המרכז האקדמי רופין
Ruppin Academic Center
המחלקה להנדסת חשמל ומחשבים
Department of Electrical and Computer Engineering

Version 7.0

16/01/2022

CYGHT

Marvel

Scanning and parsing artifacts to extract executables from remote computers through communication protocols

Progress Report – Final Project תשפ"ב

Supported by the company CYGHT

Table of Contents

1 Abstract

1.1 Abstract in Hebrew

התקפות סייבר הפכו לקשות יותר לזיהוי עם חלוף השנים, הזמן הממוצע לזיהוי פריצה בשנת 2020 היה 228 ימים ואף הזמן הממוצע להכלת הפריצה היה 80 ימים [1]. כל מתקפה נהיית יותר מתוחכמת וממוקדת, דבר המותיר כל סביבה שאינה מוכנה כראוי כפשוטה לפריצה. בכדי לפתור בעיה זו, ארגונים רבים יצרו או רכשו מערכות אנטי וירוס וחומות אש איכותיות. למרות זאת הופתעו כשגילו שהרשת שלהם הוצפנה על ידי מפעיל תוכנת כופר [2]. לעיתים קרובות, ניתוח ריצת תוכניות הינו גורם משמעותי במהלך חקירות דיגיטליות, זאת על מנת לגלות אילו תוכנית רצו במערכת במהלך אותה הפריצה, ואף היסטוריית הביצוע שלהן [3]. לכן ישנו צורך להבין כיצד התרחשה התקיפה, אילו קבצים היו מעורבים, כך הפרויקט שלנו נכנס לתמונה.

Marvel הינה תוכנה אשר מתחברת למחשבים ברשת אזורית בעזרת פרוטוקולי תקשורת שונים, ולאחר מכן תסרוק ותנתח ארטיפקטים (טביעות רגל של הפורץ), לבסוף תחלץ קבצי הרצה שנמצאו במחשב מרוחק אל המחשב הראשי, כדי שניתן יהיה להמשיך בחקירתם. תהליך זה יעזור לצוות IR (תגובה לאירוע) להשקיע זמן רב יותר בחקירת קבצי ההרצה ולא להתעקב בחיפושם, דבר העלול לקחת זמן רב. כתוצאה מכך, תהליך החקירה יהיה יותר מהיר ויעיל, ויעזור להתמודד עם תקריות סייבר.

הפרויקט פותח ב-Microsoft Visual Studio עם C#. החלק הראשון של הפרויקט הינו החלק התקשורתי. המחשב הראשי יסרוק אחר כתובות IP באזורינו על ידי שיטות שונות שונות, לאחר מכן יתחבר המחשב הראשי אל אותן כתובות IP באמצעות פרוטוקולי תקשורת כגון: WMI [4], SMB [5], SSH [6] ו-WinRM [7]. בכדי לנפות באגים ולהבין בעיות תקשורת נשתמש בתוכנה Wireshark. החלק השני של הפרויקט הינו סריקת המחשבים המרוחקים, ניתוח וחילוץ קבצי הרצה. תהליך זה יעבוד בדומה לתוכנה KAPE [8], ולבסוף התוכנית תעביר את הקבצים למחשב הראשי.

בעזרת תוכנית זו ארגונים יכולים להתאושש ממתקפות סייבר בצורה מהירה ויעילה הרבה יותר, מה שיעזור לנו לחיות בעולם בטוח יותר.

1.2 Abstract in English

Cyber-attacks became harder to detect, the average time to identify a breach in 2020 was 228 days and the average time to contain a breach was 80 days [1]. Each attack becomes more and more sophisticated and are better targeted, which leaves any environment that isn't prepared properly easy for an attack. To solve this problem organizations, have created or purchased quality working antivirus systems and firewalls and were surprised when they discovered their network has been encrypted by a ransomware operator [2]. Program execution analysis is often desirable in digital investigations to know the programs executed on the system and their execution history [3]. Because of this there is a need to understand how the attack occurred and what has been infected and that's where our project comes into play.

Marvel is a program that will connect to computers within a network with the help of communication protocols, and then scan and parse artifacts (a footprint of the culprit) and extract executable files found to the main computer so they can be further investigated. What this does is it helps an IR (Incident Response) team to spend more time on investigating the executable files instead of searching for them, which can be very time-consuming. In result, this leads to a much more fast and efficient way to deal with cyber-incidents.

The program was developed in Microsoft Visual Studio with C#. The first part of the program which is the communication part, will receive IP addresses over a network through various methods and then connect to them with communication protocols such as WMI [4], SMB [5], SSH [6], and WinRM [7]. To debug and understand communication issues we will use

the program Wireshark. The second part of the program which scans, parses, and extracts executable files will work similarly to the program Kape [8] and then send everything to the main computer.

With this program organizations can recover from cyber-attacks in a quicker and much more efficient way, which will help us live in a safer world.

1.3 Keyword list

Lateral movement, remote execution, communication protocols, Model-View-ViewModel, forensic artifacts, Eric Zimmerman tools, Windows error reporting, Windows forensic analysis and program execution.

2 Tables and terms

2.1 Table of figures

2.2 Table of tables

2.3 Glossary of terms

Computer Networking- A computer network is a set of computers sharing resources located on or provided by network nodes. The computers use common communication protocols over digital interconnections to communicate with each other [9].

Communication Protocols- A communication protocol is a system of rules that allows two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both [10].

Forensics Artifacts- Forensics artifacts are objects that have forensic value. Meaning any objects that contain data or evidence of something that occurred [11]. An artifact in a digital forensics' investigation includes things like registry keys, files, timestamps, and event logs – all of these are the traces we follow in digital forensic work [12].

Incident Management- Incident management refers to a set of practices, processes, and solutions that enable teams to detect, investigate, and respond to incidents. Incident management processes ensure that IT teams can quickly address vulnerabilities and issues [13].

Lateral Movement- Lateral Movement consists of techniques that adversaries use to enter and control remote systems on a network. Following through on their primary objective often requires exploring the network to find their target and subsequently gaining access to it. Reaching their objective often involves pivoting through multiple systems and accounts to gain. Adversaries might install their own remote access tools to accomplish Lateral Movement or use legitimate credentials with native network and operating system tools, which may be stealthier [14].

Execution- Execution consists of techniques that result in adversary-controlled code running on a local or remote system. Techniques that run malicious code are often paired with techniques from all other tactics to achieve broader goals, like exploring a network or stealing data. For example, an adversary might use a remote access tool to run a PowerShell script that does Remote System Discovery [15].

Windows Presentation Foundation (WPF)- Windows Presentation Foundation is a UI framework that creates desktop client applications. The WPF development platform supports a broad set of application development features, including an application model, resources, controls, graphics, layout, data binding, documents, and security [16].

Model-View-ViewModel (MVVM)- A software design pattern that helps to cleanly separate the business and presentation logic of an application from its user interface (UI). Maintaining a clean separation between application logic and the UI helps to address numerous development issues and can make an application easier to test, maintain, and evolve. The MVVM pattern is used in WPF [17].

Windows Error Reporting (WER)- A flexible event-based feedback infrastructure designed to gather information about the hardware and software problems that Windows can detect, report the information to Microsoft, and provide users with any available solutions [18].

Persistence- Persistence consists of techniques that adversaries use to keep access to systems across restarts, changed credentials, and other interruptions that could cut off their access. Techniques used for persistence include any access, action, or configuration changes that let them maintain their foothold on systems, such as replacing or hijacking legitimate code or adding start-up code [19].

3 Introduction

Cyber-attacks have become harder to detect, each attack becomes more and more sophisticated and are better targeted, which leave any environments that isn't prepared properly easy for an attack. To solve this problem organizations, have created or purchased quality working antivirus systems and firewalls and were surprised when they discovered their network has been encrypted by a ransomware operator [2]. This example is only the tip of the iceberg of many "solutions" organizations have used to prevent cyber-attacks and sadly failed, because as always nothing is safe on the internet.

Our product will help organizations deal with a cyber-incident by giving them an opportunity to understand how the attack occurred and investigate the footprints left behind by the culprit. In result, this will lead to a better recovery from a cyber-attack.

3.1 Defining the problem

Cyber-attacks have become harder to detect, the average time to identify a breach in 2020 was 228 days, the average time to contain a breach was 80 days and the global average cost of a data breach is 3.86 million dollars. It is estimated that a business will fall victim to a ransomware attack every 11 seconds by 2021 [1].

When investigating an attack, typical questions concerning its very beginning are centred on whether the attackers already left traces in form of suspicious behaviour. By knowing which traces the attackers leave, help to better understanding how an attack occurred and how to recover from an attack [20].

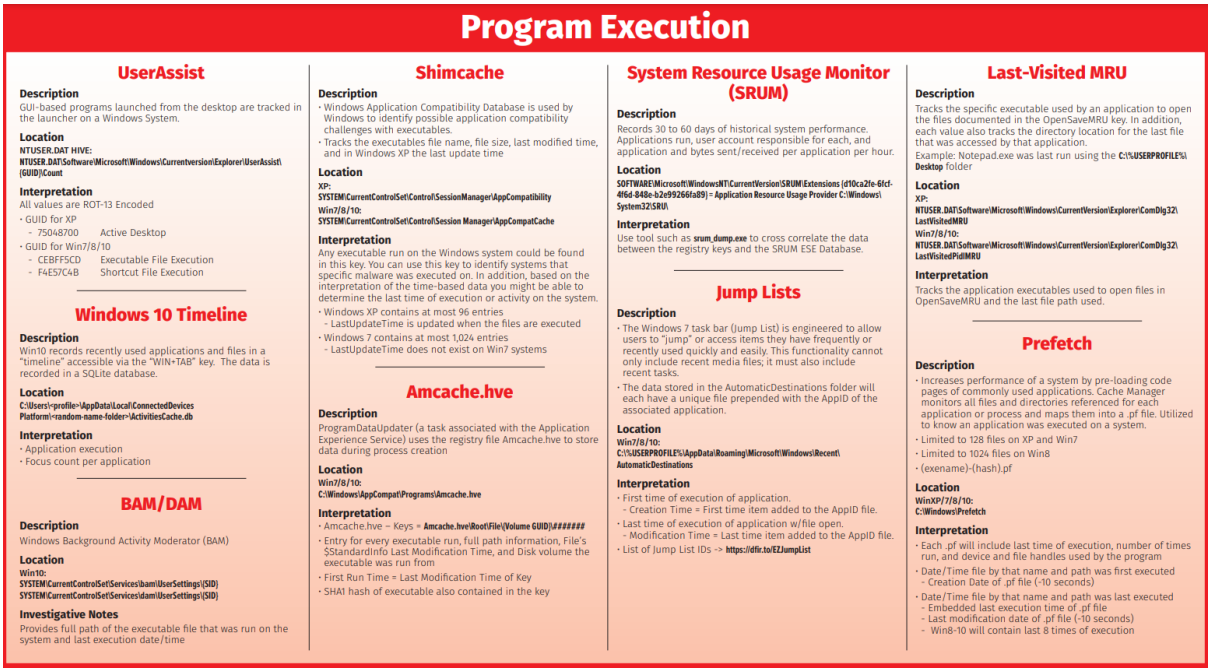
3.2 The technological problem

Our product gives an organization a much more efficient way to understand the reason of a cyber-attack. Instead of wasting time and going through every computer one by one within the organization, our product can connect to all the computers over an organization's network and in each computer scan, process, and collect any problematic files or artifacts ("footprints" left behind by the culprit) and upload them to the main computer, so they can be further investigated.

4 Ways to solve the problem

To solve the problem our program will work according to multiple methods and similarly too several tools in two parts. First the program will start with the communication part and later it will continue with the scanning, parsing, and extracting part.

To start with the first part our program has several ways to get hosts, for example the user can enter them manually, scan for them automatically or extract them from a text file. Once each host is added, the program will Ping each host to test their connectivity, scan their Ports and display their connectivity. A program which gave us a lot of inspiration in doing this is the network scanning tool Nmap [21]. Once the hosts were added into the program and the user can see their information, the user may start running commands through communication protocols like WMI [4], SMB [5], SSH [6], and WinRM [7]. Some of the commands which can be set are transferring items between the computers, getting specific directories displayed or executing a program in the remote computer. These commands were achieved by writing them ourselves or with the help of APIs like SSH.NET [22]. A program which helped us understand how to create the communication commands and what was needed is the large-scale intranet penetration scanner and Cobalt Strike, Ladon [23], which explains and shows how to execute remote commands.



Program Execution

UserAssist

Description

GUI-based programs launched from the desktop are tracked in the launcher on a Windows System.

Location

NTUSER.DAT HIVE:
NTUSER.DAT\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\{GUID}\Count

Interpretation

All values are ROT-13 Encoded

- GUID for XP
  - 75048700 Active Desktop
- GUID for Win7/8/10
  - CEBFF5CD Executable File Execution
  - F4E57C4B Shortcut File Execution

Windows 10 Timeline

Description

Win10 records recently used applications and files in a "timeline" accessible via the "WIN+TAB" key. The data is recorded in a SQLite database.

Location

C:\Users\<profile>\AppData\Local\ConnectedDevicesPlatform\<random-name-folder>\ActivitiesCache.db

Interpretation

- Application execution
- Focus count per application

BAM/DAM

Description

Windows Background Activity Moderator (BAM)

Location

Win10:
SYSTEM\CurrentControlSet\Services\bam\UserSettings\{SID}
SYSTEM\CurrentControlSet\Services\dam\UserSettings\{SID}

Investigative Notes

Provides full path of the executable file that was run on the system and last execution date/time

Shimcache

Description

- Windows Application Compatibility Database is used by Windows to identify possible application compatibility challenges with executables.
- Tracks the executables file name, file size, last modified time, and in Windows XP the last update time

Location

XP:
SYSTEM\CurrentControlSet\Control\SessionManager\AppCompatibility
Win7/8/10:
SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache

Interpretation

Any executable run on the Windows system could be found in this key. You can use this key to identify systems that specific malware was executed on. In addition, based on the interpretation of the time-based data you might be able to determine the last time of execution or activity on the system.

- Windows XP contains at most 96 entries
  - LastUpdateTime is updated when the files are executed
- Windows 7 contains at most 1,024 entries
  - LastUpdateTime does not exist on Win7 systems

Amcache.hve

Description

ProgramDataUpdater (a task associated with the Application Experience Service) uses the registry file Amcache.hve to store data during process creation

Location

Win7/8/10:
C:\Windows\AppCompat\Programs\Amcache.hve

Interpretation

- Amcache.hve – Keys = **Amcache.hve\Root\File\{Volume GUID}\######**
- Entry for every executable run, full path information, File's $StandardInfo Last Modification Time, and Disk volume the executable was run from
- First Run Time = Last Modification Time of Key
- SHA1 hash of executable also contained in the key

System Resource Usage Monitor (SRUM)

Description

Records 30 to 60 days of historical system performance. Applications run, user account responsible for each, and application and bytes sent/received per application per hour.

Location

SOFTWARE\Microsoft\WindowsNT\CurrentVersion\SRUM\Extensions {d10ca2fe-6fcf-4f6d-848e-b2e99266fa89} = Application Resource Usage Provider C:\Windows\System32\SRU

Interpretation

Use tool such as **srum_dump.exe** to cross correlate the data between the registry keys and the SRUM ESE Database.

Jump Lists

Description

- The Windows 7 task bar (Jump List) is engineered to allow users to "jump" or access items they have frequently or recently used quickly and easily. This functionality cannot only include recent media files; it must also include recent tasks.
- The data stored in the AutomaticDestinations folder will each have a unique file prepended with the AppID of the associated application.

Location

Win7/8/10:
C:\%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations

Interpretation

- First time of execution of application.
  - Creation Time = First time item added to the AppID file.
- Last time of execution of application w/file open.
  - Modification Time = Last time item added to the AppID file.
- List of Jump List IDs -> **https://dfir.to/EZJumpList**

Last-Visited MRU

Description

Tracks the specific executable used by an application to open the files documented in the OpenSaveMRU key. In addition, each value also tracks the directory location for the last file that was accessed by that application.

Example: Notepad.exe was last run using the **C:\%USERPROFILE%\Desktop** folder

Location

XP:
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU
Win7/8/10:
NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedPidMRU

Interpretation

Tracks the application executables used to open files in OpenSaveMRU and the last file path used.

Prefetch

Description

- Increases performance of a system by pre-loading code pages of commonly used applications. Cache Manager monitors all files and directories referenced for each application or process and maps them into a .pf file. Utilized to know an application was executed on a system.
- Limited to 128 files on XP and Win7
- Limited to 1024 files on Win8
- {exename}-{hash}.pf

Location

WinXP/7/8/10:
C:\Windows\Prefetch

Interpretation

- Each .pf will include last time of execution, number of times run, and device and file handles used by the program
- Date/Time file by that name and path was first executed
  - Creation Date of .pf file (-10 seconds)
- Date/Time file by that name and path was last executed
  - Embedded last execution time of .pf file
  - Last modification date of .pf file (-10 seconds)
  - Win8-10 will contain last 8 times of execution

Figure 1: Windows Forensic Analysis Program Execution table

In the second part our program will scan each selected host for forensic artifacts, parse them and finally extract an executable file. With the help of the "Program Execution" table in "Windows Forensic Analysis Poster" that is shown in Figure 1 above, which was taken from SANS [24] we have a better understanding of the type of the artifacts we are dealing with, where to locate them and how to tackle them. Under each category the location of such artifacts is written, and we can use the tools recommended in Figure 1 interpretation, use existing APIs, create our own or use Eric Zimmerman's tools [25] to parse the files. Once we parse a file, we will have the name and the directory of a used executable, which will help us locate it and extract it to the main computer with the help of our protocol commands. Each executable we find will be saved in a folder that will be named as the host's IP, indicating where it came from, and in that folder the executable will be placed into a folder named after the method that found it.

Both parts of the program can be seen in the block schemes below Figure 2 and Figure 3.

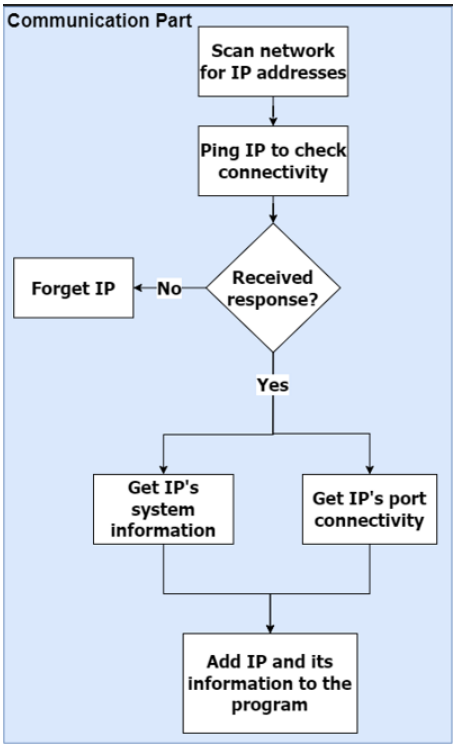


Figure 2: Communication Part Block Scheme

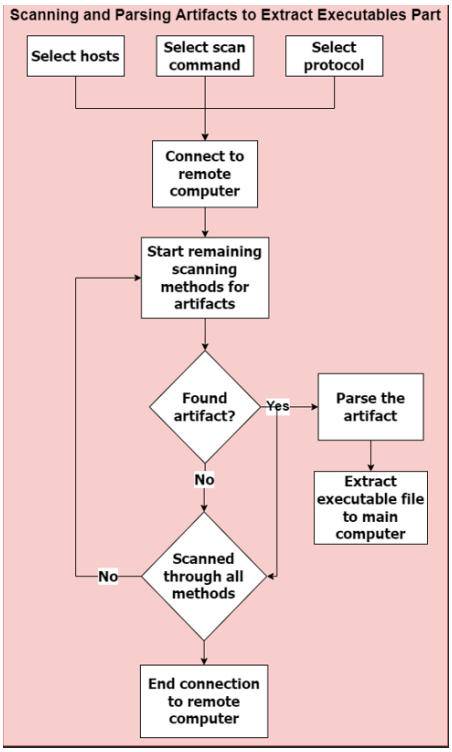


Figure 3: Scanning and Parsing Artifacts to Extract Executables Part Block Scheme

Gathering all the executables which were found in the remote computers into one main computer, saves the time of running a specific tool if not more to locate the artifacts, time which can be put into better use to further investigate any executables that were found.

These stages of operation show how our program works and how it helps IR (incident response) teams handle cyber-incidents by it being better organized and a lot less time consuming, which in result leads into a better recovery process.

5 Expected product from the project

Our product's user interface takes a lot of inspiration from the Zenmap GUI [26] which can be seen in Figure 4 below.

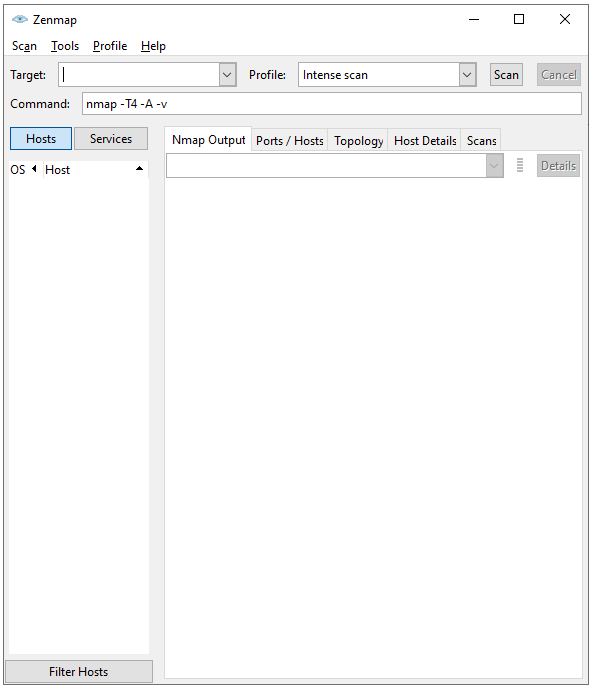


Figure 4: Zenmap GUI

The program will be set into multiple sections from top to bottom. The first section will be the host handling section, in this section the user can enter hosts by multiple ways, for example enter the host's IP, username and password manually, press a button to discover local IPs or extract hosts from a text file. The next section will be the command section, the user can choose from a combo box which communication protocol to work with and from another combo box which command to execute, below there will be two text boxes which will provide the user the possibility to choose the source and destination directories for the commands. In the last section the user can see the hosts that were added to system and information about them, their ports and commands that were executed on them. The user may be able to save things within the program like added hosts or their information in case they would like to use or see it at a different time.

Our program will be developed in Microsoft Visual Studio in the .NET 5 framework which will be coded in C#. To write the GUI of the program we will be using WPF and will work according to the software architectural pattern Model-View-ViewModel. For each host that is running a command we will start a System.Threading.Tasks so that every host can run in parallel. In our program we are going to use the communication protocols SSH, WinRM, SMB and WMI. To run the communication protocols the program will use existing APIs like SSH.NET [22], or we can run another process within our program to execute commands from tools like CMD and PowerShell. To scan and parse the artifacts within remote computers

our program may use tools like Eric Zimmerman's [25]. Finally, to extract executables found by the program we will be using again our protocol commands.

6 Describing a similar idea that can be inspiring

A similar idea to represent our project is an investigator for crime scenes. Imagine when a crime (cyber-attack) happens the police (incident response team) will call an investigator (our system), the investigator's job is to look for any evidence (artifacts) left behind by the culprit (hackers) and bring any evidence found back to the police, so they can arrest the culprit or prevent the crime from happening again.

7 Risks, uncertainties, and project constraints

In case our program won't be able to discover hosts within a local network by itself for any unexpected reason, we provide an option to input hosts manually or from a text file. If there is a problem connecting to computers over the network, we can use Wireshark to debug and understand how to solve any unpredicted communication problem. Since the program may run commands on many hosts this action can take a long time to end, to solve the problem each host that runs a command will have its own Task, this gives our hosts the option to run in parallel and save time. In case a Task is stuck and will never end our program will know how to end it and notice the user with a command failed window so they can run it again or change their settings. While studying how to use multiple communication protocols and later looking for existing APIs to use them, if an API wasn't found we achieved the communication protocol commands by starting another process within our program to use tools like CMD or Powershell so we can run the commands from them with the protocol that was needed. One of the problems with professional and experienced hackers is that they may clean their traces and for that reason we are aiming to use all the Windows Forensic Analysis Program Execution methods which can be seen in Figure 1, by doing this the perpetrators won't be able to clean their traces too easily. Completing this project in time will be a very difficult task, since everything that's being dealt with is new to us and it requires a lot of deep knowledge and investigating, but as difficult this project is it is as interesting which motivates us to reach the best of its capabilities.

8 Related works (literature survey)

Some of the related works in our field are: k8gege/Ladon [23], Nmap [26], sshnet/SSH.NET [22], KAPE [8] and Eriz Zimmerman's tools [25].

Ladon is a multi-threaded plug-in comprehensive scanning artifact for large-scale network penetration, including port scanning, service identification and many more features. The two features that interest us the most in Ladon are port scanning and executing remote commands. To execute remote commands Ladon uses protocols such as SSH, WinRM, SMB and WMI which we were interested in using in our project.

Nmap is a network scanner created by Gordon Lyon. Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses [27]. The

features that interest us in Nmap are host discovery, port scanning, version detection and the Zenmap GUI. Our project design was inspired by Zenmap.

SSH.NET is a Secure Shell library for .Net that is optimized for parallelism [22]. We used SSH.NET in our SSH commands, for example we used the "RunCommand" method to execute commands on a remote computer and the "Upload", "Download" methods to send and receive files from a remote computer.

KAPE stands for Kroll Artifact Parser and Extractor. KAPE is an efficient and highly configurable triage program that will target essentially any device or storage location, find forensically useful artifacts, and parse them [8]. A feature which interests us in KAPE is parsing capabilities and its output which is sent into a file called EvidenceOfExecution.

Eric Zimmerman's tools have many uses and the use we are most interested in is parsing artifacts. A specific tool we are looking into is Windows Prefetch Parser, this can help us parse .pf files and help us discover applications that were executed on a remote system.

Our project will include many of the features mentioned combined in one program, this will significantly help IR teams in investigating cyber-incidents. The order in which our program will function is, first we will discover hosts and get information about their ports and OS, like Nmap. Then we will make a connection with every remote computer and execute a command like Ladon and SSH.NET. Once a scan command is executed our program will scan and parse forensic artifacts with tools like Eric Zimmerman's. At last, we will extract the executable files like KAPE but instead of extracting the executables to the same computer we will extract them to the main computer with the help of protocols like SSH, WinRM, SMB and WMI.

9 Specifies Functional Requirements

9.1 Requirements from the project

The program will be able to discover any hosts in a local network, give information about the hosts and scan their ports to return their connectivity. With each host that has valid credentials the program can execute remote commands, get directories of a remote computer, transfer and execute remote files. The scan command in the program will track executables used by an application, parse them, and extract them to the main computer which will be saved in a file according to their fitting Program Execution type, for example Prefetch, UserAssist, Shimcache and Amcache.hve.

9.2 How to operate and use the project

Once the program is running the user can insert a host manually by writing the host's details in the "IP", "Username" and "Password" text boxes, then press the "Add Host" button as you can see below in Figure 5.

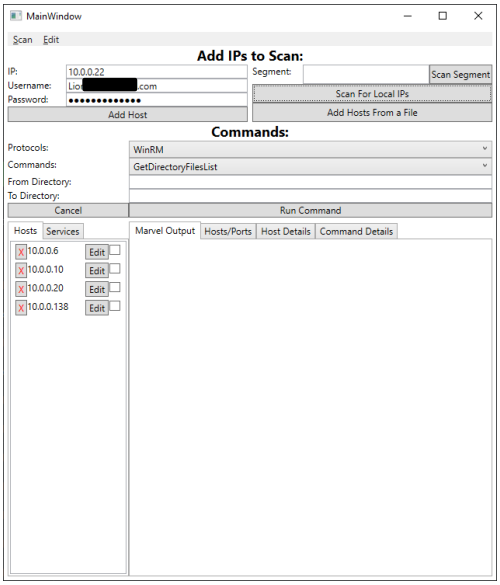


Figure 5: adding hosts by "Add Host" and "Scan for Local IPs"

The user may also add hosts from a text file or by scanning the network with the buttons "Add Hosts from a File" and "Scan for Local IPs". Once hosts were added, the user can run commands on them with different protocols through the "Protocols" combo box and choose a command with the "Commands" combo box. The user may write a directory into the "From Directory" and "To Directory" text boxes to direct the commands source and destination. Once the "Run Command" button has been pressed, every host that has its check box checked will run that set command. After the command is done the user can highlight a host by selecting it and the command output will be displayed in the "Marvel Output" tab. As you can see below in Figure 6 the use of the SSH protocol and "GetDirectory" command help us get the directory "D:\Ruppin" of the host "10.0.0.22".

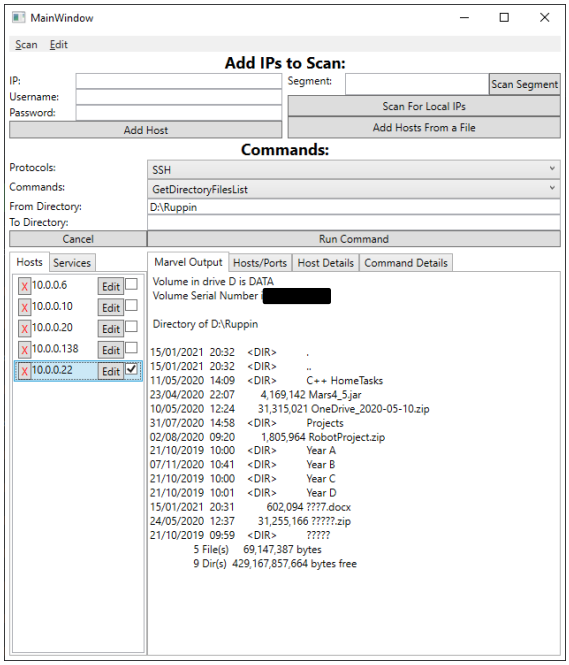


Figure 6: getting directory from remote computer with SSH protocol

Another example that can be seen is the transfer command "ReceiveItem" in Figure 8. In Figure 7 as you can see, we create a text file called "ProjectExample.txt" at the host

"10.0.0.22" computer. After we press the run command button in Figure 8 we transfer the file from the directory of the computer "10.0.0.22" to another directory of the computer "10.0.0.12" which are both inserted into the textbox. Finally, in Figure 9 we can see there is a text file called "ProjectExample.txt" in host 10.0.0.12 which indicates the transfer was successful.

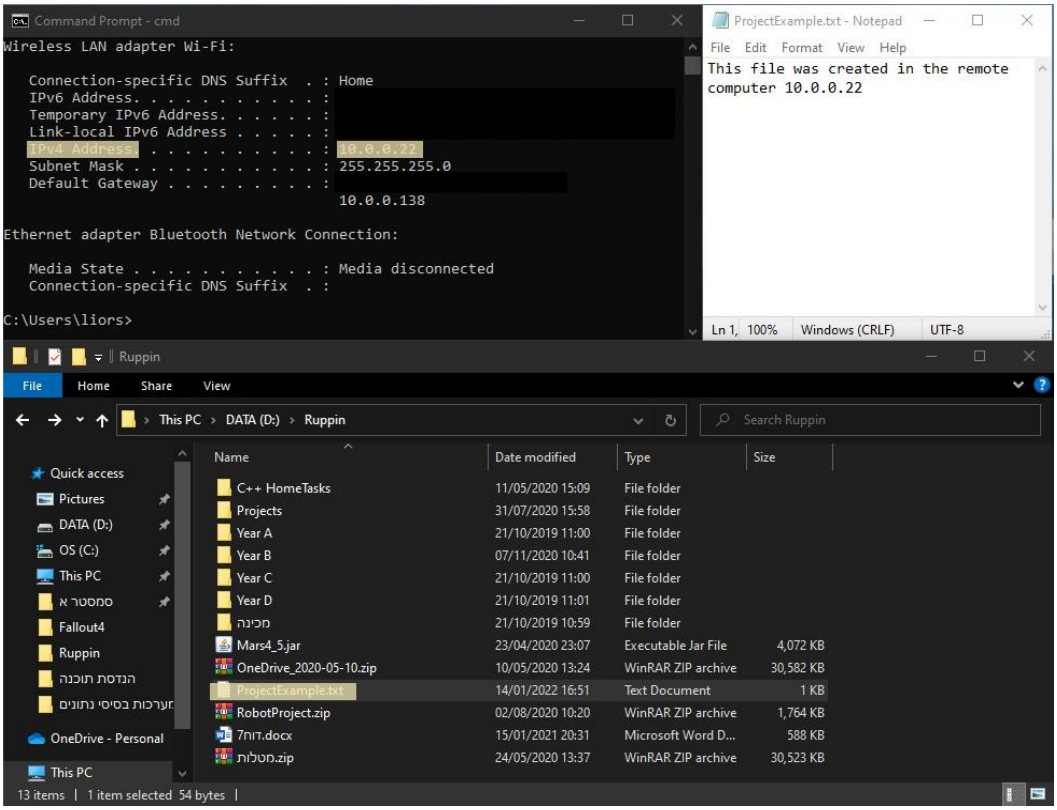


Figure 7: file "ProjectExample.txt" created in remote computer 10.0.0.22

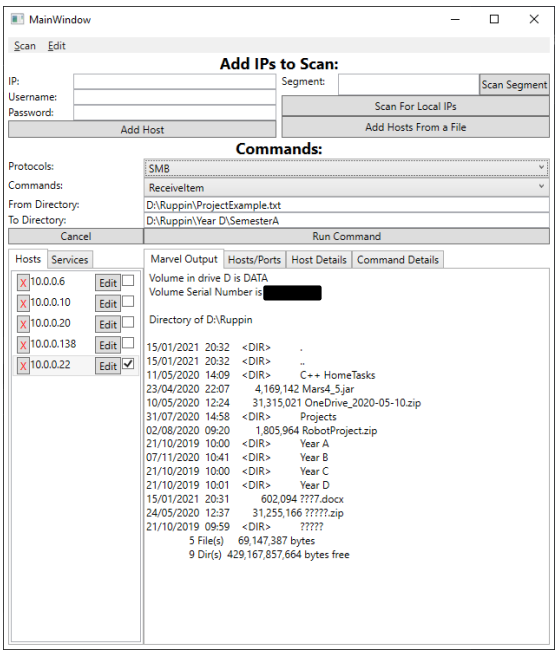


Figure 8: using "ReceiveItem" command with SMB protocol to transfer file from 10.0.0.22 to 10.0.0.12

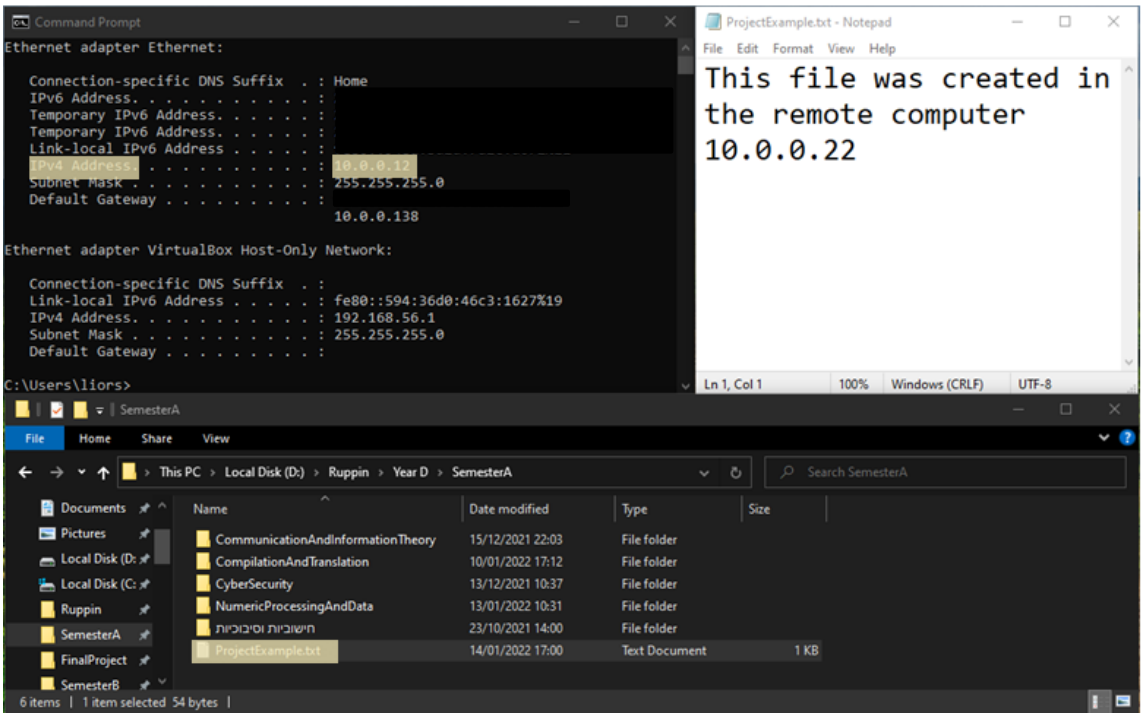


Figure 9: file "ProjectExample.txt" received in main computer 10.0.0.12

While running protocol commands, the system may run into an error where it will never finish the command for several reasons. Since each host that runs a command is set as a "System.Threading.Tasks", the system will know when to end a "Tasks" life in case it is stuck and return an error message to the user so they can run the command again or change the command.

9.3 Specification hardware requirements

The program will fully work on Windows 10, but most of its features are expected to also work on Windows 8 and lesser on Windows 7 or Windows XP. To execute remote commands and transfer files there must be a server and client in the main and remote computers respectively according to each protocol used in the program, for example OpenSSH must be installed to use SSH protocol commands. The firewall of the remote computers must be turned off or set to allow communication from the main computer to enter its ports.

10 What has been done so far in the project

When we only started the project our advisor from the industry first sent us terms like "Lateral movement", "Program Execution" and "Persistence" from MITRE ATTACK [28] to enrich our knowledge for the project we were about to start. Later we were told to further study about some of the communication protocols we might use in the project which are SMB, WMI, WinRM, RPC and SSH and the tools PsTools and telnet. Once we were done with studying communication protocols and a few tools, our advisor sent us a few courses. The first course was "Networking Fundamentals" which didn't help us much since we already knew everything to teach in that course. The second course was "Java Socket Programming", that course also was familiar to us, and we didn't learn anything new from there too. The third and final course we took was "Network Ethical Hacking", in that course we learned how

to hack in Kali Linux, we learned how to access a remote computer using a "backdoor" how to create executable files which grants us complete control over remote computers, how to access, take down and block a local networks, and how to use tools like Veil [29] and Meterpreter which are Metasploit attack payloads. With the help of the Network Ethical Hacking course, we were able to better understand our project.

After we learned everything that was needed for the project, we started investigating how to use communication protocols and PsTools, our main objective with each of them was to find a way to transfer files between computers, execute commands on remote computers and get directory details from remote computers.

The first protocol to succeed was WinRM with PowerShell, we understood how to connect to remote computers and achieve all the main objectives. After understand how to use WinRM we started looking for an existing API, but unfortunately we didn't find anything to our liking, in the end we worked with two NuGet packages the first is System.Management.Automation which helped us run a PowerShell process which we can write and add PowerShell commands to execute and Microsoft.PowerShell.SDK which we found out was needed to support the first package. The code can be seen in Appendix A - WinRMCommands Code.

The next protocol we worked with was SMB, for this protocol we learned how to run commands through the Command Prompt. After understanding how to run commands we started looking for APIs to use, but in the end, we decided to write our own commands. To do this we started another process within our program to start CMD.exe and then wrote the commands as we wrote them in the command prompt.

After achieving two protocols we decided and work with PsTools. Understanding how to use Pstools was quick, and we achieved our main objective easily. Although, the success, while working on how to implement it in our code, our advisor from the industry told us to stop working with this tool.

SSH is the final protocol which we currently added to the program. For this protocol we had a bit difficulty at start by understanding how to get the server side to accept client requests. After understanding how to accept client requests we started looking for APIs and luckily, we found an API which stood up to all our desires that is called SSH.NET [22]. Implementing the code for this protocol with the help of SCP was the easiest.

The two protocols that weren't added to our program are RPC and WMI. With WMI we had many failures with looking for existing APIs, how to use them and how to use the protocol itself, although we are still determined in adding it in the future. The RPC protocol we decided to leave out because we found enough protocols that can achieve what this protocol does.

After finishing designing our protocol commands code, we started working on how to achieve IPs. The most convenient way to gain a list of IPs is from an existing list of text file or with the scan for local hosts command, both can be seen in Figure 5. To scan for local hosts, we started a CMD.exe process within our program with the command "arp -a" which returns a list of IPs that we parsed into our own IP list. Once an IP is being added to the IP list our program gives us information about that IP, like ports connectivity and operating system (OS). To get the ports connectivity we used the TcpClient Class and tried to connect to the desired ports in our project and return an answer according to the result, to get the OS we ran through a CMD.exe process the "systeminfo /s IP" command.

After achieving all the goals above we decided to start designing our GUI. Our GUI took a lot of inspiration from the Zenmap GUI which can be seen in Figure 4. After deciding what we want and need from our own GUI we decided how to create it, that's when WPF came in mind and from there we had to learn how to write in WPF and what is the software architectural pattern Model-View-ViewModel. Once we fully understood how to work with

WPF we designed a skeleton of our application and later attached all the past works, which resembles like building a puzzle.

Finally, we completed all our goals for the first part of the project, which is the computer communication part and by that we could start the second part, which is scanning, parsing, and extracting artifacts. The second part we will achieve by working according to Windows Forensic Analysis Program Execution table which can be seen in Figure 1 and the help of Eric Zimmerman's tools [25].

During the whole time of working on our project we used GitHub to upload our project and work on different branches, Asana to set tasks to each other and communicated through WhatsApp, Zoom and Google Meet.

11 Planning

11.1 Milestones table

Table 1: Milestone Table

Milestone Number	Describe Milestone	End Date	Total Man-hours	Measurable Product
1	Preparatory report	24/10/2021	25	Preparatory report
2	The system GUI is ready. Users can enter IPs and get details about them. Connect to remote computers to transfer files, execute commands, and get directories.	10/1/2022	270	1st part of the project is done. Communication part
3	Progress report	16/1/2022	50	Progress report
4	The system runs through each computer to scan and parse artifacts and extract executable files to the main computer in a similar fashion to Kape	3/7/2022	300	2nd part of the project is done. Scan and parse artifacts and extract executable files.

5	Project day + Practical demonstration	12/7/2022	25	Poster + Presentation + POC
6	Improve any element in the project, polish the project's user interface and fix minor bugs	1/9/2022	150	Project is complete
7	Protections	15/9/2022	80	Project book + Working project

11.2 Task table

Table 2: Task Table

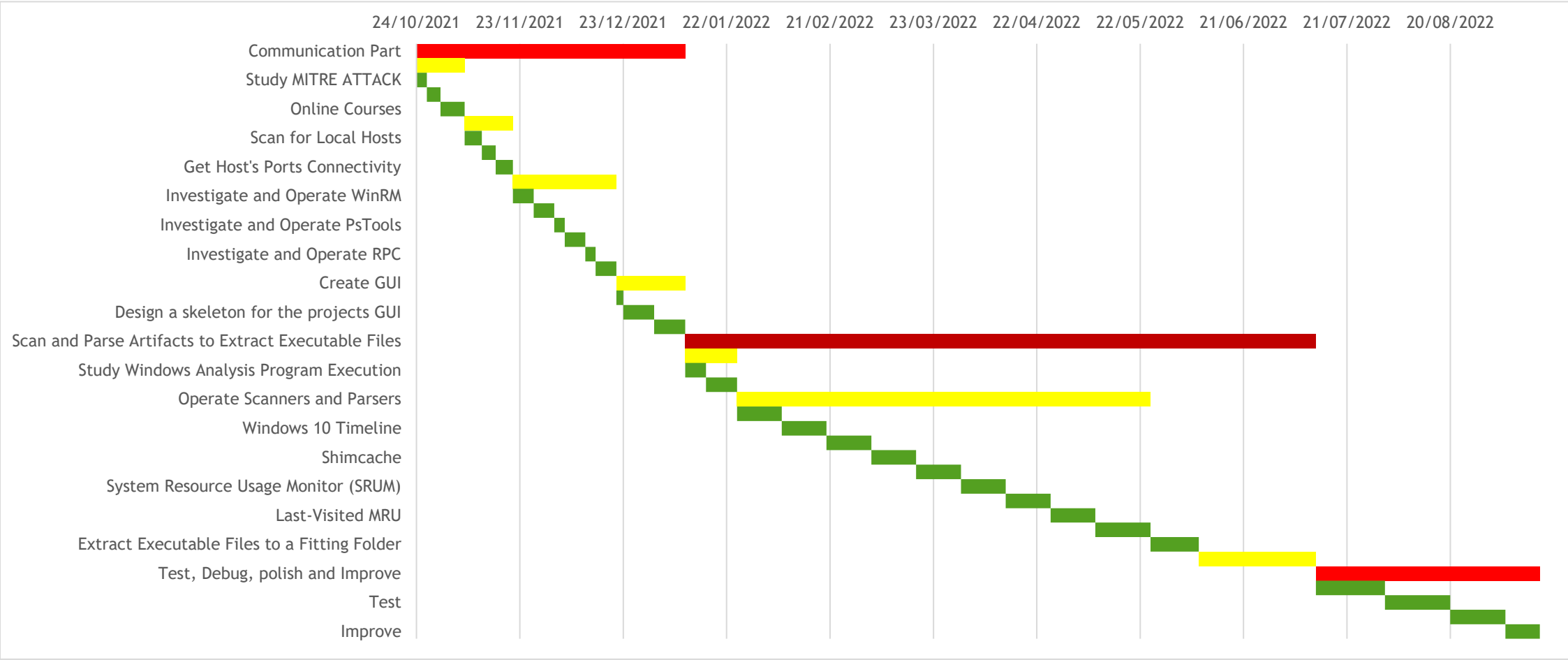
Landmark Number (Under Milestone)	Landmark Name (Under Milestone)	Task Number	Task Name	Start Date	Expected Due Date	Expected Hours	Intermediate Product	Due Date	Total Hours
1.1	Studying the Project	1	Study MITRE ATTACK	24/10/2021	27/10/2021	5	We understood better the project's goal.	27/10/2021	5
		2	Study Communication Protocols	27/10/2021	31/10/2021	10	Further enriched our knowledge in SMB, WMI, WinRM, RPC and SSH.	31/10/2021	10
		3	Online Courses	31/10/2021	07/11/2021	10	Learned how to use Kali Linux and a few Metasploit payload tools.	07/11/2021	10
1.2	Discover Hosts	1	Scan for Local Hosts	07/11/2021	12/11/2021	20	Wrote a code which can scan for local hosts with the help of "arp -a" in cmd.exe.	12/11/2021	20
		2	Get Hosts System Info	12/11/2021	16/11/2021	10	Wrote a code which can get info of a host with the help of "systeminfo /s IP" in cmd.exe.	16/11/2021	10
		3	Get Host's Ports Connectivity	16/11/2021	21/11/2021	15	With the use of TcpClient a connection is made to a desired port to check its connectivity.	21/11/2021	15
1.3	Investigate and Operate Protocols	1	Investigate and Operate WinRM	21/11/2021	27/11/2021	20	Wrote a code for WinRM with the help of System.Management.Automation and Microsoft.PowerShell.SDK for the projects desired commands.	27/11/2021	20
		2	Investigate and Operate SMB	27/11/2021	03/12/2021	20	Wrote a code for SMB for the projects desired commands.	03/12/2021	20

		3	Investigate and Operate PsTools	03/12/2021	06/12/2021	15	Abandoned according to industry advisors	06/12/2021	15
		4	Investigate and Operate SSH	06/12/2021	12/12/2021	20	Wrote a code with the help of the library SSH.NET for the projects desired commands.	12/12/2021	20
		5	Investigate and Operate RPC	12/12/2021	15/12/2021	5	Abandoned since no need in it.	15/12/2021	5
		6	Investigate and Operate WMI	15/12/2021	21/12/2021	20	Stopped, will continue in the second part of the project.	21/12/2021	20
1.4	Create GUI	1	Design according to Zenmap	21/12/2021	23/12/2021	5	Saw and used Zenmap to design our own GUI.	23/12/2021	5
		2	Design a skeleton for the projects GUI	23/12/2021	01/01/2022	65	Created a desktop application with WPF according MVVM pattern.	01/01/2022	65
		3	Attach Everything Together to the GUI	01/01/2022	10/01/2022	30	Attached all the commands written in the past to give functionality to the project.	10/01/2022	30
2.1	Study Forensics Artifacts and Eric	1	Study Windows Analysis Program Execution	10/01/2022	16/01/2022	10	Understand how to find artifacts and extract them with a parser.	16/01/2022	10

	Zimmerman's Tools	2	Study Eric Zimmerman's Tools	16/01/2022	25/01/2022	10	Understand how to use Eric Zimmerman's tools to parse artifacts.		
2.2	Operate Scanners and Parsers	1	UserAssist	25/01/2022	07/02/2022	16	Create a scanner and parser for UsserAssist.		
		2	Windows 10 Timeline	07/02/2022	20/02/2022	16	Create a scanner and parser for Windows 10 Timeline.		
		3	BAM/DAM	20/02/2022	05/03/2022	16	Create a scanner and parser for BAM/DAM.		
		4	Shimcache	05/03/2022	18/03/2022	16	Create a scanner and parser for Shimcache.		
		5	Amcache.hve	18/03/2022	31/03/2022	16	Create a scanner and parser for Amcache.hve.		
		6	System Resource Usage Monitor (SRUM)	31/03/2022	13/04/2022	16	Create a scanner and parser for System Resource Usage Monitor (SRUM).		
		7	Jump Lists	13/04/2022	26/04/2022	16	Create a scanner and parser for Jump Lists.		
		8	Last-Visited MRU	26/04/2022	09/05/2022	16	Create a scanner and parser for Last-Visited MRU.		
		9	Prefetch	09/05/2022	25/05/2022	16	Create a scanner and parser for Prefetch.		
2.3	Extract Executable Files to a	1	Extract Executable Files to a Fitting Folder	25/05/2022	08/06/2022	30	Execute a file from the remote computer to the main computer and place it in a folder with a fitting path Marvel/RemoteIP/Program Execution		

	Fitting Folder						Type, for example: Marvel/10.0.0.12/Prefetch.		
2.4	Attach Everything Together to the GUI	1	Attach Everything Together to the GUI	08/06/2022	12/07/2022	106	Add new functionality to the GUI.		
3.1	Test, Debug, polish and improve	1	Debug	12/07/2022	01/08/2022	50	Debug the project to find out any bugs.		
		2	Test	01/08/2022	20/08/2022	50	Test any edge cases.		
		3	Polish	20/08/2022	05/09/2022	30	Polish the program to run more efficiently.		
		4	Improve	05/09/2022	15/09/2022	20	Add new functionalities in case there is spare time.	15/09/2022	

11.3 Gantt


24/10/2021
23/11/2021
23/12/2021
22/01/2022
21/02/2022
23/03/2022
22/04/2022
22/05/2022
21/06/2022
21/07/2022
20/08/2022
Communication Part
Study MITRE ATTACK
Online Courses
Scan for Local Hosts
Get Host's Ports Connectivity
Investigate and Operate WinRM
Investigate and Operate PsTools
Investigate and Operate RPC
Create GUI
Design a skeleton for the projects GUI
Scan and Parse Artifacts to Extract Executable Files
Study Windows Analysis Program Execution
Operate Scanners and Parsers
Windows 10 Timeline
Shimcache
System Resource Usage Monitor (SRUM)
Last-Visited MRU
Extract Executable Files to a Fitting Folder
Test, Debug, polish and Improve
Test
Improve

12 Reading Sources

[1] R. Sobers, "varonis.com/blog/data-breach-statistics/," Varonis, 16 4 2021 .[מקוון] . Available: https://www.varonis.com/blog/data-breach-statistics/#:~:text=The%20average%20time%20to%20identify,was%2080%20days%20(IBM).[21 10 2021 -התבצעה גישה ב] ..

[2] J. Myllyla ו A. Costin, "Reducing the Time to Detect Cyber Attacks ",*Combining Attack Simulation With Detection Logic* .2021 ,

[3] B. Singh ו U. Singh, "Program execution analysis in Windows ",*A study of data sources, their format and comparison of forensic capability* ,74 כרך ,pp. 94-114, 12 1 2018 .

[4] A. Green, "varonis.com/blog/wmi-windows-management-instrumentation/," varonis, 29 1 2021 .[מקוון] .Available: https://www.varonis.com/blog/wmi-windows-management-instrumentation.[2021 10 2 -התבצעה גישה ב] ./

[5] R. Sheldon ו J. Scarpati, "techtarget.com/searchnetworking/definition/Server-Message-Block-Protocol," TechTarget, 8 2021 .[מקוון] .Available: https://www.techtarget.com/searchnetworking/definition/Server-Message-Block-Protocol.[2021 10 2 -התבצעה גישה ב] .

[6] "openssh.com/," 26 September 2021 .[מקוון] .Available: https://www.openssh.com 15 -התבצעה גישה ב] ./January 2022.[

[7] "microsoft.com/en-us/windows/win32/winrm/portal," Microsoft .[מקוון] ,Available: https://docs.microsoft.com/en-us/windows/win32/winrm/portal -התבצעה גישה ב] . .[2021 10 2

[8] "kroll.com/en/services/cyber-risk/incident-response-litigation-support/kroll-artifact-parser-extractor-kape .[מקוון] ",Available: https://www.kroll.com/en/services/cyber-risk/incident-response-litigation-support/kroll-artifact-parser-extractor-kape . .[2021 10 2 -התבצעה גישה ב]

[9] "wikipedia.org/wiki/Communication_protocol#Bibliography," Wikipedia, 10 1 2022 . .[מקוון]Available: https://en.wikipedia.org/wiki/Communication_protocol#Bibliography התבצעה] . .[2022 1 11 -גישה ב

10] ["wikipedia.org/wiki/Communication_protocol," Wikipedia, 10 1 2022 .[מקוון] . Available: https://en.wikipedia.org/wiki/Communication_protocol התבצעה גישה] . .[2022 1 11 -ב

11] [N. Bencherchali, "medium.com/windows-forensics-analysis-windows-artifacts-part-i-c7ad81ada16c," nasbench.medium, 15 September 2019 .[מקוון] .Available:

https://nasbench.medium.com/windows-forensics-analysis-windows-artifacts-part-i-c7ad81ada16c 2022 -ב גישה התבצעה] .January 2022.[

12] ["tetradefense.com/digital-forensics-services/what-are-forensic-artifacts-my-favorite-artifacts-part-0/," Tetradefense .[מקוון] ,Available: https://www.tetradefense.com/digital-forensics-services/what-are-forensic-artifacts-my-favorite-artifacts-part-0.[2022 1 11 -ב גישה התבצעה] ./

13] [OnPage Corporation, 31 August 2020 .[מקוון] .Available: https://www.onpage.com/incident-management-process-5-steps-to-effective-resolution 11 -ב גישה התבצעה] ./January 2022.[

14] ["attack.mitre.org/tactics/TA0008/," The MITRE Corporation, 17 October .2018 .[מקוון]Available: https://attack.mitre.org/tactics/TA0008 11 -ב גישה התבצעה] ./ January 2022.[

15] ["attack.mitre.org/tactics/TA0002/," The MITRE Corporation, 17 October .2018 .[מקוון]Available: https://attack.mitre.org/tactics/TA0002 11 -ב גישה התבצעה] ./ January 2022.[

16] ["microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2022," Microsoft, 12 February 2021 .[מקוון] .Available: https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2022 11 -ב גישה התבצעה] .January 2021.[

17] ["microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm," Microsoft, 7 August 2021 .[מקוון] .Available: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm 11 -ב גישה התבצעה] .January 2021.[

18] ["microsoft.com/en-us/windows/win32/wer/about-wer," Microsoft, 23 August 2019 . .[מקוון]Available: https://docs.microsoft.com/en-us/windows/win32/wer/about-wer 12 -ב גישה התבצעה] .January 2022.[

19] ["attack.mitre.org/tactics/TA0003/," The MITRE Corporation, 17 October .2018 .[מקוון]Available: https://attack.mitre.org/tactics/TA0003 12 -ב גישה התבצעה] ./ January 2022.[

20] [S. Florian ו P. Timea , "Under false flag ",*using technical artifacts for cyber attack attribution* .2020 3 20 ,

21] [L. F. Gordon , "nmap.org/," insecure, 1997 .[מקוון] .Available: https://nmap.org ./ .[2021 10 20 -ב גישה התבצעה]

22] [sshnet, "github.com/sshnet/SSH.NET," 14 December 2021 .[מקוון] .Available: https://github.com/sshnet/SSH.NET 8 -ב גישה התבצעה] .January 2022.[

23] [k8gege, "github.com/k8gege/Ladon," 30 October 2021 .[מקוון] .Available: https://github.com/k8gege/Ladon 22 -ב גישה התבצעה] .December 2021.[

24] [L. Rob, "sans.org/posters/windows-forensic-analysis/," Sans, 25 March 2021 . .[מקוון]Available: https://www.sans.org/posters/windows-forensic-analysis ./ 5 -ב גישה התבצעה]January 2022.[

25] [E. Zimmerman, "ericzimmerman.github.io/#!index.md .[מקוון] ",Available: https://ericzimmerman.github.io/#!index.md 28 -התבצעה גישה ב] .October 2021.[

26] [G. F. Lyon, "nmap.org/zenmap/," Nmap .[מקוון] ,Available: https://nmap.org/zenmap.[2021 10 18 -התבצעה גישה ב] ./

27] ["wikipedia.org/wiki/Nmap," Wikipedia .[מקוון] ,Available: https://en.wikipedia.org/wiki/Nmap 13 -התבצעה גישה ב] .January 2022.[

28] ["attack.mitre.org/," The MITRE Corporation, 2015 .[מקוון] .Available: https://attack.mitre.org 2 -התבצעה גישה ב] ./October 2021.[

29] ["github.com/Veil-Framework/Veil," Github .[מקוון] ,Available: https://github.com/Veil-Framework/Veil 1 -התבצעה גישה ב] .November 2021.[

30] [SSI, "insider.ssi-net.com/insights/what-is-an-artifact-in-cyber-security," Insider.SSI, 2 3 2021 .[מקוון] .Available: https://insider.ssi-net.com/insights/what-is-an-artifact-in-cyber-security.[2021 10 2 -התבצעה גישה ב] .

31] [E. Zimmerman, "ericzimmerman.github.io/#!index.md#Requirements_and_troubleshooting," GitHub .[מקוון] ,Available: https://ericzimmerman.github.io/#!index.md#Requirements_and_troubleshooting . .[2021 10 2 -התבצעה גישה ב]

32] [D. Athanasios, I. Nenad, K. Boonserm ו M. Ioannis, "D4I ",*Digital forensics framework for reviewing and investigating cyber attacks* .2019 12 26 ,5 כרך ,

33] ["attack.mitre.org/tactics/TA0008/," attack.mitre, 17 October .[מקוון] .2018 Available: https://attack.mitre.org/tactics/TA0008 11 -התבצעה גישה ב] ./January 2022.[

Appendices

List of appendices

Appendix A - WinRMCommands Code

```
using Marvel.Enum;
using Marvel.Model;
using System;
using System.Collections.ObjectModel;
using System.Management.Automation;

namespace Marvel.ProtocolCommands
{
    class WinRMCommands
    {
        public string Commands(Host host, string fromDirectory, string toD
irectory, CommandsEnum selectedCommand)
        {
            PowerShell ps = PowerShell.Create();

            // Set credentials
            ps.AddScript("$password = ConvertTo-SecureString '" +
                host.Password + "' -AsPlainText -Force");
            ps.AddScript("$cred = New-
Object System.Management.Automation.PSCredential" +
                " (\"" + host.Username + "\", $password)");

            // Start Session
            ps.AddScript("$session = New-PSSession -
ComputerName " + host.IP + " -Credential $cred");

            switch (selectedCommand)
            {
                case CommandsEnum.GetDirectoryFilesList:
                    ps.AddScript("Invoke-Command -Session $session -
ScriptBlock { Get-ChildItem " + fromDirectory + " }");
                    break;
                case CommandsEnum.RunItem:
                    ps.AddScript("Invoke-Command -Session $session -
ScriptBlock {" + fromDirectory + " /silent}");
                    break;
                case CommandsEnum.ReceiveItem:
                    ps.AddScript("Copy-Item -
FromSession $session " + fromDirectory + " -Destination " + toDirectory);
                    break;
                case CommandsEnum.SendItem:
                    ps.AddScript("Copy-Item -
ToSession $session " + fromDirectory + " -Destination " + toDirectory);
                    break;
                default:
                    return null;
            }

            Collection<PSObject> result;
```

```
            try
            {
                result = ps.Invoke();
            }
            catch (Exception e)
            {
                return e.Message;
            }

            string resultString = "";

            foreach (PSObject item in result)
            {
                resultString += item + "\n";
            }

            return resultString;
        }
    }
```