**YS-5-RED Senior Project Cybersecurity in Malware Analysis**

***Website:*** *https://www.sites.google.com/view/ys5red*

*Authors: Connor Skidmore, Jonathan Tarrant, Andy Guo  
Project Owner: Yong Shi*

**8/20/2022**

**Abstract**

With society becoming more computerized each day, the paradigm of cybersecurity is emerging as a crucial field to the maintenance of the quality of life we all have come to enjoy. Every aspect of our lives seems to involve some sort of human-machine interaction, from the constant checking of our mobile phones to our growing reliance on smart appliances and devices. Industry has shown an even more eager attitude towards the adoption of automated systems. All this is to say, if our computer systems go down, life as we know it ceases to exist. Malicious actors have taken note of this as well and have begun to innovate new and more effective ways to hinder or shut down their target systems, threatening the stability of our society. We propose five algorithms to detect five different types of malwares, with a focus on the network traffic anomalies of Distributed Denial of Service (DDoS) attacks and Port Scanning Attacks. Our other modules include Internet of Things (IoT) Attacks, Artificial Intelligence (AI) Attacks, and Crypto Jacking attacks. This paper will explore the background of each of these kinds of malware, as well as an overview of the research we conducted and an exploration of how Artificial Intelligence and Machine Learning is being used to stop malware attacks today***.***

**Table of Contents**

Requirements

Analysis

Development of Software

Results

Contract

Planning and Management

Conclusion

References

IEEE Research Paper

4

5

6

14

16

17

24

25

27

**Requirements**

1. 2 research modules related to “Network traffic anomaly detection using AI and ML approaches”, in specific DDoS and Port Scanning.
2. Three modules, one of each in AI attacks, Crypto-Jacking, and Internet of Things (IoT) attack. This makes a total of 5 research modules.
3. 2 coding stimulations and solution of an AI/ML program, one for DDoS and Port Scanning.
4. 3 non-AI/ML coding stimulation and solution, one for AI attacks, Crypto-Jacking, and Internet of Things attack.
5. An IEEE research paper.
6. A documentation of the Project Plan report, which includes but not limited to Agenda, Gantt chart, Source code, Dataset, Milestones, Weekly Reports, Website link, and References.
7. GitHub page includes all researched modules with substantial material.

**Analysis**

**Tech platform**

* Colab
  + Short for Colaboratory, this is a product from Google Research. By using Colab our team can effectively write and execute python code through the browser. It allows easy access to team members, which makes collaborating more accessible.
* PyTorch
  + An library that provides powerful tools for Machine learning and Artificial Intelligence. In short, it is an incredibly comprehensive library to use for programming, training, and running AI and Machine Learning Programs.
* GitHub
  + Provides a remote and flexible workplace for our team to store, track, and collaborate on software projects. It also stores our documentations for coding and research for our senior project.
* Google Pages
  + Google owned site, its simplistic but effective layout and tools allow us to make a webpage that is visible to the public when we need it and is sufficient in presenting our results.
* Discord
  + Communication software that provides effective communication and allows us to keep track of files through different chat functions. It’s voice chat and text function allow us to communicate near real time even if we are not on campus.

**Development of Software**

***DDoS and Port Scanning***

During the first phase of development our team was not aware of the requirement to develop artificial intelligence and machine learning stimulations of the malware attacks. After a few weeks, our team formatted a contract and sent it to our project owner, Yong Shi.

We were then notified to attend a meeting and was assigned the DDoS and Port Scanning module. At that point we got a completely different requirement sheet. It had more requirements and overall remained somewhat vague. Professor Yong Shi then explained in the next meeting that our team was responsible for creating an AI/ML stimulation. Essentially make a mock DDoS and Port Scanning attack.

***AI Attack***

The situation our team has chosen in relation to this attack is the ability of AI to mimic voices. First, we implemented a text to speech to mimic three potential but popular calls, which are Bank, Internet, and Mobile. Using a package, we can record an audio file with the robotic sound of google to make it more realistic. Also, implementing a speech to text function where we get the user’s voice when they make a response to the specified service. This completes the attack.

Overall, this module was more doable the libraries online and tutorials helps tremendously in creating the code. The difficult parts were the ways to implement the audio files into Colab and access them within the code.

***Internet of Things Attack***

The situation our team has chosen is a user who has been a victim of a phishing attack, that attack is aiming to take the login information on the application of a fictious smart lightbulb company. First, we implemented the login function for the smart lightbulb. In this case we made an email phishing attempt which sends an email from our email account, it will ask for the user’s name and password as an update to the system and require the company to take in the previous login information. The solution to the issue is taking a list of the recent email phishing and comparing it to the official email accounts the company uses. The main distinction being the domain like, .gov, .net, .org, etc.

***Crypto Jacking***

The situation our team has chosen is a computer that uses a certain amount of memory it may fluctuate. With the implementation of a crypto currency which steals RAM thereby making the computer slower but at the same time only taking a small amount of RAM to mine cryptocurrency. The user is taking the place of the malicious party, and is asked how much of the total system memory they want to steal from the victim of the attack. Once they give the value, the simulation is run, and the time it takes for the program to run is tracked. Once the simulation is complete, the code calculates the amount of currency was mined using the victimized system during the execution time.

**AI Attack Source Code**

!pip install gTTS #Install through Command Prompt

from gtts import gTTS #Import Google Text to Speech

from IPython.display import Audio #Import Audio method from IPython's Display Class

#language to use

language = "en"

#random variable

num = input("Please enter a number 0, 1, 2: ")

#Choice of speech depending on number

if num == "0":

  speech = "Welcome to XYZ bank, please say your account number and pin."

elif num == "1":

  speech = "Welcome to XYZ internet, please say your account number and pin"

elif num == "2":

  speech = "Welcome to XYZ mobile, please say your account number and pin"

else:

  speech = "Invalid account number and/or pin"

#Provide the string to convert to speech

tts1 = gTTS(text = speech, lang = language, slow =False)

tts1.save('1.wav') #save the string converted to speech as a .wav file

sound\_file = '1.wav'

Audio(sound\_file, autoplay=False)

#Autoplay = True will play the sound automatically

#If you would not like to play the sound automatically, pass Autoplay = False.

#Cant run in Colab, using cmd to run

!pip install SpeechRecognition pydub

!pip install pyaudio

import speech\_recognition as sr

speech\_r = sr.Recognizer()

#Loading audio file and converting speech to text

with sr.Microphone() as source:

    print('Talking...')

    # read the audio data from the default microphone

    audio\_data = speech\_r.record(source, duration=5)

    print("Recognizing...")

    # convert speech to text

    text = speech\_r.recognize\_google(audio\_data)

    print(text)

**Internet of Things Attack (IoT Attack) Source Code**

def login():

  #Creation of user1 with a 6 digit password

  user1 = "Andy"

  pw1 = "123a@S"

  #Key to exit while loop and error key to stop new login

  key = True

  ekey = 0

  #Loop for login

  while key == True:

    print("Enter your login information")

    user = input("Enter your username: ")

    pw = input("Enter your password: ")

    if user == user1 and pw == pw1:

      key = False

      print("login successful... Welcome to your SMART LIGHT")

    else:

      print("Error username or password incorrect...")

      ekey+=1

    if ekey == 3:

      key = False

      print('Login failure')

login()

#importing the Yagmail library

!pip install yagmail #Install through Command Prompt

import yagmail

def sende():

  try:

    #initializing the server connection

    yag = yagmail.SMTP(user='ys5red2@gmail.com', password='ltbcomkwymjrwqfd')

    #sending the email

    yag.send(to='ys5red2@gmail.com', subject='Smart Life - Smart Living',

             contents='<img src="https://images.unsplash.com/photo-1606812667169-0e1991ed3742?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1170&q=80" alt="Lightbuld" width="70" height="60"> <br></br>' +

             '<h2> Due to the unexpected breach, we are requiring a mandatory password change. ' +

             '</h2> <br></br> Please verify username and old password to recive a temporary password link to reset your password.')

    print("Email sent successfully")

  except:

    print("Error, email was not sent")

#Uncomment as needed

sende()

#Importing modules/packages

import smtplib

import imaplib

import email

import traceback

#Defining email used to send phishing attack

FROM\_EMAIL = "ys5red2@gmail.com"

#Using APP password to log into gmail

FROM\_PWD = "ltbcomkwymjrwqfd"

#Connecting to Internet Message Access with gmeil.com

SMTP\_SERVER = "imap.gmail.com"

SMTP\_PORT = 993

#Reading the emails in gmail

def read\_gmail():

    try:

        mail = imaplib.IMAP4\_SSL(SMTP\_SERVER) #Connecting to server, using the logins from above, and targeting 'inbox

        mail.login(FROM\_EMAIL,FROM\_PWD)

        mail.select('inbox')

        data = mail.search(None, '(SUBJECT "Smart Life - Smart Living")') #Sifting emails for targeted subject

        mail\_id = data[1]

        id\_list = mail\_id[0].split()

        first\_email = int(id\_list[0]) #Making a list of the emails

        latest\_email = int(id\_list[-1])

        for i in range(latest\_email, first\_email, -1): #Searching through emails and fetching requirements

            data = mail.fetch(str(i), '(RFC822)' )

            for response in data:

                array = response[0]

                if isinstance(array, tuple): #Traversing data list of emails

                    msg = email.message\_from\_string(str(array[1],'utf-8')) #Pulling email message

                    e\_subject = msg['subject'] #Pulling subject

                    e\_from = msg['from'] #Pulling from whom

                    e\_body = str(msg.get\_payload(0)) #Pulling email body

                    print('From : ' + e\_from + '\n')

                    print('Subject : ' + e\_subject + '\n')

                    print('Body: ' + e\_body + '\n')

    except Exception as e:

      print(str(e))

#Solution algorithm to authenticate emails

def solve\_email():

    try:

        mail = imaplib.IMAP4\_SSL(SMTP\_SERVER) #Connecting to email server and logging in

        mail.login(FROM\_EMAIL,FROM\_PWD)

        mail.select('inbox')

        data = mail.search(None, '(SUBJECT "Smart Life - Smart Living")') #Selecting suspicious emails

        mail\_id = data[1]

        id\_list = mail\_id[0].split() #Creating list of emails from subject

        first\_email = int(id\_list[0])

        latest\_email = int(id\_list[-1])

        for i in range(latest\_email, first\_email, -1): #Traversing email list

            data = mail.fetch(str(i), '(RFC822)' )

            for response in data:

                array = response[0]

                if isinstance(array, tuple): #Checking if message from component is credible

                    msg = email.message\_from\_string(str(array[1],'utf-8'))

                    e\_from = msg['from']

                    #Checking to see if the domain is from an organization or business

                    if e\_from.\_\_contains\_\_('.inc') or e\_from.\_\_contains\_\_('.org') or e\_from.\_\_contains\_\_('.net'):

                      print('Credible')

                    else:

                      print('DO NOT TRUST ' + e\_from)

    except Exception as e:

      print(str(e))

#Calling functions

read\_gmail()

solve\_email()

**Crypto Jacking Attack Source Code**

from random import sample

from time import time, sleep

from math import floor

from copy import deepcopy

class CPU:

    RAM = 24

    startTime = 0

    sortList = list()

    def \_\_init\_\_(self, memory, sortList) -> None:

        self.memory = memory

        self.sortList = deepcopy(sortList)

    def makeArray(self, infection):

        multiplier = 1 if infection < 1 else infection

        # print(multiplier)

        nums = sample(range(0,10000\*multiplier), (9000\*multiplier))

        for x in range(len(nums)):

            self.sortList.append(nums[x])

        size = len(self.sortList)

        self.startTime = time()

        self.quickSort(self.sortList, 0, size - 1)

        # print("Array Sorted!")

    def partition(self, array, low, high):

        pivot = array[high]

        i = low - 1

        for j in range(low, high):

            if array[j] <= pivot:

                i = i + 1

                (array[i], array[j]) = (array[j], array[i])

        (array[i + 1], array[high]) = (array[high], array[i + 1])

        return i + 1

    def quickSort(self, array, low, high):

        # print("Sorting")

        if low < high:

            pi = self.partition(array, low, high)

            self.quickSort(array, low, pi - 1)

            self.quickSort(array, pi + 1, high)

class CryptoJacker:

    cryptoRam = 0

    run = True

    coinsMined = 0

    def \_\_init\_\_(self) -> None:

        pass

    def stealRAM(self):

        self.cryptoRam = input("How much RAM do you want to use? ")

        return int(self.cryptoRam)

    def mineCoins(self, seconds):

        coinsPerSec = float(self.cryptoRam) \* 0.05

        self.coinsMined += coinsPerSec \* seconds

        print(f"Total ShiCoins mined: {self.coinsMined}")

class Detector:

    def \_\_init\_\_(self) -> None:

        pass

    def detect(self, expectedTime, actualTime):

        if actualTime != expectedTime:

            print("\n-------------------------------------------\n" +

                  "  WARNING: CRYPTOJACKER RUNNING ON SYSTEM\n" +

                  "-------------------------------------------\n" +

                  f"Expected Execution Time: {expectedTime}\n" +

                  f"Actual Execution Time: {actualTime}")

def Menu():

    print("\n------Main Menu------\n" +

          "1. Run Program\n" +

          "2. Exit Program\n")

    return int(input("Enter an Option: ") or 0)

run = True

sortList = list()

cpu = CPU(10, sortList)

crypto = CryptoJacker()

cpu.makeArray(0)

expectedTime = time() - cpu.startTime

while(run):

    userInput = Menu()

    if userInput == 1:

        del cpu

        cpu = CPU(10, sortList)

        detector = Detector()

        stolen = crypto.stealRAM()

        infection = floor((stolen / cpu.RAM) \* 250)

        cpu.makeArray(infection)

        totalTime = time() - cpu.startTime

        crypto.mineCoins(totalTime)

        formatTime = "{:.2f}".format(totalTime)

        print(f"Exectution Time: {formatTime} sec")

        detector.detect("{:.2f}".format(expectedTime), formatTime)

    elif userInput == 2:

        print("Done! Thank you")

        run = False

    else:

        print("Please input a valid menu option (1 or 2)")

**Results**

**DDoS**

DDoS, or Distributed Denial-of-Service is a growing and concerning trend in both academic and industrial settings concerned with cybersecurity and malware. A DDoS attack is, quite simply, an orchestrated take-down of an organization's computer networks and infrastructure using a flood of requests, typically HTTP requests, to a server or network. The reason why the topic of malware comes into play here is that most of these attacks are accomplished using many separate subordinate computing devices and/or servers that have been infected with a program, or piece of malware, to effectively turn them into DDoS bots that the originator of this attack can use to create this flood of HTTP requests, hence, the word "distributed" in the DDoS acronym. The result of a successful DDoS attack is always either slowing the target server/network down to a speed at which it can no longer respond to legitimate user requests, or the crashing of the server/network completely. Once one sees the dire consequences of such an attack, especially for critical networks and servers that support medical, law enforcement, or national security related industries, it becomes necessary to investigate how to stop and prevent DDoS attacks from happening.

**Port Scanning**

Till this day the use of the internet is still growing. Many applications have been transformed into some sort of digital product. For example, purchasing from online stores, making appointments, and various services. For a computer to communicate and transfer requests, it requires a tunnel to send that data. The tunnel is uses is ports, there are 65,536 ports numbered from 0 to 65,535. One growing concern is the usage of port scanning. Port scanning is a method for determining what ports on a network are operating. But what makes port scanning so important? All computers must use ports to communicate and transfer digital data. This is the key, with computers being so abundant and mandatory in today’s age, data is constantly being transferred. This creates opportunities for hackers to steal and send information from and to ports. One can imagine that if an organization that relies on accurate information, such as a hospital, if false information is provided it would lead to lethal consequences. That is just one possibility of what port scanning could cause. With this potential in mind, it becomes necessary to investigate how to prevent port scanning attacks.

**AI Attacks**

The category of malware known as AI Attacks is indeed a broad category, as it covers any malware that utilizes Artificial Intelligence to carry out some sort of cyber-attack on a system or to gain unauthorized access to information. Artificial Intelligence is a fast-growing and powerful field in the world of computing, so malicious actors are beginning to realize its potential to target everyone from large agencies and corporations down to individual users.

**Crypto Jacking**

Crypto Jacking is the practice of using a person's computer to mine cryptocurrencies without the person's authorization. It makes use of the victim's hardware to mine minuscule amounts of the desired cryptocurrency. There are a couple of symptoms of Crypto Jacking that can alert victims to the presence of malware on their system. Victims might experience a slowdown in their device's performance, overheating of components such as batteries and processors, or unexpected shutdowns due to insufficient computing resources. This can cause outages and issues for non-infected users if the victim is a server that hosts web services.

**Internet of Things Attack**

An IoT Device Attack is any malware attack whose target is an IoT device. Such devices include smartphones, home assistants, and any "smart" device (light bulbs, TVs, door locks, switches, appliances, etc.). Many of these devices have vulnerabilities that are constantly being exploited by malicious parties. In today's technological space, smartphones are the prime target for these kinds of attacks. Many home-based IoT devices are controlled through apps on a person's phone. This means that gaining control over a smartphone with attached IoT devices gives the malicious party access to all the devices without having to hack each one.

Graphical user interface, text, application

Description automatically generated**Contract**

Text

Description automatically generated**Project Planning and Management**

**#1 Milestone due date 10/22/2022**

* 09/29/2022 Thursday 2:00pm in person
* 10/06/2022 Thursday 2:00pm in person
* 10/13/2022 Thursday 2:00pm in person
* 10/20/2022 Thursday 2:00pm in person

**#2 Milestone due date part 1 - 11/19/2022**

* 10/27/2022 Thursday 2:00pm in person
* 11/03/2022 Thursday 2:00pm in person
* 11/10/2022 Thursday 2:00pm in person
* 11/17/2022 Thursday 2:00pm in person

**#3 Milestone due date part 2 - 12/01/2022**

* 11/24/2022 Thursday 2:00pm in person
* 12/01/2022 Thursday 2:00pm in person
* 12/08/2022 Thursday 2:00pm in person
* 12/15/2022 Thursday 2:00pm in person

Weekly Reports and Notes in Chronological Order

**YS5-Red Cybersecurity in Malware Analysis Weekly Report**

Connor Skidmore, Jonathan Tarrant, Andy Guo

Date: 9/23/2022

**Meeting Notes from This Week:**

**Meeting 1 (Thursday 9/22/2022 2 PM):**

In this meeting we discussed the following:

• Created Google Site Framework to contain both research and Github links for source code programs.

• Discussed deliverable due dates and planned out workload distribution throughout the semester

• Settled on which malware modules we would be completing for the project

• Discussed the project scope with Dr. Perry and determined that we would be unable to do AI/ML programs for all 5 of the modules

• Discussed and looked at various sources for data sets of malware attacks, including Kaggle

**Meeting 2 (Friday 9/23/2022 12 PM)**

In this meeting we discussed the following:

• Discussed project scope with Dr. Perry

• Discussed need to do source code program for all 5 modules

• Discussed using non-AI/ML source code for 3 of the Modules

• Decided on doing in-depth research and AI/ML analysis/detection for DDoS and Port Scan attacks

• Decided on doing non-AI/ML source code program for other 3 modules

**Goals for Next Week:**

• Begin filling in Google Site with preliminary research for DDoS, Port Scan, Cryptojacking, AI Attacks, and IoT Attacks

• Continue searching for and gathering data sets on these malware topics from various sites such as Kaggle

• Continue learning about ML/AI, how they are used to detect malware, as well as how they apply to our specific modules

Next Meeting: Thursday 9/29/2022 2 PM

**YS5-Red Cybersecurity in Malware Analysis Weekly Report**

Connor Skidmore, Jonathan Tarrant, Andy Guo

Date: 9/30/2022

**Meeting Notes from This Week:**

**Meeting 1 (Thursday 9/29/2022 2 PM):**

In this meeting we discussed the following:

• Discussed progress on website

• Discussed AI Attacks and DDoS modules

• Discussed deadlines and organization of deliverables (possible Gantt chart development)

• Went over research conducted so far

• Discussed prototype presentation on 10/18/2022

**Goals for Next Week:**

• Getting one research module done for DDoS attacks

• Continuing to plan and discuss AI and non-AI code development

• Plan for prototype presentation to class

Next Meeting: Thursday 10/6/2022 2 PM

**YS5-Red Cybersecurity in Malware Analysis Weekly Report**

Connor Skidmore, Jonathan Tarrant, Andy Guo

Date: 10/7/2022

**Meeting Notes from This Week:**

**Meeting 1 (Thursday 10/6/2022 2 PM):**

In this meeting we discussed the following:

• Discussed upcoming progress presentations for Dr. Perry’s class

• Further hashed out workload distribution and discussed work that needed to be done over the weekend

• Went over research completed so far and discussed the plan for further research to be completed over the weekend.

**Goals for Next Week:**

• Being fully prepared to present our work up to this point to Dr. Perry’s class

• Getting a significant amount of research completed and up on the website to show both the class and Dr. Shi at our next meeting.

• Continuing to work with PyTorch to explore the development of our source code/programs to detect malware.

• Continuing to collect and search for data sets on our assigned Malware topics using sites such as Kaggle.

Next Meeting: Thursday 10/13/2022 2 PM

**YS5-Red Cybersecurity in Malware Analysis Weekly Report**

Connor Skidmore, Jonathan Tarrant, Andy Guo

Date: 10/14/2022

**Meeting Notes from This Week:**

**Monthly Meeting with Dr. Shi (Wednesday 10/12/2022 1 PM):**

In this meeting we discussed the following with Dr. Shi:

• Need to have working demonstrations ready by week of November 7th.

• Need to finish research by end of October

• Be ready to demonstrate prototypes to Dr. Shi by our next monthly meeting (11 November 2022)

• Additional deliverable required now: IEEE format research paper

• Reiterated AI/ML approaches for Network Traffic Anomalies DDoS and Port Scanning

• Reiterated Conventional non-AI/ML approaches for other 3 modules.

• Reiterated using Python/Pytorch/Colab for code/environment.

**Meeting 2 (Thursday 10/13/2022 2 PM)**

In this meeting we discussed the following:

• Discussed progress with research

• Discussed progress with code

• Assigned work for the IEE Research paper among the 3 group members

• Discussed preparation for milestone 1 demonstration.

**Goals for Next Week:**

• Prepare for presentations

• Start work on IEEE paper

• Work on Research so that it can be done by the end of October.

Next Meeting: Tuesday 10/18/2022 2 PM

**YS5-Red Cybersecurity in Malware Analysis Weekly Report**

Connor Skidmore, Jonathan Tarrant, Andy Guo

Date: 10/21/2022

**Meeting Notes from This Week:**

**Meeting 1 (Tuesday 10/18/2022 2 PM):**

In this meeting we discussed the following:

• Progress on IEEE paper

• Need to add Dr. Shi to google site as editor

• Jonathan’s progress on Pytorch to use for our AI/ML Programs

**Meeting 2 (Thursday 10/20/2022 2 PM)**

In this meeting we discussed the following:

• Continued to work on research and code

• Discussed current state of deadlines and workload

• Discussed goals for next week

**Goals for Next Week:**

• Near completion of research for our first milestone

• Have some AI/ML code completed for our Network Traffic Anomaly modules

Next Meeting: Monday 10/24/2022 5 PM

**YS5-Red Cybersecurity in Malware Analysis Weekly Report**

Connor Skidmore, Jonathan Tarrant, Andy Guo

Date: 10/29/2022

**Meeting Notes from This Week:**

**Meeting 1 (Monday 10/24/2022 5 PM):**

In this meeting we worked on the following:

• **Continued work on IEEE research paper finishing first few sections**

• **Jonathan continued work on understanding PyTorch to apply to AI/ML code**

• **Andy continued work on IoT Attack non-AI code**

**Meeting 2 (Tuesday 10/25/2022 5 PM)**

In this meeting we discussed the following:

• Continued to work on IEEE research paper

• Jonathan continued work on DDoS/Port Scanning

• Andy Continued work on IoT device attacks

**Meeting 3 (Thursday 10/25/2022 2 PM)**

In this meeting we discussed the following:

• Continued to work on IEEE research paper

• Jonathan continued work on learning AI/ML code/programming

• Andy Continued work on his phishing code for IoT device attacks

**Goals for Next Week:**

• Near completion of source code

• Have C Day Submission ready by Monday

• Have IEEE paper near complete or complete by end of next week

• Revamp website to showcase updated research/code

Next Meeting: Tuesday 11/1/2022 2 PM

**YS5-Red Cybersecurity in Malware Analysis Weekly Report**

Connor Skidmore, Jonathan Tarrant, Andy Guo

Date: 11/4/2022

**Meeting Notes from This Week:**

**Meeting 1 (Tuesday 11/1/2022 2 PM):**

In this meeting we discussed the following:

• How we can work towards the deadline of the 9 November meeting

• Finalizing the 5 modules to demonstrate to Dr. Shi

• Finalizing the website to show to Dr. Shi

• Finalizing the IEEE paper to show to Dr. Shi

**Meeting 2 (Thursday 11/3/2022 2 PM)**

In this meeting we discussed the following:

• Continued to finalize our modules for demonstration next week.

• Continued to finalize the website to show to Dr. Shi

• Continued to work on and finalize the IEEE paper to show to Dr. Shi

**Goals for Next Week:**

• Have modules ready to demonstrate to Dr. Shi

• Have website ready to show to Dr. Shi

• Have IEEE paper draft ready to show to Dr. Shi

Next Meeting: Tuesday 11/8/2022 2 PM

**YS5-Red Cybersecurity in Malware Analysis Weekly Report**

Connor Skidmore, Jonathan Tarrant, Andy Guo

Date: 11/11/2022

**Meeting Notes from This Week:**

**Meeting 1 (Tuesday 11/8/2022 2 PM):**

In this meeting we discussed the following:

• Finalizing items to present to Dr. Shi tomorrow

• Finalizing Code and draft of IEEE paper for Dr. Shi

• Determining how meeting and presentation will go tomorrow

**Meeting 2 (Wednesday 11/9/2022 1 PM):**

In this meeting we discussed the following with Dr. Shi:

• Demonstrated working code for non-AI modules

• Showed work so far on IEEE paper

• Showed sources

• Discussed plans and suggestions for what to add to project and how to modify project

**Meeting 3 (Thursday 11/10/2022 2 PM)**

In this meeting we discussed the following:

• Discussed Dr. Shi’s feedback on our project from yesterday.

• Discussed how workload for next deliverables will be distributed.

• Further worked on suggested improvements to project

**Goals for Next Week:**

• Implement Dr. Shi’s suggestions

• Continue working on finishing project for Nov 30 Meeting with Dr. Shi

• Prepare for Milestone 2 presentation next week.

Next Meeting: Tuesday 11/15/2022 2 PM

A picture containing table

Description automatically generated

Summary / Conclusions and feedback

***What can be done better***

-Senior project needs to have clear requirements and scope from the beginning of the project.

-Mandatory and more frequent meetings with project owner will be useful to see progress and keep on track.

-A well described product by project owner that can be subjected to questions by the students.

-An increased team size to minimal of 4 to reduce workload throughout semester.

-Introduction page to project needs to have specific requirements

1. Amount of team members minimal of 3

2. Detailed description of project

***What was frustrating***

-Throughout the semester the project deliverables were changed.

-Scope was defined but not achievable in a single semester long project.

-Knowledge related to project needs to be clearly stated.

-Each meeting needed to reexplain last meeting’s information.

-Project owner refused to alter requirements to better suit our team’s abilities.

-After contract was signed, other requirements were added that were not included in the original contract.

References

“2022 cyber security statistics trends & data,” *PurpleSec*, 17-Oct-2022. [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/. [Accessed: 29-Nov-2022].

B. Guembe, A. Azeta, S. Misra, V. C. Osamor, L. Fernandez-Sanz, and V. Pospelova, “The emerging threat of AI-Driven Cyber Attacks: A Review,” *Applied Artificial Intelligence*, vol. 36, no. 1, 2022.

D. Tang and X. Kuang, “Distributed denial of service attacks and defense mechanisms,” *IOP Conference Series: Materials Science and Engineering*, vol. 612, no. 5, p. 052046, 2019.

Address false positives/negatives in Microsoft defender for endpoint,” *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/defender-endpoint-false-positives-negatives?view=o365-worldwide. [Accessed: 29-Nov-2022].

E. Chickowski, “What is cryptojacking? how to prevent, detect, and recover from it,” *CSO Online*, 20-Jun-2022. [Online]. Available: https://www.csoonline.com/article/3253572/what-is-cryptojacking-how-to-prevent-detect-and-recover-from-it.html?page=2. [Accessed: 29-Nov-2022].

H. H. Hajri, B. M. Mughairi, M. I. Hossain, and A. M. Karim, “Crypto jacking a technique to leverage technology to mine crypto currency,” *International Journal of Academic Research in Business and Social Sciences*, vol. 9, no. 3, 2019.

H. Kumar, “Survey on Malware Detection Methods.”

K. Hahn, “The real reason why malware detection is hard--and underestimated,” *G Data Software*, 15-Jul-2022. [Online]. Available: https://www.gdatasoftware.com/blog/2022/06/37445-malware-detection-is-hard. [Accessed: 29-Nov-2022].

M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Surveying port scans and their detection methodologies,” *The Computer Journal*, vol. 54, no. 10, pp. 1565–1581, 2011.

O. Aslan and R. Samet, “A Comprehensive Review on Malware Detection Approaches,” *IEEE Access*, Jan. 2020.

Valeo Networks, “How is the internet of things (IOT) being impacted by malware?,” *Valeo Networks*, 10-Sep-2021. [Online]. Available: https://www.valeonetworks.com/how-is-the-internet-of-things-iot-being-impacted-by-malware/. [Accessed: 29-Nov-2022].

*Python : How to parse the body from a raw email , given that raw email does not have a "body" tag or anything*. Stack Overflow. Retrieved December 1, 2022, from https://stackoverflow.com/questions/17874360/python-how-to-parse-the-body-from-a-raw-email-given-that-raw-email-does-not

*Send email using yagmail in python*. GeeksforGeeks. (2021, October 7). Retrieved December 1, 2022, from https://www.geeksforgeeks.org/send-email-using-yagmail-in-python/

Rockikz, A. (2019, October 15). *How to convert speech to text in Python*. Python Code. Retrieved December 1, 2022, from https://www.thepythoncode.com/article/using-speech-recognition-to-convert-speech-to-text-python

An Analysis of Malware Detection Methods: DDoS, Port Scanning, Cryptojacking, AI Attacks, and IoT Device Attacks

College of Computing and Software Engineering

Kennesaw State University

Marietta, USA

Contact: [cskidmo6@students.kennesaw.edu](mailto:cskidmo6@students.kennesaw.edu), [jtarrant@students.kennesaw.edu](mailto:jtarrant@students.kennesaw.edu), [aguo@students.kennesaw.edu](mailto:aguo@students.kennesaw.edu), AND [yshi5@kennesaw.edu](mailto:yshi5@kennesaw.edu)

***Abstract –* With society becoming more computerized each day, the paradigm of cybersecurity is emerging as a crucial field to the maintenance of the quality of life we all have come to enjoy. Every aspect of our lives seems to involve some sort of human-machine interaction, from the constant checking of our mobile phones to our growing reliance on smart appliances and devices. Industry has shown an even more eager attitude towards the adoption of automated systems. All this is to say, if our computer systems go down, life as we know it ceases to exist. Malicious actors have taken note of this as well, and have begun to innovate new and more effective ways to hinder or shut down their target systems, threatening the stability of our society. We propose five algorithms to detect five different types of malware, with a focus on the network traffic anomalies of Distributed Denial of Service (DDoS) attacks and Port Scanning Attacks. Our other modules include Internet of Things (IoT) Attacks, Artificial Intelligence (AI) Attacks, and Cryptojacking attacks. This paper will explore the background of each of these kinds of malware, as well as an overview of the research we conducted and an exploration of how Artificial Intelligence and Machine Learning is being used to stop malware attacks today.**

***Keywords— Cybersecurity, Malware, Artificial Intelligence, Machine Learning.***

1. INTRODUCTION

       Malware is a portmanteau of the words “malicious” and “software.” In short, it is code written for malicious or illegal purposes, often to illegally gain access to a host server or computer and either disable it, encrypt it for monetary gain, or use it as a launching pad for further malware attacks [1]. While there is no accurate figure for the amount of monetary damage that malware causes each year, some estimates set the figure at around 6 Trillion Dollars in financial damage to industries and consumers annually.

       In this paper, we will first discuss existing technologies and approaches to detecting and eliminating malware threats. We will then discuss research on the benefits and drawbacks of certain technologies currently in use to detect, prevent, and stop malware attacks. Finally, we will discuss our approaches and source code to detect certain types of malware.

1. RELATED WORK

       Aslan and Samet define malware detection as the simple act of looking at a program or piece of software and determining whether or not it was written with malicious intent [10]. This is often much more difficult than it might seem. Many legitimate programs might behave in the same way as a malware program. The difference here is intent. For example, Respondus’ Lockdown Browser program, a test proctoring program that teachers and professors commonly use for tests in online courses, acts in a similar way to how a malicious software program might act. Once downloaded and upon asking for administrator privileges on a student’s computer, Lockdown Browser inserts itself into the kernel of a student’s operating system and when activated, monitors processes on the computer at a very low level and sends any suspicious activity in the form of a report to both Respondus’ database and the student’s instructor. While these actions might be very similar to what a malware program might do, Lockdown Browser is not malware due to one simple reason: it does not have malicious intent. As stated above, students consent to Lockdown Browser’s insertion on their OS kernel when they download the program, and the purpose of the program is not to collect data for nefarious purposes, but to collect data to maintain academic integrity.

       Herein lies the issue with modern malware detection: How does one write a program that can differentiate between good and bad intent by the developer of the code it is examining? Artificial Intelligence and Machine Learning come to mind as two powerful approaches to determining this intent. AI-based malware detection programs can be fed examples of previous malware attacks to teach them how to detect future attacks. This presents its own problem, though. AI Attacks are one branch of malware that will be further discussed later in this paper, and one type of AI Attack occurs when a malicious actor writes code to feed an AI program false information so as to hamper or disable its ability to understand real information [7]. This can be used against malware detecting AI by utilizing a malicious AI to feed the malware detecting AI examples of malware that are much easier to detect. Once the malware detecting AI is sufficiently accustomed to the easier-to-detect malware, the attacker launches an attack with a much more complex malware program that the AI has been deliberately trained not to detect.

1. FALSE POSITIVES AND FALSE NEGATIVES

       One of the largest issues with any sort of malware detection, whether it’s in the area of DDoS, Port Scanning, AI Attacks, Cryptojacking, IoT attacks, or others, is the prospect of a false positive or false negative. A false positive is when a malware detection algorithm determines a legitimate piece of software to be malware when in fact it is not. A false negative is when a malware detection algorithm determines a piece of malware to be legitimate software. When it comes to Artificial Intelligence and Machine learning approaches, dealing with false positives and false negatives is not only very important, but very difficult. After all, by the very nature of a false positive or false negative, it has fooled the algorithm into deeming it the opposite of what it actually is. So, we present two solutions. The first solution, put forward by the likes of Microsoft in their Microsoft Defender malware detection suite, involves using human input to train the AI algorithm on which of its decisions are false positives and which are false negatives [4].

Graphical user interface, application

Description automatically generated

**Figure 1: Microsoft Defender’s approach to dealing with false positives and false negatives**

At first glance, this seems as though it would be the most straightforward approach. Here, a computer has failed at its task to detect, identify, and stop a malware attack, and therefore human intervention is required to not only stop the malware attack itself, but to also train the AI algorithm to not make these kinds of mistakes in the future.

        Another option would be to approach the problem of false positives and false negatives from a strictly AI based method, leaving out the human element entirely. What this approach would look like practically is one or more layers of AI algorithms built as a sort of redundant platform to catch the mistakes of earlier AI algorithms, but this would still present the vulnerability that a well-developed piece of malware could make it through all layers of the AI detection program. For this reason, it is unlikely that a fully autonomous AI algorithm will be developed in the near future that is able to achieve a 0% false positive and false negative rate without any human intervention.

        To understand why achieving a 0% false positive and false negative rate is so important, we must consider the ramifications of even a low false positive/false negative rate. As Karsten Hahn notes, there has already been proof that a perfect malware detection algorithm is impossible. In 1984, Fred Cohen put forth the hypothetical argument of a perfect malware detection program juxtaposed with a piece of malware that was written to only show signs of being malware once the perfect malware detection program had decided that it was not malware [8].

Diagram

Description automatically generated

**Figure 2: A diagram of Cohen’s hypothetical perfect malware program, where a piece of malware is written to only become infectious when the perfect malware detection program deems it not to be malware.**

        So, we can surmise based on this proof that achieving a 100% accuracy rate on an AI program to detect malware is impossible with the current state of our advancement in computer technology. The question then becomes, what accuracy rate is acceptable for modern malware detection? Does this acceptable accuracy rate change based on the industry in which the program is applied (i.e. defense, healthcare, finance, etc.)?

       Hahn explains that as computer systems and technology become more advanced, his theory is that malware detection accuracy rates will approach, in asymptotic fashion, 100% but never actually make it to 100% due to Cohen’s perfect malware detection theorem [8]. Along the same lines, though, designers of malware detection products must not become satisfied with accuracy rates that seem quite high, but in reality are unacceptable for even the personal computer user.

Chart, scatter chart

Description automatically generated

**Figure 3: Hahn’s graph of the accuracy rates of his malware detection program. The black squares represent the true positive rate, and the black crosses represent the false positive rate.**

        For example, Hahn asks us to consider a situation where a malware detection algorithm has a false positive rate of 5%, which to the average user would seem quite good. Also it is worth noting again that in this case, we are examining false positives, which are instances where the algorithm deems an innocent program to be malware and intercepts the innocent program. Hahn then asks us to consider an average C drive folder on a machine running the Windows 10 Operating System, which is a folder with over 500,000 files. If one were to apply that malware detection algorithm with the 5% false positive rate to the Windows 10 C drive, that would result in 25,000 innocent files being deleted, which is an act that would almost surely crash the machine entirely. All of that would occur under a malware detection algorithm with a 95% accuracy rate [8]. Hahn then poses the question of “How many files would you be willing to bet on?” So it becomes very clear why having an accuracy rate that approaches 100% is crucial.

        Hahn goes on to further distinguish the necessary false positive rates based on whether or not the AI-based malware detection program is autonomous. Hahn contends that if a malware detection program not only detects but also automatically stops malware, then the false positive rate must be as close to 0 as possible [8]. In critical applications, this is especially true, as even one instance of a false positive or false negative could cripple an entire computer network or system, as a crucial file would be deleted. For this reason, we typically do not see automated malware interception programs being used in the most critical of software applications. Instead, these industries employ malware detection algorithms, that then must alert a human to the presence of malware and obtain permission to take any sort of mitigating action against it.

        So, Hahn asks, what is the best approach to creating the most accurate malware detection program? It goes back to a principle discussed at the beginning of this section, the layering principle. Hahn calls it the “Swiss Cheese Model of Malware Defense” [8]

Diagram

Description automatically generated

**Figure 4: An example diagram of Hahn’s Swiss Cheese Model of Malware defense from his article on malware detection [8]**

        As shown in the diagram, this model employs a layered approach in which the vulnerabilities in each layer are backed up by another layer which has its own vulnerabilities, but such vulnerabilities do not overlap or replicate the vulnerabilities from the previous layer, leading to a situation where each piece of malware will, at some layer, be caught. You can see here why, although still an issue, false negatives are not nearly as common in malware detection as false positives. The main worry for designers of malware detection algorithms is not whether malware is slipping through the cracks, but whether their software is labeling innocent programs as malware, thereby threatening the integrity of the system as a whole.

1. DDoS ATTACKS

        Our first subsection of malware that we will discuss is Distributed Denial-of-Service attacks, or DDoS attacks. A DDoS attack is differentiated from a Denial of Service, or DoS, attack because of both the number of devices involved and the method which the attacker uses to transmit the attack. A DoS attack is typically launched from a single machine and consists of an overloading amount of data packets sent to the victim machine. The DoS model allows for the attacking machine to either be an unwitting machine that is infected with malicious software, or a machine that is under the direct possession and use of the DoS attacker. The anatomy of a DDoS attack, however, is much different. A DDoS attack requires that all the attacking machines be under the unwitting influence of the DDoS attacker.

Icon

Description automatically generated

**Figure 5: A diagram demonstrating the anatomy of a Distributed Denial of Service Attack.**

        This is typically accomplished through the use of some sort of malware that the DDoS attacker has installed on numerous different machines, sometimes across distances that span continents or even further. This malware is often installed using another type of malware attack, which we will discuss later in this paper, known as an email phishing attack. Oftentimes when a user clicks an unknown link in their email, that link will download several malware files to that user's computer which are arranged in such a way as to not arouse the suspicion of the user. This malware may lie dormant for months or even years until the DDoS attacker has amassed enough infected devices under their control to launch the DDoS attack. It is worth noting as well that these attackers often use the concept of a “Zero Day,” in which the attacker takes advantage of a flaw in the victim’s DDoS prevention software in order to launch the attack without warning. Once the attack is launched, each of the devices infected with this malware will begin to send large amounts of data packets to a certain IP address or internet server. If the server is not able to stop this flood of data packets, then it will almost invariably crash, causing untold financial damage, and in some cases, costing lives.

        As noted before, DDoS attacks can have devastating effects on our society, especially in the sectors of healthcare, first responder technology, and national security. It is no wonder that a blossoming industry has developed around the question of how to detect, stop, and prevent DDoS attacks. Artificial Intelligence and Machine Learning methods have been widely deployed to assist in solving this problem, but like anything in cyberspace, they must continually evolve to keep up with the latest advancements in malware.

Diagram

Description automatically generated

**Figure 6: Diagram of Tang and Kuang’s proposed taxonomy of DDoS Detection and Defense Mechanisms, modified to include our proposed creation of the Dependent component of Extent of Cooperation to house the interdependent and cooperative components [3].**

        In their paper, “Distributed Denial of Service Attacks and Defense Mechanisms,” Dan Tang and Xiao Kuang propose a taxonomy of DDoS Detection and prevention methods. This taxonomy has eight components, with each component belonging to one of three subsections. The first subsection is *Activity Level*. This subsection contains the components of “Reactive” and “Preventative” DDoS defense mechanisms. This is noteworthy as all DDoS protection algorithms can be categorized as either preventative or reactive [3]. Preventative DDoS protection systems are proactive in their methods, looking for ways to shore up their system’s defenses before an attack is ever launched. This might consist of a sort of “benevolent worm” program that works its way through the different ports on the host server, looking for vulnerabilities and patching them. Reactive DDoS defense mechanisms, on the other hand, sit on the end server and sniff for suspicious traffic. Only once a DDoS attack has already begun do they spring into action, either stopping the nefarious traffic by blocking it, or by creating another virtual server to move the host server to for the duration of the attack, in effect, “evacuating the server” until the coast is clear [3].

        The next subsection that Tang and Kuang propose is Extent of Cooperation, which has to do with the level of outside involvement needed to sustain the DDoS detection algorithm [3]. Although Tang and Kuang propose three components to this subsection, in reality two components would be more appropriate and less redundant. The two components that are more appropriate are autonomous and dependent DDoS detection systems (this is because both interdependent and cooperative DDoS detection systems require external input in order to function properly). The distinction between these two components is important because in theory, as discussed above, no malware detection algorithm can be truly autonomous, lest it become obsolete as soon as malicious actors update their attack methods. What Tang and Kuang are referring to is a DDoS detection system that runs almost entirely autonomously with human intervention only required for periodic updates to the system. On the other hand, dependent DDoS detection systems require constant human intervention to run properly [3].

        The final subsection that Tang and Kuang refer to, is the deployment location of the DDoS detection system. For this subsection, they divide it into three components. These components are the source network, the victim network, and the intermediate infrastructure. At first glance, these locations seem self-explanatory. Tang and Kuang have indeed covered the three possible locations where DDoS detection systems may reside. However, a closer look at these three categories reveals some noteworthy benefits and drawbacks to each [3]. The most notable benefit to having DDoS prevention infrastructure installed at the victim’s network or server would be that the most interested party in preventing a DDoS attack, namely the victim, would have full control over the system. For DDoS prevention infrastructure installed at the source network however, not only would the victim not have control over this system, the cost to implement such a system would be astronomical, as DDoS attacks often involve tens of thousands of attacking devices under the control of a centralized malicious actor. For DDoS prevention infrastructure installed on the intermediate infrastructure (i.e. internet infrastructure) the cost is also great [3]. When considering the best location to install DDoS prevention infrastructure, it is clear that victim-side installations will have the greatest control and benefit for the user. However, it must also be noted that the three pronged approach to DDoS prevention locations would be optimal, if cost were no factor [3].

1. PORT SCANNING ATTACKS

        Port scanning attacks are unique among our selected malware topics in that they themselves do not cause harm to the device or system being attacked. Instead, malicious actors use port scanning as a sort of intelligence-gathering tool in order to launch a separate attack. Cybersecurity scholars differ in their opinions of whether the term “port scanning” refers to the simple act of a piece of malware looking for vulnerable ports on the victim server, or if the term refers to the entire attack from start to finish [9]. For the purposes of this analysis, we will use the term “port scanning” to refer to just the act of looking for vulnerable ports, as this both meets the definition of a malware attack and deserves a section devoted to it entirely.

Diagram

Description automatically generated

**Figure 7: Diagram of Port Scanning attack from start to finish, adapted from *Surveying Port Scans and Their Detection Methodologies*****by Bhuyan. Note that our analysis will only cover the reconnaissance and scanning portions.**

Port scanning, like DDoS, is an almost exclusively internet-based form of malware attack, as it relies on transmitting over TCP/IP connections [9]. A port itself is just a name for the terminal location of an internet connection. Take a web server, for example, that is hosting the website of a large banking corporation. This server will have many ports on it that each serve as a sort of gate to the server. TCP/IP internet traffic will use these gates to connect to the server. At any given time, most of these ports are occupied by a connection. What a port scanning attack attempts to do is to scan these ports for unused or open ports, and then further probe those open ports for vulnerabilities. The question then turns to what kind of vulnerabilities these malicious actors are scanning for. Like any classic tale of malware prevention and detection, it is a constant war of attrition between the defending server and the attacking entity. The port scanning attacker is checking these unused ports to see how robust the intended victim’s defenses are. The attacker does this by masking their identity and sending seemingly innocuous TCP/IP messages to each port, and then measuring the response from each port to determine whether or not the victim is also regularly scanning their ports for possible intrusion [9].

        Like with DDoS Attacks, Machine Learning and Artificial Intelligence-based approaches have found great success in detecting and stopping port scanning attacks. This is resultant from the fact that the pattern of a port scanning attack is quite predictable. A trained AI algorithm can easily detect when many ports on its server are being pinged from the same originating location. The problem arises when the attacker attempts to either mask their true identity, or distribute the port scanning attack across several originating locations. This kind of attack profile demands the swiss-cheese approach to cybersecurity, as discussed above. Designers of anti-port scanning systems must first have a robust detection algorithm in place to look for suspicious network traffic. Once that is in place, the unused ports must be closed to incoming traffic, or if that is not possible, they must be secured in such a way that they cannot be exploited to gain access to the larger system. If this is not possible, then the defending system must vigorously check its open ports so that any probing attackers will receive returns from their port scanning attempts that demonstrate their intended victim is closely guarded.

    While not the main intent of this section, it is important to also discuss the end goal of a port scanning attacker. The port scanning attack itself is not the dangerous part. After learning which ports are unguarded and vulnerable, the attacker will then attempt to exploit any known vulnerabilities through that port. Once the attacker gains access to the server or system through that port, they will then attempt to gain the trust of the server they just hacked into,  giving themselves administrator privileges. While maintaining access to the system, they will use these administrator privileges to create their own private vulnerability or tunnel through which to easily access the server or system in the future. It is through this tunnel that the true damage of a port scanning attack occurs. Attackers may use their access to view and steal confidential data. They may use it to destroy or weaken the system as a whole, or sit quietly on the server and eavesdrop on the packets being processed through the server.

1. AI ATTACKS

        The section devoted to AI attacks is undoubtedly a broad section because by definition, an AI malware attack is any malware attack that uses Artificial Intelligence in its execution. Because such a broad category would go beyond the scope of this paper, we have chosen to focus our research on AI attacks to AI-based email phishing attacks. As a brief background, email phishing is where a malicious actor will send an email to their victim that impersonates a legitimate email from a corporation or trusted entity. The email will often prompt the victim to click a link or button. This link or button will then either inconspicuously download malicious software, or prompt the victim to enter in their personal information to be later used by the attacker. The growing problem with this sort of attack, at least from the attacker’s perspective, is that the general population is becoming wiser and more skilled at identifying these suspicious emails. It is for this occasion that attackers have turned to artificial intelligence to create more realistic and believable phishing emails.

Text

Description automatically generated

**Figure 8: A screenshot of our AI Attack code demonstration. Our demonstration shows how AI can be used to mimic a human voice and deceive a victim into believing the machine they are talking to is actually human.**

However, phishing attacks are not the only branch of AI Attacks. In our demonstration, we show how AI can convincingly portray a human voice, thereby convincing the victim to give up their personal information.

        With AI attacks, just as the category of malware itself is very broad, the approaches to detecting and stopping these attacks are wide ranging as well. Perhaps the most effective method for preventing these sorts of attacks is education about them among the general population. If potential victims are armed with the necessary knowledge to spot the difference between a legitimate email or phone call and one generated by an AI, then the success of these kinds of attacks will be greatly diminished. When all else fails, and the victims fail to identify potential AI attacks, we must turn to Artificial Intelligence itself. Part of the solution to this lies in the email and phone infrastructure. Email service providers and phone providers using Voice Over Internet Protocol (or VOIP) share a great deal of responsibility when it comes to preventing this sort of attack [2]. As we saw before in Tang and Kuang’s taxonomy, the best location to install anti-malware systems is on the potential victim’s device, or on the intermediate infrastructure that could possibly be carrying the attack [3]. This is why these service providers share the main responsibility for the technical detection and prevention of AI attacks. The main responsibility for preventing these attacks, though, still lies with the user.

Graphical user interface, text

Description automatically generated with medium confidence

**Figure 9: A type of captcha method that detects the mouse patterns of the user and uses them to predict whether or not the user is a bot. (ReCAPTCHA 2022)**

Another method for preventing and stopping these attacks is through the use of robust captcha, or “Completely Automated Public Turing test to tell Computers and Humans Apart,” as many of these types attacks rely on bots taking advantage of services meant for humans, so placing a gate on that service that only allows humans to utilize it would prevent these sort of attacks from taking place. The only question is how to best design systems that tell computers and humans apart [2]. While a discussion of captchas is not meant for this section, it is important to note that bots and AI have been learning over the years how to defeat these captchas, so continuous research and development is needed in order to keep up with such advances.

1. CRYPTOJACKING

        Cryptojacking is much different from the previous types of malware in that it is a much narrower category of malware attack. Cryptojacking occurs when a malicious actor installs malware on a victim device or server that is designed to innocuously utilize a small portion of the device’s graphics card in order to mine cryptocurrency for themselves [5]. Ideally, this use of another’s GPU is intended to go unnoticed, so that the victim never knows they have been attacked. The way malicious actors make this type of attack profitable is by distributing it across tens of thousands, or even millions, of devices. While it may not be profitable to use a slice of one’s GPU small enough to not produce a noticeable drawback in performance when taken away, if done across millions of devices, it can become quite profitable for the attacker [5]. There are many ways that the attacker can propagate their malicious software to such a wide net of devices. One way is through email phishing, although this may not reach as many devices as other methods.

Timeline

Description automatically generated

**Figure 10: A diagram from imperva.com showing the anatomy of a typical Cryptojacking attack. (Imperva 2022)**

The most common method for an attacker to spread their cryptojacking malware across a multitude of victim devices is through the use of embedded malware or scripts on unscrupulous websites. When going to unverified or unsecure websites, it is common to encounter ads that cover the page, hidden ads, or ads that mimic legitimate download buttons or other such buttons [5]. In this way, the attacker induces the victim to download their malicious script, which then turns the victim device into a miniature cryptocurrency mining operation. The victim, if all goes well for the attacker, will never be the wiser. Each time a new bit of cryptocurrency is mined by the victim device, it is sent to and housed in the cryptocurrency wallet of the attacker, along with the cryptocurrency being sent by the tens of thousands of other infected devices.

        While at first glance, the only harm coming to the victims is the fact that their computer has been made into a slave unit for the attacker to do with as he pleases, it actually goes much deeper than this. Firstly, this attack is a form of cybercrime and is prosecuted aggressively by the authorities. Secondly, long term use of a slice of your GPU in this manner to mine cryptocurrencies can slow down the performance of your computer to a noticeable level, creating not only a nuisance, but almost a sort of theft. The attacker is stealing your computer performance so that they can make money, essentially. When one considers fully the impact of cryptojacking attacks, it becomes clear why detecting, stopping, and preventing these sorts of attacks is crucial.

Text

Description automatically generated

**Figure 11: A screenshot of our code to simulate a cryptojacking attack**

        Detecting cryptojacking can prove to be quite difficult, as these attackers specifically design their malware to be impervious to detection over long periods of time, as that is what allows these attacks to be profitable. Once the malware is downloaded onto the victim’s computer, it is virtually undetectable. This is why the main method for detecting and stopping cryptojacking is to monitor the network traffic for suspicious activity [6]. As a component of a cryptojacking attack, the attacker must have some sort of internet connection between their victim devices and their cryptocurrency wallet. This connection must send packets on a semi-regular basis, and this is something that an AI program can be trained to look for and either shut it down autonomously, or alert the victim to the fact that their device is being used in a cryptojacking attack. In addition, if the devices under attack are part of a large organization or corporation, then that corporation’s tech or IT department can monitor their users’ devices for unusual reports of decreases in performance [6].

        When a cryptojacking attacker’s stealth code begins to become more obvious, one of the first symptoms is an unusual or new slowdown of the GPU or of the device as a whole. Responding quickly to these symptoms can help ensure a cryptojacking attack does not spread throughout an organization’s IT infrastructure. In addition, as mentioned above, JavaScript is one way that cryptojacking attackers insert their code onto a victim machine, so implementing robust JS security measures on web browsers is an excellent way to prevent these sort of attacks, along with general education among end users of the risks of cryptojacking attacks and how to avoid falling victim to them. [6]

1. IoT ATTACKS

        IoT, or Internet of Things, Attacks are another example of a broad category of malware attacks that we decided to narrow down to focusing on IoT Login Attacks, but there are numerous vulnerabilities with IoT devices and subsequently numerous types of IoT attacks. One main issue with IoT devices is that almost all IoT products are designed with the device functionality as the foremost priority, with internet connection and cybersecurity oftentimes an afterthought. This is, on its face, bad design, as these devices are meant to be introduced to the network of either a domestic or commercial end user. As one node on the larger network, an unsecured IoT device could provide a perfect loophole through which to launch an attack on the larger network. Unsecured IoT devices can even be used to launch DDoS attacks, as discussed above, because they are capable of sending packets of data through the network they are connected to [11].

Chart, pie chart

Description automatically generated

**Figure 12: A graphic from Help Net Security shows the most commonly attacked IoT Devices.**

        One key security flaw with IoT Devices is that many of them come with a preset, factory standard login username and password. Malicious actors know this too, and the fact that the majority of end users do not change this factory-set login information means that these devices are highly vulnerable to attacks [11]. Once malicious actors gain access to one device, their access can spread from device to device, and then from device to network. Once the malware has spread to the network, there is little stopping it from attacking more critical devices on that network, such as servers or end user computers. When it comes to IoT attacks, the old adage that a team is only as strong as its weakest link rings true, and can be applied to any network which houses IoT devices. Due to the necessity of securing the weakest link (namely the IoT devices), it is crucial to explore how these attacks can be detected, stopped, and prevented.

        As discussed before, one of the main prevention methods for all types of malware attacks is simple education of users. If users understand the vulnerabilities in their devices and how to mitigate those vulnerabilities, that takes away one avenue for an attacker to use. Take the previously discussed login attacks for IoT devices, for example. If users are educated on the importance of changing the factory login information on their IoT devices, then that vulnerability is closed immediately. For a systems-based approach to detecting and stopping IoT attacks, many are unaware that it is possible to install malware detection and prevention algorithms directly onto the IoT device itself.

Text

Description automatically generated

**Figure 13: A screenshot of our IoT Login Attack code which is designed to entice a user to give out their IoT login by spoofing a legitimate email from their IoT service provider.**

        In addition to this approach, it is important to secure the network where the IoT Devices will be connected to. This can be done with a robust firewall or commercial antivirus software. What is noteworthy about these kinds of software is that they already implement the latest Artificial Intelligence based methods to stopping these attacks. Another large risk with IoT devices is the fact that they deal largely with unencrypted data, so if the network on which they reside deals with sensitive information, it is crucial to keep unencrypted IoT devices on a separate network from devices that rely on encryption for data privacy and security.

1. CONCLUSION

          One main theme that has been clear throughout this exploration of various malware categories, attacks, and detection methods, is that the world of malware is an ever-changing and ever-evolving world. Cybersecurity analysts must constantly stay vigilant as malicious actors look for new ways to exploit vulnerabilities. Across and during our survey and research of these malware categories, we came across a few common principles. Firstly, in every malware attack, the attacker is looking for holes, weaknesses, and vulnerabilities to exploit. If the potential victim is also looking for their own weaknesses, then many times these attacks can be stopped before they even start, when the potential victim is able to patch those weaknesses. Second, lack of awareness of malware is both one of the most dangerous vulnerabilities, and the easiest to fix. A user who is oblivious to the fact that, for example, clicking on suspicious ads on unscrupulous websites is dangerous online behavior could very well bring their entire company’s network down. Simple education and familiarization with malware, vulnerabilities, and how to prevent them pays dividends when it comes to cybersecurity and malware. Finally, across all of these categories, Artificial Intelligence is a valuable tool, however potential users and vendors of AI systems must be vigilant so as to not allow their AI to fall behind in the times. Artificial Intelligence algorithms must be continually taught, and most importantly, guarded against false information and fake teaching by the very same malicious actors the AI is attempting to protect against. If these principles are adopted and adhered to, many malware attacks happening today could be stopped, preventing such AI malware detection algorithms from being overwhelmed with suspicious incidents and attacks.

WORKS CITED

[1]    “2022 cyber security statistics trends & data,” *PurpleSec*, 17-Oct-2022. [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/. [Accessed: 29-Nov-2022].

[2]    B. Guembe, A. Azeta, S. Misra, V. C. Osamor, L. Fernandez-Sanz, and V. Pospelova, “The emerging threat of AI-Driven Cyber Attacks: A Review,” *Applied Artificial Intelligence*, vol. 36, no. 1, 2022.

[3]    D. Tang and X. Kuang, “Distributed denial of service attacks and defense mechanisms,” *IOP Conference Series: Materials Science and Engineering*, vol. 612, no. 5, p. 052046, 2019.

[4]    Address false positives/negatives in Microsoft defender for endpoint,” *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/defender-endpoint-false-positives-negatives?view=o365-worldwide. [Accessed: 29-Nov-2022].

[5]    E. Chickowski, “What is cryptojacking? how to prevent, detect, and recover from it,” *CSO Online*, 20-Jun-2022. [Online]. Available: https://www.csoonline.com/article/3253572/what-is-cryptojacking-how-to-prevent-detect-and-recover-from-it.html?page=2. [Accessed: 29-Nov-2022].

[6]    H. H. Hajri, B. M. Mughairi, M. I. Hossain, and A. M. Karim, “Crypto jacking a technique to leverage technology to mine crypto currency,” *International Journal of Academic Research in Business and Social Sciences*, vol. 9, no. 3, 2019.

[7]    H. Kumar, “Survey on Malware Detection Methods.”

[8]    K. Hahn, “The real reason why malware detection is hard--and underestimated,” *G Data Software*, 15-Jul-2022. [Online]. Available: https://www.gdatasoftware.com/blog/2022/06/37445-malware-detection-is-hard. [Accessed: 29-Nov-2022].

[9]    M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Surveying port scans and their detection methodologies,” *The Computer Journal*, vol. 54, no. 10, pp. 1565–1581, 2011.

[10]    O. Aslan and R. Samet, “A Comprehensive Review on Malware Detection Approaches,” *IEEE Access*, Jan. 2020.

[11]    Valeo Networks, “How is the internet of things (IOT) being impacted by malware?,” *Valeo Networks*, 10-Sep-2021. [Online]. Available: https://www.valeonetworks.com/how-is-the-internet-of-things-iot-being-impacted-by-malware/. [Accessed: 29-Nov-2022].