Identifying Operating System Using Flow-based Traffic Fingerprinting

Tomáš Jirsík and Pavel Čeleda

Institute of Computer Science, Masaryk University,
Botanická 68a, Brno, Czech Republic
{jirsik|celeda}@ics.muni.cz

Abstract. Many vulnerabilities are operating system specific. Information about the OS of all hosts in a network represents a valuable asset for network administrators. While OS detection in small networks is an easy task, expanding the same process on a large scale becomes a challenge. The weak performance, high speed traffic and large amount of hosts for OS detection are issues to overcome. In this paper we propose a flow based framework for large scale OS detection. Furthermore, we describe the framework implementation into a flow probe, provide performance comparison and share remarks on deployment in a real world network.

Keywords: OS fingerprinting, passive, high-throughput, p0f, flow

1 Introduction

Being aware of all operating system (OS) communicating in a network means an advantage for administrators protecting network security. Hosts with vulnerable OSs pose a serious threat to security as they could be misused as an entry-point to the network. This awareness also improves network management as any host with an outdated system can be easily identified (e.g., the recent termination of Windows XP support). Since 1994, when the main concept was introduced in [3], two main approaches to OS detection have emerged: active and passive.

The existing active OS detection tools such as *nmap* [5] launch a set of TCP, UDP and ICMP probes to a scanned host and detect OS system based on responses they receive. This approach provides high accuracy, but is also time demanding especially in large networks as we need to scan each host. Furthermore, it inserts other traffic into network, which increases bandwidth. This approach has the form of a network scan which, without permission, is legally questionable.

On the other hand, passive OS detection does not insert any packets into the network traffic. The OS type is determined based on the traffic captured from the network. One widely used passive OS detection tool is p0f [6], which employs data extracted from packet L3-L4 headers and even utilizes the content of application-level payload to detect OS. p0f is a cornerstone for other tools such as Ettercap, Disco, Yaf or Satori [6]. The advantages of passive fingerprinting are that it does not leave any traces and does not modify the traffic. This approach

is suitable for deployment in large networks as there is no necessity to actively probe each host.

We identify following the criteria for a OS detection tool working on large scale: easy deployment and use, passive detection, and high performance. High performance is desired in order to be able to scan large networks entirely and handle traffic incoming at high speeds. Passive detection is preferred to active because it does not affect the traffic in any way and is capable of monitoring large amount of hosts at the same time. Ease of deployment and use are required to save network administrators' time.

In this paper, we present a flow based OS detection framework for use on a large scale. We implement the framework into a flow probe, compare its performance with other OS detection tools and discuss some remarks on its deployment in a real network.

2 Flow Based OS Detection

In this section we describe a flow based framework for OS detection in large networks. Such an approach to OS detection has been chosen for following reasons. First, flow based monitoring [2] is widely used standard for monitoring large network infrastructures. This fact eases the deployment of our detection framework as there is no need to introduce a new monitoring infrastructure. Second, observation points for flow based monitoring are suitably situated. Usually, the points are placed on the main network traffic hubs and network borders. Therefore all hosts can be monitored at the same time. Moreover, we are able to detect the OS of all hosts in a network without accessing them. Lastly, the concept of flow based network traffic monitoring is designed to process network traffic even at high speeds of up to 100 Gbits/s, which makes it suitable for deployment in high speed networks.

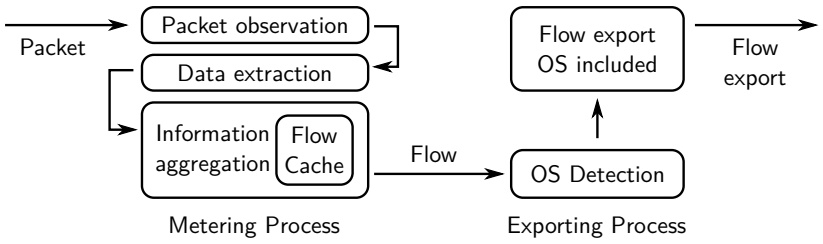

Fig. 1: Architecture of flow based OS detection framework

The proposed framework for large scale OS detection is described in Fig. 1. The most utilized part of the framework is the Metering Process as millions of packets per second are processed there. For this reason, we postpone all logic related to OS detection to less utilized processes. During the Metering Process packets are observed at the observation point (e.g., using TAPs). Each packet

is processed and relevant data are extracted. Apart from the basic set of values needed for flow creation, we extract data necessary for OS detection, namely TTL, SYN packet size, initial size of TCP window and User Agent field from HTTP protocol. The number of values extracted from the packet can be easily increased by adding new rules into the Data extraction process. The gathered information is then aggregated in Flow Cache into flows. Anytime the inactive or active timeout is triggered, the flows are passed further onto the Exporting process. So far, there has been no logic to detect OS. This logic is part of the Exporting Process. The delay of logic for OS detection decreases the computational demands as the number of flows to process is significantly lower than the number of packets. The logic for OS detection can be represented by the comparison of gathered OS specific information with a database of OS fingerprints. The improvement of logic for OS detection is left for future work. When an OS is detected, OS information is added to flows and flows are exported to collector.

We have implemented the framework into FlowMon probe [4] as a set of processes and filter plugins.[1] Furthermore, we have benchmarked the tool at a set of 1068 packets (TCP SYN packets : HTTP GET packets : other TCP/UDP packets = 1 : 1 : 1), which were loaded in loop from memory to probe[2]. The FlowMon probe running without detection plugins processed 18.328 Mp/s. Using only OS detection based on User Agent field, the throughput dropped by 50.42 % to 9.087 Mp/s. Running the detection based on all fields (i.e. TTL, SYN size, TCP win. size) the performance decreased by another 6.49 % to 8.497 Mp/s.

The main contribution of our work is the framework design with high performance OS detection in large networks. Other passive OS detection tools such as p0f (or k-p0f) are able to process up to 0.05 Mp/s (or 0.6 Mp/s) [1]. Our approach benefits from flow monitoring framework specially designed for deployment in large networks. Therefore, we were able to increase the performance of OS detection remarkably. We do not evaluate the precision of the OS detection, since we collect the same entries from network traffic as (k-)p0f tool in [1]. Therefore the precision of detection is dependent solely on the quality of the fingerprint database.

3 Deployment and Further Remarks

We deployed the detection tool in an university campus network and collected data for two hours. During this period we observed 10.221M flows from 12 897 hosts in the campus network. 33.5 % (3.425M) of all monitored flows contained all the information needed for OS detection which represented 70.33 % (9 072) of all hosts. We observed that in some cases more than one OS was detected for one IP. The cause of this behavior could be dynamic addressing in networks. Therefore we removed all dynamically addressed subnets from evaluation.

The results (see Table 1) show, that the portion of the IP addresses with more than one detected OS has decreased after the removal of dynamically addressed

[1] Description and sources available at `http://is.muni.cz/th/359565/fi_b`

[2] Configuration: Intel Xeon CPU E31230 @ 3.20GHz, 15GB RAM, and Linux (64 bit).

networks. However, still 4 % of IP shows characteristics of two or more OS. This fact can be explained by the presence of more devices with different OS using the same IP address. This implies the presence Network Address Translation (NAT) devices. Therefore, the OS detection can be used also as NAT detection assuming that only a static addressed network is monitored.

# of unique OS	# of IP in A	% of all A	# of IP in B	% of all B
1	7898	87.059	3996	95.989
2	1071	11.806	159	3.819
3	80	0.882	7	0.168
> 3	23	0.253	1	0.024
Total	9072	100 %	4163	100 %

Table 1: Number of unique OS detected at one IP:
A - whole network, **B** - dynamically addressed subnets removed

The OS detection framework presented in this paper represents a useful tool for network administrators. It meets all previously defined requirements for deployment in large scale networks: easy deployment and use, passive monitoring, and high performance. The ease of deployment and use is ensured by taking advantage of existing flow based monitoring infrastructure commonly used in such networks. The design of the framework also ensures the flexibility and, as it has been shown, the increase in performance. In future work, we will focus on the OS detection logic. We would like to enlarge the database of fingerprints by adding OS specific elements and provide a new approach to fingerprints correlation, which should increase the precision of OS detection.

Acknowledgments. This material is based upon work supported by Cybernetic Proving Ground project (VG20132015103) funded by the Ministry of the Interior of the Czech Republic.

References

1. Barnes, J., Crowley, P.: k-p0f: A high-throughput kernel passive os fingerprinter. In: Architectures for Networking and Communications Systems (ANCS), 2013 ACM/IEEE Symposium on. pp. 113–114 (Oct 2013)
2. Claise, B., Trammell, B., Aitken, P.: RFC 7011: Specification of the IPFIX Protocol for the Exchange of Flow Information (Sep 2013)
3. Comer, D., Lin, J.C.: Probing tcp implementations. In: USENIX Summer. pp. 245–255 (1994), `http://dblp.uni-trier.de/db/conf/usenix/usenix_su94.html#ComerL94`
4. INVEA-TECH: FlowMon Exporter – Community Program (Apr 2013), `http://www.invea-tech.com`, [cited 2014-04-15]
5. Lyon, G.F.: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure, USA (2009)
6. Zalewski, M.: p0f v3, `http://lcamtuf.coredump.cx/p0f3/`, [cited 2014-04-15]