UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LAUREN SILVA ROLAN SAMPAIO

An Overview of AI-enabled Attacks: concepts, state-of-the-art, and evaluation of prototypes

Advisor: Prof. Dr. Mariana Recamonde Mendoza
Coadvisor: Assoc. Lect. Alain Lebret
Coadvisor: Rsr. Karel Mittig

Porto Alegre
May 2021

“I was taught that the way of progress
was neither swift nor easy.”
— Marie Curie

ACKNOWLEDGMENTS

I would like to thank my tutors Karel Mittig and Jean-François Misarsky, and my colleagues from Orange Labs, as well as my ENSICAEN tutor Alain Lebret for their availability, friendliness, and professionalism. Besides them, I thank my Brazilian advisor Mariana Recamonde Mendoza for her availability and expertise, and professor Alberto Egon Schaeffer Filho for introducing me to the cybersecurity area. I also would like to thank CAPES for the financing of my third year at École Nationale Supérieure d'Ingénieurs de Caen, and my UFRGS tutors Fernanda Kastensmidt, Ricardo Reis and André Reis, as well as professor Raphael Brum, for their quick responses and all the help they gave me.

Finally, I thank my friends and family for the support that lead me to the achievement of my internship and final paper.

ABSTRACT

As new technologies appear given the evolution of computer science, unpredicted cyber threats are created. The high performance of artificial intelligence algorithms in the most diverse areas of knowledge gets the attention of researchers, and shows new possibilities of intelligent network attacks. One way to avoid possible damage caused by these attacks, which are still poorly known, is the study of available tools, and the elaboration of such threats in a controlled environment. Thus, in this study a state-of-the-art containing multiple prototypes implementing such threats was compiled, them being at both academic and private levels. Each prototype was briefly commented, besides having a counter-measure estimated, since the main objective is to mitigate such attacks. In addition to that, some prototypes were selected to a deeper analysis, where the code and results were verified. Finally, a prototype was developed by myself in a domain not much explored by other authors, and then analyzed. I verified that all tested tools were still in early development phase, and did not reach yet their objectives, even though they show a great theoretical potential. My password-breaking tool prototype also did not have an optimal result, but it had similar achievements to those of traditional password-breaking tools.

Keywords: Machine learning. artificial intelligence. cyberdefense. cyberattack. cybersecurity. brute-force. CAPTCHA. phishing. natural language processing. computer vision.

Um Resumo sobre Ataques Guiados por IA: conceitos, estado da arte e avaliação de protótipos

RESUMO

Conforme novas tecnologias surgem com a evolução da computação, ameaças cibernéticas, até então não previstas, são criadas. A boa performance de algoritmos de inteligência artifical nas mais diversas áreas chama a atenção de pesquisadores, e abre espaço para a criação de ataques de rede inteligentes. Uma maneira de evitar os possíveis danos destes ataques, ainda pouco compreendidos, é o estudo das ferramentas disponíveis e a elaboração de tais ameaças em ambiente controlado. Assim sendo, neste estudo coletou-se um estado da arte contendo vários protótipos a nível acadêmico e privado que se propõem a implementar esse tipo de ameaças. Cada protótipo foi brevemente comentado, além de ter uma possível contra-medida estimada, uma vez que o objetivo final é impedir tais ataques. Além disto, alguns protótipos foram selecionados para estudo mais profundo, onde o código e resultados foram analisados. Finalmente, um protótipo de minha autoria foi desenvolvido em uma área pouco explorada pelos outros autores, e então analisado. Verificamos que todas as ferramentas testadas ainda estão na fase mais inicial de prototipagem e ainda não alcançaram seus objetivos, embora demonstrem ter grande potencial. O protótipo desenvolvido por mim para a quebra de senhas também não obteve resultado ótimo, mas teve desempenho semelhante ao de ferramentas tradicionais desta área.

Palavras-chave: aprendizado de máquina, inteligência artificial, cyberdefesa, ciberataques, cybersegurança, força bruta, CAPTCHA, phishing, processamento de linguagens naturais, visão computacional, exploit.

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
NLP	Natural Language Processing
PCA	Principal Component Analysis
GAN	Generative Adversarial Network
CNN	Convolutional Neural Network
DBSCAN	Density-based spatial clustering of applications with noise
TLD	Top-Level Domain
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
LSTM	Long-Short Term Memory
DL	Deep Learning
DoS	Denial of Service
DDoS	Distributed Denial of Service
OS	Operational System
URL	Uniform Resource Locator

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

From the replicants of *Blade Runner* to the complex military system of Skynet in *Terminator*, Artificial Intelligence (AI) has been a part of popular culture since the early years of the 20th century. Once it is usually linked to robots, most films and books treat these two technologies as interdependent. However, in the real world, they evolve in separated paces. Most of the current AIs are disembodied, as presented in the movie *Her*.

One common denominator between most of these works is the use of AI in attacks. Indeed, with the ease of automation, such applications became evident even before their development in real-world solutions. AI attacks are faster, stealthier, and more accurate than human-operated ones. It did not take long before military services started observing the benefits of developing such solutions (NSCAI, 2019; O'CONNOR, 2017; ALLEN, 2019).

Besides the impact on the military, AI is currently used in social and economic applications, as in the decision making of some countries and companies (AHMED; WRIGHT, 2019, Chapter 19). Data analysis combined with learning techniques allows governments that endorse censorship to better control what is being published online in real-time (YENALA et al., 1970; YANG, 2018).

This document aims to take an overview on the subject of cyberattacks enabled by the use of Artificial Intelligence, from their first appearance in the global network to the recently developed tools. This knowledge will allow us to understand better the risks related to these threats, anticipate the possible evolution of these attacks, and, finally, give us an idea of possible counter-measures.

Besides this theoretical study, I analyzed some tools in the prototyping phase that relate to the concepts presented, such as CAPTCHA breaking, phish generation, and automatic exploit generation. Furthermore, I developed a set of algorithms to analyze a database of passwords and, based on the results of such analysis, evaluate the best attack using the acquired knowledge.

The organization of the remainder of this work is as follows: chapter 2 describes some essential concepts and definitions in the fields of AI and cybersecurity; chapter 3 presents the state-of-the-art of AI-enabled tools, be they in the prototyping phase or commercial phase. Chapter 4, some tools are evaluated, and one experiment involving CAPTCHA is done. Finally, in chapter 5, our conclusions are presented, as well as possible directions for future works.

2 CONCEPT DEFINITIONS

This study focuses on the intersection of cybersecurity in connected networks and artificial intelligence algorithms. This chapter describes the main concepts from these research fields underlying this work's goals.

2.1 Artificial Intelligence

Artificial Intelligence (AI) is the ability of a computer to perform actions that usually would require human intelligence to solve. One of the most successful and ancient systems in AI is the expert systems (JACKSON, 1998). These methods emulate the decision-making procedure of an actual human being in a sort of "question-making" analysis. The method analyses multiple factors of a sample and attributes a class to the sample based on prior knowledge. One limitation of this method is that it does not learn from experience: it only applies the rules given to it. Another issue is the difficulty of maintaining the system's coherence over time, given that the rules accumulate and end up conflicting with each other.

A subdomain of AI is Machine Learning (ML), where besides being able to perform these activities, the machine is also capable of learning from the execution of such actions. One example of an ML algorithm is the decision tree (SHALEV-SHWARTZ; BEN-DAVID, 2017, Chapter 18). It analyses a subset of samples and discovers which rules must be applied to that set to obtain an ideal classification. It is similar to the expert system since it is based on rules; however, it can be adapted to the dataset.

Finally, Deep Learning (DL) is a set of specific algorithms based on the concept of deep neural networks. Multiple nodes of calculus – neurons – are interconnected in a fashion that it is possible to extrapolate complex functions from it. DL is usually applied to problems where we do not need to explain the process that took us to a particular conclusion mathematically. Some applications are image and text classification, as well as image and text generation.

2.1.1 Types of learning

An ML algorithm can be classified into four types:

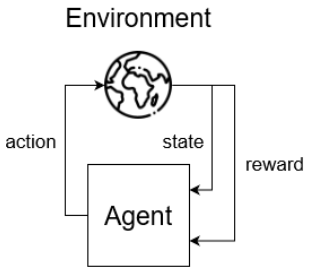
Supervised learning, where each sample has a label indicating the class to which it belongs, and the model is trained based on the database labels. One example of an algorithm that uses supervised learning is the decision tree, as cited before in (SHALEV-SHWARTZ; BEN-DAVID, 2017, Chapter 18).

Unsupervised learning, in which there is no labeled sample, and the objective of the algorithm is to group the samples into classes that share similarities observed in their attributes or better understand how the attributes relate to each other. The k-means algorithm (STEINLEY, 2006) is one example of such type of learning, and its naive version consists of verifying the positions of the k centroids given as input. Iteratively the algorithm recalculates the position of a centroid until there is no big change in its position.

Semi-supervised learning, in which some samples are labeled while others are not. It is seen as a middle-term between the two previous methods and has a classification objective. Generative models (ZHU, 2008) can be used for this purpose, where it is possible to learn the distribution of the unlabeled data based on knowledge of the labeled data. We intend to discover the conditional probability of an attribute X given that its sample belongs to a class Y (mathematically described as $P(X|Y = y)$).

Reinforcement learning, where an algorithm performs an action into the world and receives a reward or a punishment, depending on the consequences of its acts (KAELBLING; LITTMAN; MOORE, 1996). One example is the Q-Learning algorithm (WATKINS; DAYAN, 1992), in which the agent acts on the environment, receives a reward, and updates a table (Q-Table) containing all the combinations of states and possible actions. An example is shown in Figure 2.1.

Figure 2.1 – Representation of Q-Learning algorithm.



Action / State	A1	A2	A3	A4	A5
S1	0	10	20	-5	0
S2	0	0	1	0	0
S3	-23	0	3	2	5
S4	0	0	0	0	0
S5	0	7	0	0	0
S6	0	0	0	-3	0

Source: the author

2.1.2 Tasks where AI is used

AI algorithms can solve multiple tasks, which can be grouped in the following:

Classification, in which an agent decides to which class a specific unlabeled sample belongs. One common application is image classification, in which an input image has to be identified as belonging to a particular category (VISHAL et al., 2016). These techniques rely upon separating the input space into regions, such that each region represents a class.

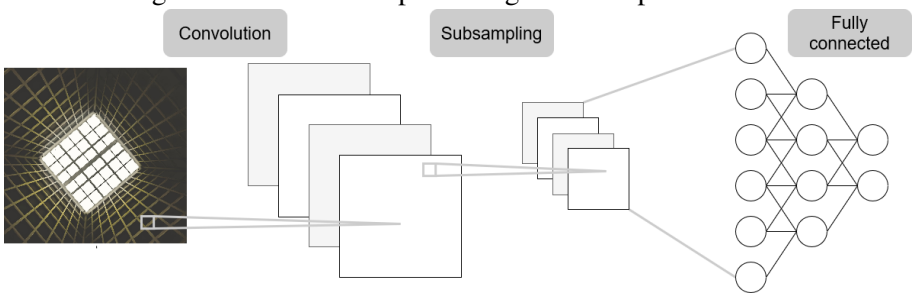
Regression, were given a data distribution, the model has to estimate the relationship between the attributes of the samples. Financial forecasting (GATELY, 1995) is one sample where a regression model is used to predict the behavior of the market given its past states.

Clustering consists of grouping samples into similar clusters of data. It can be based on the continuity of a cluster, its density, or its center (RAI; SINGH, 2010). Clustering is used when we try to understand the similarities between different data samples and usually do not rely on labeled data.

2.1.3 Some specific Neural Networks

There are multiple architectures of DL, each having a better performance when applied to a certain problem. An example is the Convolutional Neural Networks (CNN) (LIU et al., 2016), which combine multiple layers of convolutions and subsampling before passing the information to a fully connected network in order to process images, as shown in Figure 2.2.

Figure 2.2 – Schema representing the concept of a CNN.

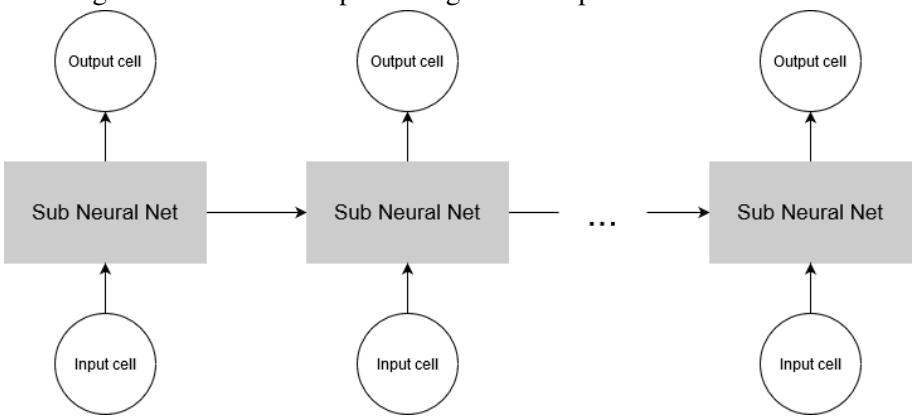


Source: the author

On the other hand, Long Short-Term Memory networks (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) can "remember" words and letters, and therefore are used in

text processing (Figure 2.3). In a translation context, each input cell in Figure 2.3 can be interpreted as a word from the original language and the output cells as words from the target language.

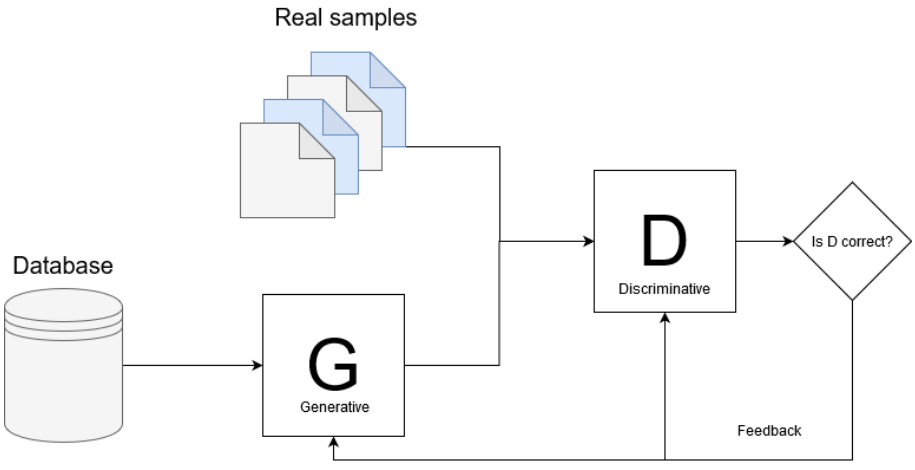
Figure 2.3 – Schema representing the concept of a LSTM network.



Source: the author

The most common method in adversarial learning is the Generative Adversarial Networks (GAN) (GOODFELLOW et al., 2014), which are composed of a pair of networks called generative (G) and discriminative (D). The objective of network G is to create synthetic examples that cannot be differentiated by network D from actual examples. One schema in Figure 2.4 shows the basic architecture of such a network.

Figure 2.4 – GAN architecture



Source: the author

The database, in a cybersecurity scenario, could be represented by the collection of different traces of previously registered attacks. These attacks would be used as input of the network G, which would generate similar examples following the database distribution. The output of G would be used as input of the network D, mixed with some real

samples. The objective of network D is to tell apart the samples that were generated by G from the real samples.

2.2 Cybersecurity

First used by the science-fiction writer William Gibson, the word cyberspace (an amalgam of "cybernetics" and "space") describes a "graphic representation of data abstracted from the banks of every computer in the human system" (SINGER; FRIEDMAN, 2014). The Internet is seen as the most successful and complete type of cyberspace we know nowadays, but it was not the first to be invented.

Between 1960-1969, the ARPANET (Advanced Research Projects Agency Network) led the way in defining the basic aspects of the Internet's network topology and some of its initial protocols, such as NCP (Network Control Protocol, which later would be substituted by TCP/IP).

In 1990 ARPANET was decommissioned, and the Internet started being developed, with HyperText Transfer Protocol (HTTP) and Hypertext Markup Language (HTML), which are used until today (even though in upgraded versions). Some protocols, however, are not used anymore since the WorldWideWeb was introduced, as the telnet protocol (CARR, 1969). Developed in a trustworthy environment, this protocol did not encrypt the transmitted data and had no authentication procedure.

Some of these bugs were known by the epoch, and in 1988 the first worm was created by Robert Morris (FBI, 2018). Worms are self-replicating programs that can spread through the network without any specific target software, unlike viruses, which must have a host program. Morris worm, as it got known, launched a Distributed Denial of Service (DDoS), and during several days the entire Internet – composed of around 60,000 computers – was clogged up (MURPHEY, 2019).

Since then, cybersecurity became an official preoccupation to researchers and Internet users. After the Morris attack, the Computer Emergency Response Team (CERT) (CMU, 2021), which is still working in the field, was created to prevent such events from happening again.

2.2.1 Some common attacks

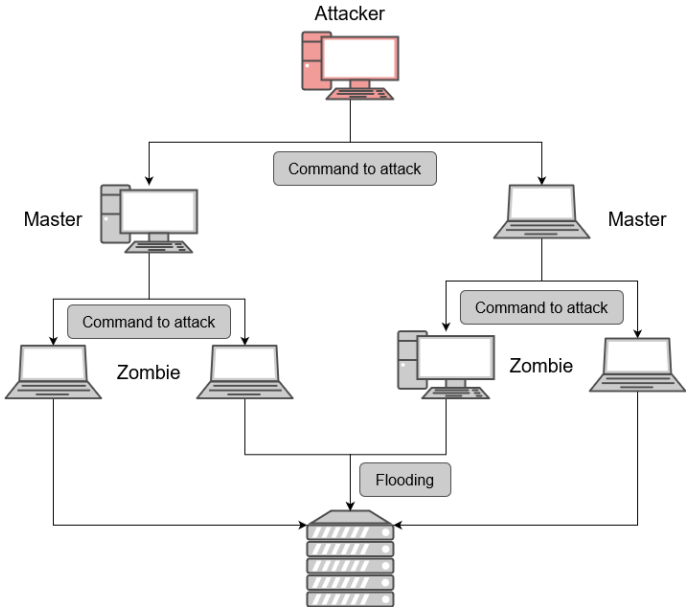
In this section, we are going to investigate some of the most common attacks on networks, from their first appearance in the network to more complex and elegant forms developed through the years.

There are relevant cyber threats that are not specified in this section. These types of attacks were considered out of the scope of this study, given the fact none of the tools found implemented them.

2.2.1.1 DoS and DDoS

Attacks of Denial of Service (DoS) have as objective to overload the capacity of response of a victim in such a way it can no longer respond to legitimate demands. There are two main types of DoS attacks (RUSSELL et al., 2001): (i) **resource consumption**, where the attacker targets bandwidth by using a number of packets that exceed the victim's capacity (known as *flooding*), and (ii) **malformed packet attacks**, which uses the fact that some operating systems are not able to process malformed headers, and may enter an unstable state.

Figure 2.5 – DDoS architecture.



Source: the author

One example of resource consumption is the SYN flood attack, in which the attacker takes advantage of a characteristic of the TCP three-way handshake, in which a server must wait for an answer from a client before establishing a connection. If the client

does not respond, a timeout is set, and the connection is closed. An attacker could send several packets using a fake source address (*spoofed*), and the server would try to connect to a non-existing host, wasting time and resources.

Malformed packets can occur in different layers and with different protocols. An attacker may send a package larger than the supported size (as it happens in Ping of Death attacks), or in a fashion that, during reassembly, the target system will not be able to reassemble the package (as happens with Teardrop and IP packets, and as Boink and UDP datagrams).

A Distributed DoS attack (DDoS) combines all the vulnerabilities cited above with a large network of controlled computers called zombies. A DDoS network has the structure shown in Figure 2.5, where an attacker sends commands to compromised computers called "masters" that, in turn, send commands to the "zombie" computers. The latter launches the attack against a specific target.

2.2.1.2 Brute force

The objective of brute force attacks is to find a password, passphrase, or another type of personal identification in order to get access to a system in a further operation. It also can be used to lockout an account since some services define a limited number of tries to access them before blocking any further attempts.

There are three types of brute force attacks, accordingly with (IBM, 2012): (i) **dictionary-based attacks**, which use a dictionary file to generate possible combinations of login and password, (ii) **search attacks**, which use all possible combinations of characters and lengths, and (iii) **rule-based search attacks**, which use a set of rules to generate possible passwords.

Some measures can be used to prevent this type of attack, accordingly with (COMMUNITY, 2020a):

- Add a pause in between authentication tries, once most brute force attacks are time-based.
- Insert key-phrases in HTML comments to fool automated analysis. Modifying the phrase used to inform the user the password is invalid at each attempt is also possible.
- Use a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart, (AHN et al., 2003a)), which are AI hard problems used to prevent

automated attacks.

- Allow login from a specific IP address, or assign unique URLs to some sets of users, so not all of them can access the service from the same URL.

Combining these methods is highly recommended, as well as not locking out completely an account, but rather limiting the user access.

2.2.1.3 Phishing

Phishing consists of sending fake messages to a target in order to obtain interaction, usually to spread malware and steal sensitive information (KNOWBE4, 2017). The most common method is spam emails, which are unwanted messages promoting some type of service or even pretending to be from a certain trustworthy company, but it can take other formats, such as a bogus login web page or a completely fake bank site (ALEROUD; ZHOU, 2017).

The authors of (ALEROUD; ZHOU, 2017) identify three phases in a phishing attack:

(i) **attack preparation**, when the attacker chooses the media they are going to use and the target devices. Based on that, they choose the ideal attack technique and prepare the material (forged website, email format) it is going to be used.

(ii) **attack execution**, in which there is material distribution, data collection, and finally target resource penetration.

(iii) **attack results exploitation**, when the collected credentials are used to impersonate the victim.

The authors also identified the following media as possible attack vectors: (i) emails, (ii) websites, (iii) instant messaging, (iv) online social networks, (v) blogs and forums, (vi) mobile apps, and (vii) voice over IP. Websites and forums, for instance, have a passive role, while emails and other active media are used to lead users to these compromised domains. Mobile apps can use some intrinsic frailties of a certain operational system, such as floating attacks in Android, where an application may have its UI on top of another, using this to steal sensitive data like login information.

One subgroup of phishing attacks is spear phishing. This technique aims at specific organizations and uses the personnel's personal data in order to obtain sensitive information. It can collect information from social networks and, from there, create an

attack vector that looks legitimate, as an email impersonating a superior. It is possible to use Natural Language Processing (NLP) to convince victims to rely on malicious content (SHROPSHIRE, 2018). This technique is based on taking a large text sample of a target and mimicking their writing style.

2.2.1.4 Cross-Site Request Forgery and Cross-Site Scripting

Accordingly, with the OWASP Top Ten of 2020 (COMMUNITY, 2017), Cross-Site Scripting (XSS) is in seventh place in a list of most common web application security risks. It is an improvement, once it was given second place in 2010 by the same organization, (SENTAMILSELVAN, 2013). In this same classification in 2010, Cross-Site Request Forgery (CSRF) was also identified as a menace, being given fifth place.

Frequently seen as the same type of attack, XSS and CSRF use, in fact, different techniques. CSRF forces a victim to perform unwanted actions on a logged site without their knowledge. The most common CSRF technique is to use an HTML tag, such a <*img*> with a modified source, to pass to the server some type of command. Let's say a user is connected to an email provider called *samplemail.com* and access the attacker's site *attacker.com/attack.html*. The HTML of *attacker.com/attack.html* consists of multiples tags, from which one is the following:

```
<img src="samplemail.com/user/changepassword.php?password=h4ck3d"/>
```

When the page *attacker.com/attack.html* is loaded, the password is changed without the knowledge of the user. It takes advantage of a characteristic of the HTTP protocol, which is sending a cookie to the user once they are logged in to a server. This cookie is then used by the attacker to forge a request and have the victim's access right to the website's sensitive functions.

Cross-Site Scripting, on the other hand, is more of an active attack since it uses injected scripts to execute privileged functions (COMMUNITY, 2021). There are three types of XSS (COMMUNITY, 2020b):

- **Stored XSS (Persistent/Type I)** which happens when the user input is stored in the server (as in a database) and then rendered on the browser.

- **Reflected XSS (Non-persistent/Type II)** which returns the user input directly in an error message or in another form of response and is shown in the browser as-is.

- **DOM Based XSS (Type 0)**, where the dataflow happens entirely on the browser,

and the source of malicious data and the method that processes these data are in the same DOM tree.

2.2.1.5 Ransomwares

This malware (malicious software) has as objective to deny the access of a user to the data stored in a hard drive, giving access back only after ransom payment. It can be done in two ways (RICHARDSON; NORTH, 2017): (i) encrypting data or (ii) locking the computer.

Crypto ransomware encrypts important files on the victim's computer, so the user has no access to them. However, it is still possible to execute some basic services. Lockers, on the other hand, prevent the victim from accessing their devices, rendering them useless. That is the reason why locker ransomware usually charges their ransom by voucher systems (RICHARDSON; NORTH, 2017), while crypto ones use Bitcoin (BöHME et al., 2015) for its anonymity and difficulty to trace.

Some of the most common infection vectors are phishing emails, websites, and web applications (RUBENS, 2017). Other methods include malvertising (advertisements that deliver malware when interacted with), social media and messaging apps, and exploit kits.

3 STATE-OF-THE-ART

This chapter contains a brief description of two critical concepts further discussed in the state-of-the-art: the application of AI in a defense context and the usage of AI in attacking. Once these concepts are defined, the state-of-the-art is separated into an academic side, based on papers published by researchers, and an industrial side, where tools developed by individuals and enterprises are aggregated. These tools were not designed as weapons but are included here, instead, by their potential to cause harm.

The tools presented here were first selected based on my advisors' initial set of papers due to their relevance to this research. Based on the references of these papers, as well as further analysis in research databases such as IEEE Xplore (IEEE, 2017), I gathered the collection of works contained in this study. It is important to note that the literature review is focused on articles that explicitly described cyberattacks enabled by AI algorithms. Many results focused on defense techniques rather than the attack structure; these articles were not included in this state-of-the-art description.

3.1 AI applied to defense

As soon as the computer networks started being used, the need for protection was noticed. Simple authentication systems were not enough, as some administrators discovered, and more complex techniques started being used. One of the first methods was the Intrusion Detection System (IDS), developed by Dorothy Denning and Peter Neumann in 1986 (BRUNEAU, 2001). It consisted of a rule-based expert system and served as a basis for the Next-Generation Intrusion Detection Expert System (NIDES).

First theorized in 1949 by John von Neumann in his paper *Theory of Self-Reproducing Automata* (SCHWARTZ; NEUMANN; BURKS, 1967), viruses only started circulating on computer systems in 1971, with the Creeper virus. Immediately after its release, Ray Tomlinson was able to create a program responsible for deleting Creepers, called The Reaper (COREWAR, 1971). It was the first known prototype of an antivirus. However, the first commercial antivirus was developed by G DATA Software in 1987 and targeted Atari ST systems (GDATA, 2020).

However, none of these solutions were artificially intelligent: they were based on a static set of rules and could not adapt to the data. This scenario changed in 2002 when Paul Graham created the first spam detector, based on a Bayesian system (GRAHAM,

2002). It was able to detect with a 99.5% accuracy rate, with a rate of false positives near zero.

Current defense systems already use Machine Learning (ML) techniques to diagnose and avoid common virtual attacks, as botnet detection (CHIGOZIE-OKWUM; AJAH, 2019), ransomware detection (VINAYAKUMAR et al., 2019; CHEN et al., 2018) and intrusion detection (KHAN et al., 2019). These solutions are based on the analysis of previous data. They can be of two types: **anomaly based**, where unusual behavior is reported as a possible threat, and **pattern based**, when a situation is compared with previous attacks registered.

In some cases, the fragility of AI systems can be used against itself, as happens with CAPTCHA tests (AHN et al., 2003b). These tests are based on the current difficulty of identifying images using machine learning algorithms and are considered a reliable system to differentiate between users and automated tools. However, with the advancement of technology and techniques, machines are obtaining higher scores in detecting such images, which leads us to believe that sooner this type of protection will not be enough.

ML techniques can also be used in an indirect way to detect attacks. One example is shown in (CHAMBERS; FRY; MCMASTERS, 2018), where a Natural Language Processing (NLP) algorithm analyzes social media texts to detect denial-of-service attacks. This technique takes into account the **symptom stage**, when users observe anomalies on the service; the **inference stage**, when some guesses about the symptoms are presented; the **confirmation phase**, where the website confirms the attack; and the **resumption phase**, when the service is restored.

3.2 AI applied to attack

It did not take a long time before attackers could see the potential of AI-enabled attacks. Automation has always been one of the most used techniques by attackers, especially when dealing with large-scale attacks. However, automation may not be enough when dealing with complex systems or unknown environments: some intelligence and analysis are needed.

In the early days of AI applied to cybersecurity, a self-learning worm was proposed in (CHEN; JI, 2005). It was based on importance scanning and learned about the environment in which it was without exchanging information with other infected hosts.

Due to this rogue behavior, the authors propose distributed detection systems to detect it during the learning phase and disable the server before the importance scanning.

Nowadays, with the advance of AI, new solutions targeting these same systems are appearing. For example, intelligent botnets can use Genetic Algorithms (GA) to create strategies that cannot be easily identified by the defender system (HENRIQUES; DANZIGER, 2017). Furthermore, AI is starting to return relevant results when applied to tasks designed to fool machines, such as CAPTCHAs (SIVAKORN; POLAKIS; KEROMYTIS, 2016; GOLLE, 2008).

In the following subsections, we will examine some of the studies done by the academy and the industry. We will compare some of their basic features, such as the ML model used if they aimed at attack or evasion and if the tools developed are open to being accessed by the public.

Before that, however, we present the Cyber Kill Chain framework to localize each solution during an attack better.

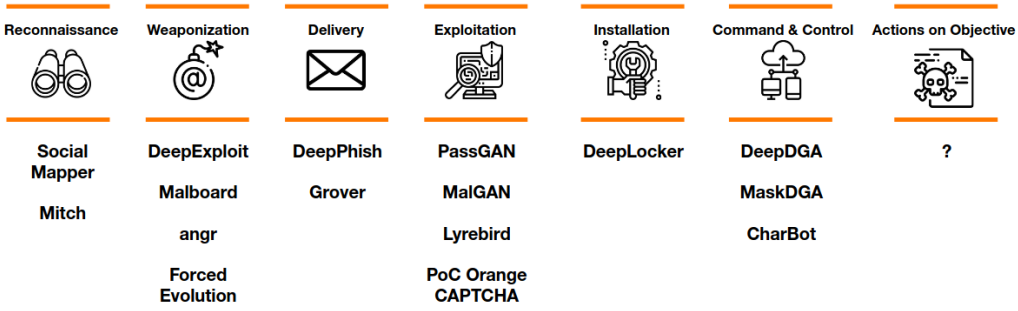
3.2.1 Cyber kill chain

A cyberattack can be divided into several steps, and each of them is classified by the Cyber Kill Chain framework, developed by Lockheed Martin (MARTIN, 2021). There are seven steps in this framework: **(i) reconnaissance**, where the attacker gathers information about the target; **(ii) weaponization**, where an exploit is coupled with a payload; **(iii) delivery** of the payload to the victim through an attack vector; **(iv) exploitation** of the vulnerabilities on the victim's system; **(v) installation** of the malware; **(vi) command and control (C2)**, where a command channel is created to remotely manipulate the victim's system; and **(vii) actions on objectives**, when the intruder can accomplish their original goal.

Using this framework, we can classify the following tools into the categories shown in Figure 3.1.

We verify that the further we are in the framework, the rarest are intelligent tools available. I particularly did not find any Action and Objectives tools, only some prototypes that were not fully developed, or just papers without source code available. Nevertheless, some of these tools were tested by me, as marked in sections 4.2, 4.1, 4.3 and 4.4.

Figure 3.1 – How each tool can be classified in the Cyber Kill Chain framework.



Source: the author

3.2.2 The academic side

In this subsection, we will explore some techniques proposed by researchers in the cyber defense area that can be used in an attack.

DeepExploit (TAKAESU, 2018)

- **What:** a framework that chooses the best combination of payloads to be sent to a target based on previous training.
- **How:** there are two phases: the training phase, where multiple exploits are used against an exploitable system; and the test phase, where the best exploits selected previously are sent to the target in an HTTP request.
- **Counter-measures:** since this tool is based on attacking open ports available for HTTP/HTTPS protocols, a basic sanitization protocol should be enough. Keeping the OS updated to the later fixes provided by the distribution must also be a habit. Also, this tool replicates learned attacks on vulnerable systems. Therefore honeypots could be used to detect this abnormal behavior quickly.

MalGAN (HU; TAN, 2017)

- **What:** a generative adversarial network trained on samples of malware to bypass a black-box detector based on ML techniques.
- **How:** the system consists of a generative network trained on a database of malware binaries and a substitute detector, which provides gradient information necessary to train the generator. The generative network creates adversarial malware examples that are fed to the black-box detector, as well as benign samples. This detector

outputs each sample's labels that will feed the substitute detector in supervised learning.

- **Counter-measures:** adopting ML techniques that cannot have its derivative calculated, such as random forests and decision trees, slightly improve the true positive rate. The authors also propose some solutions, such as retraining the models to deal with adversarial input (LI; VOROBEYCHIK; CHEN, 2016) and using auto-encoders to clean up the data (GU; RIGAZIO, 2014).

PassGAN (HITAJ et al., 2017)

- **What:** a generative adversarial network whose objective was to generate passwords that matched a distribution.

- **How:** the generative network was trained on two databases of leaked passwords and had an equivalent performance of the state-of-the-art. It could generate a larger quantity of passwords with a higher quality (human-likeness) than other methods.

- **Counter-measures:** most passwords generated were easily deducible (*e.g.* 12345, and first names). Therefore, using more complex passwords with a combination of numbers, lower and uppercase letters, and special characters are the best alternative.

Mitch (CALZAVARA et al., 2019)

- **What:** an ML tool to detect Cross-Site Request Forgery (CSRF) vulnerabilities. It focuses on visibly sensitive HTTP requests such as social media actions and online transactions.

- **How:** Mitch is a browser extension based on Random Forests that must be used in two stages: (i) the attack stage, where the security tester navigates the target website searching for sensitive requests, and (ii) the victim stage, where the security tester logs in as another user and Mitch exploits the prior knowledge as to replicate a CSRF attack. If a request made by the victim matches the one received by the attacker, a vulnerability is reported.

- **Counter-measures:** typical defenses against CSRF can be applied to the website, as well as developing these tools considering possible security flaws.

DeepLocker (KIRAT; JANG; STOECKLIN, 2018)

- **What:** the victim downloads this ransomware as a benign conference application, and it behaves like so until the webcam identifies the target. When activated, the ransomware acts like some other famous viruses, encrypting the victim's hard disk.
- **How:** The virus processes the image into an encryption key using a Deep Neural Network (DNN) and uses this key to activate the ransomware. The authors do not consider DeepLocker as an AI-enabled attack, but rather an AI-embedded attack once the trained model is sent to the victim with the application.
- **Counter-measures:** There is not yet a known method to avoid this type of attack, rather than being cautious on choosing which applications will have access to the webcam. One solution could be the use of adversarial noise automatically added to the real-time image, such that no software can identify the victims' image.

DeepPhish (BAHNSEN et al., 2018a)

- **What:** A DL solution to create more effective phishing URLs.
- **How:** It uses a Long Short-Term Memory Network to learn from previous successful phishing attacks and, from there, generate its URLs. The algorithm was able to improve the effectiveness of an attack on both scenarios analyzed.
- **Counter-measures:** this is a passive attack, where the attacker lures the victim with a sound-looking URL. There are two possible ways to avoid this attack: detecting these forged URLs as quickly as possible and blacklisting them. It can be done by an URL classifier, based on machine learning or not.

Malboard (FARHI; NISSIM; ELOVICI, 2019)

- **What:** method that captures the user keystroke pattern and reproduces it.
- **How:** It uses a clustering technique that analyzes the behavior of a user and, after a short period, can reproduce it. It is used in keystroke injection attacks and is difficult to identify due to its human-like speed.
- **Counter-measures:** Once this device is read as a legitimate keyboard, one possible solution is to have a two-factor authentication system. Detecting the abnormal behavior of the user can also be indicative that an attack is occurring.

MaskDGA (SIDI; NADLER; SHABTAI, 2019)

- **What:** a neural network tool that adds adversarial noise to a generated domain name, evading detection.

- **How:** instead of generating domain names from scratch, MaskDGA adds a mask to previously obtained names. These names are issued from a specific Domain Generation Algorithm (DGA).

- **Counter-measures:** as shown in DeepDGA, this tool itself could be used as a counter-measure. However, the authors further indicate that the algorithm could use other features besides character-level information to train a more robust detector.

DeepDGA (ANDERSON; WOODBRIDGE; FILAR, 2016)

- **What:** a domain name generator neural network designed to trick deep learning detection tools.

- **How:** first, an auto-encoder is trained on a database of domain names. After training, the auto-encoder is reassembled in a generative (encoder) and discriminative (decoder) network in a GAN fashion.

- **Counter-measures:** DeepDGA is, by itself, a counter-measure. The authors identify the fact that the algorithm could use a database containing legitimate domain names and synthetic ones to train a detector. This detector was proven to have high accuracy when trained with the hardened version of the dataset (from 68% to 70%).

CharBot (PECK et al., 2019)

- **What:** another domain name generator based on adversarial approach.

- **How:** unlike DeepDGA and MaskDGA, this tool does not use a simple algorithm to generate a DGA based on typosquatting (when some characters are randomly changed). It tries to minimize an adversary cost function at each step.

- **Counter-measures:** the authors have proven that CharBot could not be used to enlarge the training dataset to build a more robust detector. Instead, the authors proposed two feasible solutions: using a white-list system, where a domain is only accessed if permitted, and using side-information to train a detector, such as IP address, how often the IP is queried, etc.

3.2.3 The industry side

In this subsection, we analyze some approaches taken by the industry.

Lyrebird (LYREBIRD, 2017)

- **What:** a speech processing tool able to clone the voice of a target;
- **How:** attacks that depend on the execution of voice commands could use this tool. One example is the article *Using AI to Hack IA: A New Stealthy Spyware Against Voice Assistance Functions in Smart Phones* (ZHANG et al., 2019), where the voice of a target is used to unlock private data from their phone. With the augmentation of voice assistants, this type of attack is likely to increase.
- **Counter-measures:** there are two possible solutions when dealing with such attacks: avoiding suspicious downloads that may contain this malware and use two-factor authentication to access certain functionalities and data.

angr (UCSB; ASU, 2015)

- **What:** a framework for analyzing binaries, enabling the user to find vulnerabilities and generate exploits.
- **How:** developed in Python, it can load a binary, disassemble it, execute it symbolically and analyze its control-flow and data-dependency.
- **Counter-measures:** angr can be used, in fact, as a defense tool if the binary is analyzed by it before its release.

Grover (GROVER, 2019)

- **What:** a neural tool for generating and detecting automatically generated fake news.
- **How:** the researchers understood that the best neural network for detecting artificially-generated news is, in fact, a generator of such content. Therefore, Grover can tell apart the great majority of human-written and machine-written news.
- **Counter-measures:** tools as Grover can be used against themselves once it is demonstrated that it can identify this type of text.

Social Mapper (WILKIN, 2018)

- **What:** an intelligence tool used to gather information on social networks of a target given their image. Teaming operations can use it in reading, and it produces reports that a human reader can understand.
- **How:** works in three stages: (i) target parsing, where the target's image and name are furnished, (ii) social media search, when the previous data is used in several social media to find the best matches, and (iii) report generation, when all data is compiled in a CSV or HTML file.
- **Counter-measures:** it is recommended to keep the quantity of information about yourself in social networks to a minimum.

PoC Orange CAPTCHA (CHENU, 2019)

- **What:** a NodeJS application specifically designed to break the Orange CAPTCHA. This CAPTCHA consists of clicking, in the right sequence, the corresponding images.
- **How:** it uses two APIs: Google Vision to identify the correct class of each image displayed, and Puppeteer, which allows the manipulation of a web page as a human would through a browser.
- **Counter-measures:** the addition of adversarial noise on the images, as proposed in the DeepLocker counter-measure, is one manner to avoid this type of attack.

Forced Evolution (SOEN, 2013)

- **What:** a tool that finds the best string to apply in a SQL/command injection attack.
- **How:** it uses a genetic algorithm that identifies, at each new epoch, the best string to launch an injection attack against a target.
- **Counter-measures:** classical defenses against SQL and command injections must be applied to all implementations, as is recommended by common digital hygiene. These defenses are input validation, parameterized queries, and escaping techniques;

4 EVALUATIONS AND EXPERIMENTATIONS

This chapter dedicates itself to analyzing seven different projects developed in three distinct areas of AI-enabled cybersecurity. Five of them are grouped in the CAPTCHA breaking domain, and I chose them based on documentation and accessibility.

A second area analyzed was the intelligent malware generation, represented by a promisor algorithm called DeepExploit. I selected this tool based on the impact its presentation had on the BlackHat Europe Conference in 2018.

The third area, phishing generation, is represented by DeepPhish and is one of the first algorithms that use adversarial learning, without a GAN, to generate convincing phishing URLs. It also had a significant impact on this same conference and is considered the state-of-the-art in this domain.

Given the current stage of such technologies, most tools are still prototypes and can have lower accuracy than predicted.

Besides these tools, created by other authors and researchers, I prototyped an experimental tool on password breaking and natural language processing. The idea was to explore an area of AI-enabled attacks that I had not seen in any articles that I had analyzed, which is password guessing based on a password distribution.

4.1 CAPTCHA solver case study

This section analyzes several intelligent tools designed to break CAPTCHA to verify if an AI-enabled attack is feasible in this domain.

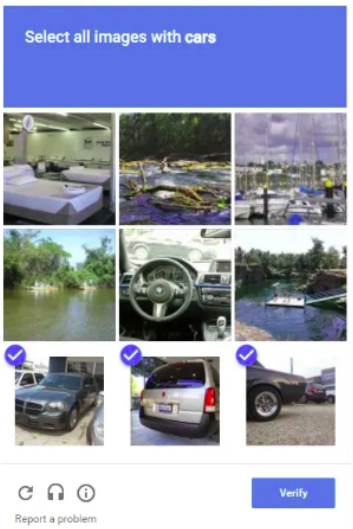
Firstly, we explain the concept of CAPTCHA, and different types of tests will be presented. Each type of test requires a specific machine learning solution with distinct algorithms. Then, we show the results of each tool and synthesize a conclusion on the state of such attacks.

4.1.1 What are CAPTCHA

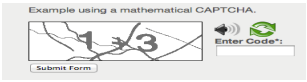
CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) are tests developed to prevent scripts and other types of automatic attacks from accessing specific pages and data. The most common type of these tests is the visual

Figure 4.1 – Four types of CAPTCHA commonly found on the internet.

(a) Identify images

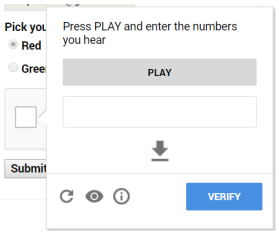


(b) Math problem



(c) Check click

(d) Identify phrase



Source: the author

CAPTCHA, which asks the user to identify objects in a noisy image. However, not all users can solve this type of test since some are visually impaired.

Based on that, the system gives alternative audio CAPTCHA. Since voice recognition is an area of machine learning that is highly developed, these tests are easily bypassed by automatic tools.

There are also several alternatives used in such tests (as seen in Figure 4.1), as math problems or word recognition (when a user is prompted to decipher a given text). These problems were once hard to solve by a computer but nowadays have fallen in disuse due to technological advances.

4.1.2 Tools and results

I tested five tools in this context, and I will further resume them in this Subsection.

Puppeteer reCAPTCHA Solver (GATIS, 2020) is a tool developed by Daniel Gatis, using an API called Wit. He took advantage of the fact that reCAPTCHA can use an audio test instead of a visual one since audio recognition algorithms have higher efficacy than image recognition. He used a JavaScript framework called Puppeteer to automatically choose the audio challenge, fill it up and submit. The API used can convert audio to text, and it is free.

However, I was able to realize only one successful test. The Google reCAPTCHA probably identified that my connection was automatized (using other means) and refused

all further tries to access the challenge.

unCAPTCHA (ECTHROS, 2018) is a tool specifically designed for an older version of reCAPTCHA present on the login page of Reddit. Since this page was deactivated, and the Wayback Machine version does not support external party software, I could not test it correctly.

reBreakCAPTCHA (EASTEE, 2017), developed by Yair Mizrahi from East-Ee, has a similar approach to the one taken by Puppeteer reCAPTCHA Solver: it selects and solves the audio challenge instead of the visual one. However, as happened with the previous tool, Google reCAPTCHA was able to detect the automatic connection by other means, and the algorithm could not solve any challenge.

captcha-break (LIU, 2018) is a set of 6 programs developed by Kalen Liu to break different visual CAPTCHA. These challenges consist of a string of letters and numbers modified in a way that a simple character recognition algorithm cannot detect all the symbols. The broken CAPTCHA are **csdn**, **jikexueyuan**, **submail**, **weibo.cn** and **weibo.com**. The algorithm used a basic image of characters for comparison.

PoC Orange Captcha (CHENU, 2019), developed by Thibeault Chenu, was used to attack a CAPTCHA system designed by Orange. This system consists of clicking, in the correct order, the images indicated by tags. The algorithm to solve this system used two Google services (Google Vision and Google Translate) first to classify all pictures, translate their tag to a set of french synonyms, and, finally, trigger the correct order of tags.

This PoC was developed for version 2 of Orange CAPTCHA. However, I was able to adapt it to version 3. It has 100% accuracy and takes a couple of seconds to solve the challenge.

4.1.3 Status and perspectives

I identified multiple tools and publications concerning the CAPTCHA breaking subject, and some of them have already some years since their first publication. Based on these facts, we can conclude that AI already has a practical application in these types of attacks, which needs only to be refined and adapted in some instances.

There are some CAPTCHA breaking services available, and they usually use human labor to solve these tests. Knowing about the constant evolution in this field, we predict that soon similar services will be available, this time using artificial intelligence.

4.2 Brute-force attack case study

The database BreachCompilation contains more than 1.5 billion tuples (email, password) obtained in multiple breaches throughout the years. This database allowed me to analyze different distributions on the passwords, given that each Top-Level Domain (TLD, the name that comes after '@' in an email) refers to a specific country and a specific language and culture.

The first part of this study dedicates itself to analyzing the distribution of the raw data; in the second part, synthetic passwords were generated using two methods (an LSTM network and a random generator); finally, in the third part, the performance of the generated passwords was compared to HashCat (STEUBE; GRISTINA, 2015), a password cracking tool. The objective of this study was to verify if an AI-based approach could be more effective than traditional methods.

4.2.1 Dataset distribution

In this phase, I selected two subsets belonging to the database: the one referring to *orange.fr* TLD and the one referring to *ufrgs.br* TLD. The first one belongs to a French telecommunications company, and the second one is from UFRGS. These TLDs were chosen by the fact that they belong to different countries (France and Brazil). Therefore, if the users create passwords containing words from their mother tongues, the password distribution would vary between TLDs from different countries.

Furthermore, these TLDs identify these institution's users but do not indicate that the breaches come from them. Many users utilize their professional/educational emails on other services (like social media), which can be breached.

To acquire features, I did a pre-processing treatment on these data, and 28 features were identified as best-performing: length, whether a password begins with a digit, the number of vowels, consonants, digits, hexadecimal and special characters, the size of the biggest stream of vowels, consonants, digits, hexadecimal and special characters, the Shannon / minimal / collision entropies, the Gini and Simpson indexes applied to 1, 2, and 3-grams). These features were pre-selected based on a previous study conducted by a colleague.

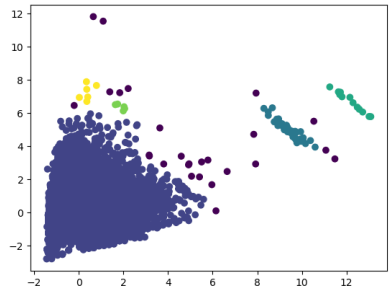
A PCA was used to obtain a visual representation of the data, converting this 28-dimension dataset into cardinal points in a plane. PCA is a method that projects high-

dimensionality data to a lower-dimensional space by maximizing the variance of each dimension. It finds the two ”best-fitting lines” representing the data, which builds an orthogonal basis (the principal components).

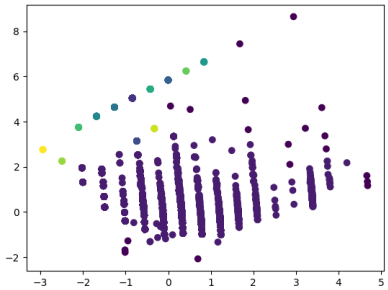
After applying the PCA, a DBSCAN clustering algorithm was used on the resulting data, obtaining different cluster formations and identifying outliers. A cluster contains points that have at least n-neighbors (core points) or are reachable by these points given a minimum distance. The other instances are considered outliers (noise).

In Figure 4.2 the distribution of the dataset composed by *orange.com* passwords are presented, and four types of passwords are compared: letters only (*e.g.* **mypassword**), digits only (*e.g.* **123456**), alphanumeric passwords (*e.g.* **mypassword123**), and symbols only (*e.g.* **$!**&@**). We verify that the alphanumeric and letters only passwords follow a natural language-like distribution, while the numeric passwords tend to be more sequential. Finally, the symbols only passwords do not show a significant distribution, and look random.

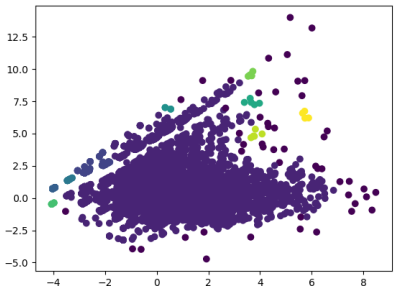
Figure 4.2 – Distribution and clustering of passwords given their composition. Axes refer to PC1 and PC2.



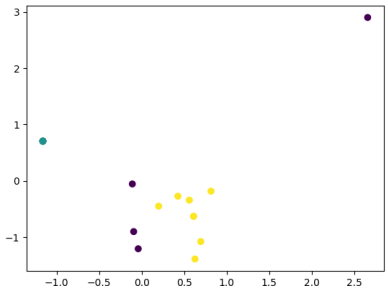
(a) Automatic, 4 neighbors, alphanumeric.



(b) Automatic, 4 neighbors, digits.



(c) Automatic, 4 neighbors, letters.



(d) Automatic, 4 neighbors, symbols.

Source: the author

4.2.2 Password generation

At this stage, an LSTM model was used to generate synthetic passwords that followed the distribution of the original dataset. I used the two previous datasets, and the difference was latent. When I converted the data to a low-dimensional form using PCA, the plotted results showed that the LSTM model had the most similar distribution, unlike the random generation.

I used two algorithms for password generation:

1. the first three symbols of one random sample in the training dataset were taken as a seed. For instance, if the password *123456* were in the training set, a viable seed to generate a new password would be *123*. In the following analysis, I call this "constant seed";

2. analysis the frequency of each letter, and random suggestion constrained by probability. One example would be the following: *a* has a frequency of 0.75, *b* of 0.15 and *c* of 0.10. A possible seed would be a random choice considering these probabilities (such as *aba*). In this section, this is called "random seed."

There was no significant difference between these two implementations, the first one being slightly better than the second. To verify if there was indeed a difference between the LSTM output and random strings, I plotted these two scenarios in comparison to the original data, as seen in Figure 4.3.

4.2.3 Comparison between methods

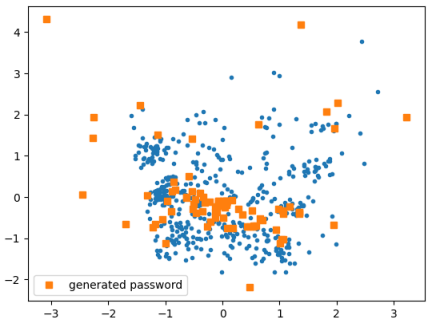
In this phase, I chose two dictionaries: the RockYou dataset (ROCKYOU, 2020), with approximately 14 million passwords; and the output of an LSTM model trained on a subsample of the target database.

There were four ways of generating data, as seen in Figure 4.4:

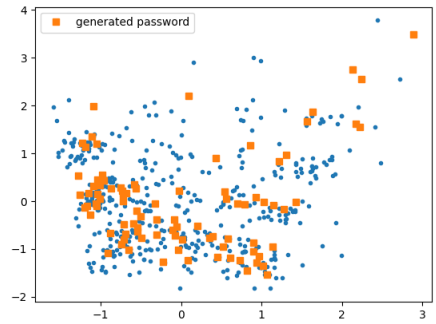
- **Scenario 1**: the attacker uses their knowledge on the distribution of passwords (represented by $n\%$ of the database) and uses an LSTM to generate fake passwords. The output is passed to HashCat as a dictionary, which then uses a hybrid attack to find matches.

- **Scenario 2**: the attacker has the same information about the distribution and applies

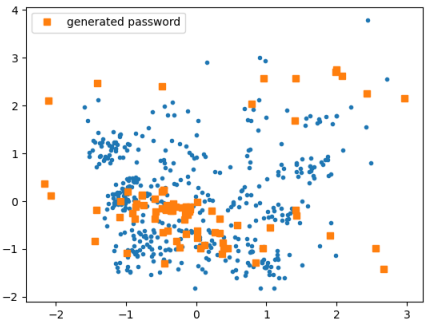
Figure 4.3 – Comparing generated and original passwords. Axes refer to PC1 and PC2.



(a) PCA of random strings and original data.



(b) PCA of constant seed-generated strings and original data.



(c) PCA of random seed-generated strings and original data.

Source: the author

it to the same LSTM. However, the output of the LSTM is directly compared with the rest of the database.

- **Scenario 3**: using the RockYou database, the attacker tries to find matches between the original passwords and the ones contained in this database.

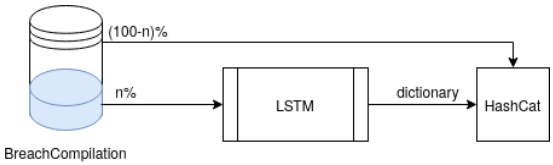
- **Scenario 4**: once again, the attacker uses the RockYou database as a dictionary in a hybrid attack. The rules of this attack are the same as **Scenario 1**.

I tested two types of cracking: direct collisions, where both dictionaries were used as-is, hashed, and then had their hashes compared to the target set; and HashCat dictionary-based hybrid attacks, where the dictionaries would be combined to some rules (such as adding integers at the end of the original password or modifying the case of some letters), using the HashCat tool as a middleman.

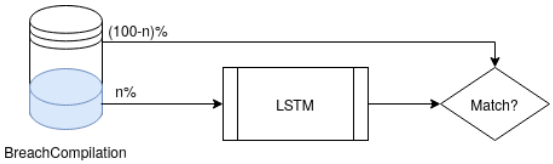
When testing in an extensive database, like the one provided by the TLD *orange.fr*, I took a small fraction of the original set (0.001%). However, when training the neural

Figure 4.4 – Four scenarios implemented.

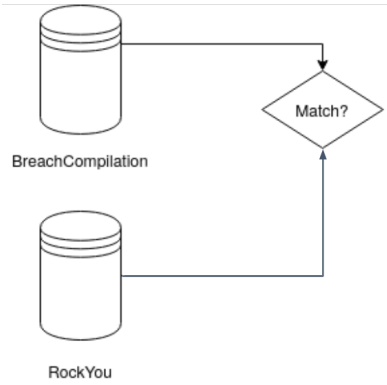
(a) Scenario 1: LSTM output as dictionary



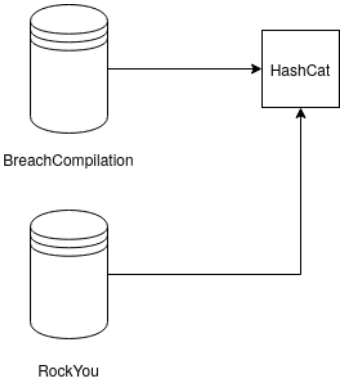
(b) Scenario 2: LSTM output for direct comparison



(c) Scenario 3: RockYou for direct comparison



(d) Scenario 4: RockYou as dictionary



Source: the author

network with a smaller dataset as the *ufrgs.br*, I used a more significant sample (5%). The idea, in both situations, was to have a basis on the distribution of data.

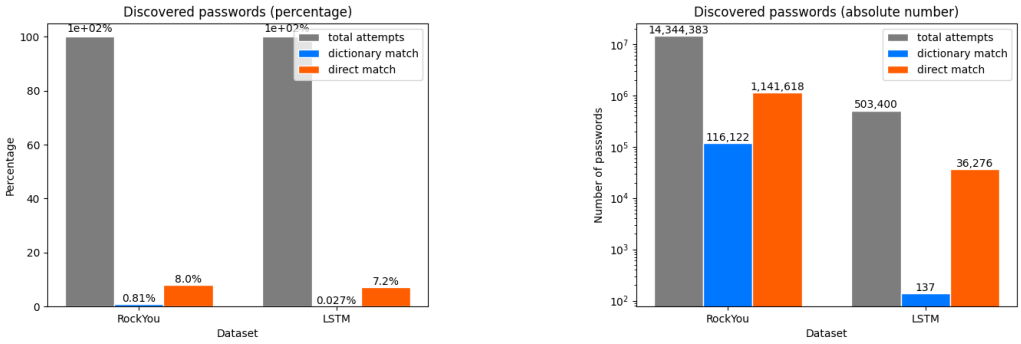
When comparing the number of samples generated by the LSTM and the number of entries in the RockYou database, we verify that the former is significantly smaller than the latter. Nevertheless, when comparing their performance on the four scenarios, the direct comparison of LSTM outputs (Scenario 2, Figure 4.4b) was proportionally more effective than the RockYou database used as a dictionary (Scenario 4, Figure 4.4d). Still, analyzing the numerical results, the RockYou dictionary provided more collisions when used with HashCat.

4.2.4 Status and perspectives

Brute-force attacks have two possible forms: online attacks and offline attacks. An offline attack allows the exhaustive search of the correct passwords, while online attacks may be interrupted after a certain number of tries. It is safe to assume, based on the results

shown in Figure 4.5, that an online attack would benefit from both the machine learning technique presented here, given that the distribution of the passwords is known and a large static dataset such as RockYou. However, in offline attacks, dictionaries can still have a more significant impact.

Figure 4.5 – Comparing techniques by percentage and magnitude.



(a) Percentage relative to each dataset and technique.

(b) Absolute magnitude of each dataset and technique.

Source: the author

In Table 4.1 we can verify the data represented in the graphic above. As we can see, the RockYou database is almost six times larger than the dataset containing unknown passwords, while the LSTM outputs represent only 20% of the total.

Table 4.1 – Relative numbers of each dataset.

	Unkown passwords	**RockYou passwords**	**RockYou direct matches**	**RockYou dictionary matches**	**LSTM passwords**	**LSTM direct matches**	**LSTM dictionary matches**
Absolute number	2,514,030	14,344,383	1,141,618	116,122	503,400	36,276	137
% of successful tries	-	100%	8.00%	0.81%	100%	7.2%	0.03%
% relative to original dataset	100%	570.57%	45.41%	4.61%	20.02%	1.44%	0.005%

Source: the author

It is essential to notice some critical points in the algorithm implementation: I trained the LSTM model on passwords with fixed size (in this study, eight characters). Given that passwords with different lengths tend to have different patterns, this constraint could limit the model's efficiency. Therefore the results seen here could have been improved by a more dynamic model.

Future works could apply several improvements to the algorithm, the main ones being:

- Combining other NLP techniques with the LSTM generation, since most of the passwords are alphanumerical and, as seen in Figure 4.2a, these passwords tend to

have a natural language-like distribution.

- Using a variable sequence length LSTM during training. Therefore, passwords containing few characters would be as precise as passwords containing multiple characters. During the training phase, I used a sequence length of 8, given that most passwords in the sample had this size, even though it could vary from 1 up to 32.

- Using a different set of features. The 28 features identified were proposed by a former colleague that worked on a similar project. However, none of us had previous knowledge of natural language processing, so a better combination of features could be more effective.

- Using another AI-based algorithm to select the ideal features for a given dataset automatically. Some of the most common algorithms used in this type of selection are genetic algorithms and Random Forest classifiers.

- Using algorithms to identify the best hyperparameters for the LSTM. The original network comprises an LSTM containing 256 units, a dropout layer with a dropout rate of 0.2, and a dense layer containing one unit per selected feature. Other hyperparameters were left as the Keras default once the idea of the project was to analyze the feasibility of such technique rather than optimize it. Creating a more refined network using well-tuned parameters will certainly increase accuracy and precision metrics and its F1-score.

- Combining several breaches. In this study, only one TLD was used to generate passwords. However, if we used more than one TLD from the same region, the number of matches would increase once the training set would be larger. TLDs from the same region should work in this case once they usually share the same language. For instance, we could combine the breaches of *ufrgs.br* and *edu.pucrs.br*, since both organizations are based in Porto Alegre/RS, Brazil.

4.3 DeepExploit

DeepExploit is a proof-of-concept tool developed by Isao Takaesu, presented at the BlackHat Europe 2018 conference. This framework selects the best payload combination to launch an attack against a target after some training on a server.

The training phase of this algorithm happens against an exploitable system to which the attack has full access, such as a Metasploitable machine. After identifying the main frailties of this system, the attacker uses this knowledge to launch a specific attack against the final target. A Nmap scan is done to determine open ports, and the algorithm sends the chosen payload through an HTTP request to the target.

In this study, three scenarios were analyzed. The results and conclusions are in the following section.

4.3.1 Results

In this study, four machines were used: **(i)** a Metasploitable machine, which is a 32-bit Debian deigned to have multiple exploitable features; **(ii)** a Microsoft Windows 8.1 Pro, x64-based; **(iii)** a 64-bit Ubuntu 16; and **(iv)** a 64-bit Kali Linux, which was the attack launcher.

The first scenario consisted of training the framework on the Metasploitable machine and testing it on Windows 8. Multiple frailties were found in the Metasploitable machine during the training phase, as expected, but when testing, only five of them suited the Windows OS. The attack could not be launched against Windows 8, even though all targeted ports were open to connections. The framework failed the first scenario.

The second scenario trained and tested the framework on the same Windows 8 machine. Once again, the Nmap identified all open ports, but the framework could not attack the system, even during the training phase. We conclude this framework is not ready to deal with Windows OS.

Finally, the third scenario was represented by a training phase on a Metasploitable and an attack phase on a Ubuntu 16. The training worked as predicted, but the attack phase failed at first because the open ports allowed only TCP connections instead of HTTP connections. After fixing the configurations on Ubuntu 16 to accept HTTP connections, the attack failed once again.

4.3.2 Conclusion

Even though some researchers have indeed found and fixed some bugs in the code of this framework, obtaining favorable results after that, I could not test the features of

this tool once it failed all tests.

The premise of this framework is interesting. However, it is counter-intuitive to assume that training on an OS that is finally different from the actual target will lead to successful exploitation. An exploit that works in a Debian 32-bit will most probably not work on a Windows 64-bit machine.

One possible application of such a framework would be its application on company networks, where there is usually a standard on the choice of OS and protocols. Suppose one machine in this network was vulnerable to external access. In that case, the attacker could use its configurations in a training server and then launch an attack against the entirety of the network.

DeepExploit is a new, non-mature approach to this type of challenge and should be further explored.

4.4 DeepPhish

This tool also had its first appearance in BlackHat Europe 2018, and the briefing had the same name as its paper (*DeepPhish: Simulating Malicious AI* (BAHNSEN et al., 2018b)). DeepPhish acts in four steps: **(i)** scores URLs contained in a phishing database from 0.0 (safe) to 1.0 (phishing); **(ii)** trains an LSTM using these URLs and scores; **(iii)** LSTM outputs URLs following a similar distribution to low-scored inputs; and **(iv)** the generated URLs are evaluated, and the accuracy of the tool is calculated.

4.4.1 Evaluation

I tested the demo available in (BAHNSEN, 2018), but I was not able to run the code as-is once it used a paid third-party software to classify the URLs in safe/phishing. I chose a free version, CheckPhish (CHECKPHISH,). The latter is still in a development phase, and its results were not suitable for this experiment.

CheckPhish classified all URLs as safe. This anomaly led to a 100% acceptance rate of the URLs produced by DeepPhish since the detection algorithm used to tune the LSTM was incorrect. Therefore, I could not verify the actual efficiency of DeepPhish.

Using the original external service, the authors had shown that 0.69% and 4.91% of the original URLs from two threat actors were effective. They improved these rates up

to 20.9% and 36.28%, respectively, by training DeepPhish on these datasets.

Such improvements, up to 30 times the initial accuracy, indicate that this approach is functional and practical.

4.4.2 Conclusion

Two solutions are possible: develop a personal URL classifier or find a paid version with higher accuracy. This study was, in fact, a quick analysis of this tool. Therefore I did not implement any of these solutions.

There are two possible counter-measures to this type of attack: URL analysis, as proposed by the paper, and a deep analysis of the page's content. The first one is quicker but less accurate than the second.

Still, on the URL analysis, one could implement a simple system of black- and white-lists, only allowing the user to access specific URLs (white-list) or preventing them from accessing other (black-list). Another possibility would be training detectors by using the outputs of such attacking tools in an adversary methodology.

5 SYNTHESIS

5.1 Conclusion

The question to be answered by this study was if one could use artificial intelligence to propel cyberattacks, and if so, how to take measures against such threats. I found multiple tools, and some were successfully tested, indicating that this type of attack is possible. Many vulnerabilities were found in these threats, also, which gives us indications of possible patches.

We verified that the farther we are in the Cyber Kill Chain framework (from Reconnaissance to Action & Objectives), the fewer prototypes are available for testing, even when considering ongoing prototypes. Such scarcity indicates how early in developing such technologies are.

Another point noticed is that the great majority of tools found are, in fact, in their early development stages. They contain several bugs and inefficiencies and cannot be applied as-is in real situations. It could indicate two possible cases: these tools are being developed by private companies that are not releasing their findings, or most solutions are in their infancy.

5.2 Perspective

A possible improvement to this study would be prototyping a countermeasure to one of the case studies, such as the brute-force attack on BreachCompilation. Even though the goal of conducting this work was to analyze these tools, the main objective of such study is, in the end, to be able to design and improve defense systems against these types of attacks.

Several domains were identified as possible study subjects, but three domains, in particular, were identified as more promising for consideration in further research, given their state-of-the-art.

5.2.1 Intelligent spear phishing

Previous works have proven that the generation of phishing emails is possible using ML techniques if the correct features are selected. Then, it is possible to extend this technique to cover more directed attacks such as spear phishing. A possible improvement to this study would be prototyping a countermeasure to one of the case studies, such as the brute-force attack on BreachCompilation. Even though the goal of conducting this work was to analyze these tools, the main objective of such study is, in the end, to be able to design and improve defense systems against these types of attacks.

Several domains were identified as possible study subjects, but three domains, in particular, were identified as more promising for consideration in further research, given their state-of-the-art.

A tool such as Social Mapper could be used as information gathering, and a Natural Language Generator could use these data as input to generate a targeted phishing mail. The exercise here would combine the information obtained by the information gathering tool and the text generated by the Natural Language Generator algorithm convincingly.

Authors of (DAS; VERMA, 2019) did an initial study on this subject, but the performance of such a tool was not very relevant. However, if combined with the text analysis done in (YAO et al., 2017), many improvements could be made. (YAO et al., 2017) demonstrates a text generation technique that a natural text can easily mistake.

The connection between a tool such as Social Mapper and a text generator in a manner it could generate tricky texts is a subject that can be a source of a complete study.

5.2.2 Exploit generator

One possible application of genetic algorithms is the generation of exploits adapted to a particular environment. The environment would be considered a combination of the machine's configurations, and the building blocks of each exploit would be the genome.

The example of the genetic algorithm given by Forced Evolution (SOEN, 2013), as mentioned above, could be used as a basis. However, instead of using a string to launch an injection attack, this tool would convert a string to a set of building blocks.

A first try has been done with the development of AEG (Automatic Exploit Generator), as explained in (AVGERINOS et al., 2011). However, AEG is environment-centered, which means finding a bug and generating a specific exploit to it. A possible

approach would be to expand the attack surface of such methods.

5.2.3 Automated network compromising

This mechanism would identify the best attack route inside a network, given it had access to one machine belonging to it. It can be thought of as a virus that infects a network and keeps communication between infected devices to exchange data about possible forms of expansion.

It would be a Command & Control solution once the algorithm would do the infection of the network in an automatized manner. The basis algorithm could be a genetic algorithm, as proposed previously, or a reinforcement learning type, as Q-learning. The idea is to have some software able to adapt itself to each machine according to its configurations.

The authors of (FANG et al., 2019) use Deep Reinforcement Learning to evade anti-malware detection systems. Still, the idea here would mainly infect other computers, with a possible concern about detection systems. A study on hivenets could be an entry to this subject, as proposed by Fortinet in 2018 (MANKY, 2017). Even two years later, not a single academic article about such an attack has been widely available.

5.3 Resources

Some sections of this document were based in previous documents I wrote. The links to the original versions are available in:

- Section 4.1, *CAPTCHA solver case study*, is available at (SAMPAIO, 2021a);
- Section 4.2, *Brute-force attack case study*, is available at (SAMPAIO, 2021b);
- Section 4.3, *DeepExploit*, and section 4.4, *DeepPhish*, are available at (SAMPAIO, 2021d).

I also implemented the parsing, analysis, and visualization tools to be applied in BreachCompilation, which are kept in a public GitHub repository (SAMPAIO, 2021c).

Thus, this study will make it possible to investigate new research tracks, as foreseen in the initial objective.

REFERENCES

AHMED, S.; WRIGHT, N. D. **Artificial intelligence, China, Russia, and the global order: technological, political, global, and creative perspectives**. Alabama, USA: Air University Press, 2019.

AHN, L. von et al. Captcha: Using hard ai problems for security. In: BIHAM, E. (Ed.). **Advances in Cryptology — EUROCRYPT 2003**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 294–311. ISBN 978-3-540-39200-2.

AHN, L. von et al. Captcha: Using hard ai problems for security. In: BIHAM, E. (Ed.). **Advances in Cryptology — EUROCRYPT 2003**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 294–311. ISBN 978-3-540-39200-2.

ALEROUD, A.; ZHOU, L. Phishing environments, techniques, and countermeasures: A survey. **Computers Security**, v. 68, p. 160 – 196, 2017. ISSN 0167-4048. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167404817300810>.

ALLEN, G. C. **Understanding China's AI Strategy**. 2019. Disponível em: <https://www.cnas.org/publications/reports/understanding-chinas-ai-strategy>.

ANDERSON, H. S.; WOODBRIDGE, J.; FILAR, B. Deepdga: Adversarially-tuned domain generation and detection. **Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security - ALSec '16**, 2016.

AVGERINOS, T. et al. Aeg: Automatic exploit generation. In: NDSS. **Proceedings of the Network and Distributed System Security Symposium, NDSS 2011**. San Diego, California, USA, 2011. v. 57.

BAHNSEN, A. C. **DeepPhish Demo**. 2018. Disponível em: <https://github.com/albahnsen/DeepPhish_BlackHat_Demo>.

BAHNSEN, A. C. et al. Deepphish : Simulating malicious ai. 2018.

BAHNSEN, A. C. et al. Deepphish : Simulating malicious ai. In: **BlackHat Europe 2018**. London, United Kingdom: BlackHat, 2018.

BRUNEAU, G. **The History and Evolution of Intrusion Detection**. 2001. Disponível em: <https://www.sans.org/reading-room/whitepapers/detection/history-evolution-intrusion-detection-344>.

BöHME, R. et al. Bitcoin: Economics, technology, and governance. **Journal of Economic Perspectives**, v. 29, n. 2, p. 213–38, May 2015. Disponível em: <https://www.aeaweb.org/articles?id=10.1257/jep.29.2.213>.

CALZAVARA, S. et al. Mitch: A machine learning approach to the black-box detection of csrf vulnerabilities. In: . Stockholm, Sweden: IEEE, 2019.

CARR, C. S. **Network subsystem for time sharing hosts**. 1969. Disponível em: <https://tools.ietf.org/html/rfc15>.

CHAMBERS, N.; FRY, B.; MCMASTERS, J. Detecting denial-of-service attacks from social media text: Applying nlp to computer security. In: **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)**. New Orleans, Louisiana, USA: Association for Computational Linguistics, 2018. p. 1626–1635.

CHECKPHISH. **Url Scanner to Detect Phishing in Real-time**. Bolster. Disponível em: <https://checkphish.ai/>.

CHEN, L. et al. **Towards resilient machine learning for ransomware detection**. 2018.

CHEN, Z.; JI, C. A self-learning worm using importance scanning. **Proceedings of the 2005 ACM workshop on Rapid malcode - WORM 05**, 2005.

CHENU, T. **PoC Orange CAPTCHA**. 2019. Disponível em: <https://github.com/tchenu/poc-orange-captcha>.

CHIGOZIE-OKWUM, C.; AJAH, I. Botnet identification using machine learning techniques: A survey. 07 2019.

CMU, C. M. U. **The CERT Division**. 2021. Accessed on 28.04.2021. Disponível em: <https://sei.cmu.edu/about/divisions/cert/index.cfm>.

COMMUNITY, O. **OWASP Top Ten**. 2017. Accessed on 28.04.2021. Disponível em: <https://owasp.org/www-project-top-ten/>.

COMMUNITY, O. **Blocking Brute Force Attacks**. 2020. Accessed on 28.04.2021. Disponível em: <https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks>.

COMMUNITY, O. **Types of Cross-Site Scripting**. 2020. Accessed on 28.04.2021. Disponível em: <https://owasp.org/www-community/Types_of_Cross-Site_Scripting>.

COMMUNITY, O. **XSS**. 2021. Accessed on 28.04.2021. Disponível em: <https://owasp.org/www-community/attacks/xss/>.

COREWAR. **Core War: Creeper and Reaper**. 1971. Accessed on 28.04.2021. Disponível em: <https://corewar.co.uk/creeper.htm>.

DAS, A.; VERMA, R. Automated email generation for targeted attacks using natural language. 2019.

EASTEE. **reBreakCAPTCHA repository**. 2017. Disponível em: <https://github.com/eastee/rebreakcaptcha>.

ECTHROS. **unCAPTCHA repository**. 2018. Disponível em: <https://github.com/ecthros/uncaptcha>.

FANG, Z. et al. Evading anti-malware engines with deep reinforcement learning. **IEEE Access**, v. 7, p. 48867–48879, 2019.

FARHI, N.; NISSIM, N.; ELOVICI, Y. Malboard: A novel user keystroke impersonation attack and trusted detection framework based on side-channel analysis. **Computers Security**, v. 85, p. 240 – 269, 2019. ISSN 0167-4048. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167404818309957>.

FBI. **The Morris Worm**. 2018. Disponível em: <https://www.fbi.gov/news/stories/morris-worm-30-years-since-first-major-attack-on-internet-110218>.

GATELY, E. **Neural Networks for Financial Forecasting**. USA: John Wiley Sons, Inc., 1995. ISBN 0471112127.

GATIS, D. **Puppeteer reCAPTCHA Solver repository**. 2020. Disponível em: <https://github.com/danielgatis/puppeteer-recaptcha-solver>.

GDATA. **About G DATA**. 2020. Disponível em: <https://www.gdatasoftware.com/about-g-data>.

GOLLE, P. Machine learning attacks against the asirra captcha. In: **Proceedings of the 15th ACM Conference on Computer and Communications Security**. New York, NY, USA: Association for Computing Machinery, 2008. (CCS '08), p. 535–542. ISBN 9781595938107. Disponível em: <https://doi.org/10.1145/1455770.1455838>.

GOODFELLOW, I. J. et al. Generative adversarial nets. In: **Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2**. Cambridge, MA, USA: MIT Press, 2014. (NIPS'14), p. 2672–2680.

GRAHAM, P. **A plan for spam**. 2002. Disponível em: <http://www.paulgraham.com/spam.html>.

GROVER. **Grover: a State-of-the-Art Defense against Neural Fake News**. 2019. Accessed on 28.04.2021. Disponível em: <https://rowanzellers.com/grover/>.

GU, S.; RIGAZIO, L. **Towards Deep Neural Network Architectures Robust to Adversarial Examples**. 2014.

HENRIQUES, M.; DANZIGER, M. Attacking and defending with intelligent botnets. In: . São Pedro, São Paulo, Brazil: SBRT, 2017.

HITAJ, B. et al. **PassGAN: A Deep Learning Approach for Password Guessing**. 2017.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HU, W.; TAN, Y. **Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN**. 2017.

IBM. **IBM Docs: Brute force attacks**. 2012. Accessed on 28.04.2021. Disponível em: <https://www.ibm.com/docs/en/snips/4.6.0?topic=categories-brute-force-attacks>.

IEEE. 2017. Accessed on 28.04.2021. Disponível em: <https://ieeexplore.ieee.org/Xplore/home.jsp>.

JACKSON, P. **Introduction to Expert Systems**. 3rd. ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1998. ISBN 0201876868.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. **Reinforcement Learning: A Survey**. 1996.

KHAN, R. et al. An improved convolutional neural network model for intrusion detection in networks. In: **2019 Cybersecurity and Cyberforensics Conference (CCC)**. [S.l.: s.n.], 2019. p. 74–77.

KIRAT, D.; JANG, J.; STOECKLIN, M. P. **DeepLocker: Concealing Targeted Attacks with AI Locksmithing**. 2018.

KNOWBE4. **What Is Phishing?** 2017. Accessed on 28.04.2021. Disponível em: <https://www.phishing.org/what-is-phishing>.

LI, B.; VOROBEYCHIK, Y.; CHEN, X. **A General Retraining Framework for Scalable Adversarial Classification**. 2016.

LIU, K. **captcha-break**. 2018. Disponível em: <https://github.com/nladuo/captcha-break>.

LIU, W. et al. A survey of deep neural network architectures and their applications. **Neurocomputing**, v. 234, 12 2016.

LYREBIRD. 2017. Accessed on 28.04.2021. Disponível em: <https://www.descript.com/lyrebird>.

MANKY, D. **Fortinet FortiGuard Labs 2018 Threat Landscape Predictions**. 2017. Disponível em: <https://www.fortinet.com/blog/business-and-technology/fortinet-fortiguard-2018-threat-landscape-predictions>.

MARTIN, L. **Cyber Kill Chain®**. 2021. Accessed on 28.04.2021. Disponível em: <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>.

MURPHEY, D. **A history of information security - IFSEC Global: Security and Fire News and Resources**. IFSEC Global | Security and Fire News and Resources, 2019. Disponível em: <https://www.ifsecglobal.com/cyber-security/a-history-of-information-security/>.

NSCAI. **National Security Comission on Artificial Intelligence Interim Report**. 2019. Disponível em: <https://www.nscai.gov/wp-content/uploads/2021/03/Full-Report-Digital-1.pdf>.

O'CONNOR, T. **Russia is building a missile that can makes its own decisions**. 2017. Disponível em: <https://www.newsweek.com/russia-military-challenge-us-china-missile-own-decisions-639926>.

PECK, J. et al. **CharBot: A Simple and Effective Method for Evading DGA Classifiers**. 2019.

RAI, P.; SINGH, S. A survey of clustering techniques. **International Journal of Computer Applications**, v. 7, n. 12, p. 1–5, Oct 2010.

RICHARDSON, R.; NORTH, M. M. **Ransomware: Evolution, Mitigation and Prevention**. Kennesaw, Georgia, United States: Faculty Publications, 2017. 4276 p.

ROCKYOU. Wikimedia Foundation, 2020. Disponível em: <https://en.wikipedia.org/wiki/RockYou>.

RUBENS, P. **Understanding Ransomware Vectors Key to Preventing Attack**. 2017. Disponível em: <https://www.esecurityplanet.com/malware/prevent-ransomware-attack.html>.

RUSSELL, R. et al. Chapter 2 - ddos attacks: Intent, tools, and defense. In: RUSSELL, R. et al. (Ed.). **Hack Proofing Your E-Commerce Site**. Rockland: Syngress, 2001, (The Only Way to Stop a Hacker is to Think Like One). p. 45 – 118. ISBN 978-1-928994-27-5. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9781928994275500055>.

SAMPAIO, L. S. R. **Analysis of some audio CAPTCHA-breaking tools**. 2021. Accessed on 28.04.2021. Disponível em: <https://github.com/LaurenRolan/TCC/blob/main/documents/CAPTCHA_Analysis.pdf>.

SAMPAIO, L. S. R. **Analyzing BreachCompilation passwords and training an LSTM with them**. 2021. Accessed on 28.04.2021. Disponível em: <https://github.com/LaurenRolan/TCC/blob/main/documents/Breach_Analysis.pdf>.

SAMPAIO, L. S. R. **GitHub repository**. 2021. Accessed on 28.04.2021. Disponível em: <https://github.com/LaurenRolan/TCC>.

SAMPAIO, L. S. R. **Verification of DeepPhish and DeepExploit**. 2021. Accessed on 28.04.2021. Disponível em: <https://github.com/LaurenRolan/TCC/blob/main/documents/DeepPhish_DeepExploit.pdf>.

SCHWARTZ, J. T.; NEUMANN, J. V.; BURKS, A. W. Theory of self-reproducing automata. **Mathematics of Computation**, v. 21, n. 100, p. 745, 1967.

SENTAMILSELVAN, K. Survey on cross site request forgery (an overview of csrf). In: **IEEE - International Conference on Research and Development Prospects on Engineering and Technology (ICRDPET 2013)**. IEEE, 2013. Disponível em: <https://www.researchgate.net/publication/281583832_Survey_on_Cross_Site_Request_Forgery_An_Overview_of_CSRF>.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning: from theory to algorithms**. Cambridge, United Kingdom: Cambridge University Press, 2017.

SHROPSHIRE, J. Natural language processing as a weapon. In: **WISP 2018 Proceedings**. Alabama, USA: SIGSEC, 2018. p. 26.

SIDI, L.; NADLER, A.; SHABTAI, A. **MaskDGA: A Black-box Evasion Technique Against DGA Classifiers and Adversarial Defenses**. 2019.

SINGER, P.; FRIEDMAN, A. **Cybersecurity: What Everyone Needs to Know**. OUP USA, 2014. (What Everyone Needs To Know). ISBN 9780199918096. Disponível em: <https://books.google.fr/books?id=f_lyDwAAQBAJ>.

SIVAKORN, S.; POLAKIS, J.; KEROMYTIS, A. D. I'm not a human: Breaking the google recaptcha. In: . Marina Bay Sands, Singapore: BlackHat, 2016.

SOEN. **Forced Evolution: Evolving Exploits through Genetic Algorithms**. 2013. Accessed on 28.04.2021. Disponível em: <https://github.com/soen-vanned/forced-evolution>.

STEINLEY, D. K-means clustering: A half-century synthesis. **British Journal of Mathematical and Statistical Psychology**, v. 59, n. 1, p. 1–34, 2006. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1348/000711005X48266>.

STEUBE, J.; GRISTINA, G. **HashCat: Advanced Password Recovery**. 2015. Accessed on 28.04.2021. Disponível em: <https://hashcat.net/hashcat/>.

TAKAESU, I. **DeepExploit**. 2018. Accessed on 28.04.2021. Disponível em: <https://github.com/13o-bbr-bbq/machine_learning_security/tree/master/DeepExploit>.

UCSB; ASU. **angr**. 2015. Disponível em: <https://angr.io/>.

VINAYAKUMAR, R. et al. Ransomware triage using deep learning: Twitter as a case study. In: **2019 Cybersecurity and Cyberforensics Conference (CCC)**. [S.l.: s.n.], 2019. p. 67–73.

VISHAL, T. V. et al. A survey and comparison of artificial intelligence techniques for image classification and their applications. **International Journal of Science and Research (IJSR)**, v. 5, n. 4, p. 187–193, May 2016.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. **Machine Learning**, v. 8, n. 3, p. 279–292, May 1992. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/BF00992698>.

WILKIN, J. **Mapping Social Media with Facial Recognition: A New Tool for Penetration Testers and Red Teamers**. 2018. Disponível em: <https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/mapping-social-media-with-facial-recognition-a-new-tool-for-penetration-testers-and-red-teamers/>.

YANG, Y. **Artificial intelligence takes jobs from Chinese web censors**. Financial Times, 2018. Disponível em: <https://www.ft.com/content/9728b178-59b4-11e8-bdb7-f6677d2e1ce8>.

YAO, Y. et al. Automated crowdturfing attacks and defenses in online review systems. In: **Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security**. New York, NY, USA: Association for Computing Machinery, 2017. (CCS '17), p. 1143–1158. ISBN 9781450349468. Disponível em: <https://doi.org/10.1145/3133956.3133990>.

YENALA, H. et al. **Deep learning for detecting inappropriate content in text**. Springer International Publishing, 1970. Disponível em: <https://link.springer.com/article/10.1007/s41060-017-0088-4>.

ZHANG, R. et al. Using ai to attack va: A stealthy spyware against voice assistances in smart phones. **IEEE Access**, v. 7, p. 153542–153554, 2019.

ZHU, X. Semi-supervised learning literature survey. **Comput Sci, University of Wisconsin-Madison**, v. 2, 07 2008.