Passive OS Fingerprinting Prototype Demonstration

Martin Laštovička*†, Daniel Filakovský*†
*Masaryk University, Institute of Computer Science, Brno, Czech Republic
†Masaryk University, Faculty of Informatics, Brno, Czech Republic
Email: lastovicka@ics.muni.cz, filakovsky@mail.muni.cz

***Abstract*—Operating system identification of communicating devices plays an important part in network protection. However, current networks are large and change often which implies the need for a system that will be able to continuously monitor the network and handle changes in identified operating systems. In this paper, we propose an architecture of an OS fingerprinting system based on passive network monitoring and a graph-based data model to store and present information about operating systems in the network. We implemented the proposed architecture and tested it on the backbone network of Masaryk University. Our results suggest that it is suitable for monitoring a large network with tens of thousands of actively communicating devices.**

I. Introduction

In our previous work, we have shown that OS fingerprinting methods are viable in the environment of wireless networks and will outlast adaptation of new network protocols [1]. However, modern networks are constantly changing and evolving environment which poses new challenges on any OS fingerprint system. We identified them as follows:

- Dynamic IP assignment – a large number of devices connect without a static address, but rather use address assigned during connection. The fingerprinting system needs to keep track of the actual relation between device and address.
- Rapid changes in the network – devices connect to the network and disconnect from it freely. In our measurement, the median time for device connection was 8.6 minute, after which another device can take the IP address. The fingerprinting system must be able to update its state within the span of minutes and update it for the whole network.

In this paper, we propose a system architecture to detect operating system of every actively communicating device in the network. We use the methods of passive OS fingerprinting presented in our previous work [1] which suit the continuous monitoring requirements. Our system solves the first challenge by maintaining timestamps for each observed fingerprint. This brings the possibility to track device operating system in the present as well as in the past. The second challenge is solved implicitly by the passive monitoring architecture which in our implementation can bring the update of whole network state down to 30 seconds.

II. System Architecture

Our architecture is based on four basic components – flow data collection, data processing, results storage, and web interface. Their interconnection is depicted in Fig. 1 and they are described in detail further in this section.

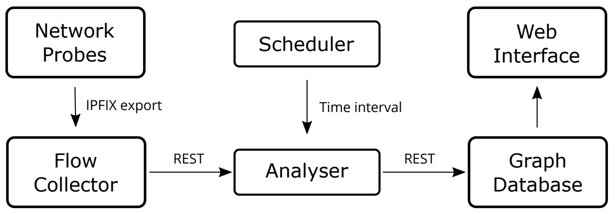

Fig. 1. Proposed architecture

A. Flow monitoring

In our architecture, we aim to monitor large networks using extended network flow technology [2]. In the prototype settings, probes monitor data from network tap mirroring 40 GE backbone link which provides visibility into communication between the monitored network and the Internet. This probe location and connection does not affect OS fingerprinting much as the methods require such communication or are able to identify OS from it.

The flow export must have capabilities to enrich flows with TCP/IP stack parameters (initial window size, size of first SYN packet, time to live value), HTTP User-agent, requested hostname (HTTP hostname or SNI value). We use IPFIX protocol for the data transfer.

B. Data Processing

Two components are responsible for the processing of primary IPFIX data. The first one, scheduler, is a controlling component which invokes data analysis. It is important to schedule the analysis to match collector data storage interval so that each run of analysis has new data to process.

The second component, analyser, is responsible for obtaining data from collector, OS detection, and storing results in the database. The primary data transfer from the collector is realized via REST API through which analyser requests data from the last time interval and filter them so that only flows with the source from subnets of interest are transferred. Primary data may contain sensitive user information, and thus the connection must be encrypted and both collector and analyser should authenticate to each other.

Upon receiving data, an operating system is determined for every active IP address in the network according to the methods introduced in our previous paper [1]. This IP address and OS are bound and stored in the graph database after every run of the analyser.

C. Graph Database

A graph is a natural representation of a network and graph databases allow to store them efficiently and are easily extensible with new types of data or attributes.

We have proposed a lightweight data model to store results with two types of nodes and two of edges, The nodes are:

- **Host** – node representing an element in the network. Has one attribute corresponding to its IP address for identification during OS detection.
- **Operating System** – node to represent one specific version of OS. Has name attribute filled by our hierarchy [1] format *OS name*, *Major version*, *Minor version*. If the level of details is missing, the corresponding fields are left empty.

Types of edges in our model are as follows:

- **Has_OS_Actual** – this edge represents the last discovered relationship of Host to OS and carries time attribute to check when the last calculation was triggered. One Host node can have at most one adjacent Has_OS_Actual edge.
- **Has_OS_History** – edges to track the Host to OS relation in time. Every time OS is detected for a host, its current OS_Actual edge is replaced by OS_history with the same timestamp attribute and a new OS_actual edge is created.

D. Web Interface

We use the Neo4j[1] database frontend in our prototype to access the data and manipulate them. Its built-in visualization engine is suitable for demonstration of both big picture of operating systems in the network and individual host details.

III. System Prototype

We have created a prototype implementation of the proposed architecture and run it on Masaryk university backbone network to evaluate its performance. All source codes are publicly available on GitHub[2].

We used already deployed passive monitoring infrastructure based on Flowmon Networks[3] products for data collection. Our network probes are located at the backbone links connecting the university to outside world which consist of two 40 GE optical routes. For OS fingerprinting we filter the traffic so that only flows with source IP from the university network are processed. For the data processing, we implemented the scheduler and analyser in Python and used Neo4j graph database for results storage and graphical presentation.

For prototype demonstration, we deployed analyser, scheduler and database on a virtual VMware server which used 4 cores of Intel(R) Xeon(R) CPU E5-2680 v2 2.80GHz processor, 8GB RAM and 1TB hard drive. The schedule was

[1]https://neo4j.com
[2]https://github.com/CSIRT-MU/PassiveOSFingerprint
[3]https://www.flowmon.com

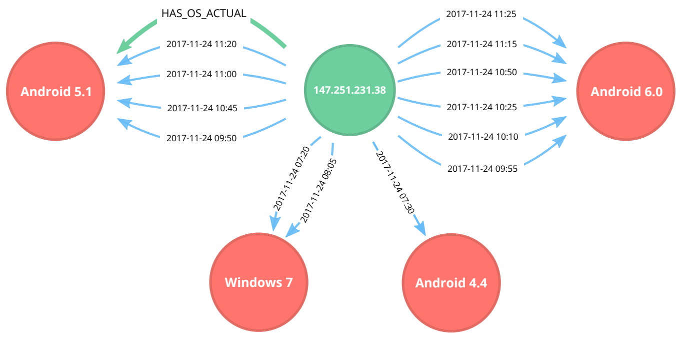

Fig. 2. Screenshot of database filled with data from real traffic

set to 5 minutes typical for flow processing. Even if the server performance parameters are quite low for high-speed monitoring, it was able to do the whole processing cycle from obtaining data to database updates within 2 to 3 minutes. Example of the resulting database is depicted in Fig. 2.

IV. Demonstration Structure

We are going to present the prototype capabilities on live network traffic. First, we will discuss monitoring probes location and its impacts on OS fingerprinting. Then we are going to go through flow data transfer, processing and impacts of time schedule settings. Finally, we plan to demonstrate interaction with Neo4j database and use it to present statistics about operating systems in our network.

V. Conclusion

We introduced a system architecture for continuous passive OS fingerprinting of devices in large dynamic networks. This architecture is designed to deal with the rapid changes in real networks and our prototype demonstration showed it is capable of processing data from 40 GE uplink with more than 22 thousands of concurrently communicating devices.

Acknowledgement

This research was supported by the Security Research Programme of the Czech Republic 2015 - 2020 (BV III / 1 VS) granted by the Ministry of the Interior of the Czech Republic under No. VI20172020070 Research of Tools for Cyber Situation Awareness and Decision Support of CSIRT Teams in the Protection of Critical Infrastructure.

Martin Laštovička is Brno Ph.D. Talent Scholarship Holder – Funded by the Brno City Municipality.

References

[1] M. Lastovicka, T. Jirsik, P. Celeda, S. Spacek, and D. Filakovsky, "Passive OS Fingerprinting Methods in the Jungle of Wireless Networks," in *Network Operations and Management Symposium (NOMS), 2018 IEEE*, 2018, [To appear].

[2] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2037–2064, Fourthquarter 2014.