University of Victoria
Engineering & Computer Science Co-op
Work Term Report
2020 - Spring

Hacking Your Own Network for Hardware Information

Andrew Sheret Limited
IT Department
Victoria, British Columbia, Canada

Stefan Esquivel
V00904819
W-1
Software Engineering
sesquivel@uvic.ca
04/28/20

In partial fulfillment of the academic requirements of this co-op term

Supervisor's Approval: To be completed by Co-op Employer

Signature: [illegible] Position: it Director Date: 04/27/20

Name (print): DC Manning E-Mail: david.manning@sheret.com

For (Company Name) Andrew Sheret Limited

Stefan Esquivel
3612 Tillicum Rd
Victoria, British
Columbia V8Z 4H4

April 27, 2020

Dr. Imen Bourguiba, Coop Coordinator
Faculty of Engineering University of Victoria
Victoria, B.C. V8W 2Y2

Dear Dr. Imen,

Please read the following Work Term Report titled, "Hacking Your Network for Hardware Information."

This report outlines my research, discovery and deployment of a new way to identify untracked hardware in the Andrew Sheret Limited (ASL) company. Since ASL is transitioning to an enterprise-level company, they are just now exploring ways to track the number of company devices they have sent out. Hardware tracking is especially important for future department budgeting. Since I have had previous experience with network security, I wanted to utilize those skills to help identify a few of these devices remotely. Discovering printer devices would be the highest priority since Ricoh, a printer brand our company uses, incorrectly charged the company service fees for printers that are no longer in use.

While working on this project, I overcame many significant challenges and learned a lot about enterprise-level server solutions. This project has introduced me to a broader range of network competencies that I can use in the future. It was indeed an honour to help grow the ASL company and develop my knowledge with the IT team.

Please read this insightful report underlining my thought process, project management, and accomplishments in completing this project.

Sincerely,

[illegible]

Stefan Esquivel

Table of Contents

Equations, Figures, and Tables

Summery

Andrew Sheret Limited (ASL) is a local company in Victoria BC, responsible for distributing plumbing and heating equipment across Canada [1]. Mr. Andrew Sheret first opened the business in 1892, and now it has expanded across Canada and into the online world, with its "Plumbing Online" store. Recently, they adopted Frontier Plumbing and Heating Supply in Alberta, adding seven more distribution branches to the business. ASL is transitioning from a small to enterprise-level company. This transition involves adding new members and employees, as well as building technical capabilities. ASL's IT development team has accomplished significant upgrades to the company network to allow more users and services for their employees and customers. However, with the increased number of users comes more internal company hardware, including printers, NUCs**,** and terminals. While small companies need not track internal hardware, it has become financially beneficial for ASL to do so at this size. The IT development team is now focusing on better tracking, servicing and distributing these new devices. This report explores how hacking and network security tools help support IT in their effort.

Glossary

ARP	Stands for Address Resolution Protocol, which is used for identifying neighboring devices on a network
ASL	Short for Andrew Sheret limited, a Victoria located plumbing and heating equipment distribution company
BASH	Short for Bourne Again Shell and is a modern UNIX shell
Confluences	Atlassian Software that allows teams to store FAQs, How-Tos, or an instruction set of solved problems
CSV	Short for Comma Separated Values and is a file type in which a comma separates each value or entry
DHCP	Stands for Dynamic Host Configuration Protocol, which a server dynamically assigns **IP**s
grep	Regular expression tool for **BASH**
GUI	Short for Graphical User Interface and uses virtual pictures to help the user interact with an application
Homebrew	A package manager for Mac **OS** X or Linux
IP	Stands for Internet Protocol and is used as a unique Identifier on a network
Jira	Atlassian Software that allows teams to manage and schedule tasks. Offers a robust block **UI** for easy task management
Mac	Short for Media Access Control, and stored in the hardware of your device as an identification on a network
OS	Short for Operating System
ping	Command to test the reachability of a host
RFID	Stands for Radio Frequency Identification, and is a device used for tracking and theft prevention

SNMP	Stands for Simple Network Management Protocol, used for collection and managing device information
stdout	Stands for standard output
TCP	Stands for Transmission Control Protocol, which controls which network interactions can be made between two hosts
TL; DR	Short for Too Long; Didn’t Read and is identical to a concise summary
TUI	Short for Text User Interface and allows user to interact with software through terminal commands
UDP	Stands for User Datagram Protocol
UI	Short for User Interface

1. Introduction

The purpose of this report is to outline a project developed using hacking tools to help locate hardware information within an enterprise-level company, Andrew Sheret Limited (**ASL**) [1]. Due to **ASL**'s size, it is now essential to track and budget office hardware around the company from a financial aspect. This project mainly focuses on printers, but further research and development can be used for other office devices as well. This paper explores the engineering process to build an early-stage local network hardware tracker using hacking tools like nmap, snmpwalk, and arp, from its initial planning to deployment.

2. Software Planning

For this project, it was essential to outline specifications, set clear deadlines, and identify constraints.

2.1 Background

The IT Department at **ASL** is in charge of servicing and distributing all the office devices our employees use daily. Over the term, the department sent out several untracked items and even had a few small devices go missing or arrive late. Due to this occurring issue, this was a topic of discussion at one of the IT development meetings. After further investigation, a few solutions where suggested. For future tracking of office technology, the IT department considered using **RFID** tags marking everything moving in and out of the Head Office and storing it on an inventory database [2]. This idea is a valid strategy and would work quite effectively as it allows quick access to our inventory through a computer system. However, lots of hardware is currently out in use and is still in service for another three to ten years. These become obsolete over time, and the IT department replaces them. Nevertheless, these devices became a concern for **ASL** when a billing invoice from Ricoh had several overcharges for printers out of service.

Ricoh is a company that exports, manufactures, and services household, office, and industrial printers, including other imaging devices like cameras, conferencing systems, and interactive whiteboards [3]. At **ASL**, Ricoh services around 95% of the printers in the company. The incorrect billing of out of service printers made it evident that the company needed to track all of the active Ricoh printers to correct this billing error.

With further investigation of the Ricoh Printer Admin Panel, there were many oddities, including printers that Ricoh billed for as if the devices were multi-function. In reality, they were print only models. To resolve the error, Ricoh required the IT department to track down all printers in the company and report there **IP** address, device serial number, make, and model. The best way to determine this information would be to use an automated process in the form of a script.

2.2 Project Specifications

The bulk of this project splits itself into two different sections. The first section includes probing information, including branch number, **IP**, **Mac**, device label, manufacture name, device model and serial number. The next section is in charge of organizing, storage and **UI,** and future implementations, possibly a **GUI**. Below outlines the detailed specs and requirements of each section.

2.2.1 General

1. Must run relatively quick (under 5 minutes)
2. Must be significantly documented and uploaded to **Confluences** with instructions, examples, and **TL; DR**
3. Must be modular, for readability [4], and to support additional changes and upgrades
4. Must run natively, on a Mac **OS** X device connected to the **ASL** Head Office network
5. Cannot allow for any interruptions in the workplace or cause downtimes
6. Must be testable

2.2.2 Probing Information

1. Must accept an input **IP** or **IP** range to search for printers within the **ASL** network
2. Must detect printers accurately
3. Must allow information accessibility in a variable or file
4. Must allow effective iteration of printer devices
5. Must run relatively quick, taking into account the latency caused by the distance of some of the branches

2.2.3 Organization and Storage

1. Data storage must allow add, search, remove, and sort functions, based on any printer attributes
2. Must allow output to a **CSV** file format
3. Must include user options to probe for any or all of the following information
    a. Branch
    b. **IP**
    c. Label
    d. **Mac**
    e. Model
    f. Name
    g. Serial
4. Early-stage must have **TUI** with appropriate help and instructions to allow users to navigate and operate the script

2.3 Proposition

Python 3 and **BASH** languages can satisfy all of the general requirements of this project. For retrieving information remotely, **BASH** contains many tools that are non-interruptible, compatible with Mac **OS** X, and have extensive documentation [5]. During the scripting process, individual lines and or units of **BASH** code can be easily tested and run from the command line. This feature allows mutual exclusion and modularity over each **BASH** script. **BASH** scripts also can be run with Python 3 code using the subprocess module [6]. Python 3 is perfect for the Organization spec as it is an object-orientated language [7]. This feature allows us to build and modify a printer object with attributes including branch number, **IP**, **Mac**, device label, manufacture name, device model and serial number, with ease. The language contains many features, including tools for building **TUI** and allowing for future implementations [8], such as a **GUI**. With the correct permissions, the entire script can also run from the command line.

2.3.1 File Organization

A modular file organization style allows for easy modification and readability [4]. It is thus vital that this project uses the same design. This assignment settles on using three **BASH** subscripts, one output **CSV** file, and four Python 3 files for the organization, as well as one text file, to hold detected **IP**'s. See below the descriptions of each file:

Table 1. File Descriptions

File Type	Name	Description
Bash	arp-pr.sh	This file contains an executable script that uses **ping** then arp to retrieve printer information
	nmap-pr.sh	Detects active printer **IP**s on the network by port scanning a range of **IP**s
	snmp-pr.sh	Retrieves the serial number, name, make and model of the printer
CSV	printers.csv	Module that allows read and write commands with a .csv file
Python	Printer.py	Python Module Contains the printer class that packages all the desired printer attributes
	Printer_List.py	Python Module that holds printer objects and outputs to either **stdout** or .csv file
	Printer_to_csv.py	The core of the script contains all the methods to create objects and call **BASH** scripts to retrieve information
	Test_Driver.py	File to run the code as well as do testing contains the argument parser for quick modification
Text	active-ips.txt	Contains a list of active branch printer **IP**s created by nmap-pr.bash

2.3.2 Order of Operations

An essential part of project planning is to define how each module and subscript work together to produce a **CSV** file with the wanted printer information. Here are the underlined steps of the script, from execution to termination:

1. Upon execution, Test_Driver.py calls Printer_to_csv.py to prep the script
2. Printer_to_csv.py then calls nmap-pr.sh to search for active printers
3. nmap-pr.sh outputs the active-ips.txt that contains a list of detected printer IPs
4. active-ips.txt inputs into Printer_to_csv.py for printer iteration
5. Printer_to_csv.py creates a printer object for each detected IP

6. Printer_to_csv.py adds each printer object into the printer list
7. For each printer object, arp-pr.sh is called, then updates the label and **mac**
8. After which, snmp-pr.sh is called for each printer object, updating the model, name, and serial
9. Lastly, Printer_List.py outputs the **CSV** file and the program is terminated

Below is a visual representation of the steps above:

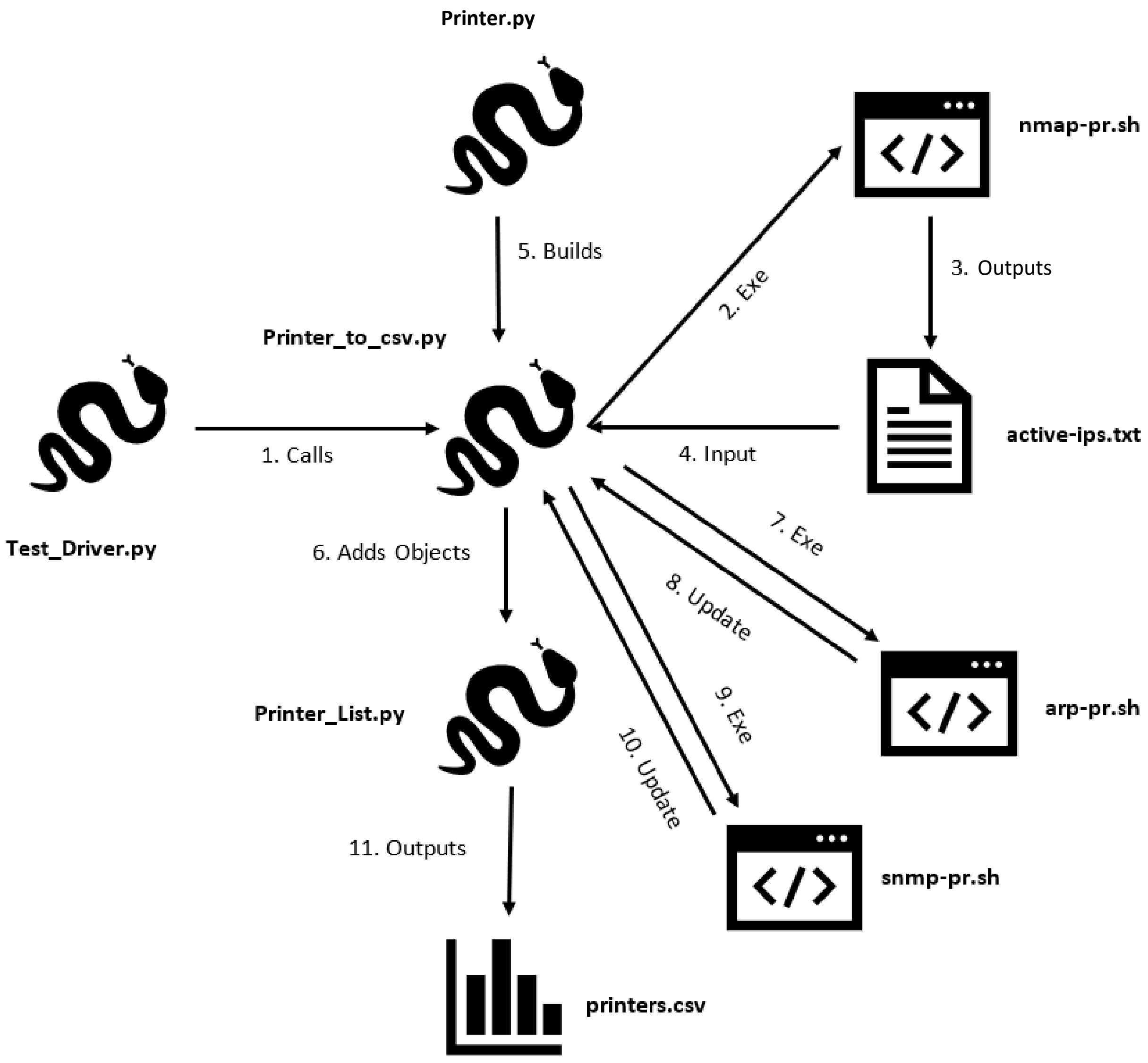


Figure 1. File Network Diagram

2.4 Necessary Commands and Modules

After deciding the appropriate languages and establishing the file network, the corresponding tools are selected.

2.4.1 Bash Commands

Although **BASH** has many useful commands prebuilt in a new Mac Pro, the following tools need to install using **Homebrew** [9]. After the installation, these commands can then run from the command line. Below is the table explaining each **BASH** tool and their significance in the project:

Table 2. BASH Commands Used in This Project

Tools	**Description**	**Project Usage**	**Project Location**
arp	Used to find the device address (**mac**) of a neighbouring device and domain associated [10]	• Retrieve **Mac** Address • Retrieve Printer Label	arp-pr.sh
nmap	A network port scanner used to check for open and active **IP**s or Hosts [11]	• Scan for active printers • Provide the **IP** addresses of active printers	nmap-pr.sh
snmpwalk	An application that searches through the **SNMP** port on a specific device and returns the information it finds [12]	• Retrieve Serial Number • Retrieve Model • Retrieve Name	snmp-pr.sh

Although not included in the above table, this project may use additional **BASH** tools.

2.4.2 Python Modules

Python 3 has many built-in modules that do not require an additional install in contrast to the BASH tools. The following table goes through all the essential modules that this project uses and highlights each one's significance:

Table 3. Python Modules in This Project

Modules	**Description**	**Project Usage**	**Project Location**
argparce	Module that allows a python executable to have organised	• Adds a **TUI** to modify outputs • -h argument gives detailed instructions	Test_Driver.py

	command line arguments [8]		
csv	Module that allows read and write commands with a .csv file [13]	• Writes a file type that can be opened up in excel or libre office • Builds the printer.csv	Printer_List.py
subprocess	Tool used to allow **BASH** scripts to run from the Python environment [6]	• Used to execute arp-pr.sh, nmap-pr.sh, and snmp-pr.sh • Receives printer information from the **BASH** scripts	Printer_to_csv.py

Although not included in the above table, this project may use additional Python 3 modules.

2.5 Time and Task Management

ASL uses Atlassian Software [14], which provides tools for team documentation and project management. **Jira**, an Atlassian tool, is used daily at Andrew Sheret to manage IT help tickets, and larger IT development Tasks. Each script and file have a different deadline and **Jira** stages to avoid interruptions.

3. Developing Bash Subscripts

In this section, the project dives into the **BASH** scripts underlining the hacking tool, Nmap. Also, the steps to retrieve the printer information from snmpwalk and arp, are outlined.

3.1 Introducing Nmap

Nmap is an open-source, free network scanner created by Gordon Lyon. This application consists of network managing, hacking, and penetration testing tools [11]. It works by sending internet packets to devices' network ports to determine if they are open or active. This tool has made a significant name for itself and was a critical factor in several black hat hacking crimes. Nmap appears in several different Hollywood films as well, including the Matrix [15]

The Nmap tool supports single or a range of **IP** port scans. For single **IP** port scans, the syntax has the following form, "$nmap <IP>" where IP is the address of the device. The command to scan a range of IPs has the following form "$namp <IP>/x," where x is the number of excluded bits of the IP, left-justified. This notation is called CIDR [16]. For example, the command "$namp 255.255.5.0/24" results in the following operations:

Equation 1. IP AND Gate Example

$$\begin{array}{r} 11111111.11111111.00000101.00000000 \\ AND\ \underline{11111111.11111111.11111111.00000000} \\ 11111111.11111111.00000101.00000000 \end{array}$$

Equation 2. IP XNOR Gate Example

$$\begin{array}{r r} & 11111111.11111111.00000101.00000000 \\ XNOR & \underline{11111111.11111111.11111111.00000000} \\ & 11111111.11111111.00000101.11111111 \end{array}$$

Notice there are 24 ones in the AND gate. The resulting **IP** is 255.255.5.0. The XNOR gate produces the upper bound of the range, 255.255.5.255. In this case, the Nmap tool scans all the **IP**’s from 255.255.5.0 to 255.255.5.255, totalling 255 different **IP**s. Performing a scan from Head Office at **ASL**, each **IP** takes around 1-5 seconds to complete a scan. Therefore, it is essential to make sure that scans are minimal.

3.2 ASL Printer organization

Printers at **ASL** are organized efficiently on the internal network and use the following IPv4 scheme: X.b.Y.n, where b is the branch number and n, is the printer number. X and Y are constant 8bit numbers; however, this report excludes this information for confidentiality purposes. For example, X.6.Y.3 is the third printer at branch six. This organization makes it exceptionally simple to find an active printer **IP**. However, not all printers take the next available **IP** as configured in the **DHCP**. Often, the next **IP** could be unavailable, resulting in gaps in the available **IP** range, some reaching as high as the X.b.Y.99 **IP**. Taking this into consideration, all needed in the "nmap-pr.sh" script is a simple iterator for all branches and a call to scan the ports in a broad enough IP range. Since there are no more than 20 printers per branch at ALS, by using the following command templet "$namp <IP>/x," we can set x to 25. Using the X.6.Y.3 example:

Equation 3. IP AND Gate Used

$$\begin{array}{r r} & X \quad .00000110. \quad Y \quad .00000000 \\ AND & \underline{11111111.11111111.11111111.10000000} \\ & X \quad .00000110. \quad Y \quad .00000000 \end{array}$$

Equation 4. IP XNOR Gate Used

$$\begin{array}{r r} & X \quad .00000110. \quad Y \quad .00000000 \\ XNOR & \underline{11111111.11111111.11111111.10000000} \\ & X \quad .00000110. \quad Y \quad .01111111 \end{array}$$

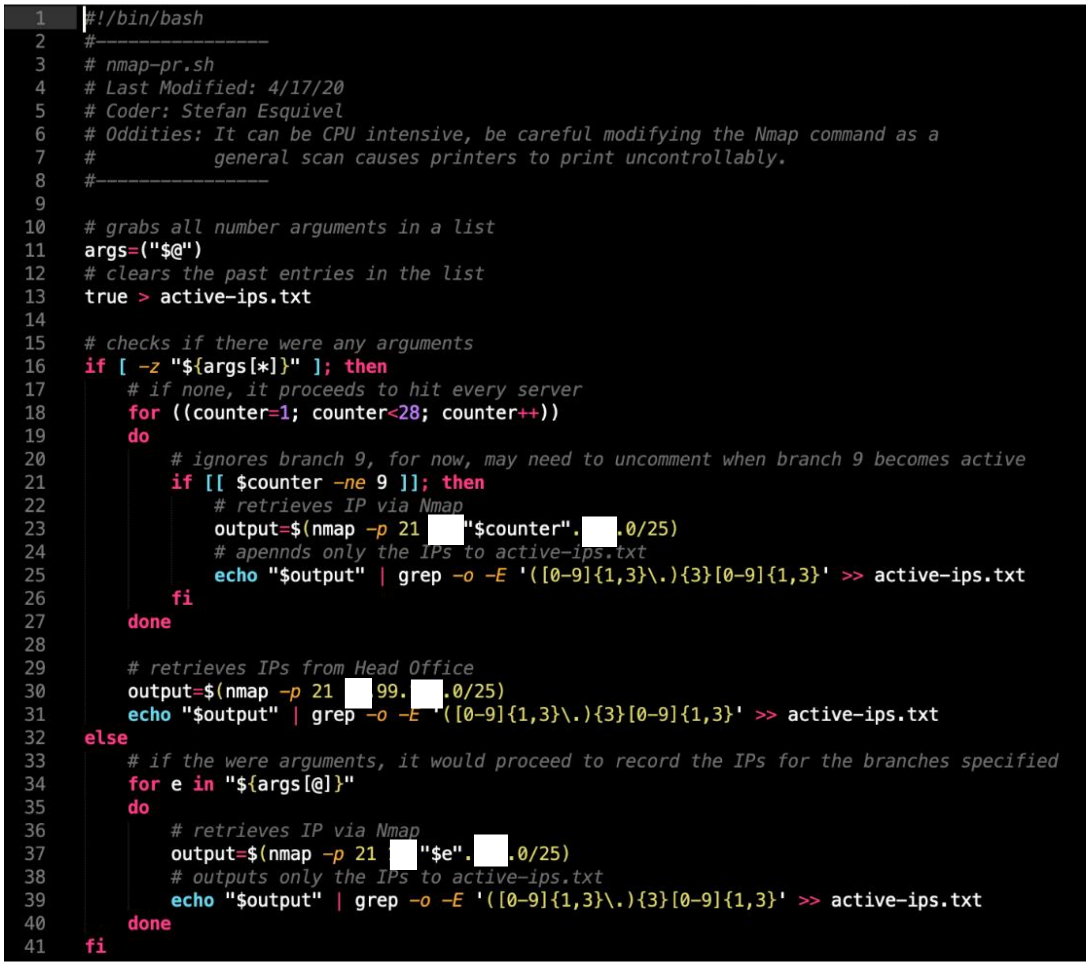
The tool will now scan the **IP** range from X.6.Y.0 to X.6.Y.127. This scan range will encompass all printers at branch 6 and even include possible outliers.

3.3 Banner Grabbing

Nmap is a standard tool used for Banner Grabbing [17]. A banner refers to a text message received from the host when connecting to a **TCP** port. Banners usually contain information about a specific device, including **OS** versions, device weaknesses, and vulnerabilities. For hackers, this is a valuable tool as they can learn about a device remotely. Using this technique, we can determine which printer is active. A Ricoh device's TCP port 21 contains the printer's name and model. For example, "|_banner: RICOH IM 510". Using the **grep BASH** tool, it simple to isolate the name and model of the printer. However, this shown to be inefficient as 10 of the 258 printers detected would time out on this port and return with no or incorrect information. And many of them are inconsistent with name and model format, making it difficult to automate the process. However, since port 21 is an open port, we can specify the port we want to scan on the printer by adding the -p flag to the Nmap command, resulting in the form "$namp -p 21 <IP>/x" [18]. Since a port is specified, the Nmap tool can save time by only scanning one port per **IP**.

3.4 Final Solution for Printer Detection

The final draft for nmap-pr.sh, scans all the branch **IP**'s and stores every printer detected into a text file named, active-ips.txt. A few features include scanning a specified branch or scanning all of them. See the completed subscript below:



```
#!/bin/bash
#----------------
# nmap-pr.sh
# Last Modified: 4/17/20
# Coder: Stefan Esquivel
# Oddities: It can be CPU intensive, be careful modifying the Nmap command as a
#           general scan causes printers to print uncontrollably.
#----------------

# grabs all number arguments in a list
args=("$@")
# clears the past entries in the list
true > active-ips.txt

# checks if there were any arguments
if [ -z "${args[*]}" ]; then
    # if none, it proceeds to hit every server
    for ((counter=1; counter<28; counter++))
    do
        # ignores branch 9, for now, may need to uncomment when branch 9 becomes active
        if [[ $counter -ne 9 ]]; then
            # retrieves IP via Nmap
            output=$(nmap -p 21 [redacted]"$counter".[redacted].0/25)
            # apennds only the IPs to active-ips.txt
            echo "$output" | grep -o -E '([0-9]{1,3}\.){3}[0-9]{1,3}' >> active-ips.txt
        fi
    done

    # retrieves IPs from Head Office
    output=$(nmap -p 21 [redacted]99.[redacted].0/25)
    echo "$output" | grep -o -E '([0-9]{1,3}\.){3}[0-9]{1,3}' >> active-ips.txt
else
    # if the were arguments, it would proceed to record the IPs for the branches specified
    for e in "${args[@]}"
    do
        # retrieves IP via Nmap
        output=$(nmap -p 21 [redacted]"$e".[redacted].0/25)
        # outputs only the IPs to active-ips.txt
        echo "$output" | grep -o -E '([0-9]{1,3}\.){3}[0-9]{1,3}' >> active-ips.txt
    done
fi
```

Figure 2. nmap-pr.sh (Censored)

Now that there is a solution to detect printers, the next step of this project is to grab the rest of the printer information. Tools that are available to us are snmpwalk and arp.

3.5 Introducing SNMP and ARP

SNMP stands for Simple Network Management Protocol and is typically used to monitor hardware stats and features remotely [12]. The **SNMP** agent on the printer's network interface device receives requests on **UDP** port 161 [12], [19]. The **SNMP** port contains information about the device and stores it in a tree-like structure for quick access. Information consists of uptime, pages printed, driver information, and more. Typically, this port is closed for security purposes. However, several different services use this port for maintenance metrics. Hackers exploit this vulnerability to penetrate local area networks at coffee shops and malls [20]. The command-line tool, snmpwalk, reads all the information from port 161 and dumps it out onto the **stdout**. Since port 161 is open for all the printers at the branch, this project uses this to grab the serial number, name, and model.

ARP stands for Address Resolution Protocol and works by storing recently-interacted **mac** addresses into a table-like format [21]. The arp is a command-line tool that returns this table and outputs it onto the **stdout**. Here the user has access to **mac** addresses with the associated device labels. Since **ARP** populates the table with recent network interactions, a **ping** command must be performed on the device before an arp. Hackers typically use this tool to change their hardware address to gain unauthorised internet access in a technique called Mac Spoofing [22]. Using this tool, retrieving information on a device like the label and **mac** is trivial.

3.6 Retrieving Additional Information

The printer's serial, name, and model have a unique location on the information tree, labelled with a key from the **SNMP** [12]. By specifying the key, the snmpwalk command returns the desired information. This key differs between the Mac **OS** X and Linux operating systems. Below shows the table of keys associated with the different printer attributes:

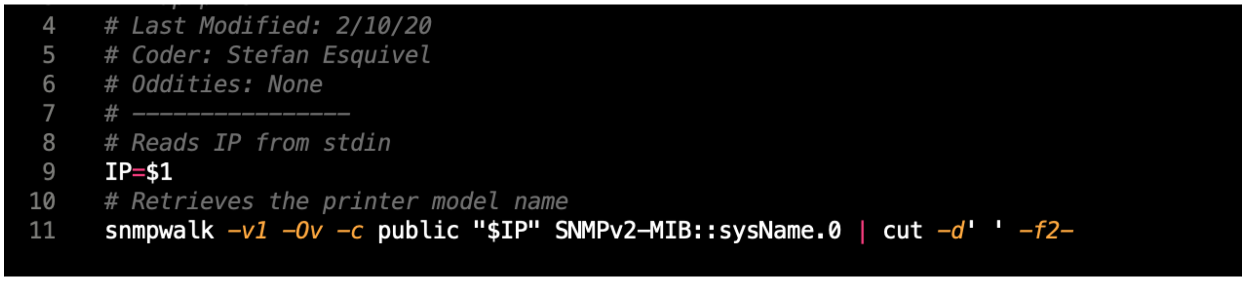
Table 4. Mac SNMP Keys

Attribute	**Key**
Model	SNMPv2-MIB::sysDescr.0
Name	SNMPv2-MIB::sysDescr.0
Serial	mib-2.43.5.1.1.17.1

Model and name use the same key and contain one space between them. Since the attributes follow a similar pattern, a simple **grep** command can isolate the required information [23]. This method also works for retrieving the **mac** address and label from the **ARP** table.

3.7 Information Retrieval Solution

In the initial planning, each type of command had its own and separated **BASH** script. Since the user can select which information they would like on the **CSV**, each information retrieval module is separate to save runtime. Each script takes an **IP** and outputs the information found. The image below shows the code design used across each of these files:



```
# Last Modified: 2/10/20
# Coder: Stefan Esquivel
# Oddities: None
# ----------------
# Reads IP from stdin
IP=$1
# Retrieves the printer model name
snmpwalk -v1 -Ov -c public "$IP" SNMPv2-MIB::sysName.0 | cut -d' ' -f2-
```

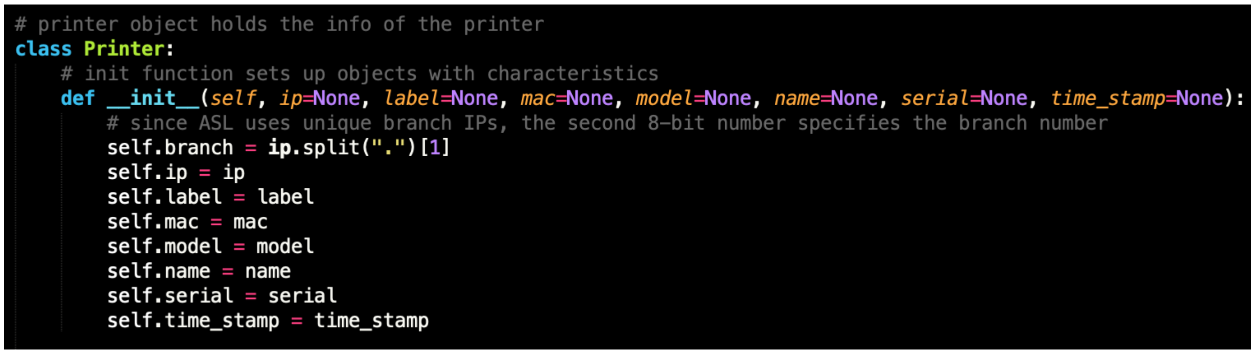
Figure 3. Example Coding Structure for Printer Name Retrieval

4. Developing Python Files

Python 3 syntax organizes, retrieves, and stores all of the printer information provided by the **BASH** scripts. This section uses four Python 3 modules to achieve this. These modules contain the printer object architecture, list to store the objects, methods to execute the **BASH** scripts, and helpful **UI**.

4.1 Printer Object

The printer class contains all of the attributes that output onto the **CSV**. Each of these attributes has its necessary accessors and mutators for future changes and easy access to the data. When the Printer_to_csv.py module makes a printer object, the constructor sets all the instance variables to none. In this script, new objects have the **IP** initialized. Since the **IP** contains the branch number, the constructer initially sets the branch number to a substring of the **IP** as seen below:



```
# printer object holds the info of the printer
class Printer:
    # init function sets up objects with characteristics
    def __init__(self, ip=None, label=None, mac=None, model=None, name=None, serial=None, time_stamp=None):
        # since ASL uses unique branch IPs, the second 8-bit number specifies the branch number
        self.branch = ip.split(".")[1]
        self.ip = ip
        self.label = label
        self.mac = mac
        self.model = model
        self.name = name
        self.serial = serial
        self.time_stamp = time_stamp
```

Figure 4. Printer Constructor

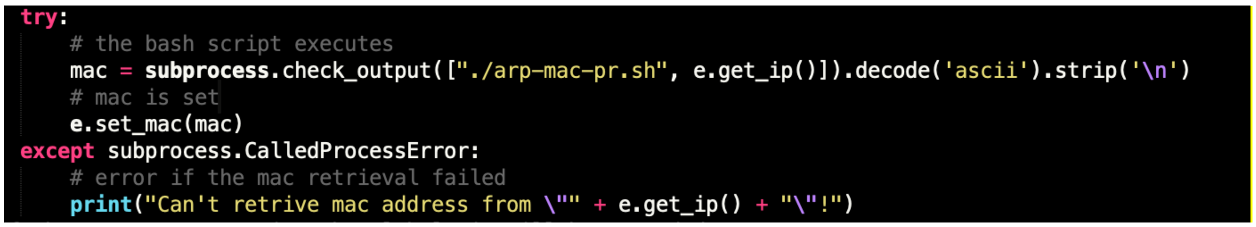
Each of these objects are then placed in a printer list located in the Printer_List.py module.

4.2 Printer List

The printer list is a class that contains a simple Python 3 array, that all the printer objects are stored. This list allows for the usual add, remove, search, and find operations. This object is made iterable by the __next__ () method in the printer list class [24]. The printer list also has functions to output all of its contents to a **CSV**.

4.3 Printer To CSV

In the first stages of this project, Python 3 handled all of the regular expression code and information parsing. However, sectioning the information in the later stages of the process made it extremely hard to visually parse, debug, and test the information for accuracy. After using the **grep** command to move the regular expression to the **BASH** scripts, programming this section became easier. Now the only tasks of this module are to retrieve and organize the information from the **BASH** scripts. Each **BASH** scripts operated by a subprocess command as seen below:

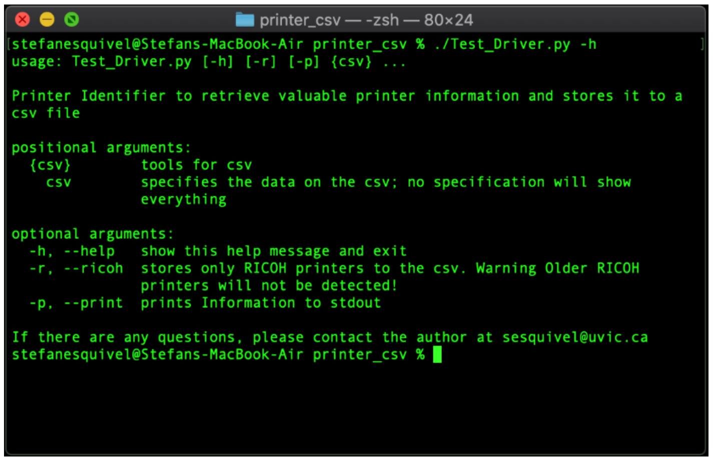


```
try:
    # the bash script executes
    mac = subprocess.check_output(["./arp-mac-pr.sh", e.get_ip()]).decode('ascii').strip('\n')
    # mac is set
    e.set_mac(mac)
except subprocess.CalledProcessError:
    # error if the mac retrieval failed
    print("Can't retrive mac address from \"" + e.get_ip() + "\"!")
```

Figure 5. Subprocess Command to Retrieve Mac Address

4.4 Test_Driver.py

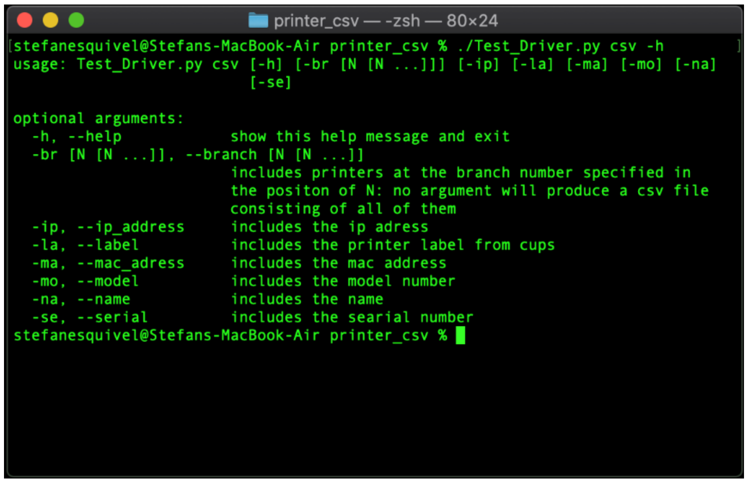
The Test Driver module is used for testing and facilitating where the code executes. Argparse is used extensively in this section to provide an instruction set for the user to operate the script [8]. Running the "./Test_Driver -h." Command results in the following:



```
[stefanesquivel@Stefans-MacBook-Air printer_csv % ./Test_Driver.py -h
usage: Test_Driver.py [-h] [-r] [-p] {csv} ...

Printer Identifier to retrieve valuable printer information and stores it to a
csv file

positional arguments:
  {csv}        tools for csv
    csv        specifies the data on the csv; no specification will show
               everything

optional arguments:
  -h, --help   show this help message and exit
  -r, --ricoh  stores only RICOH printers to the csv. Warning Older RICOH
               printers will not be detected!
  -p, --print  prints Information to stdout

If there are any questions, please contact the author at sesquivel@uvic.ca
stefanesquivel@Stefans-MacBook-Air printer_csv %
```

Figure 6. General Help

This script gives the user option to print to the **stdout** and only retrieve Ricoh printers. It also has a **CSV** sub parser that gives the user options to select the information they want on the **CSV**. This help can be accessed by the command “./Test_Driver csv -h.”:



```
stefanesquivel@Stefans-MacBook-Air printer_csv % ./Test_Driver.py csv -h
usage: Test_Driver.py csv [-h] [-br [N [N ...]]] [-ip] [-la] [-ma] [-mo] [-na]
                          [-se]

optional arguments:
  -h, --help            show this help message and exit
  -br [N [N ...]], --branch [N [N ...]]
                        includes printers at the branch number specified in
                        the positon of N: no argument will produce a csv file
                        consisting of all of them
  -ip, --ip_address     includes the ip adress
  -la, --label          includes the printer label from cups
  -ma, --mac_adress     includes the mac address
  -mo, --model          includes the model number
  -na, --name           includes the name
  -se, --serial         includes the searial number
stefanesquivel@Stefans-MacBook-Air printer_csv %
```

Figure 7. CSV Help

Overall developing the Python 3 code ran seamlessly and ties the script together.

5. Deployment and Performance

The following information underlines vital performance stats and runtimes, under several different settings and commands.

5.1 Performance Tests

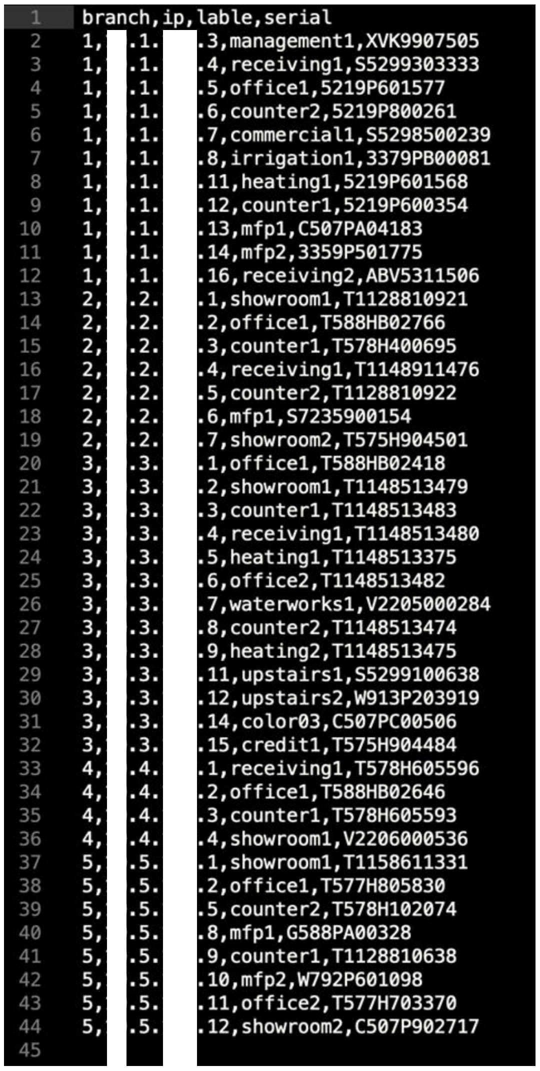
Following the criteria set in the project planning stage, the program is tested according to a few quantifiable performance metrics, including runtime, **CPU** load percentage and #Of errors, as seen in the table on the next page:

Table 5. printer_csv Performance Tests

Command	**Description**	**Runtime (s)**	**Max CPU utilization (% on intel i5-5200U)**	**Errors**	**Error Description**
./Test_Driver.py	Outputs all printer information discovered to **stdout**	183	25	5	Printer too old: no information found from snmpwalk
./Test_Driver.py csv	Outputs all printer information discovered to printers.csv	192	23	5	Printer too old: no information found from snmpwalk
./Test_Driver.py -p -r csv	Outputs all Ricoh printer information discovered to printers.csv and **stdout**	168	23	1	Printer too old: no information found from snmpwalk
./Test_Driver.py -p -r csv -ip	Outputs the **IP** of the printers to the printers.csv	78	24	5	Printer too old: no information found from snmpwalk
./Test_Driver.py -p -r csv -br 12 3 1 -mc	A test consisting of some user options	44	23	0	No errors
./Test_Driver.py -p -r csv -br 1 2 3 4 5 -ip -la -ma -mo -na -se	A test consisting of many user options	67	24	0	No errors

5.2 Usage

This project was used in practice to submit a **CSV** file to Ricoh with the printer information necessary to resolve the incorrect billing error, mentioned at the beginning of this report. After two weeks of use, there were no bugs that appeared in the code. However, the code would occasionally break with user error, such as a call to search for printers at a branch that does not exist. Since this project is in the early stages of development, future updates can resolve this issue. Another flaw was the runtime, as seen in the performance table. Since the **ASL** IT department operates in a fast-paced environment, future versions must tighten the runtimes. To the right is an example of a **CSV** file that we used to see which printers were active:



```
branch,ip,lable,serial
1,[redacted].1.[redacted].3,management1,XVK9907505
1,[redacted].1.[redacted].4,receiving1,S5299303333
1,[redacted].1.[redacted].5,office1,5219P601577
1,[redacted].1.[redacted].6,counter2,5219P800261
1,[redacted].1.[redacted].7,commercial1,S5298500239
1,[redacted].1.[redacted].8,irrigation1,3379PB00081
1,[redacted].1.[redacted].11,heating1,5219P601568
1,[redacted].1.[redacted].12,counter1,5219P600354
1,[redacted].1.[redacted].13,mfp1,C507PA04183
1,[redacted].1.[redacted].14,mfp2,3359P501775
1,[redacted].1.[redacted].16,receiving2,ABV5311506
2,[redacted].2.[redacted].1,showroom1,T1128810921
2,[redacted].2.[redacted].2,office1,T588HB02766
2,[redacted].2.[redacted].3,counter1,T578H400695
2,[redacted].2.[redacted].4,receiving1,T1148911476
2,[redacted].2.[redacted].5,counter2,T1128810922
2,[redacted].2.[redacted].6,mfp1,S7235900154
2,[redacted].2.[redacted].7,showroom2,T575H904501
3,[redacted].3.[redacted].1,office1,T588HB02418
3,[redacted].3.[redacted].2,showroom1,T1148513479
3,[redacted].3.[redacted].3,counter1,T1148513483
3,[redacted].3.[redacted].4,receiving1,T1148513480
3,[redacted].3.[redacted].5,heating1,T1148513375
3,[redacted].3.[redacted].6,office2,T1148513482
3,[redacted].3.[redacted].7,waterworks1,V2205000284
3,[redacted].3.[redacted].8,counter2,T1148513474
3,[redacted].3.[redacted].9,heating2,T1148513475
3,[redacted].3.[redacted].11,upstairs1,S5299100638
3,[redacted].3.[redacted].12,upstairs2,W913P203919
3,[redacted].3.[redacted].14,color03,C507PC00506
3,[redacted].3.[redacted].15,credit1,T575H904484
4,[redacted].4.[redacted].1,receiving1,T578H605596
4,[redacted].4.[redacted].2,office1,T588HB02646
4,[redacted].4.[redacted].3,counter1,T578H605593
4,[redacted].4.[redacted].4,showroom1,V2206000536
5,[redacted].5.[redacted].1,showroom1,T1158611331
5,[redacted].5.[redacted].2,office1,T577H805830
5,[redacted].5.[redacted].5,counter2,T578H102074
5,[redacted].5.[redacted].8,mfp1,G588PA00328
5,[redacted].5.[redacted].9,counter1,T1128810638
5,[redacted].5.[redacted].10,mfp2,W792P601098
5,[redacted].5.[redacted].11,office2,T577H703370
5,[redacted].5.[redacted].12,showroom2,C507P902717
```

Figure 8. CSV Example (Censored)

6. Conclusion

As discussed, this project successfully tackled the central issue of the incorrect billing invoice from Ricoh. It showed how hacking and network security tools could successfully locate hardware information within an enterprise-level business. This project helped **ASL** prevent incurring costs due to out of date devices being incorrectly charged. According to the performance metrics, it was suitable in most of the sections; however, it seemed weak for runtime.

7. Recommendations

Building this project opens a discussion for additional improvements and recommendations. The three most important sections for improvement include cross-platform compatibility, increased speed, and more user arguments and options. Improving cross-platform compatibility is the first step to improve integration with Linux. Adding a feature to detect the **OS** can allow the code to adapt to the **SNMP** keys to work with Linux. This integration opens up the opportunity to run the script on more devices, including the Head Office Server at **ASL**. Lowering the runtime is the next recommendation. Since Nmap is open-source, exploring the tool and tweaking it to improve the run time can better tailor the script for a fast IT environment. Exploring different port scanning tools other than Nmap could also allow for quicker runtime. Lastly, adding more arguments and features to improve user experience can also make this tool dexterous. Additional features include building a **GUI**, adding options for different file types, and including more printer features like page count and uptime. All of these recommendations improve the scalability and user experience and reduce learning times, making it a better application.

References

[1] "Company," Andrew Sheret Limited, 2020. [Online]. Available: https://www.sheret.com/company. [Accessed 23 April 2020].

[2] "What is RFID and How Does RFID Work?," AB&R, [Online]. Available: https://www.abr.com/what-is-rfid-how-does-rfid-work/. [Accessed 25 April 2020].

[3] "Ricoh at a Glance," Ricoh, [Online]. Available: https://www.ricoh.com/about/at-a-glance/field/. [Accessed 23 April 2020].

[4] "Importance Of Modularity In Programming," AOSD, 18 January 2018. [Online]. Available: http://aosd.net/importance-of-modularity-in-programming/. [Accessed 26 April 2020].

[5] N. Hamilton, "The A-Z of Programming Languages: BASH/Bourne-Again Shell," Computerworld, 30 May 2008. [Online]. Available: https://www.computerworld.com/article/3481039/the-a-z-of-programming-languages-bash-bourne-again-shell.html. [Accessed 13 April 2020].

[6] "subprocess," Python Software Foundation, 26 April 2020. [Online]. Available: https://docs.python.org/3/library/subprocess.html. [Accessed 26 April 2020].

[7] S. L. Margaret Rouse, "object-oriented programming (OOP)," Tech Target, April 2020. [Online]. Available: https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP. [Accessed 26 April 2020].

[8] "argparse," Python Software Foundation, 26 April 2020. [Online]. Available: https://docs.python.org/3/library/argparse.html. [Accessed 26 April 2020].

[9] R. P. M. M. D. L. Max Howell, "Homebrew," [Online]. Available: https://brew.sh/. [Accessed 16 January 2020].

[10] Computer Hope, "Linux arp command," Computer Hope, 4 May 2019. [Online]. Available: https://www.computerhope.com/unix/arp.htm. [Accessed 26 April 2020].

[11] G. Lyon, "Nmap," [Online]. Available: https://nmap.org/. [Accessed 25 4 2020].

[12] T. Keary, "snmpwalk Examples for Windows and Linux," Comparitech Limited, 21 June 2019. [Online]. Available: https://www.comparitech.com/net-admin/snmpwalk-examples-windows-linux/. [Accessed 23 04 2020].

[13] "csv," Python Software Foundation, 26 April 2020. [Online]. Available: https://docs.python.org/3/library/csv.html. [Accessed 26 April 2020].

[14] "https://www.atlassian.com/," [Online]. Available: https://www.atlassian.com/. [Accessed 7 January 2020].

[15] "Matrix mixes life and hacking," BBC News, 19 May 2003. [Online]. Available: http://news.bbc.co.uk/2/hi/technology/3039329.stm. [Accessed 26 April 2020].

[16] "What Is CIDR Notation?," What Is My IP Address, [Online]. Available: https://whatismyipaddress.com/cidr. [Accessed 27 April 2020].

[17] Security Trails Team, "What is banner grabbing?," Security Trails, 14 November 2019. [Online]. Available: https://securitytrails.com/blog/banner-grabbing. [Accessed 26 April 2020].

[18] "Port Specification and Scan Order," NMAP.ORG, [Online]. Available: https://nmap.org/book/man-port-specification.html. [Accessed 26 April 2020].

[19] "Simple Network Management Protocol (SNMP)," Wire Shark, 1 November 2019. [Online]. Available: https://wiki.wireshark.org/SNMP. [Accessed 15 April 2020].

[20] O. T. Web, "How to Exploit SNMP for Reconnaissance," Wonder How To, 3 January 2014. [Online]. Available: https://null-byte.wonderhowto.com/how-to/hack-like-pro-exploit-snmp-for-reconnaissance-0150181/. [Accessed 10 April 2020].

[21] "Linux arp command," Computer Hope, 4 May 2019. [Online]. Available: https://www.computerhope.com/unix/arp.htm. [Accessed 23 April 2020].

[22] "What is MAC spoofing?," Digital Guide, 24 August 2017. [Online]. Available: https://www.ionos.ca/digitalguide/server/know-how/what-is-mac-spoofing/. [Accessed 10 April 2020].

[23] "How To Use grep Command In Linux / UNIX," nixCraft, 24 April 2020. [Online]. Available: https://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/. [Accessed 25 April 2020].

[24] T. Hunner, "How to make an iterator in Python," Trey Hunner, 18 June 2018. [Online]. Available: https://treyhunner.com/2018/06/how-to-make-an-iterator-in-python/. [Accessed 2 April 2020].