TOWARDS AN AGENT-BASED APPROACH TO SIMULATING HUMANS FALLING FOR PHISHING ATTACKS

by

Anibal J. Robles Perez

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Technology

Charlotte

2018

Approved by:

Dr. Thomas Moyer

Dr. Heather Lipford

Dr. Aidan Browne

Dr. Weichao Wang

ABSTRACT

ANIBAL J. ROBLES PEREZ. Towards an Agent-based Approach to Simulating Humans Falling for Phishing Attacks. (Under the direction of DR. THOMAS MOYER)

In this project, we explore methods of implementing an autonomous agent that simulates human behaviour, in order to produce common mistakes that create vulnerabilities in enterprise networks. The objective of this research is to aid in testing the security of existing networks when vulnerabilities are created by the actions of their users. The research was be split into two stages. First, we investigated different possibilities of implementing the autonomous behaviour and which types of actions should be supported. The second stage was to create a virtual machine (VM) that can be implemented in either physical or virtual networks, and test the agent in different scenarios by running it against anti-phishing tools. The tool downloads files from a website using two different methods, and we record if, and when, the anti-phishing tools blocks those actions. This autonomous agent can help administrators test their network's protection against unwanted downloads. It can also be used to see which anti-phishing tool is better at protecting their network by trying different configurations against the agent.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

These days, everyone has become a target of cyber attacks: from major companies and government departments, to users in their homes [3]. Around 79% of companies have faced at least one cyber attack last year [4], where 13% of total attacks were against government agencies, 7% were to financial companies, and 33% against to home users [5]. To protect themselves, companies hire penetration testers (pentesters) to test the security of their networks [6]. Pentesting is "the simulation of an unethical attack of a computer system or other facility in order to prove the vulnerability of that system in the event of a real attack."[7] Although pentesters use special tools such as vulnerability scanners [8] to gain access to a network, one of the greatest vulnerability that a pentester can exploit are the users themselves. Users in a network can, and do, make mistakes,which attackers have learned to use for their advantage.

Among the greatest causes of cyber attacks are human error, which accounts for around 90% [9] (found in one study by Willis Towers Watson [10]). These human errors can lead to damaging vulnerabilities in their networks, such as the famous Sony phishing attack in 2014 and the HIV status of patients in 2015 [11]. Among the types of errors users make are:

- having common and/or insecure passwords,
- not installing or deactivating anti virus,
- not updating their software,
- downloading sensitive information to their personal (and not secure) devices,
- opening untrustworthy emails and attachments. [12] [13]

Attacks that exploit weaknesses in the users are called social engineering attacks [14]. Within theses ones, spear-phishing attacks (or phishing attacks, if done on a wider

scale [15]) are attacks used to trick users into installing malware on their computers. With this project, we create an agent that simulates human behaviour that allows social engineering attacks, such as spear phishing, to be successful. In order to test the capabilities of the agent, we tested several anti-phishing tools. The goal is not to compare and contrast the capabilities of the anti-phishing tools. Instead, the purpose is to provide an example of how the agent can be used to test different network and system configurations. This will help us see what improvements can be made to the autonomous agent, and see how anti-phishing tools that are used in real-world scenarios will react to its actions.

1.1 PROBLEM STATEMENT

- Can we create an agent that simulates risky user behavior?
- Can we use this agent to test different network and system configurations, allowing administrators to understand the impact of varying configurations?

CHAPTER 2: BACKGROUND

With the invention of the internet, users can access information wherever a connection can be made. In "A brief history of the internet "[16], the internet is defined by the FNC (Federal Networking Commission) as:

> '"Internet" refers to the global information system that – (i) is logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons; (ii) is able to support communications using the Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP compatible protocols; and (iii) provides, uses or makes accessible, either publicly or privately, high level services layered on the communications and related infrastructure described herein.'

With the same technology used for the internet, companies create internal networks to share resources between employees and protect them from outside intrusion. However, it did not take long for attackers to gain unauthorized access to the private information. This was due to company employees needing access to the internet while working with company data.

2.1 NETWORK ATTACKS

According to S. Hansman & R. Hunt [17], any internet connected computer is at risk of acquiring some form of malware (malicious software). In an IBM Security report [18], insiders were responsible for 55% of network attacks in 2015. Out of that, 23.5% were unsuspecting users. This involved leaving company equipment, such as laptops, unattended where attackers could take them, using insecure internet websites, and making weak passwords that are easy to guess. Another issue is phishing attacks, which has become the preferred method of attack by 71% of organized groups in 2017

[19]. The attack occurs when users open an innocent looking email/IM and downloads content thinking it came from IT or some other internal department. Emails can contain worms or trojan horses that can cause serious breaches in the network [20]. One recent phishing attacks was carried out against Breitbart [21]. The perpetrator impersonated Steve Bannon in emails leading insiders to discuss sensitive information, even when there were inconsistencies in the emails they received. And by far one of the most damaging attacks, suspected of being propagated by phishing, was the WannaCry incident that affected over 230,000 people in more than 100 countries [21].

2.2 NETWORK SECURITY

Companies have to analyze how at risk their assets are and use security measures to manage that risk [22]. There are different approaches to securing a network, including managing internet connections. This is handled by various hardware and software components such as firewalls, intrusion prevention systems, content security, secure wireless networks, identity management, and compliance management [23]. But even these measures can't control the actions of employees. Companies are trying to implement various means of mitigating the issues brought on by employees. Some of the measures are through policies such as setting password strength, reuse, and replacement.

2.3 HUMAN FALLIBILITY

There is enough evidence that shows that users have to safeguard themselves against the threats of phishing attacks when using the internet, since no one is exempt from being a victim. Security professionals have seen an increase in social engineering attacks delivering malicious software directed at employees in enterprises each year [24]. McAfee alone detected 57.6 million new samples in their third quarter of 2017 [25]. Even with all this evidence, users are either unaware or trust the internet too much to safeguard themselves, which increases the risk of falling victim to

cyber threats. Much of this can be attributed to the users having little knowledge about how the internet works, as evidenced by R. Kang in [26]. Users may think that popular services such as Amazon and Gmail are always secure, or base their trust in the website on their real life experience of a physical store. This is an example of a response a participant gave:

> "I talk to the employees there in person a lot, and they just seem to have a level head on their shoulders. I don't think they would give out their information to anybody over the phone without verifying who they were with some kind of credential verification." [26]

Having blind trust upon the services may lead users to not take preemptive measures such as using antivirus software, installing anti-phishing browser extensions, or not following links in emails before confirming their destination. People who are unaware of the dangers of cyber attacks should be of high concern to employers. It is recommended to train employees about cyber threats and test them on their knowledge afterwards [27], but still many employees do not follow through with what they learn. These days, many features are present to try to protect users from dangerous or insecure websites. Browsers now show users when their connection to a website is not secure (e.g.. not an encrypted connection in websites that have password or credit card input fields) [28] [29].

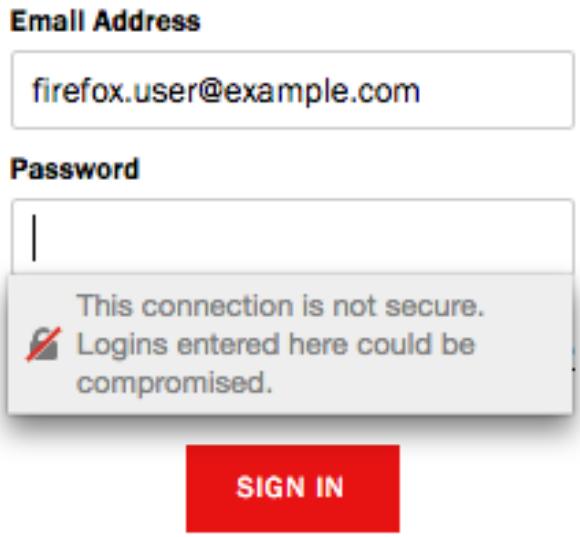

Figure 2.1: Firefox's warning of unsecure login.

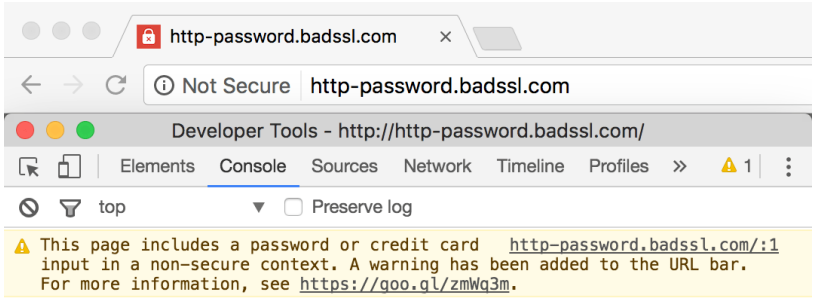

Figure 2.2: Google Chrome's warning of unsecure website.

In a telemetry data study done by the Mozilla Corporation [30] [1], login forms detected by Mozilla from 2016 to 2017 were found to have increased the use of HTTPS from 40% to 70%.

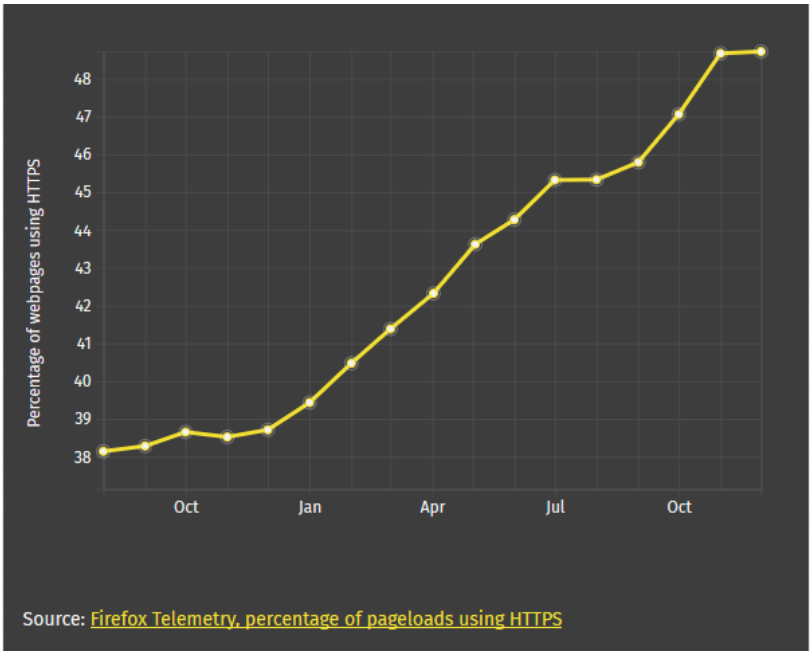

Figure 2.3: Mozilla Telemetry Data [1]

By Mozilla Firefox and Google Chrome, which are among the most popular desktop browsers [31], deciding to mark any website as "Not Secure" if it wasn't using HTTPS [1][32], it has convinced website owners to implement security in their website so as to not appear as a suspicious websites to potential visitors. Google also announced that they have seen an increase in protected traffic: 68% in Windows and Android, and 78% in Chrome OS and Mac.

In a 2005 study by R. Dhamija, J.D. Tygar, and M. Hearst called "Why Phishing Works"[33], participants were shown 20 websites (only seven being legitimate websites) and out of the 22 participants, 90% failed to identify phishing websites. According to their cognitive walkthrough on approximately 200 sample attacks within the "Phishing Archive"(maintained by the Anti Phishing Working Group), phishing websites exploit the user's lack of technical knowledge, use visual deception, and the user's lack of attention, to trick them into falling for the attack. Among these techniques are:

Forging an email header to appear as being sent from another entity [34]. Overlay-

ing images of hyperlinks over real links to redirect the user when clicked. Images that look like browser windows so users click on them. Pages that imitate real websites (perceptive users might notice that the URL is different). Replacing letters in an URL with similar looking characters (‘1’for ‘I’, or ‘O’for ‘0’), or inserting characters that are not printed in the address bar.

But recent studies, such as a 2013 study D. Akhawe [2] about clickthrough rates in users who saw and bypassed Mozilla Firefox and Google Chrome’s security warnings, there was evidence that in the majority of cases, more than half of users did heed the warnings.

Operating System	Malware		Phishing	
	Firefox	Chrome	Firefox	Chrome
Windows	7.1%	23.5%	8.9%	17.9%
MacOS	11.2%	16.6%	12.5%	17.0%
Linux	18.2%	13.9%	34.8%	31.0%

Figure 2.4: Clickthrough rates based on Operating System. [2]

Channel	Malware		Phishing	
	Firefox	Chrome	Firefox	Chrome
Stable	7.2%	23.2%	9.1%	18.0%
Beta	8.7%	22.0%	11.2%	28.1%
Dev	9.4%	28.1%	11.6%	22.0%
Nightly	7.1%	54.8%	25.9%	20.4%

Figure 2.5: Clickthrough rates based on Release Version. [2]

This was also the case in a 2015 study by M. Alsharnouby [35], where participants only detected 53% of phishing websites, although it was found that the technical expertise of the participant was not related to the success/failure rate, but rather how much time they spent looking at the security indicators in the browser.

But recently, attackers are using HTTPS for their own advantage. In an Anti-Phishing Working Group 2017 report [36], it is shown that new phishing websites are

using HTTPS encryption to lure users in with a false sense of security. The example shown on their report shows that the phisher obtained their encryption certificate from Let's Encrypt, a website which provides free certificates [37]. It is stated in the report that from PhishLabs' 54,631 unique phishing sites they examined in the third quarter of 2017, almost a quarter were using HTTPS.

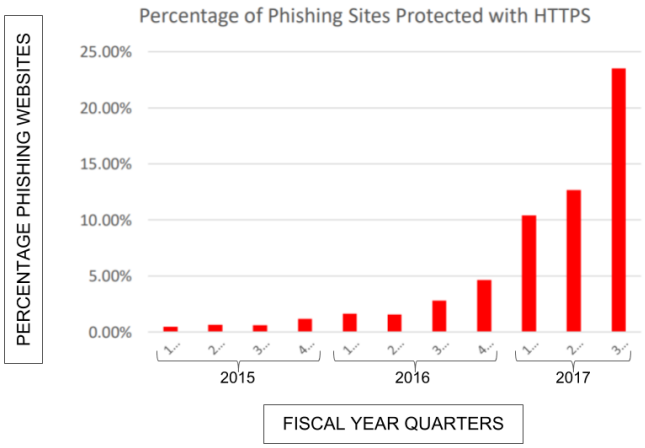

Figure 2.6: Percentage of phishing sites using HTTPS.

We can conclude from these studies that although there is an increase in user awareness, thanks to browsers taking measures to enforce internet security, there is still a large percentage (more than 40% in one case of the clickthrough rates) of users who still fall for phishing websites. From the 2005 to the 2015 study, we can see that there was an increase of users who were able to distinguish phishing websites from safe ones, and we can attribute that to more 'intrusive' safety warnings from the browsers. Still, the clickthrough rate study was just in 2013, and the 2015 paper saw that users only detected up to 53% of phishing sites, which still leaves 47% of phishing sites that the users fell for. There is also evidence that attackers are catching up to internet browsers'efforts to protect users, which means that just relying on the warning is not enough, and a combination of anti-phishing tools, such as antivirus software, firewalls, and user privileges; and common sense is needed to protect network users.

2.4 PHISHING PREVENTION TOOLS

One method for preventing phishing attacks is to use two factor authentication [38]. Needing an external device (such as a cellphone) to be able to login to a system provides another barrier for an attacker to access the system. Another method is to teach users to identify and avoid phishing attacks, such as using a video game that teaches users how to identify fraudulent websites [39]. A third method is to simulate spear phishing, ransomware, and malware infected attachment attacks on users to teach those who fell victim how to avoid future attacks [40]. A fourth method that has been used to prevent phishing attacks is to create virtual users to protect the real users logged in [41].

CHAPTER 3: METHODOLOGY

3.1 TOOL IMPLEMENTATION & TESTING

3.1.1 SETUP

For this tool to be able to simulate user mistakes, we first decided what causes vulnerabilities in the network. From our research, we discovered that the majority of the vulnerabilities created are caused by executing or opening files that have embedded a malicious payload to it. This means that the tool has to be able to download files from a website to run automatically afterwards. The address of this website was entered manually. This simple website is an HTML file with four links which download different types of files, which include an image file, a python script, either a Bash or Powershell file (depending on what OS the test is done), and a text file. Bash/Powershell files were used instead of executable files because the website used to host our custom site did not permit executables to be stored for download. This website was hosted two different ways. First, it was hosted on 000WebHost [42], where it is hosted using HTTPS, and then hosted on a local Apache server installed with XAMPP [43]. This is to simulate both HTTP and HTTPS scenarios for testing the tool, in case that the anti-phishing tools reacted differently to the agent based on the connection. Since websites hosted on a server should be enclosed within their directories and not be able to access local directories in the client, our theory was that the anti-phishing tools will detect the downloaded files as not coming from the same user, and either block them or allow downloading them based on if they are downloaded using HTTP or HTTPS. The tool downloads files from it in two different methods: using the Mechanize library [44] by finding the HTML elements for links

and downloading them, and with the PyAutoGUI library [45] for simulating mouse and keyboard actions by moving through each link and 'clicking' them. Although the Mechanize library doesn't open links through the default browser but instead uses Python's urllib library [46], we were testing to see if the agent' s download of files will be interrupted by the anti-phishing tools, which would mean that they monitor internet activity for file downloads; or if the agent will only be interrupted when trying to run the downloaded files, which would mean that downloads through applications are not a loophole that can be exploited. PyAutoGUI on the other hand, was used to test how well the agent works against the anti-phishing tools' protection against user mistakes. A challenge we faced while testing the PyAutoGUI functions was that we discovered that every browser behaves differently to the Tab key, which is used for navigating through the links in the website. This meant that depending on the browser used, we had to change if the tool used only the Tab key, or Shift plus Tab key to navigate through the links. Finally, to do the testing more efficiently, we set up VMs that contain the autonomous tool. The tool was developed and tested on both Linux and Windows, using Python 2.7.15 [47]. Python was chosen since the autonomous tool has to be portable, which means it can be installed on both Windows and Linux machines. It was setup in Oracle VirtualBox [48], where it can be exported as an OVF to use in different networks if needed. The VMs that we used were an Ubuntu 18.04.1 LTS [49] and a Windows 7 x86 [50].

3.2 AUTONOMOUS AGENT ARCHITECTURE

Due to how it was developed, the autonomous agent was able to be deployed on both Windows and Linux machines, as mentioned before. The agent will interact with a website hosted both on a remote server that is accessed through the internet, and hosted on a local server running on the same machine that the tool is being run. The locally hosted website will be on the Windows VM, so the tests that are going to be done on HTTP will only be run on the Windows VM. The tool will both

send requests to the website by simulating mouse clicks, and save the file directly by saving the data directly from the file link into a local file. Figure 3.1 shows the context diagram of how the autonomous agent interacts with the hosted websites.

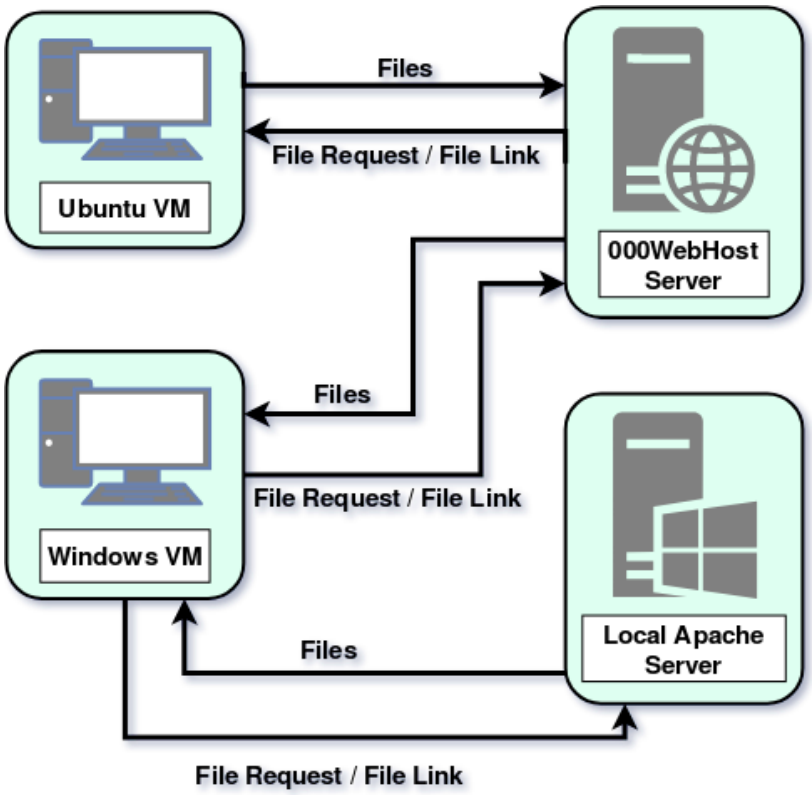

Figure 3.1: Context Diagram for Autonomous Agent.

In a real-world scenario, the tool will be able to do file requests and save the files directly on any hosted website. It detects the amount of file links and it will do both types of download actions that same number of times. This means it will be compatible with different websites besides the one used for testing in this thesis.

3.3 SCRIPT DEVELOPMENT

The tool was built with an object-oriented approach, with each class having separate functionality and functions that can be tested individually and expanded upon. PyAutoGUI and Mechanize were installed, and they each download files by simulating keyboard and mouse actions, and using Python's urllib library, respectively. Mecha-

nize's functionality was a modified version of a sample code of how to download files from a webpage, given in StackOverflow [51]. This Mechanize code finds the links in the HTML code of the website and saves the ones that match certain file types. I expanded the file type list to include more types of image, video, executable, and script files. This same method is used in the PyAutoGUI code to detect the number of links that have to cycled through for clicking. For running the files, the tool had to be able to check any changes in the Download folder. For this, the tool saved the paths of the contents of the folder in a text file, and before running the files, it cross-checked the current files in the folder and the contents of the text file to see if there are any new files and update the file if necessary. To run the files, it uses the paths of the files saved in the text file and runs them using the appropriate Bash or Windows CMD command. Separate commands had to be used so that each file was run with the appropriate program used to open them. For the Mechanize tests, it has to download the files in a different folder than the default Download folder so that the same files are not run again. But since the Mechanize code is downloading the files using the urllib library instead of the browser, it was easy to save the file in a different location, and update the download folder text file to save the contents of this different folder instead. Although the tool is able to run both functions autonomously, we made a function for running each (PyAutoGUI and Mechanize) separately so that recording the output of the anti-phishing tools would be easier.

3.4 EXPERIMENT

The tests are divided into two sections: one where the anti-phishing tools are going to be tested against the tool downloading files using HTTPS, and another using HTTP. The anti-phishing tools tested are:

1. AVG [52]
2. Avast [53] [54]
3. MalwareBytes [55] [56]

4. Download Virus Checker [57]

These tools were chosen on two factors: one, their popularity and ratings; and two, their features. If it seemed an antivirus software only protected users by blocking ads and pop-ups, then it was discounted as a viable security tool. Avast, MalwareBytes, and Download Virus Checker are available as add-ons for Mozilla Firefox, and they were installed that way. Avast, MalwareBytes, and AVG are available as standalone scanners for protecting the machine itself (instead of just browser interactions), and both Avast and MalwareBytes provide their own browsers with custom software. Using the two prepared VMs (Ubuntu and Windows), we will take a snapshot of each VM, install one of the anti-phishing tools, run the application, and record the results. In the Ubuntu VM, we will install the extensions for the Mozilla Firefox browser (2 to 4) and test them against the website hosted on 000WebHost; in the Windows VM we will redo the tests done in Ubuntu, and also install the antivirus standalone scanners (1 to 3), with AVG and Avast installing their custom web browsers, and testing them against the website hosted on the Apache server.

3.5 EXPECTED RESULTS

For the autonomous agent, we believe that it will successfully download files using both methods. Based on small tests of the modules used for the agent's individual functions: clicking and pressing keys with PyAutoGUI, and testing the Mechanize code snippet with the sample page that is provided in the StackOverflow; Python and its modules are powerful enough to do all the tasks required for this thesis. Since we were developing it in both Windows and Ubuntu, we also believe that it will be easy to integrate this tool into a real network using VMs.

In terms of the anti-phishing tools, we believed that they will warn the users when downloading the Python script, and for both the Powershell and Bash scripts, but not for the image and text files. The warnings should be different depending if the website is running in HTTPS or just HTTP: the tools will warn the users before downloading

files from the Apache server (since it's not secure), than when downloading from the secured website. The anti-phishing tools that are add-ons for the browser will only warn the user for downloading the files, while the stand-alone applications will warn the user when running the Python and the Powershell/Bash script.

CHAPTER 4: RESULTS

With the first set of test, which is using PyAutoGUI for going through the links in the web page and clicking them, only Download Virus Checker provided any notification regarding the downloads. Since it uses VirusTotal[58] to scan downloads, it did show that it was queuing the files for scanning with VirusTotal.

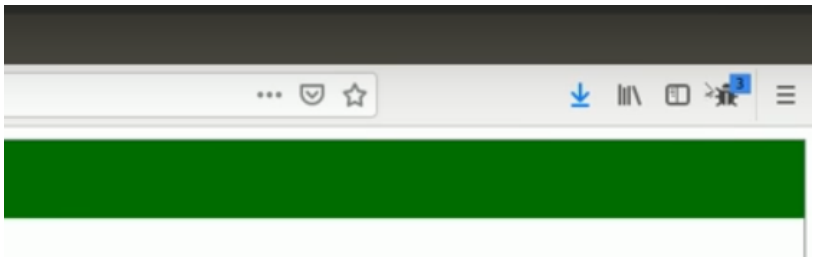

Figure 4.1: Download Virus Checker scanning files while downloading.

So that it would scan all the downloads, I removed all the file types in it's whitelist.

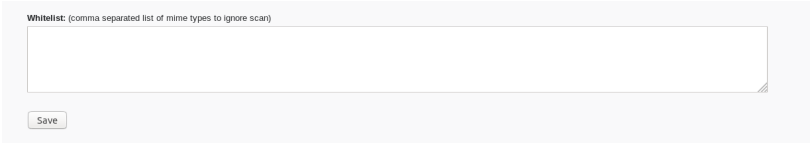

Figure 4.2: Empty whitelist of filetypes to ignore when scanning.

Avast's and MalwareBytes'browser extensions did not show any notification when the files were downloaded, and none of them showed any notifications when the files were executed. On the second set of tests, where files are downloaded with the Mechanize library, none of the extensions showed any warnings when the files were downloaded and executed. We can conclude from the first one that the browser extensions tested are only monitoring internet activity within their own browser instance, and any other internet activity outside of that is either ignored or cannot be accessed

from within that browser instance. The reason of the last one is because these browser extensions are designed to only monitor activity within the scope of interactions with the browser itself, while standalone antivirus applications are supposed to monitor for activity throughout the user's session in their computer. This is the default behaviour for these browser extensions, but some vulnerabilities have been discovered that've led to extensions being able to get access to more parts of the computer, such as monitoring keyboard strokes and access to the webcam and microphone [59].

In the first test where we were recreating the Ubuntu tests, the results were the same: only Download Virus Checker scanned the files using VirusTotal, while the others didn't show a warning. In the second test, when downloading the files using using PyAutoGUI, both AVG and Avast's custom browsers reacted the same way, by warning the user of downloading both script files (the Python and Powershell scripts.)

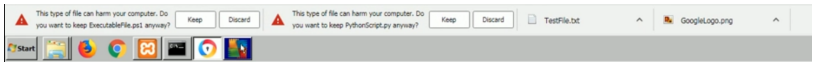

Figure 4.3: Chromium browser warnings for script files.

Since the way the notification looks is the same and we know that Avast's secure browser is built upon Chromium, the open-source browser provided by Google, we can assume that AVG's custom browser is also based on Chromium, hence the same warnings to the same files. Besides these notifications, AVG, Avast, and MalwareBytes did not give a warning for running the files or downloading through Mechanize.

CHAPTER 5: CONCLUSIONS

From our tests, we conclude that the agent was successful as a test software for the anti-phishing tools used in this experiment. This means that this tool can be used on a larger scale to test the protection of networks against phishing attacks. But, our testing revealed that the tool would need some changes to be compatible with different scenarios in larger scale experiments. Since we discovered that every browser behaves differently to the Tab key, for future versions of the agent, we need to configure it to do different actions depending on the browser. Another problem we faced was that for some types of files, the browser always asks for confirmation for downloading. This was not to warn the user of the dangers of the file type, but because the browser cannot be configured to download them automatically. For this, adding computer vision functionality so that the agent detects when a window pop-ups so it can confirm the download would help in being able to download a wider range of filetypes. Ultimately, the tool was able to download files in two different ways, which can be used for testing different types of network protection. The tool can also be easily deployed on networks by exporting the VM, since it was already developed on a VM.

In conclusion, we believe that modern antivirus software should be stricter in terms of notifying the user of the dangers. we don't think they should be strict by default, but at least have settings for being able to monitor more internet activity. For example, based on how none of the anti-phishing tools showed any notifications when the Mechanize library was downloading files, we can conclude that they are only monitoring for some internet activity and a large portion is ignored. Of course, if the antivirus software were to monitor all activity, it would lead to it interfering with other appli-

cations (such as applications that are streaming data) and maybe interrupting their connection. This would also lead to the anti phishing tool needing more resources from the computer to monitor and scan all the activity, and there are better tools for controlling the data going in, like firewalls. But we would prefer that anti-phishing tools would provide either a separate application or more options on their default application that allowed them to be more strict in that regard. It would help both users who are not tech savvy but are conscious of cyber attacks, and users who are tech savvy but want a quicker and portable alternative than setting different tools for protecting their devices. Seeing as how most of the browser extensions, except Download Virus Checker, did not warn the users of the downloaded files because of their limitations, we can conclude that most of them are not a good alternative to having a standalone antivirus application running on the computer. Still, in the Windows only test, the standalone anti-phishing tools did not provide any warnings when downloading the files: the browsers were the only ones that gave warnings, and although that was a positive sign towards more secure web surfing, it doesn't mean that the same tools that are designed to protect the users don't come with faults. For example, Avast's browser had a vulnerability that allowed attackers to use it as a custom file explorer of the victim's local directory [60]. Based on all these results, users shouldn't rely on a single anti-phishing tool for protection since there's not an individual one that covers every situation. Instead, users should use multiple tools to cover different vulnerabilities, and use common sense when using the internet, such as scanning individual files before running them, scheduling regular antivirus scans of the machine, and always use websites who use HTTPS. For future work, setting up a testing environment where real malware can be introduced on the website and tested against the antivirus would be closer to a real-world scenario. Also, expanding upon the list of antivirus software used here and include more of them; and also use other tools to test, such as firewalls against the autonomous tool.

Finally, we believe this tool, due to its portability, can be implemented in virtual networks to test its security. Pentesters can insert this tool into a virtual network that represents a real network, where it would serve the purpose of a user making mistakes, and test more efficiently without the need of human intervention for activating payloads within the network. This means that pentesters who want to test new technologies for infiltrating a network, network administrators who want to test tools to protect a network, and students who want to learn pentesting or network administration, can all take advantage of this tool and use it for testing virtual networks with different configurations.

REFERENCES

[1] P. D. T. Vyas, "Communicating the dangers of non-secure http," jan 2017.

[2] D. Akhawe and A. P. Felt, "Alice in warningland: A large-scale field study of browser security warning effectiveness," in *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, (Berkeley, CA, USA), pp. 257–272, USENIX Association, 2013.

[3] L. L. M. I. of Technology, "The sage air defense system," 2017.

[4] J. Goldman, "79 percent of companies faced severe cyber attacks in q3 2017," dec 2017.

[5] P. Technologies, "Cybersecurity threatscape q3 2017," tech. rep., Positive Technologies, 2017.

[6] E. Tittel, "Protect your data by hiring the right penetration test vendor," apr 2003.

[7] K. M. Henry, *Penetration Testing: Protecting Networks And Systems*. IT Governance Publishing, jun 2012.

[8] R. Das, "The top 5 pen testing tools you will ever need," apr 2018.

[9] R. Kelly, "Almost 90% of cyber attacks are caused by human error or behavior," mar 2017.

[10] "2017 cyber risk survey report," jun 2017.

[11] L. Morgan, "Five damaging data breaches caused by human error," feb 2016.

[12] A. S. Horowitz, "Top 10 security mistakes," jul 2001.

[13] T. Sammel, "Incident response teams find common pitfalls in network security," apr 2015.

[14] N. Lord, "Social engineering attacks: Common techniques & how to prevent an attack," sep 2018.

[15] N. Giandomenico, "What is spear-phishing? defining and differentiating spear-phishing from phishing," sep 2018.

[16] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, "A brief history of the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 22–31, Oct. 2009.

[17] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," *Comput. Secur.*, vol. 24, pp. 31–43, Feb. 2005.

[18] I. X.-F. Research, "Reviewing a year of serious data breaches, major attacks and new vulnerabilities," tech. rep., IBM Security, aug 2016.

[19] Symantec, "Executive summary, 2018 internet security threat report, volume 23," tech. rep., Symantec Corporation, mar 2018.

[20] T. K, "How human error compromises network security," aug 2016.

[21] E. Benishti, "Devastating phishing attacks dominate 2017," apr 2018.

[22] G. Stoneburner, A. Y. Goguen, and A. Feringa, "Sp 800-30. risk management guide for information technology systems," tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, United States, 2002.

[23] Cisco, "Business network security checklist."

[24] B. Hat, "The 2017 black hat atendee survey," tech. rep., Black Hat, jul 2017.

[25] E. P. R. S. C. S. D. S. B. S. Niamh Minihane, Francisca Moreno, "Mcafee labs threat report," tech. rep., McAfee Labs, dec 2017.

[26] R. Kang, L. Dabbish, N. Fruchter, and S. Kiesler, ""my data just goes everywhere:" user mental models of the internet and implications for privacy and security," in *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, (Ottawa), pp. 39–52, USENIX Association, 2015.

[27] R. 7, "Phishing awareness training."

[28] M. C. Support, "How do i tell if my connection to a website is secure?."

[29] E. Lawrence, "Avoiding the not secure warning in chrome."

[30] Technopedia, "Telemetry."

[31] Statista, "Market share held by leading desktop internet browsers in the united states from january 2015 to november 2017."

[32] E. Schechter, "A secure web is here to stay," feb 2018.

[33] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, (New York, NY, USA), pp. 581–590, ACM, 2006.

[34] C. Hoffman, "How scammers forge email addresses, and how you can tell," sep 2016.

[35] M. Alsharnouby, F. Alaca, and S. Chiasson, "Why phishing still works: User strategies for combating phishing attacks," *International Journal of Human-Computer Studies*, vol. 82, pp. 69 – 82, 2015.

[36] A.-P. W. Group, "Phishing activity trends report, 3rd quarter 2017," tech. rep., Anti-Phishing Working Group, feb 2018.

[37] I. S. R. Group, "About let"s encrypt."

[38] B. Parno, C. Kuo, and A. Perrig, "Phoolproof phishing prevention," in *Proceedings of the 10th International Conference on Financial Cryptography and Data Security*, FC'06, (Berlin, Heidelberg), pp. 1–19, Springer-Verlag, 2006.

[39] S. Sheng, B. Magnien, P. Kumaraguru, A. Acquisti, L. F. Cranor, J. I. Hong, and E. Ferrall-Nunge, "Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish," in *SOUPS*, 2007.

[40] Confense, "Confense phishme, employee conditioning for resiliency against phishing."

[41] D. Shiloh, "Method and system for securing user identities and creating virtual users to enhance privacy on a communication network," mar 2000.

[42] "Absolutely free web hosting with php, mysql and no ads.."

[43] "What is xampp?."

[44] K. Goyal, "mechanize 0.3.7."

[45] A. Sweigart, "Pyautogui 0.9.38."

[46] "20.6. urllib2 - extensible library for opening urls."

[47] "Python 2.7.15 documentation."

[48] "Welcome to virtualbox.org!."

[49] C. Ltd., "The leading operating system for pcs, iot devices, servers and the cloud | ubuntu."

[50] Microsoft, "Free virtual machines from ie8 to ms edge - microsoft edge development."

[51] robert king, "Download all the links(related documents) on a webpage using python," may 2011.

[52] "Avg 2019 | free antivirus, vpn & tuneup for all your devices."

[53] A. Software, "Avast online security."

[54] "Avast | download free antivirus & vpn | 100% free & easy."

[55] M. Inc., "Malwarebytes browser extension."

[56] "Free antivirus replacement & anti-malware tool | malwarebytes."

[57] J. Fray, "Download virus checker."

[58] "Virustotal."

[59] P. Gupta, "Chrome permission flaw: Extensions can remotely monitor user activity," 2017.

[60] H. T. Casey, "Avast's secure web browser was anything but safe," feb 2016.

APPENDIX A: CODE SNIPPETS

A.1 DOWNLOAD FILES WITH MECHANIZE

```
def downloadLinksOnPage(self, br, downloadLinks, fileTypes):
        """Get links and download files from it."""
        # Get root folder of PC.
        rootFolder = os.path.expanduser('~')

        if self.mOS == "Linux":
            rootFolder += "/TempFiles/"
        elif self.mOS == "Windows":
            rootFolder += "\\TempFiles\\"

        # Go through links received.
        for x,aLink in enumerate(downloadLinks):
            time.sleep(1)

            # Create path to save file in.
            fileToSave = rootFolder + aLink.text + fileTypes[x]
            print "This is the file to save " + fileToSave
            downloadFile=open(fileToSave, "w")

            # Go to link.
            br.follow_link(aLink)
            # Save file from link.
            downloadFile.write(br.response().read())
            print aLink.text + " has been downloaded"
```

```
            # Navigate back to main page.
            br.back(1)
            time.sleep(1)


    def getDownloadLinksInPage(self, webpageLink):
        """Get webpage link, and get all download links from it."""

        # Create browser object to interact with webpage.
        browserObject = mechanize.Browser()
        aResponse = browserObject.open(webpageLink)
        print "This is the response: " + str(aResponse.getcode())

        # If there was an error getting to the webpage, log error.
        if aResponse.getcode() >= 400:
            logging.error("Error. Problem in getting to website.")
            return
        else:
            # Keywords for types of files to download from webpage.
            keywordList=[".ps1", ".zip", ".py", ".txt", ".exe",".tar.gz",".png",".
            myFiles=[]
            fileTypes=[]

            # Get all links from webpage that contain one of the keywords.
            for aLink in browserObject.links():
                for aType in keywordList:
```

```
            if aType in str(aLink):
                myFiles.append(aLink)
                fileTypes.append(aType)
        # Send links for downloading the files.
        self.downloadLinksOnPage(browserObject, myFiles, fileTypes)
```

A.2 DOWNLOAD FILES BY SIMULATING CLICKS

```
def openBrowserAndFocus(self, aUrl):
    """ Open new browser instance and click on it to focus all other inputs in

    # Browser will be opened with the url provided.
    if self.mOS == "Linux":
        subprocess.Popen(["xdg-open", aUrl])
    elif self.mOS == "Windows":
        os.system("start " + aUrl)
    time.sleep(3)

    # Move mouse to center of screen.
    pyautogui.moveTo(self.screenWidth/2, self.screenHeight/2)
    # Click on center to focus keyboard on browser window
    pyautogui.click()

def clickOnBrowserElements(self):
    """Click on all available links in the webpage by using Tab and Enter"""

    # Depending on the browser, it changes how Tab and Enter works.
    # If it's Mozilla Firefox, the Tabs go from the next link close
    # to the mouse, which would be an advertisement for 000webhost in
    # the lower right corner. Because of this, it has to first Tab into
    # the ad, and then Shift+Tab back to the other links in the browser.
    # Google Chrome starts from the first link element,
    #regardless of the mouse position.
    # But when retesting Firefox\textquotesingle s addons, we needed
```

```
# to change the Windows section to be like the one from Linux.
if self.mOS == "Linux":
    pyautogui.press('tab')
    for iteration in range(self.linksInBrowser):
        pyautogui.keyDown('shift')
        pyautogui.press('tab')
        pyautogui.keyUp('shift')
        time.sleep(2)
        pyautogui.press('enter')

elif self.mOS == "Windows":
    for iteration in range(self.linksInBrowser):
        #pyautogui.keyDown('shift')
        pyautogui.press('tab')
        #pyautogui.keyUp('shift')
        time.sleep(2)
        pyautogui.press('enter')
```

VITA

Anibal Robles Perez was born in October in Arecibo, Puerto Rico. He graduated with a Bachelor's degree in Computer Science at the University of Puerto Rico at Arecibo, and is expected to graduate in December 2018 with a Master in Information Technology at the University of North Carolina at Charlotte. He has worked as a Graduate Research Assistant in UNCC since Fall 2017.