "www.cinebel.dhnet.be", ...],
#     "Ecriture": ["www.overleaf.com"],
#     "Train": ["www.interrail.eu", "www.belgiantrain.be", "www.thalys.com/be", ...],
#     ...
# }
```

3.2.2 Periods of use spreading

The GHOSTS framework allows to set up one or more *handlers*. These define the periods in which traffic will be generated according to the specified activities. In this case, each activity zone will be linked to a handler. The beginning of a zone will mark the beginning of a handler and the same goes for the end of each zone.

Each handler is composed of a header and a body. Firstly, the header defines the basis of the activities that will be performed during the handler period. In the context of this traffic generation, only the start and end time will be subject to change to suit each area of activity. All other parameters will remain unchanged. Secondly, the body contains the set of activities to be performed as well as some actions that can be performed (typically clicking on a link). For the purpose of this traffic generation, only the simple *browse* option will be used. Here is the beginning of the timeline.json file automatically generated from the information collected.

```
# {
#     "TimeLineHandlers": [
#         {
#             "HandlerType": "BrowserFirefox",
#             "Initial": "about:blank",
#             "UtcTimeOn": "08:31:00",
#             "UtcTimeOff": "08:57:00",
#             "Loop": "True",
#             "TimeLineEvents": [
#                 ...
#             ]
#         },
#         {
#             "HandlerType": "BrowserFirefox",
#             "Initial": "about:blank",
#             "UtcTimeOn": "09:04:00",
#             "UtcTimeOff": "10:13:00",
#             "Loop": "True",
#             "TimeLineEvents": [
#                 ...
#             ]
#         },
#         ...
#     ]
# }
```

The parameters that will remain unchanged are.

- HandlerType : this parameter defines which search engine is used to perform the internet search(s). For efficiency reasons, Firefox has been chosen.
- Initial : this parameter is set to about:blank to ensure that the very first page generated when the Firefox application is opened is a blank neutral page.
- Loop : since the generation of activities is a Markov chain, there is no room for random generation of activities. All websites that will be visited to generate traffic are planned in advance and will be hardcoded in the *TimeLineEvents* parameter.

3.2.3 Activities spreading

The different activities were distributed in each handler following the user's timeline. There are as many activities in each handler as in each zone. The time between each activity is set to the time taken by the equivalent activity in the analysed network trace. The use of the group method and the Markov chain allows a great diversity and therefore rarely obtain two identical timeline files. Furthermore, all activities occur at logical times because they are all statistically linked through the Markov chain.

Possible improvement

An important enhancement to the spreading of activities would be to adjust the timing of each activity more precisely and to make the activity 'live' by using the GHOSTS features for clicking on links or entering text.

CHAPTER

4

RESULTS

This chapter is the culmination of this dissertation, which will compares the results obtained when the methods were practiced. Two sections will make up this chapter. Firstly, the results related to the first part of the methods, i.e. the information extraction phase. Secondly, the results related to the second phase which is the information spreading phase.

It can be noted that both objectives of the thesis were successfully met. Firstly, with a script to extract global and more user-specific data from pcaps files. And secondly, with a second script that used the previously retrieved data to create several simple `timeline.json` files that could be used with the GHOSTS framework to generate realistic web traffic.

4.1 Information extraction results

As a reminder, this phase allowed the extraction of several characteristics highlighted in the human behaviour analysis part of the literature review chapter. Some features were therefore extracted and stored in a raw format and others were extracted before being interpreted by the script to help better understand the human actions behind them.

There are two different files that the script produces. Firstly, the `global-statistics.json` file which is generated once and contains features on the whole network trace analysed without taking into account the actions of the users. Secondly, the `<MAC address>-statistics.json` file(s) which is a digest of a lot of data relating to a particular user who has that MAC address.

The results of this phase are conclusive because, firstly, all the important data listed in the table 2.2 are present in the output files of the script (in raw form or already interpreted if necessary). The points below summarise each of them.

- In the `global-statistics.json` file :
  - Bits statistics (ingoing, outgoing, total, rate)
  - Packets statistics (ingoing, outgoing, total, rate)
  - Inter-packet time statistics (ingoing, outgoing, total, rate)
  - Protocols statistics (ingoing, outgoing, total)
  - Domain names statistics (domain contacted, number contacted, unduplicated contacted, rate)
- In the `<MAC address>-statistics.json` file(s) :
  - Bits statistics (ingoing, outgoing, total, rate)
  - Packets statistics (ingoing, outgoing, total, rate)
  - Domain names statistics (domain contacted, number contacted, unduplicated contacted, rate)
  - Zones statistics (domain name)
  - Activities statistics (IP address, domain name, ports, bits, packets)
  - Markov chain (activities)
  - Timeline (activities)

And secondly because the extracted data correspond to what was done during the day the traffic was generated.

4.2 Information spreading results

This second phase allowed, from the different `json` files generated during the first phase, to automatically generate different simple `timeline.json` files. The files created by this second program are intended to be used with the GHOSTS framework, but the data generated during the first phase are quite general and not very oriented towards the framework, so they can be used in a completely different context. However, in this thesis, it is for the generation with the framework that the last script has been made.

The generated files are quite simple because they only contain the "browse" command. However, the framework offers a multitude of commands in order to generate highly customisable traffic. However, these first simple files are already able to generate interesting things.

- Firstly, the generation zones are precisely defined, which allows to have traffic only at the time of the day when it is necessary. As this part of the `timeline.json`

file below shows, the handlers have a period of activity similar to the periods of the different zones defined in the information extraction phase. As a reminder, zone 1 is between 8:31 and 8:57 and zone 2 between 9:04 and 10:13.

```
# {
#     "TimeLineHandlers": [
#         {
#             "HandlerType": "BrowserFirefox",
#             "Initial": "about:blank",
#             "UtcTimeOn": "08:31:00",
#             "UtcTimeOff": "08:57:00",
#             "Loop": "True",
#             "TimeLineEvents": [
#                 ...
#             ]
#         },
#         {
#             "HandlerType": "BrowserFirefox",
#             "Initial": "about:blank",
#             "UtcTimeOn": "09:04:00",
#             "UtcTimeOff": "10:13:00",
#             "Loop": "True",
#             "TimeLineEvents": [
#                 ...
#             ]
#         },
#         ...
#     ]
# }
```

- Secondly, the internet searches arrive in a statistically correct order according to a Markov chain established from the network trace data that serves as a model. In addition, each file generated is different thanks to the grouping of activities, which makes it possible to multiply the possibilities of activity for a user while maintaining a logical order of execution. Here is a part of two `timeline.json` files generated one after the other. You can see that they are not exactly the same in terms of the websites that will be visited.

```
# {
#     "TimeLineHandlers": [
#         {
#             "HandlerType": "BrowserFirefox",
#             "Initial": "about:blank",
#             "UtcTimeOn": "08:31:00",
#             "UtcTimeOff": "08:57:00",
#             "Loop": "True",
#             "TimeLineEvents": [
#                 {
#                     "Command": "browse",
#                     "CommandArgs": [
#                         [
#                             "www.pathe.be"
#                         ]
#                     ],
```

```
#                         ...
#                     },
#                     {
#                         "Command": "browse",
#                         "CommandArgs": [
#                             [
#                                 "www.like2sport.com"
#                             ]
#                         ],
#                         ...
#                     },
#                     {
#                         "Command": "browse",
#                         "CommandArgs": [
#                             [
#                                 "www.gstatic.com"
#                             ]
#                         ],
#                         ...
#                     },
#                     {
#                         ...
#                     }
#                 ]
#             },
#             ...
#         ]
# }
```

```
# {
#     "TimeLineHandlers": [
#         {
#             "HandlerType": "BrowserFirefox",
#             "Initial": "about:blank",
#             "UtcTimeOn": "08:31:00",
#             "UtcTimeOff": "08:57:00",
#             "Loop": "True",
#             "TimeLineEvents": [
#                 {
#                     "Command": "browse",
#                     "CommandArgs": [
#                         [
#                             "www.cinebel.dhnet.be"
#                         ]
#                     ],
#                     ...
#                 },
#                 {
#                     "Command": "browse",
#                     "CommandArgs": [
#                         [
#                             "www.lesechos.fr"
#                         ]
#                     ],
#                     ...
```

```
#                     },
#                     {
#                         "Command": "browse",
#                         "CommandArgs": [
#                             [
#                                 "www.google.be"
#                             ]
#                         ],
#                         ...
#                     },
#                     {
#                         ...
#                     }
#                 ]
#             },
#             ...
#         ]
# }
```

- Thirdly, the time spent between each website is realistic as it is directly derived from the time spent performing similar activities in the network trace used as a model. Here is a part of a `timeline.json` file showing the time elapsed between the first two activities. As a reminder, the time spent on the first activity is 121 seconds and the time spent on the second activity is 241 seconds.

```
# {
#     "TimeLineHandlers": [
#         {
#             "HandlerType": "BrowserFirefox",
#             "Initial": "about:blank",
#             "UtcTimeOn": "08:31:00",
#             "UtcTimeOff": "08:57:00",
#             "Loop": "True",
#             "TimeLineEvents": [
#                 {
#                     "Command": "browse",
#                     "CommandArgs": [
#                         [
#                             "www.cinebel.dhnet.be"
#                         ]
#                     ],
#                     "DelayAfter": "121",
                      "DelayBefore": 0
#                 },
#                 {
#                     "Command": "browse",
#                     "CommandArgs": [
#                         [
#                             "www.lesechos.fr"
#                         ]
#                     ],
#                     "DelayAfter": "241",
                      "DelayBefore": 0
#                 },
```

```
#                     {
#                         ...
#                     }
#                 ]
#             },
#             ...
#         ]
# }
```

The operation of `timeline.json` files has not been formally tested within the framework. However, the format of these files follows the format given in several tutorials. They should therefore all work with the framework without any problems.

CHAPTER

5

CONCLUSION

This thesis is the result of more than three months of research and development into realistic traffic generation technologies and models. Despite the two main obstacles that are the problems related to the generation of the Internet and those related to the simulation of human behaviour. A simple but functional model that meets the requirements of realism has been developed as a result of this paper.

This journey began by highlighting the need for this type of technology to train security professionals more effectively. This was followed by the presentation of the Cyber range which is the perfect place to train these technologies in order to avoid the risk of propagation of certain attacks during the simulations, and the GHOSTS framework which will be used to generate the realistic traffic. Finally, the first chapter ended with the presentation of the first big problem but which will finally be circumvented, the simulation of the internet.

Then came the purely theoretical aspect which consisted in carrying out the necessary research in order to study what is already being done in terms of traffic generation (realistic and non-realistic). Given the wealth of literature on this subject, the choice was made to sort the different existing technologies according to the realism of the traffic they generate. These theoretical researches have allowed to find the different relevant criteria that are useful to extract from different network traces in order to regenerate realistic traffic. The end of the second chapter of this thesis is dedicated to the study of these characteristics.

The next logical step after the study of the features to be extracted is the extraction itself. This is why the penultimate chapter is to be followed in parallel with the python

code provided in the Github linked at the beginning of the chapter. This one includes all the practical part of the model, from the extraction of data from network traces to the generation of timeline files. These are the skeleton of the traffic generation with the GHOSTS framework.

Finally, the last part of this course deals with the results obtained from the implementation of the model. These results demonstrate that the model works well. Nevertheless, improvements are always possible and given the vast arsenal of tools that the GHOSTS framework offers, it is still possible to generate ever more realistic traffic.

BIBLIOGRAPHY

[1] Jerry Glowniak. "History, structure, and function of the internet". In: *Seminars in Nuclear Medicine* 28.2 (1998). Telenuclear Medicine, pp. 135–144. ISSN: 0001-2998. DOI: https://doi.org/10.1016/S0001-2998(98)80003-2. URL: https://www.sciencedirect.com/science/article/pii/S0001299898800032.

[2] J. Clement. *Internet usage worldwide - Statistics & Facts*. Nov. 2022. URL: https://www.statista.com/topics/1145/internet-usage-worldwide/#topicHeader__wrapper (visited on Dec. 28, 2022).

[3] J. Degenhard. *Forecast of the number of internet users in the World from 2010 to 2025*. July 2021. URL: https://www.statista.com/forecasts/1146844/internet-users-in-the-world (visited on Dec. 28, 2022).

[4] Statista Research Department. *U.S. companies and cyber crime - Statistics & Facts*. Oct. 2022. URL: https://www.statista.com/topics/1731/smb-and-cyber-crime/#topicHeader__wrapper (visited on Dec. 28, 2022).

[5] Statista Research Department. *Average cost of a data breach in the United States from 2006 to 2022*. Sept. 2021. URL: https://www.statista.com/statistics/273575/us-average-cost-incurred-by-a-data-breach/ (visited on Dec. 28, 2022).

[6] Statista Research Department. *Estimated cost of cybercrime worldwide from 2016 to 2027*. Dec. 2022. URL: https://www.statista.com/statistics/1280009/cost-cybercrime-worldwide/ (visited on Dec. 29, 2022).

[7] Xavier Lambert and Danielle Welter. *Cyberattaque des hôpitaux Vivalia: toutes les consultations annulées lundi, un redémarrage par priorités*. May 2022. URL: https://www.rtbf.be/article/cyberattaque-des-hopitaux-vivalia-toutes-les-consultations-annulees-lundi-un-redemarrage-par-priorites-10993010 (visited on Dec. 29, 2022).

[8] Camryn Mottl. *What Industries are the Biggest Targets for Cyber Attacks?* Apr. 2022. URL: https://www.statista.com/statistics/221293/cyber-crime-target-industries/ (visited on Dec. 29, 2022).

[9] Rob Sobers. *Cybersecurity Data Statistics*. URL: https://www.varonis.com/fr/blog/data-breach-statistics (visited on Dec. 30, 2022).

[10] Jon Davis and Shane Magrath. *A Survey of Cyber Ranges and Testbeds*. Tech. rep. ADA594524. Edinburgh, Australia: Defense Science and Technology Organisation, Oct. 2013.

[11] U.S. Department of Commerce. *Cyber Ranges*. NIST, National Institute of Standards and Technology. Washington D.C., U.S., Feb. 2018.

[12] NICE. *The Cyber Range: A Guide*. https://www.nist.gov/system/files/documents/2020/06/25/The%20Cyber%20Range%20-%20A%20Guide%20%28NIST-NICE%29%20%28Draft%29%20-%20062420_1315.pdf. Accessed: 2023-01-22. 2020.

[13] *What is Cyber Range? - Types and Use Cases*. URL: https://www.purplesynapz.com/cyber-range-types-and-use-cases (visited on Feb. 20, 2023).

[14] B. Pandey and S. Ahmad. "Introduction". In: *Introduction to the Cyber Ranges*. 4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN: CRC Press, 2022, pp. 6–7.

[15] Nestoras Chouliaras et al. "Cyber Ranges and TestBeds for Education, Training, and Research". In: *Applied Sciences* 11 (Feb. 2021), p. 1809. DOI: 10.3390/app11041809.

[16] Thibault Debatty and Wim Mees. "Building a Cyber Range for training CyberDefense Situation Awareness". In: *2019 International Conference on Military Communications and Information Systems (ICMCIS)*. 2019, pp. 1–6. DOI: 10.1109/ICMCIS.2019.8842802.

[17] *What is orchestration?* Oct. 2019. URL: https://www.redhat.com/en/topics/automation/what-is-orchestration (visited on Jan. 1, 2023).

[18] *Un hyperviseur, qu'est-ce que c'est ?* Jan. 2020. URL: https://www.redhat.com/fr/topics/virtualization/what-is-a-hypervisor (visited on Jan. 1, 2023).

[19] Liza Poggemeyer et al. *Remote Desktop Services - Access from anywhere*. July 2021. URL: https://learn.microsoft.com/en-us/windows-server/remote/remote-desktop-services/rds-plan-access-from-anywhere (visited on Jan. 1, 2023).

[20] Wim Mees. "Cyber-situation awareness". In: *Pragmatic Cybersecurity*. 2020, pp. 137–141.

[21] Daniel Aarno and Jakob Engblom. "Chapter 5 - Networking". In: *Software and System Development using Virtual Platforms*. Ed. by Daniel Aarno and Jakob Engblom. Boston: Morgan Kaufmann, 2015, pp. 129–159. ISBN: 978-0-12-800725-9. DOI: https://doi.org/10.1016/B978-0-12-800725-9.00005-6. URL: https://www.sciencedirect.com/science/article/pii/B9780128007259000056.

[22] Sally Floyd and Vern Paxson. "Why We Don't Know How To Simulate The Internet". In: (Dec. 1997). DOI: 10.1109/WSC.1997.640988.

[23] S. Floyd and V. Paxson. "Difficulties in simulating the Internet". In: *IEEE/ACM Transactions on Networking* 9.4 (2001), pp. 392–403. DOI: 10.1109/90.944338.

[24] Staff Contributor. *Best Network Traffic Monitoring Tools*. Mar. 2020. URL: https://www.dnsstuff.com/network-traffic-generator-software (visited on Feb. 8, 2023).

[25] Harris Andrea. *10 Best Network Traffic Packet Generator Software Tools*. URL: https://www.networkstraining.com/best-network-traffic-packet-generator-tools/ (visited on Feb. 8, 2023).

[26] Marc Wilson. *Network Traffic Generator and Stress Testing Tools for LAN WAN Bandwidth*. URL: https://www.pcwdld.com/network-traffic-generator-and-stress-testing-tools (visited on Feb. 8, 2023).

[27] Luo Song and G. A. Marin. *Traffic Generators for Internet Traffic*. URL: http://www.icir.org/floyd/papers/webpages/trafficgenerators (visited on Feb. 8, 2023).

[28] Vincent H. Berk, Ian Gregorio-de Souza, and John P. Murphy. "Generating realistic environments for cyber operations development, testing, and training". In: *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense XI*. Ed. by Edward M. Carapezza. Vol. 8359. International Society for Optics and Photonics. SPIE, 2012, p. 835908. DOI: 10.1117/12.924762. URL: https://doi.org/10.1117/12.924762.

[29] Paul Simoneau. *The OSI Model: Understanding the Seven Layers of Computer Networks*. URL: https://faculty.sfcc.spokane.edu/Rudlock/files/WP_Simoneau_OSIModel.pdf (visited on Mar. 14, 2023).

[30] Rachelle Miller. *The OSI Model: An Overview*. Tech. rep. CMU/SEI-2018-TR-005. SANS Institute, 2021.

[31] Bharat Rahuldhev Patil et al. "Ostinato - A Powerful Traffic Generator". In: *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*. 2017, pp. 1–5. DOI: 10.1109/CSITSS.2017.8447596.

[32] Shalvi Srivastava et al. "Comparative study of various traffic generator tools". In: *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*. 2014, pp. 1–6. DOI: 10.1109/RAECS.2014.6799557.

[33] *PACKETH*. URL: https://packeth.sourceforge.net/packeth/Home.html (visited on Apr. 5, 2023).

[34] *iPerf - The ultimate speed test tool for TCP, UDP and SCTP*. URL: https://iperf.fr (visited on Apr. 5, 2023).

[35] S. Avallone et al. "D-ITG distributed Internet traffic generator". In: *First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings*. 2004, pp. 316–317. DOI: 10.1109/QEST.2004.1348045.

[36] Sándor Molnár, Péter Megyesi, and Géza Szabó. "How to validate traffic generators?" In: *2013 IEEE International Conference on Communications Workshops (ICC)*. 2013, pp. 1340–1344. DOI: 10.1109/ICCW.2013.6649445.

[37] *tcpreplay(1) - Linux man page*. URL: https://linux.die.net/man/1/tcpreplay# (visited on Apr. 5, 2023).

[38] Wu-chang Feng et al. "TCPivo: A High-Performance Packet Replay Engine". In: *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*. MoMeTools '03. Karlsruhe, Germany: Association for Computing Machinery, 2003, pp. 57–64. ISBN: 1581137486. DOI: 10.1145/944773.944783. URL: https://doi.org/10.1145/944773.944783.

[39] NICOLA BONELLI, STEFANO GIORDANO, and GREGORIO PROCISSI. *BRUTE: A High Performance and Extensibile Traffic Generator*. URL: https://arpi.unipi.it/handle/11568/190774 (visited on Apr. 5, 2022).

[40] Gianni Antichi et al. "BRUNO: A high performance traffic generator for network processor". In: *2008 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*. 2008, pp. 526–533.

[41] Sebastian Zander, David Kennedy, and Grenville Armitage. *KUTE – A High Performance Kernel-based UDP Traffic Engine*. Tech. rep. 050118A. Melbourne, Australia: Centre for Advanced Internet Architectures (CAIA), Jan. 2005.

[42] *Intel IXP2400 Network Processor*. INTEL. Nov. 2003.

[43] Edward Kresch and Sarvesh Kulkarni. "A Poisson Based Bursty Model of Internet Traffic". In: *2011 IEEE 11th International Conference on Computer and Information Technology*. 2011, pp. 255–260. DOI: 10.1109/CIT.2011.95.

[44] R. Jain and S. Routhier. "Packet Trains–Measurements and a New Model for Computer Network Traffic". In: *IEEE Journal on Selected Areas in Communications* 4.6 (1986), pp. 986–995. DOI: 10.1109/JSAC.1986.1146410.

[45] Chitra Javali and Girish Revadigar. "Network Web Traffic Generator for Cyber Range Exercises". In: *2019 IEEE 44th Conference on Local Computer Networks (LCN)*. 2019, pp. 308–315. DOI: 10.1109/LCN44214.2019.8990880.

[46] Kashi Venkatesh Vishwanath and Amin Vahdat. "Swing: Realistic and Responsive Network Traffic Generation". In: *IEEE/ACM Transactions on Networking* 17.3 (2009), pp. 712–725. DOI: 10.1109/TNET.2009.2020830.

[47] W3Techs. *TRex - Realistic Traffic Generator*. URL: https://trex-tgn.cisco.com (visited on May 2, 2022).

[48] Lee Rossey et al. "LARIAT: Lincoln adaptable real-time information assurance testbed". In: vol. 6. Feb. 2002, pp. 6–2671. ISBN: 0-7803-7231-X. DOI: 10.1109/AERO.2002.1036158.

[49] Timothy M. Braje. "Advanced Tools for Cyber Ranges". In: *LINCOLN LABORATORY JOURNAL*. Vol. 22. 1. 2016, pp. 24–32.

[50] Ben Carterette, Evangelos Kanoulas, and Emine Yilmaz. "Simulating Simple User Behavior for System Effectiveness Evaluation". In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. CIKM '11. Glasgow, Scotland, UK: Association for Computing Machinery, 2011, pp. 611–620. ISBN: 9781450307178. DOI: 10.1145/2063576.2063668. URL: https://doi.org/10.1145/2063576.2063668.

[51] Icek Ajzen. "Models of human social behavior and their application to health psychology". In: *Psychology & Health* 13.4 (1998), pp. 735–739. DOI: 10.1080/08870449808407426. eprint: https://doi.org/10.1080/08870449808407426. URL: https://doi.org/10.1080/08870449808407426.

[52] G.I. Webb, M.J. Pazzani, and D. Billsus. "Machine Learning for User Modeling". In: *User Modeling and User-Adapted Interaction* 11 (2001), pp. 19–29. DOI: https://doi.org/10.1023/A:1011117102175.

[53] Hyoung-Kee Choi and J.O. Limb. "A behavioral model of Web traffic". In: *Proceedings. Seventh International Conference on Network Protocols*. 1999, pp. 327–334. DOI: 10.1109/ICNP.1999.801961.

[54] Dustin D. Updyke et al. *GHOSTS in the Machine: A Framework for Cyber-Warfare Exercise NPC Simulation*. Tech. rep. CMU/SEI-2018-TR-005. Pittsburgh, PA: Software Engineering Institute - Carnegie Mellon University, Dec. 2018.

[55] Software Engineering Institute. *GHOSTS - A Framework for Realistic NPC Orchestration*. Tech. rep. Pittsburgh, PA: Carnegie Mellon University, 2018.

[56] Georgi Nikolov. *Simulate user activity with the GHOSTS framework: Client set-up and Timelines*. URL: https://cylab.be/blog/81/simulate-user-activity-with-the-ghosts-framework-client-set-up-and-timelines (visited on Apr. 10, 2023).