Passive Operating System Fingerprinting Analysis Using Artificial Intelligence Techniques

David Benenati, Zach Mueller, Dr. Ezhil Kalaimannan, and Dr. Caroline John
Department of Computer Science, Hal Marcus College of Science and Engineering, University of West Florida

Introduction

Modern enterprise networks are complex and present countless security challenges. Understanding the nature of the systems that exist within a network environment is a vital step in securing such environments. Hence, operating systems on the network must be identified, tracked, and continuously monitored. In this research, we consider the problem of detecting unauthorized operating systems on an enterprise network, which could exist as a result of the unintentional actions of an authorized user or by the unauthorized actions of internal users or external attackers. A possible scenario involves malware applications that attempt to install virtual machines on a host system within the network as a means of obfuscation and detection avoidance. We intend to utilize a neural network based classifier which will be developed using the pytorch and fastai deep learning libraries. Simulated network traffic will be generated through the implementation of a virtual network environment and the generated traffic will be passively collected and analyzed as it traverses the network boundary. The performance evaluation of the neural network classifier will also be analyzed using the collected data in this research.

Operating System (OS) fingerprinting can be used during offensive or defensive operations as shown in the Security Testing Roadmap.

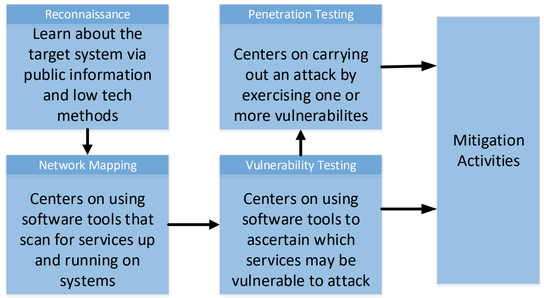

Figure 1: Security Testing Roadmap [1]

OS fingerprinting involves the capture and examination of network packets. The OS can be accurately guessed based on differing packet features.

Attackers and penetration testers use OS fingerprinting to identify and enumerate potential targets.

- Active fingerprinting initiates a scan by sending specially crafted packets to a target and examining response packets.
- Passive fingerprinting examines existing normal network traffic without generating additional traffic.

Effective Network Defense

- Begins with establishing an understanding of the systems that exist on the network.
- Passive OS fingerprinting can be used to achieve this objective without creating excess traffic on the network.

Research Objectives

This study will consider the application of passive OS fingerprinting as a means of collecting packets that traverse the network with the ultimate goal of using captured packets to detect unauthorized operating systems on a network.

How and why does fingerprinting work?

- RFC's 791 and 793 specify the required packet structure for IP and TCP protocols.
- Both RFC's leave a number of packet structure decisions up to OS developers.
- Operating systems are identified by examining these differences among TCP/IP packet features [2].
- Figures 3 and 4 display TCP/IP packet structure

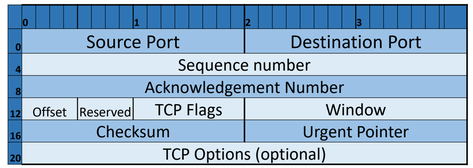

Figure 2: IPv4 Packet Header [3]

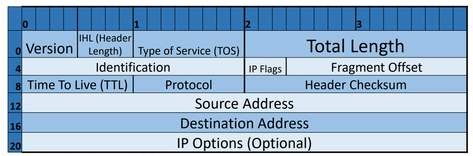

Figure 3: TCP Packet Header [4]

- Common packet features used to identify operating systems include Time To Live (TTL), Window Size, checksum, and total length [5].
- Figure 3 displays many of the TCP packet features that can be examined to identify operating systems.

Common TCP/IP Features Used in OS Fingerprinting	
TCP Features	**IP Features**
stream	checksum
options.timestamp.tsval	ttl
window_size_scalefactor	len
port	proto
srcport	opt.ra
analysis.duplicate_ack_frame	dsfield.dscp
analysis.duplicate_ack_num	frag_offset
analysis.out_of_order	
analysis.rto_frame	
flags	

Figure 4: Common Packet Features Used in OS Fingerprinting [5]

This experiment aims to identify a small dataset of packet features of different protocol packets that most efficiently and accurately identifies the operating system.

Methods

A virtual network infrastructure will be constructed specifically for the purpose of conducting the experiment. A general overview of the network infrastructure is displayed in figure 5.

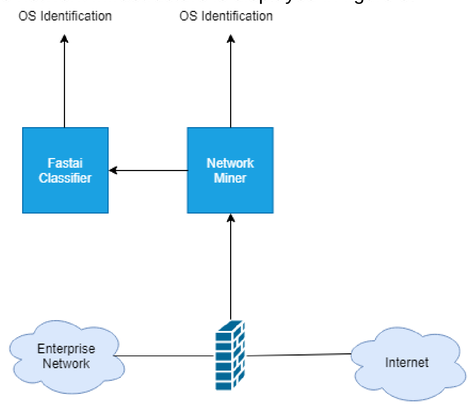

Figure 5: Common Packet Features Used in OS Fingerprinting [6]

Network traffic will be mirrored as it enters the LAN side of the gateway and packet data will be captured for analysis

- Network Miner will be used to capture packets
- Network Miner is a forensic analysis tool that can be used as a passive packet capture tool.
- The OS is identified by comparing captured packets to OS fingerprints in the Satori and p0f databases.

The experiment intends to explore the use of the fastai machine learning libraries in identifying which packet features can be best used to identify operating systems.

What is fastai?

- A set of code libraries designed to simplify writing machine learning tasks.
- Built on top of the PyTorch library.
- Allows for tasks such as running a convolutional neural network for pattern recognition [6].

Included Operating Systems

- Windows 7
- Windows 8
- Windows 10
- OpenBSD
- Linux Distributions
  - Debian
  - CentOS
  - OpenSUSE

Proposed results

Initial results will be based on operating systems as identified by Network Miner.

The performance of fastai machine learning classifiers will be evaluated as compared to the OS predictions compiled using Network Miner.

Performance Evaluation

- Accuracy - The ability to correctly predict the OS
- Efficiency - Not simply resource usage
  - OS identification based on learned patterns
  - Databases that must be manually updated are not needed.

Further consideration will be given to specific problem scenarios

- Virtual operating systems installed on physical host systems
- Verification of systems that match an approved OS actually belong on the network.
  - Will require a solution separate from fingerprinting techniques.

References

[1] D. Kim and M. Solomon, Fundamentals of Information System Security. Burlington, MA; Jones and Bartlett Learning, 2018.
[2] J. M. Allen, "OS and Application Fingerprinting Techniques," *SANS Institute Information Security Reading Room,* 2007.
[3] Information Sciences Institute , "Internet Protocol: DARPA Internet Program Protocol Specification," September 1981. [Online]. Available: https://tools.ietf.org/html/rfc791. [Accessed 8 March 2019].
[4] Information Sciences Institute, "Transmission Control Protocol: DARPA Internet Protocol Specification," September 1981. [Online]. Available: https://tools.ietf.org/html/rfc793. [Accessed 8 March 2019].
[5] A. Aksoy and M. H. Gunes, "Operating System Classification Performance of TCP/IP Protocol Headers," in IEEE 41st Conference on Local Computer Networks Workshops, 2016.
[6] fastai. (2019, April 1). fast.ai: Making neural nets uncool again. Retrieved from fast.ai: https://www.fast.ai/