Analysis of OS fingerprinting with Machine learning

Achintya Kumar, Ishan Soni, Madhur Tatiya, Sparsh Agarwal

National Institute of Technology Karnataka Surathkal, India
{achintya.191it203, ishu.191it121, mowgli.191it227, ishu.191it150}@nitk.edu.in

Abstract. Operating system fingerprinting is used to determine the operating system running on the target system. Generally, in operating system identification, the OS is identified using network packets from a rule-based matching method. However, this method has problems when the network packet information is insufficient, or the OS is relatively new. This study compares the OS fingerprinting capabilities of several machine learning methods, specifically, K-nearest neighbours (KNN), Decision Trees, Artificial Neural Network (ANN), Naive Bayes and Random Forest. Our results show that better OS identification can be achieved through Machine Learning techniques in encrypted traffic of mobile devices and notebooks connected to the wireless network to a conventional commercial rule-based method.

Keywords: We would like to encourage you to list your keywords in this section.

1 Introduction

Today, everyone around is connected to the internet, so the need to be secure from breaches and intrusions is essential. This external threat for businesses triggers them to explore multiple security options like firewalls and intrusion detection systems in order to secure themselves from hackers. The Operating system fingerprinting is a much-needed method for remotely detecting and identifying the identity of a target system by observing the TCP/IP packets generated by that system.

The most widely used method in the market currently identifies the OS using rule-based matching methods. In these methods, rules are created, and the OS is identified by using packet information transmitted and received on a network. It does not require much large amount of data and the identification speed is also fast, unlike the machine learning approach. However, when sufficient information cannot be obtained from the packets collected for identification because of network security settings, policies, newer versions etc., the OS will not be identified, or low identification accuracy will occur.

OS fingerprinting techniques are categorized into two categories, active and passive.

Active fingerprinting works by sending packets to a target and analyzing the packets that are received. Almost all the active fingerprinting these days is done with Nmap. Nmap is generally in use by network admins for security purposes. With

Nmap, they can check to make sure that all of the firewalls in their network are appropriately configured, and they can also make sure that all of the TCP/IP stacks they maintain are functioning correctly.

Passive fingerprinting works by sniffing TCP/IP ports rather than generating network traffic by sending packets. Hence, it is an effective way of avoiding detection. In the passive OS detection, the main focus is on the parameters of the packet headers, which are time to live (TTL), window size (WS), do not fragment bit (DF), and TCP options/flags. The main advantage of passive OS detection for the attackers is that they can detect the remote host without leaving any traces.

In this paper, We will take the approach of Passive Fingerprinting by analyzing TCP network packet header info as well as info from HTTP header using several machine learning methods, such as K-nearest neighbours (K-NN), Decision Trees, Artificial Neural Network (ANN), Naive Bayes and Random Forest.

The rest of the paper is organized as follows. Section II presents the literature survey; Section III explains our Problem Statement; section IV demonstrates our proposed framework, and Section V provide details of the Dataset. In Section VI, the results of our experimental work are analyzed and compared. Finally, the conclusion and future work are discussed in Section VII.

2 Literature Survey

There has been some research in the field of passive and active OS fingerprinting in the last 12-15 years.

Lanze Spitzner in [6] first identified what passive OS fingerprinting was, how it worked and the uses. He also compared passive and active fingerprinting in terms of differences and similarities.

[1] Al-Shehari T, Shahzad F. has proposed machine learning techniques combined with traditional tools to build a system where TCP/IP communication is setup between machines to capture and analyze TCP/IP packets for more accurate OS detection.

Lippmann et al. [5] introduced the idea of near-match fingerprints, they used machine learning classifiers to generate and determine the OS categories that are identifiable using OS fingerprinting. Tyagi et. al. [4] made passive fingerprinting of TCP/IP to detect unauthorized operating systems on private internal networks.

Lastly, Jinho Song et al.[2] compared the OS identification capabilities of several machine learning methods, specifically, K-nearest neighbors (K-NN), Decision Tree,Naive Bayes,Random Forest and Artificial Neural Network (ANN), to that of a conventional commercial rule-based method with 94% probability, which is higher than the accuracy of the conventional rule-based method. They found A-NN to perform best of three when the dataset was large and properly trained. K-NN was second with no bounds to datasize and performed consistently.

2.1 Our Contributions

While TCP/IP and HTTP features were previously used for passive OS fingerprinting, and TLS features have been used to fingerprint many browsers. We describe the first system that integrates all of these data types in a multi-session model to identify the major and minor versions of both Mobile and PC operating systems. We have used Conventional Machine learning algorithms such as K-nearest neighbours (K-NN), Decision Tree and Artificial Neural Network (ANN) from [2] our base paper and introduced some of the newer algorithms such as Random Forest and Bayes algorithm too in our implementation. We have also done an analysis of the probability of correct identification across different operating systems and the role of different parameters involved in the process.

Table 1 Summary of Literature survey

Authors	Methodology	Merits	Limitations
Taher Al-Shehari and Farrukh Shahzad	TCP/IP header packet info for ML and new extended tool	Simple algorithm and use in real life	accuracy is less precise.
Martin Lastovicka and 3 others	Using TLS Fingerprints for OS Identification in Encrypted Traffic	High accuracy as TLS,TCP/IP and HTTP headers are used	Did not utilise more ML models.
Jinho Song, ChaeHo Cho, Yoojae Won	Analysis of OS identification using ML techniques	Employed many ML models for higher accuracy	Not used HTTP and TLS information.

3 Problem Statement

"Using Machine Learning for OS fingerprinting"

A Objectives :

- To suggest a better way for OS identification
- To compare machine learning algorithms with existing OS fingerprinting solutions

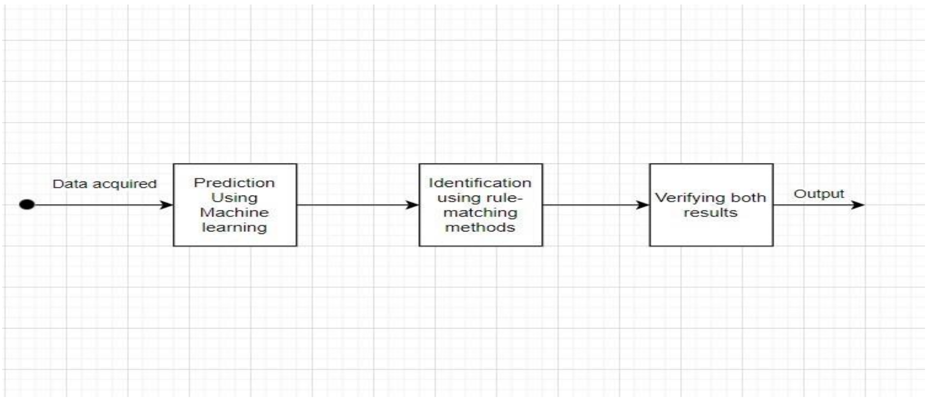
B Solved using :

- Information from TCP/IP, HTTP packets header
- Machine Learning algorithms

4 Proposed Work

As discussed earlier in our Contributions, the project is basically split into 2 sections:

A. Comparison between Conventional and ML approach


Data acquired
Prediction Using Machine learning
Identification using rule-matching methods
Verifying both results
Output

Classifier and Conventional rule-based matching methods.

Figure 1 Proposed OS Prediction mode

While TCP/IP and HTTP features have previously been used for passive OS fingerprinting, and TLS features have been used to fingerprint browsers. We describe the first system that integrates all of these data types in a multi-session model to identify the major and minor versions of both Mobile and PC operating systems
Below architecture shows how OS fingerprinting can be achieved by combining Machine Learning.

We have used Conventional Machine learning algorithms such as K-nearest neighbours (K-NN), Decision Tree and Artificial Neural Network (ANN) from [2] our base paper and introduced some of the newer algorithms such as Random Forest and Bayes algorithm too in our implementation. We have also analyzed the probability of correct identification across different operating systems and the role of different parameters involved in the process.

B. To Determine the best ML classifier and the dominant parameter.

This section deals with two different goals.

- Analyze all the different classifiers to find the most suitable classifier taking size,time taken and other costs in consideration
- Determine which parameter(s) play a major role in prediction and are necessary for fingerprinting The work is to be done in python with the help of external Python libraries that aid in Machine Learning and OS work.

5 Dataset and Model Description

The dataset was acquired from [4] paper where authors had posted the dataset on Zenodo.org. The dataset consists of data from three different sources; flow records collected from the university backbone network, log entries from the two university DHCP (Dynamic Host Configuration Protocol) servers and a single RADIUS (Remote Authentication Dial-In User Service) accounting server.

The dataset is in the form of CSV file with the following information fields important for OS identification:

- Basic flow features
  - Date flow start - timestamp of flow start
  - Date flow end - timestamp of flow end
  - Src IPv4 - source IPv4 address
  - Port - source L4 port
  - Dst IPv4 - destination IPv4 address
  - dPort - destination L4 port
- Extended TCP/IP parameters
  - SYN size - the size of the initial SYN packet of a TCP connection (in bytes)
  - TCP win - value of TCP Window size parameter
  - TCP SYN TTL - observed TTL value
- HTTP parameters
  - HTTP Host - hostname from the HTTP request
  - HTTP UA OS - OS identification based on user-agent
  - HTTP UA OS MAJ - OS identification based on user-agent
  - HTTP UA OS MIN - OS identification based on user-agent
  - HTTP UA OS BLD - OS identification based on user-agent
- Log based extensions
  - Session ID - ID of the session to match flows from one device
  - Ground Truth OS - OS name derived from log data

After careful consideration and testing, we have taken following parameters to train and test the model.

- SYN size
- TCP win
- TCP SYN TTL
- HTTP UA OS
- HTTP UA OS MAJ
- HTTP UA OS MIN
- HTTP UA OS BLD
- Ground Truth OS

5.1 Preprocessing of Datasets

The data consisted of the network packet information as showed above in Dataset Description. The values of each attribute are not uniformly distributed, and they consist of integer numbers. They are converted into decimal numbers, thereby reducing the range. We took 232391 instances of data where approximately 80% of the data is used to train the model, and the rest 20% is used to test it.

5.2 Model Setup

The model has to be set up to learn the model. In this study, the model was configured with one input layer, six hidden layers, and one output layer. For the input values, a total of 7 attribute values were used: SYN size, TCP win, TCP SYN, TTL, HTTP UA OS, HTTP UA OS MAJ, HTTP UA OS MIN, HTTP UA OS BLD.The four outputs are Windows, Linux, Android and MAC.

As shown in the results of the trained model, the loss rate was 0.001%, and the identification accuracy using the test data was 100%. However, to verify the efficiency, the model had to be evaluated with real data.

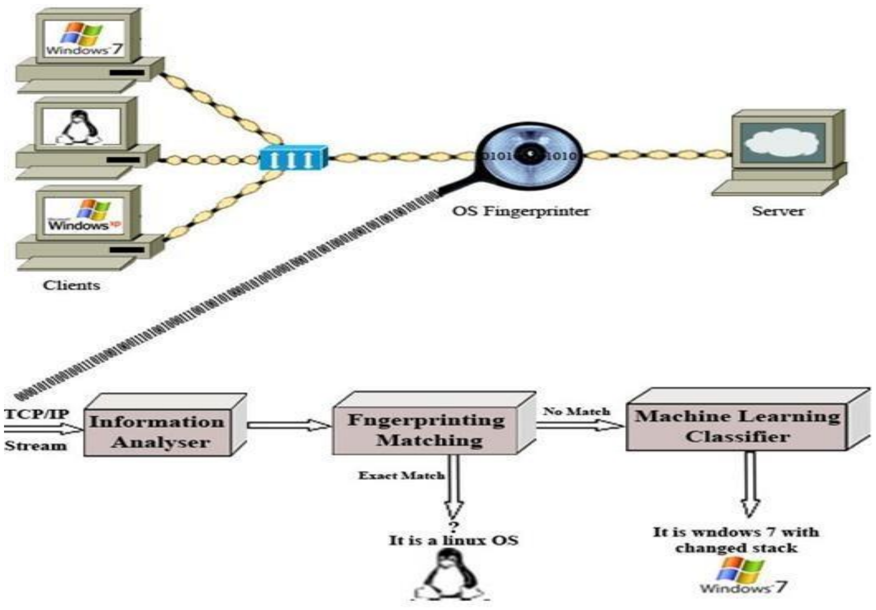


Figure 2 Architecture Proposed

6 Experimentation and Results

This study compared the K-NN, Decision Tree, Random forest, Bayes and ANN algorithms using the standard pieces of data. Below is the implementation for each of the algorithm, along with their advantages and disadvantages.

A. Decision trees

Decision Trees is a method that firstly learns, makes decision rules, and then outputs in a tree-like structure. It has the advantages of not converting data into the decimal type and the relative ease of constructing the model when the depth of the tree is specified. However, since the model changes with depth, accuracy lessens if the model is not set up perfectly. For the Decision Tree, two separate models (Depth = 4, 10) were implemented.

B. Artificial Neural Networks

ANNs are described as machine learning algorithms that are designed to acquire knowledge using patterns from the data. As shown in Fig. 3, it is divided into an input layer, a hidden layer, and an output layer. There is no limit on their numbers. An ANN divides the input dataset into training and test set. It has the advantages of high accuracy when there is a large amount of data and the capability to perform additional learning contrary to other machine learning algorithms. Overfitting is a problem when the dataset is not too large.

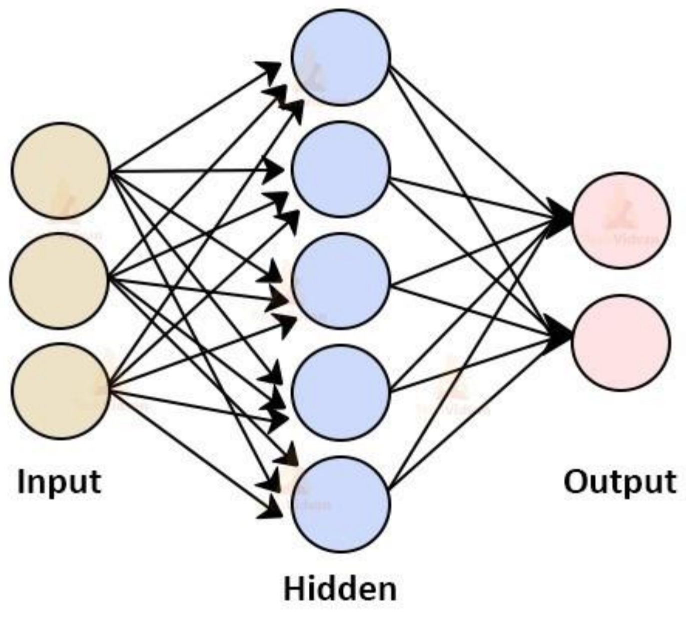


Figure 3 ANN Architecture

C. K Nearest Neighbours

K-NN is a machine learning method that uses the closest neighbours' info by measuring separation to previous data when novel data is entered. The distance is calculated using the Euclidean calculation method. The K-NN algorithm shines when it requires fast processing speed, and data to compare is not significant because learning is not required at all and ease of use. However, the drawback is that when the size of data is not small, the processing speed declines rapidly. For the K-NN model, three separate learning models K = 10, 20, 30 variations were implemented

D. Random Forest

Random Forest is based upon ensemble learning, which combines multiple classifiers to solve a complex problem and slightly improves performance. More specifically, Random Forest is a classifier that employs a number of decision trees and takes its average to improve the accuracy of the dataset. Although it combines the Decision trees, it takes a considerably big time and is not particularly good for OS that is rare to be seen.

E. Naive Bayes

Na¨ıve Bayes algorithm is based on the Bayes Theorem of probability and statistics. It is simple and yet one of the powerful ML algorithms today and is categorised as a probabilistic classifier. One of the reasons for this is that it assumes one feature in a class does not affect the other one. The Drawback of Bayes is that it does not relate all the parameter together and treats everything independent, which can prove results to be unpredictable at times.

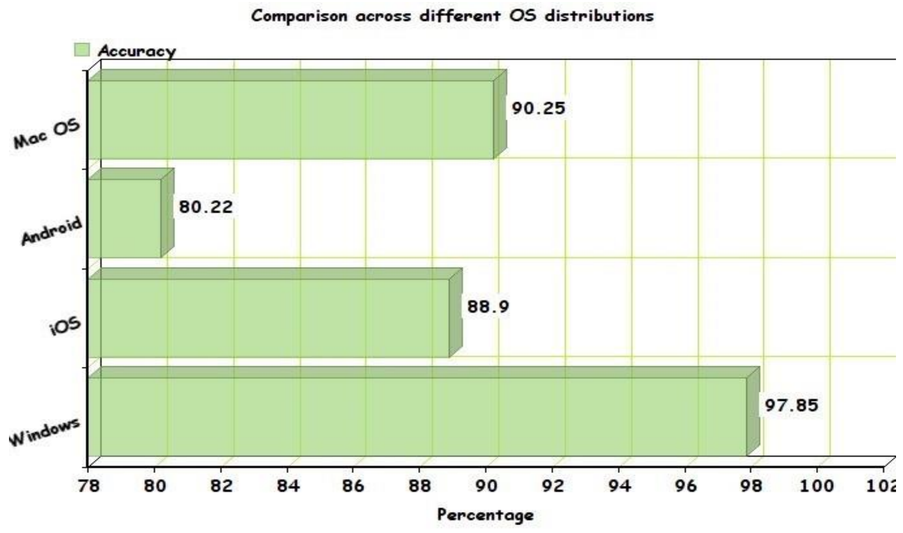


Figure 4 Comparison between different OS accuracy

F. Comparison with existing tools

We compared the KNN, which boasts of the highest accuracy, with some of the currently used commercial rule-based matching program (p0f) widely used in the market. For the comparison, Wireshark was used to extract the relevant data from the data collected in p0f, and preprocessing was performed to input them into the model. The unrecognized data also belong to the four types of operating systems (Windows, Android, Linux and MAC) used in this study. The data applied to the KNN learning model comprised 1956 identified data samples and 490 unidentified samples from p0f. The anonymous data were collected using the OS identification tool NetworkMiner. When the KNN model was applied, the accuracy was 95% probability, and in the case of unrecognizable OS, the OS was identified with 72% probability. Among 1956, p0f showed 52% precision, and the ANN model identified the OS with 79% probability.

G. Results

Here, In fig 5, We can see different models compared using different metrics such as precision, accuracy etc. Here, X-axis denotes the percentage and Y-axis the algorithm used. F1- score is scaled to 100 for percentage depiction purposes. KNN appeared as the best ML model for OS identification. Random Forest Classifier also performed well and overtook KNN when the dataset was huge. We also compared the accuracy of prediction between operating systems. We specially Trained the data for specific Ground Truth OS instances.

In fig 4. We can see the chart comparing the accuracy of predicting different operating systems. We also removed a parameter and retrained our model the parameter whose absence showed significant decline was termed as the most influencing parameter, here in our case; it is TCP SYN TTL [shows a decline of 23.65%].

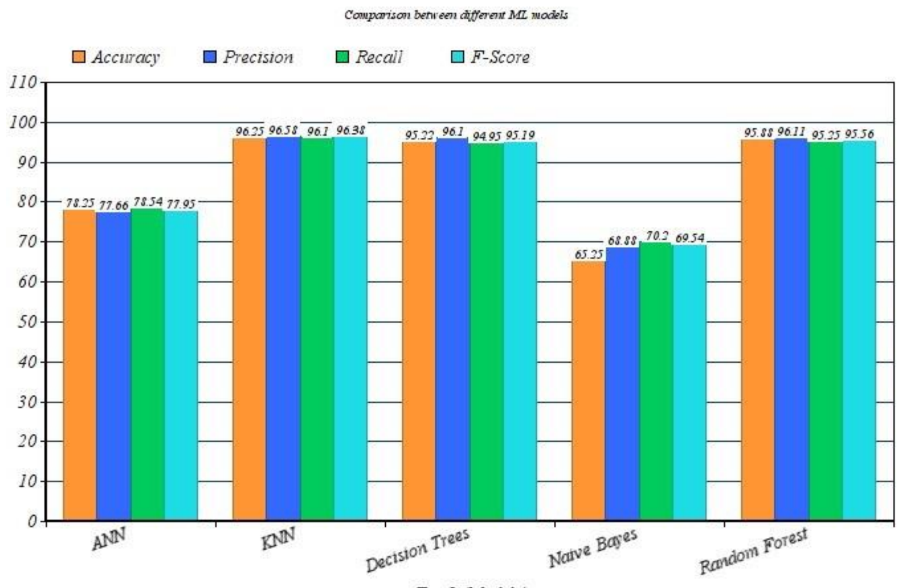


Fig 5 (Models)

7 CONCLUSION

In this study, using machine learning and OS attribute values (SYN size, TCP win, TCP SYN TTL, HTTP UA OS, HTTP UA OS MAJ, HTTP UA OS MIN, HTTP UA OS BLD), operating systems were correctly identified with more than 96% probability, which proved higher than that of a conventional rule-based method.

However, because of individual OS versions using identical attribute values, the versions could not be explicitly classified. Furthermore, when the operating systems are relatively new, as with the rule-based method, the machine learning model could not identify the OS with a very high level of accuracy. To solve this problem, a follow-up study is required to determine if the machine learning-based program is better than the programs already existing in the market by comparing them with programs such as SinFP3, p0f and Nmap with real-life data.

As we had limited parameters required for these type of programs, we had trouble analysing similar data and had to use our own systems for the task.

8 References

[1]Taher Al-Shehari and Farrukh Shahzad , Improving Operating System Fingerprinting using Machine Learning Techniques LATEX, International Journal of Computer Theory and Engineering, Vol. 6, King Fahd university of petroleum and minerals, Saudi Arabia

[2] Jinho Song, ChaeHo Cho, Yoojae Won, Computers and Electrical Engineering 78 (2019)LATEX, Chungnam National University,Korea .

[3] Blake Anderson and David McGrew, OS Fingerprinting: New Techniques and a Study of Information Gain and ObfuscationLATEX,2017 IEEE Conference on Communications and Network Security (CNS) Cisco Systems .

[4] R. Lippmann, D. Fried, K. Piwowarski, and W. Streilein, Passive Operating System Identification from TCP/IP Packet HeadersLATEX IEEE Workshop on Data Mining for Computer Security (DMSEC), 2003, pp.40–49 .

[5] L. Spitzner,Passive fingerprintingLATEX , vol. 3, pp. 1–4, May.

[6] Gregory Conti and Kulsoom Abdullah. Passive Visual Fingerprinting of Network Attack Tools. In ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC), pages 45–54, 2004.

[7] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), 2008.

[8] Zakir Durumeric, Zane Ma, Drew Springall, Richard Barnes, Nick Sullivan, Elie Bursztein, Michael Bailey, J Alex Halderman, and Vern Paxson. The security impact of https interception. In Network and Distributed Systems Symposium (NDSS17), 2017.

[9] Charles Elkan. The Foundations of Cost-Sensitive Learning. In International Joint Conference on Artificial Intelligence (IJCAI), pages 973–978, 2001.

[10] David Formby, Preethi Srinivasan, Andrew Leonard, Jonathan Rogers, and Raheem A. Beyah. Who's in control of your control system? device fingerprinting for cyber-physical systems. In 23nd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016. The Internet Society, 2016.

[11] Stephan Friedl, Andrei Popov, Adam Langley, and Emile Stephan. Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension. RFC 7301 (Proposed Standard), 2014.

[12] Daniel Gillmor. Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS). RFC 7919 (Proposed Standard), 2016.

[13] Lloyd Greenwald and Tavaris Thomas. Toward Undetected Operating System Fingerprinting. In USENIX Workshop on Offensive Technolo gies (WOOT), pages 1–10, 2007