Detection of Operating System Types

(Progress Report)

Alem Fitwi
Electrical & Computer Egineering Department
State University of New York at Binghamton
(Ph.D. Student)
Binghamton, New York, USA
afitwi1@binghamton.edu

Abstract— **The Internet has amazingly revolutionized every aspect of human life. The revolution of the Internet can be traced back to the development of the Advanced Research Projects Agency Network (ARPANET) in the early 1970s by the Department of Defense of the USA based on which the Internet, the TCP/IP protocol, that has eventually become the most widely used network interconnection protocol, was developed. However, insufficient security concerns at the beginning of the design have left a lot of weakness that could be exploited by attackers to compromise computing systems. Particularly, the vulnerabilities of the Internet protocols and Technologies are widely exploited by both internal and external users and attackers. Hence, there has been a tremendously pressing quest for the design, development, and deployment of less-resource hungry, effective intranet security solutions that ensure secure, authentic, and authorized access to enterprise resources. The prime goal of this course project is to explore the preludes to the design and implementation of strong Intranet Security Management System. Specifically, Operating Systems Type detection, a part of an intranet security solution, will be implemented using such technologies as NMAP (Network Mapper) and a powerful python network module called Scapy. It focuses on the implementation of an algorithm that will function in a way similar to network mapping tool to detect the type and version of operating systems running on local area network (LAN) hosts where the Raspberry Pi will be used to run the python script which will be responsible for the detection of the OS via packet analysis, port scanning, port filtering, and based on predefined peculiar attributes of an OS like TTL (Time To Live) and Window Size.**

Keywords— Internet, ARPANET, TCP/IP, Intranet security, Raspberry PI, security policies, TTL, NMAP, Scapy, & LAN

I. INTRODUCTION

The Internet has alarmingly revolutionized the way we communicate, the way we do business, and the way we live. The start of the revolution of the Internet can be traced back to the development of the Advanced Research Projects Agency Network (ARPANET) in the early 1970s by the Department of Defense of the USA. The ARPANET was an early packet switching network and the first network to implement the protocol suite TCP/IP, which later both technologies became the technical foundations of the Internet. In other words, along with the rapid development of the Internet, the TCP/IP protocol has become the most widely used network interconnection protocol [3]. However, due to insufficient security concerns during the design phase, the protocol has a lot of security risks. It is to be recalled that the Internet was first applied in a research environment for a few trusted user groups. Therefore, network security problems were not the major concerns at that time which has left big security holes in the TCP/IP protocol stacks. The vast majority protocols do not provide the necessary security mechanisms. Various studies show that many of the vulnerabilities of any network, be it Intranet, Extranet or Internet are compromised or affected by intentional or unintentional attacks directed from Terminals of internal users where the edge security devices like firewalls and antivirus alone can't solve [4, 5, 6, 7, 8, 9].

One of the solutions is building a strong Intranet management system capable of fully implementing an enterprise's security policy that can safeguard the intranet from possible attacks by enforcing Identity and security checks or authentications. This project focuses on one of the major preludes or prerequisite for strong Intranet security solution– the detection of operating systems and their versions. One of the major functions that any Intranet security solution should possess is security authentication. Security Authentication refers to the process of detecting the type of OS running in hosts along with their versions, and scanning for vulnerabilities so as to recommend timely correct patching and version updates. This is the main push factor that aroused my interest to work on "Operating Systems Type Detection" as part of my Hardware Based Security course hoping that it will serve as a stepping-stone or prelude to my future research works.

So far, in this project work, a review of many related papers and technologies was made. Then, in what ensues, two more sections are presented. Section II explores the major tasks accomplished to date and the results produced. Then, section III briefly outlines the remaining tasks which will be accomplished until or before May 15, 2018.

II. TASKS ACCOMPLISHED

So far, I have made a survey on how operating systems are differentiated from one another in the eyes of NMAP and the Scapy python module for network security. What is more, I have tried my level best to explore how the NMAP and the Scapy operate, and what special features they comprise. The major tasks and surveys I have accomplished so far are tersely presented in what ensues in five sub-sections.

A. *Proposed Implementation Model*

My python script is being developed with the intent of equipping it with capabilities to detect the operating systems (along with their versions) of hosts connected to any network to which the host or server running the OS detection python script is connected. For security and safe practice ethical reasons, I have decided to create my own LAN where the Raspberry PI will be the heart of it as portrayed in figure 1. Then, as many hosts running different operating systems as possible can be connected to it using a layer 2 switch as depicted in figure 1. Putting it another way, the python script I am developing exploiting the important security features of NMAP and SCAPY can be deployed in any network, but for security and simple demonstration reasons, I will run it in an isolated LAN similar to the one depicted in figure 1.

Summing it up, the architecture portrayed in figure 1 is what I have proposed for the simple but meaningful demonstration of my project. I already have the Tinker Board of Raspberry PI with a Micro SD card adapter and a micro SD card memory of size 64GB.

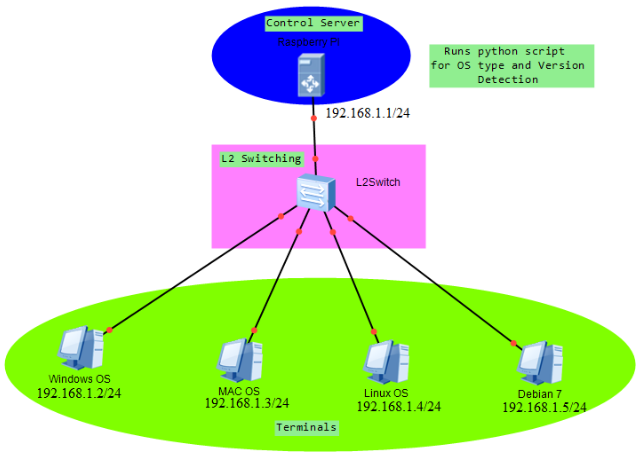

Fig.1 Simple OS Detection Model

B. *Determining Types of Operating Sysetms*

I have made a thorough survey in search of the attributes or features that could be used to distinguish one operating system from another. So far, I have found out that operating systems could be identified based on some contents of the packet they send. Specifically such attributes of a packet as TTL and Window size could be employed to identify what type of an operating system a given network host is running

Time to live (TTL) or hop limit is a mechanism that limits the lifespan or lifetime of data in a computer or network. TTL could be implemented as counter or timestamp attached to or embedded in the data. Once the prescribed event count or timespan has elapsed, data is discarded. In computer networking, TTL prevents a data packet from circulating indefinitely. In computing applications, TTL is used to improve performance of caching. TCP window size is one of the most popular options for network troubleshooting or performing an application baseline. The TCP window size, a.k.a. the TCP receiver window size, is simply an advertisement of how much data (in bytes) the receiving device is willing to receive at any point in time. The receiving device can use this value to control the flow of data, or as a flow control mechanism. Some operating systems will use a multiple of their maximum segment size (MSS) to calculate the maximum TCP window size [10].

Generally speaking, Operating systems of different vendors and architecture might have different TTL values and different window sizes the combination of which could be used as a distinguishing OS signature. Table 1 lists the various types of operating systems along with their respective TTL values and TCP window sizes, showing that the combination of the two is different for most of the operating systems.

Table 1: OS Distinguishing Features/Attributes

S/N	***Operating System***	***TTL***	***Window Size***
1	Windows XP	128	65535
2	Windows 2000/ Server 2003	128	16384
3	Windows 7/Vista/ Server 2008	128	8192
4	Linux Kernel 2.x	64	5840
5	Linux Kernel 3.x	64	14600
6	FreeBSD	64	65535
7	Chrome OS/Android	64	5720
8	MAC OS	64	65535
9	CISCO IOS 12.4	255	4128

C. *NMAP*

NMAP stands for Network Mapper. It is free, open source security scanner for auditing network that runs on most platforms and was originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich). When it is aimed at a particular host, NMAP can determine which ports are open, which operating systems and versions are running, what services are offered and what firewalls are being used [1,2].

Though NMAP is a powerful Network Discovery Tool, the output might not make sense to us human unless some apt analysis and processing is done on the discovered packets. Usually the output is presented in the form of lengthy or list (a data structure type) comprising a number of tuples as elements. Hence, the knowledge of Python data structures like Dictionary, Lists, Tuples, and the corresponding methods and functions coupled with Scapy features is extremely vital to successfully unpack packets and extract useful/insightful information. Regular Expressions are also of paramount importance in analyzing packets. Table 2 depicts some of the commonest NMAP commands [1,2].

Table 2: NMAP commands

S/N	*NMAP Command*	*Brief Description*
1	NMAP -sP	Ping scans the network, listing machines that respond to ping
2	NMAP –p –sV -sS	Full TCP port scan using service version detection
3	NMAP -sS	TCP Sync Scan
4	NMAP -F	Fragment packets (optionally w/ given MTU)
5	NMAP -sV	Probe open ports to determine services/version information
6	NMAP –O	Enable OS Detection
7	NMAP -sn	Ping scan - Disable port scan
8	NMAP –T 0-5	Set timing template-higher is faster (less accurate)

D. SCAPY

It is an interactive packet manipulation tool with which you can capture the code, analyze, create, and send network packet. Scapy is flexible that lets you create any packet you can think of and send it on the wire even if it is invalid packet. You can also use it to create or build network Scanners, Sniffers, or even attack tools for ARP poisoning or DHCP starvation [1]. Of course, we should build such kind of utilities for learning and testing purposes only as an ethical or white hacker. It has plenty of useful methods like lst() that lists the commands supported by Scapy, ls() that outputs long list of network protocols supported by Scapy, sniff() for listening or sniffing packets, and packets.show() for packet analysis.

I first fully developed my python script based on Scapy features, but at the time of testing I discovered that this script can't be run on the Raspberry PI due to the fact that the Raspberry PI Tinker Board has limited capacity, insufficient to accommodate the heavy Scapy features. Then, I opted to heavily rely on NMAP features while developing the script so as to come up with a light script suitable for the Tinker Board.

E. Python Script Development

I have been developing a python script exploiting the powers of NMAP and Scapy modules of Python Programming languages to easily detect OS's of hosts in a network. I recently tested my script on my home LAN by running it on my laptop, and it successfully detected the operating systems of all devices connected to the WLAN in my apartment area. I will continue to further refine it in the subsequent days until the deadline. Scapy is a very powerful tool not only for packet analysis but also for packet creation and fabrication. It is a very powerful and useful module but I am gravitating towards using more features of NMAP in my python script because the Scapy module is so heavy and resource hungry that I can't run it on the Raspberry PI. As a result, I am focusing on a light-weight program which can be run on the Raspberry PI.

III. FUTURE WORKS

I feel like I have accomplished the milestone or majority of the tasks but there are some tasks I should accomplish in the remaining days. The major remaining tasks include the following, which are expected to be accomplished well before the deadline, May 15, 2018.

- Refining the python script
- Setting up the Raspberry PI as to support Python
- Carrying out demonstration on isolated LAN
- Final project Presentation
- Compiling the final report and submission

IV. CONCLUSION

In conclusion, the various techniques for the detection of operating systems running in network hosts and available technologies were surveyed and studied. Particularly, data about the distinguishing features of operating systems were analyzed and identified. What is more, the powers and features of NMAP and the Scapy python module were thoroughly studied followed by the development and testing of a python script exploiting their important attributes.

REFERENCES

[1] Alan Freedman, "Computer Desktop Encyclopedi", unpublished Electronic version.

[2] https://highon.coffee/blog/nmap-cheat-sheet/, accessed on 18 April 2018

[3] Abbate, Janet Ellen. "From ARPANET to Internet: A history of ARPA-sponsored computer networks, 1966--1988." (1994): 8-13

[4] Bellovin, Steven M. "A look back at" security problems in the TCP/IP protocol suite." *Computer Security Applications Conference, 2004. 20th Annual*. IEEE, 2004: 1-2

[5] Richardson, Robert. "2011 CSI computer crime and security survey," 2011." (2010): 19-21

[6] Singh, Utkarsh, Sumit Jaiswal, and R. S. Singh. "Cloud banking." *CSI Transactions on ICT* (2016): 1-6.

[7] CeRT, MAJOR AUSTRALIAN BUSINESS. "2015 CYBER SECURITY SURVEY." (2015): 19-23

[8] White, G. K. *Simple Institutional and User Best Practices for Cybersecurity in Research Reactors*. No. LLNL-CONF-679241. Lawrence Livermore National Laboratory (LLNL), Livermore, CA, 2015: 3-7

[9] Ivanovs, Ivo, and Sintija Deruma. "Revising Cybersecurity Skills for Enterprises."

[10] Larry L.Peterson and Bruce S. Davie, "Computer Networks, A System Approach", 3rd Edition, Morgan Kaufman Publishers