Computers and Electrical Engineering 78 (2019) 1–10

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Analysis of operating system identification via fingerprinting and machine learning[☆]

Jinho Song, ChaeHo Cho, Yoojae Won[*]

Department of Computer Science Engineering, Chungnam National University, Daejeon, South Korea

ARTICLE INFO

Article history:
Received 28 September 2018
Revised 10 June 2019
Accepted 17 June 2019

Keywords:
Operating system fingerprinting
Machine learning
Artificial Neural Network
NetworkMiner
K-nearest Neighbors
Decision Tree

ABSTRACT

In operating system (OS) fingerprinting, the OS is identified using network packets and a rule-based matching method. However, this matching method has problems when the network packet information is insufficient or the OS is relatively new. This study compares the OS identification capabilities of several machine learning methods, specifically, K-nearest neighbors (K-NN), Decision Tree, and Artificial Neural Network (ANN), to that of a conventional commercial rule-based method. It is shown that the ANN correctly identifies operating systems with 94% probability, which is higher than the accuracy of the conventional rule-based method.

1. Introduction

Concomitant with the increasing pervasiveness of the Internet, a variety of Internet of Things (IoT) products, such as mobile devices, smart-home devices, and wearable computers, are being released into the market and, accordingly, the vulnerability of products is increasing. For instance, routers are being hacked and used for distributed denial of service (DDoS) attacks, and devices such as webcams and closed-circuit television (CCTV) are being infected with malware [1]. Various studies have been conducted with the objective of rectifying these issues. For example, various methods such as malicious code detection using data mining techniques [2], infection path tracing using malicious code infected systems [3], network forensic methods using domain name system (DNS) [4], and malicious code detection methods using machine learning [5] have been proposed.

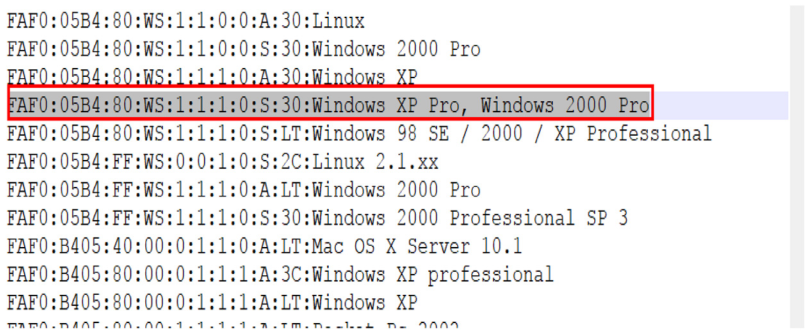
Although such studies are underway, one of the most critical issues at present is analysis and removal of any vulnerabilities in a product prior to market release. The process of searching for vulnerabilities in a system is usually performed remotely through a network. In order to identify such vulnerabilities, the operating system (OS) of the device must first be identified and device type confirmed. Identification of the OS facilitates application of countermeasures to known OS vulnerabilities, and security threats that can occur according to the operating purpose and environment of the device can be predicted.

Various OS identification techniques have been proposed. Taleck [6] presented a passive OS detection method that uses information coming from the Intrusion Detection System (IDS) environment. Wei-hua et al. [7] identified the OS by using

[☆] This paper is for CAEE special section SI-aisec. Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. James Park.

[*] Corresponding author.
E-mail addresses: jhsong91@cnu.ac.kr (J. Song), greatopen@cnu.ac.kr (C. Cho), yjwon@cnu.ac.kr (Y. Won).

https://doi.org/10.1016/j.compeleceng.2019.06.012
0045-7906/© 2019 Published by Elsevier Ltd.



```
FAF0:05B4:80:WS:1:1:0:0:A:30:Linux
FAF0:05B4:80:WS:1:1:0:0:S:30:Windows 2000 Pro
FAF0:05B4:80:WS:1:1:1:0:A:30:Windows XP
FAF0:05B4:80:WS:1:1:1:0:S:30:Windows XP Pro, Windows 2000 Pro
FAF0:05B4:80:WS:1:1:1:0:S:LT:Windows 98 SE / 2000 / XP Professional
FAF0:05B4:FF:WS:0:0:1:0:S:2C:Linux 2.1.xx
FAF0:05B4:FF:WS:1:1:1:0:A:LT:Windows 2000 Pro
FAF0:05B4:FF:WS:1:1:1:0:S:30:Windows 2000 Professional SP 3
FAF0:B405:40:00:0:1:1:0:A:LT:Mac OS X Server 10.1
FAF0:B405:80:00:0:1:1:1:A:3C:Windows XP professional
FAF0:B405:80:00:0:1:1:1:A:LT:Windows XP
```

Fig. 1. Existing personal information detection method.

the Internet Control Message Protocol (ICMP), while a number of other studies have used machine learning [8–10]. Meidan et al. [11] proposed an OS detection method that uses other protocols in addition to User Datagram Protocol, Transmission Control Protocol (TCP), and ICMP.

The most widely used method in the market currently identifies the OS using rule-based matching. In the rule-based matching method, rules are created and the OS is identified by using packet information transmitted and received on a network. It does not require large amounts of data, unlike machine learning, and the identification speed is fast. However, when sufficient OS information cannot be obtained from the packets collected because of network security settings and policies (firewall, IDS, etc.), the OS will not be identified or low identification accuracy will occur. This paper proposes a method that uses machine learning to overcome the problems of existing rule-based matching and compares the identification speed and accuracy with those of machine learning algorithms (K-NN, Decision Tree, and ANN) used in [8–10].

The remainder of this paper is organized as follows. Section 2 explains the concept of OS identification and discusses the programs, algorithms, and data information currently used. Furthermore, previous studies that used machine learning are also discussed. Section 3 outlines the pros and cons of using machine learning-based programs. Section 4 describes the data and preprocessing procedure used in the proposed method. Section 5 compares the results of the proposed method to that of previously studied algorithms (K-NN and Decision Tree) and presents concluding remarks and outlines future work.

2. Related work

OS fingerprinting identifies the OS of a device by using network information (TTL, MMS, window size, etc.) transmitted and received by the target device [12]. The currently released programs typically use a matching method that analyzes the OS and creates rules based on the transmitted/received packets, as shown in Fig. 1.

The currently available programs include NetworkMiner, Ettercap, p0f, Nmap, and Xprobe2. This paper describes NetworkMiner briefly and additionally explains the rule-based matching method.

2.1. NetworkMiner and rule-based matching

NetworkMiner is a network forensic analysis tool used in the Windows environment; its program window is shown in Fig. 2. It performs the analysis using a PCAP file and has a function that creates a file by reconstructing the transmitted file. Moreover, it uses two rule-based matching methods—Similarity Matching (SM) and Exact Matching (EM)—and identifies the OS based on a built-in database.

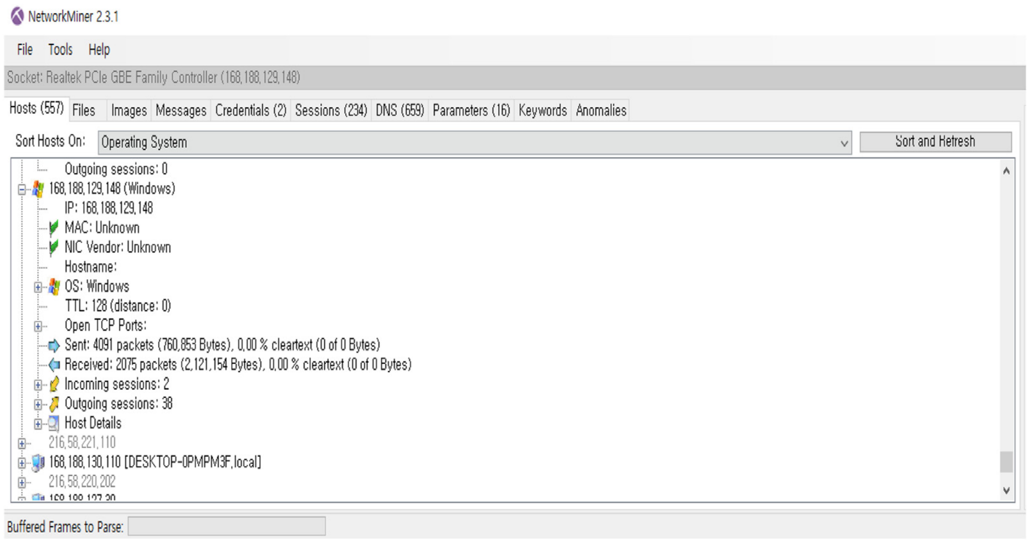
The SM method compares the field and identification information, as shown in Fig. 3, and determines the most likely OS based on the input information [13].

The EM method identifies an OS based on continuously matching data results, as shown in Fig. 4 [14]. This rule-based matching method has several problems. First, when the input information is insufficient or inaccurate, the OS is not easily identified. Second, when the input information is not found in the database, the OS cannot be identified. Finally, it requires a packet analysis process and rules in the database to apply to a new OS.

2.2. Studies based on machine learning

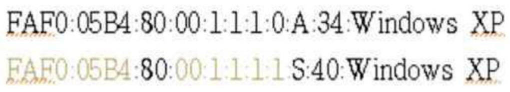
To solve the problems besetting the matching method, several studies have been conducted with the objective of identifying the OS using machine learning. Beverly [8] proposed a Bayesian algorithm-based method. Alese and Oduwale [9] investigated OS identification using an ANN. Al-Shehari and Shahzad [10] utilized Decision Trees with C4.5 algorithms.

Although Beverly [8] confirmed that the OS could be identified by using machine learning, it was concluded that the identification accuracy was similar to that of the rule-based method. Further, although Alese and Oduwale [9] proved that the OS could be identified using an ANN, they did not include comparisons with rule-based methods or other algorithms.


NetworkMiner 2.3.1
File Tools Help
Hosts (557) Files Images Messages Credentials (2) Sessions (234) DNS (659) Parameters (16) Keywords Anomalies
Sort Hosts On: Operating System
Sort and Refresh
Buffered Frames to Parse:

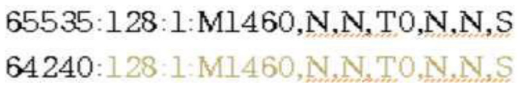
```
Socket: Realtek PCIe GBE Family Controller (168.188.129.148)
  Outgoing sessions: 0
168.188.129.148 (Windows)
  IP: 168.188.129.148
  MAC: Unknown
  NIC Vendor: Unknown
  Hostname:
  OS: Windows
  TTL: 128 (distance: 0)
  Open TCP Ports:
  Sent: 4091 packets (760,853 Bytes), 0.00 % cleartext (0 of 0 Bytes)
  Received: 2075 packets (2,121,154 Bytes), 0.00 % cleartext (0 of 0 Bytes)
  Incoming sessions: 2
  Outgoing sessions: 38
  Host Details
216.58.221.110
168.188.130.110 [DESKTOP-0PMPM3F.local]
216.58.220.202
```

Fig. 2. NetworkMiner program window.



```
FAF0:05B4:80:00:1:1:1:0:A:34:Windows XP
FAF0:05B4:80:00:1:1:1:1:S:40:Windows XP
```

Fig. 3. Comparison of field and identification information in the SM method.



```
65535:128:1:M1460,N,N,T0,N,N,S
64240:128:1:M1460,N,N,T0,N,N,S
```

Fig. 4. Matching data results in the EM method.

Table 1
Network packets used in OS Fingerprinting.

Fields	
IP	Port
IPID	Timestamp
Time to live	Total length
Maximum Segment Size	Do not fragment
Selective Acknowledgement	No operation
Window size	Window scale

In the case of Al-Shehari and Shahzad [10], although the identification accuracy was improved by using the Decision Tree algorithm, the comparison result with the rule-based method could not be confirmed.

This paper compares the processing speed and identification accuracy of the four methods presented above: the conventional rule-based method, the previously studied Decision Tree and ANN algorithms, and K-NN, i.e., a machine learning algorithm.

3. Network packet information and algorithm introduction (K-NN, Decision Tree, and ANN)

Conventional programs currently use TCP/IP packet data to identify an OS, as shown in Table 1. Although IP and Timestamp are used when collecting unique device information, they are usually excluded in most cases when performing OS identification. This is because, although IP is a fixed value, it is difficult to use as comparative analysis information when the device is changed; moreover, Timestamp cannot be used in identification because it simply signifies the time at which the packet was generated. Therefore, 10 out of 12 pieces of data are used. This study also compared the rule-based method, K-NN, Decision Tree, and ANN using the ten pieces of data.

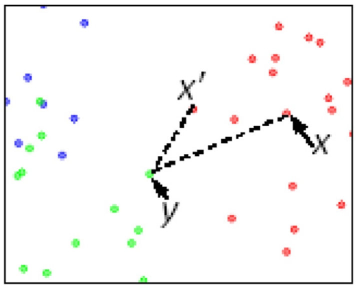


Fig. 5. K-nearest neighbors (K-NN).

$$
\begin{aligned}
X &= (x_1, x_2, \ldots, x_n)\\
Y &= (y_1, y_2, \ldots, y_n)\\
d_{(X,Y)} &= \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}\\
&= \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}
\end{aligned}
$$

Fig. 6. Euclidean distance expression.

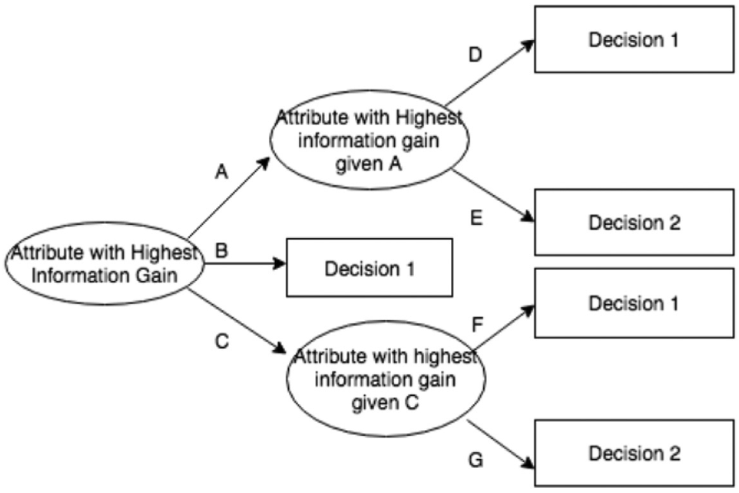

Attribute with Highest Information Gain
A
B
C
Attribute with Highest information gain given A
Decision 1
D
Decision 1
E
Decision 2
Attribute with highest information gain given C
F
Decision 1
G
Decision 2

Fig. 7. Decision Tree results structure.

3.1. K-nearest neighbors algorithm (K-NN)

K-NN is an estimation method that uses the k closest neighbors' information by calculating distances to previously possessed data when new data are inputted, as shown in Fig. 5 [15]. Therefore, the result varies depending on the distance measurement method. The distance calculation algorithms usually use the Euclidean calculation method or the Mahalanobis calculation method.

In this paper, the distances are measured by using the Euclidean distance calculation equations presented in Fig. 6. The advantages of the K-NN algorithm are fast processing speed when the amount of data to compare is small, because learning is not required, and ease of use. However, it has the drawback that when the amount of data is large, the processing speed declines rapidly.

3.2. Decision Tree

Decision Tree is a method that learns data, makes decision rules, and then displays the result in a tree-like structure.

A Decision Tree uses a top-down approach to learn the data and, as shown in Fig. 7, it builds a tree-like structure to find standards using the variables of the given dataset in each stage. The Decision Tree has the advantages of not having to

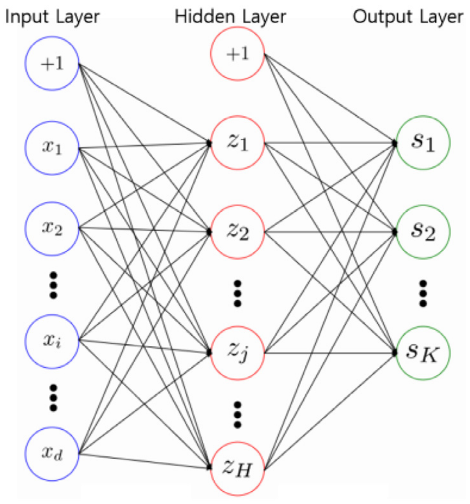


Fig. 8. Artificial Neural Network Structure.

Table 2
Advantages and disadvantages of the compared models.

Model	Advantages	Disadvantages
K-NN	Fast processing speed and learning not required.	When the amount of data is large, the process speed declines.
Decision Tree	Does not require rendering of learning data and is easily utilized.	When learning is improperly set up, accuracy suffers.
ANN(Artificial Neural Network)	Accuracy is high when there is much data.	When the amount of data is small, the accuracy rate can suffer. Overfitting can occur.

convert the learning data into numerical type, and relative ease constructing the learning model when the depth of the tree is specified. However, because the learning model changes according to the depth, accuracy declines when the model is not properly set up.

3.3. Artificial Neural Network (ANN)

An ANN is an algorithm composed of artificial neurons that are based on the neurons in the human body. As shown in Fig. 8, it is divided into input layer, hidden layer, and output layer. Moreover, it is a machine learning algorithm and uses an error back-propagation algorithm to perform learning. An ANN divides the given dataset into training set and test set and has the advantages of high accuracy when there is a large amount of data and, unlike other algorithms, capability to perform additional learning. However, it can have low accuracy when the amount of data is small and, in contrast to K-NN and Decision Tree, overfitting can occur. The advantages and disadvantages of the algorithms are summarized in Table 2.

4. Development of OS fingerprint using machine learning and network packets

4.1. Dataset collection

For the packets, the data were collected with the assistance of KISA (Korea Internet & Security Agency). In this study, approximately 50,000 data samples were acquired. The configured OS information is shown in Table 3.

To use the data in machine learning, preprocessing was first performed, in which the data were divided into training and test datasets. The training data were used to train the model, and the test data were used to evaluate the trained model. Learning was performed by allocating 90% and 10% of the data to the training and test datasets, respectively.

Table 3
List of operating systems.

Operating system	Count
Windows	20,878
AIX	15,213
Linux	10,371
MAC	5012

Table 4
Data from other operating systems.

	Operating system	
Attributes	Windows	AIX
Port	80	80
IPID	8641	31,189
Total length	44	44
Time to live	113	41
Do not fragment	1	0
Maximum segment size	1360	512
No operation	0	0
Selective acknowledgement	0	0
Window scale	0	0
Window size	8192	16,384

Table 5
Data from the same operating systems.

	Operating system	
Attributes	AIX 4.1	AIX 4.3
Port	80	80
IPID	40,589	31,189
Total length	44	44
Time to live	41	41
Do not fragment	0	0
Maximum Segment Size	512	512
No operation	0	0
Selective Acknowledgement	0	0
Window scale	0	0
Window size	16,384	16,384

Table 6
Model training result.

Model	Accuracy	Precision
ANN	95%	94%

4.2. Dataset description

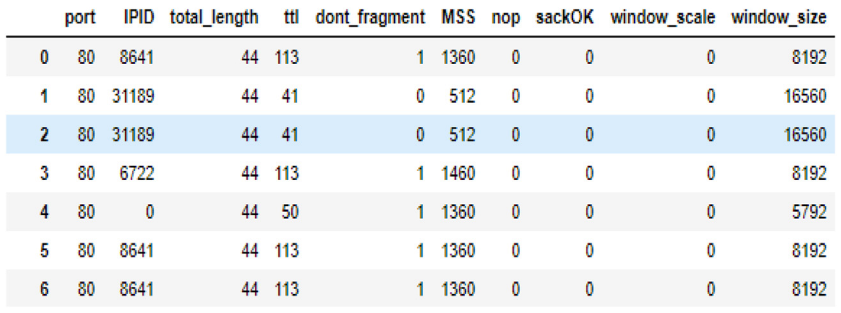
As shown in Table 1, the dataset is composed of ten attribute values among the 12 attribute values, excluding the IP and Timestamp, and the OS result value. The OSes are Windows, AIX, Linux, and MAC. In the case of Windows OS, because the versions consist of Windows XP, 7, 8.1, and 10, an attempt was made to classify them, but several problems occurred.

4.2.1. Data issues

Table 4 compares Windows and AIX.

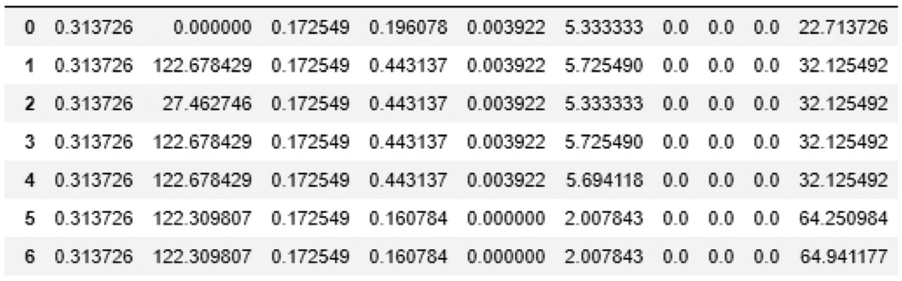
Table 4 confirms that different operating systems can be differentiated because their IPID, TTL, Do not Fragment, MMS, and Window size differ. Conversely, with operating systems that are the same (but different versions) problems occur, as shown in Table 5.

In the case of the same OS, it is confirmed that the values of attributes excluding IPID are consistent. Further, in Windows and Linux, because the attribute values are similar between different versions, although many attribute values exist, the OS can be distinguished, however it is difficult to determine specific versions. To classify the versions, more attribute values should be added. In this study, however, as the goal was simply to identify the OS, the preprocessing was performed only to classify each OS Table 6.



	port	IPID	total_length	ttl	dont_fragment	MSS	nop	sackOK	window_scale	window_size
0	80	8641	44	113	1	1360	0	0	0	8192
1	80	31189	44	41	0	512	0	0	0	16560
2	80	31189	44	41	0	512	0	0	0	16560
3	80	6722	44	113	1	1460	0	0	0	8192
4	80	0	44	50	1	1360	0	0	0	5792
5	80	8641	44	113	1	1360	0	0	0	8192
6	80	8641	44	113	1	1360	0	0	0	8192

Fig. 9. Network packet information.



0	0.313726	0.000000	0.172549	0.196078	0.003922	5.333333	0.0	0.0	0.0	22.713726
1	0.313726	122.678429	0.172549	0.443137	0.003922	5.725490	0.0	0.0	0.0	32.125492
2	0.313726	27.462746	0.172549	0.443137	0.003922	5.333333	0.0	0.0	0.0	32.125492
3	0.313726	122.678429	0.172549	0.443137	0.003922	5.725490	0.0	0.0	0.0	32.125492
4	0.313726	122.678429	0.172549	0.443137	0.003922	5.694118	0.0	0.0	0.0	32.125492
5	0.313726	122.309807	0.172549	0.160784	0.000000	2.007843	0.0	0.0	0.0	64.250984
6	0.313726	122.309807	0.172549	0.160784	0.000000	2.007843	0.0	0.0	0.0	64.941177

Fig. 10. The converted information.

4.3. Model learning method

To create a learning model, five procedures (classification of training and test datasets, data preprocessing, model configuration, model learning process setup, and model learning) were carried out. The data were divided into training set (90%) for learning the model and test set (10%). In data preprocessing, the data are converted to facilitate model learning, and the model settings include a loss function and a slope descent algorithm.

4.3.1. Preprocessing of the dataset

The data consisted of the network packet information shown in Fig. 9.

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, as shown in Fig. 10.

4.3.2. Model configuration and model learning process setup

The model has to be set up to learn the model. In this study, the model was configured with one input layer, four hidden layers, and one output layer. For the input values, a total of ten attribute values were used: Port, IPID, total length, time to live (TTL), do not fragment, Maximum Segment Size (MSS), No operation, Selective acknowledgement, Window scale, and Window size. The four outputs are Windows, AIX, Linux, and MAC. The configuration of the model is shown in Fig. 11.

The activation function of Dense 1–4 used ReLU, and in the last layer (Dense 5), SoftMax was used. For Dense 1, as the number of attribute values used was 10, ten inputs and ten outputs were set up; in the hidden layers, Dense 1–4, 128 nodes were set up, respectively to configure the model.

For the loss function and the gradient descent method, cross entropy and stochastic gradient descent were used.

4.3.3. Model learning results

As shown in Fig. 12, 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 efficacy of the model, it had to be evaluated with real data.

4.3.4. Model prediction performance evaluation

Accuracy and precision are used as evaluation indices for the performance estimation of a model. Accuracy is an index regarding the probability percentage at which the OS is identified, whereas precision is an index that is evaluated using the test data. The accuracy is calculated by using the real value and the predicted value, as shown in Fig. 13.

Precision is calculated by dividing the number of correctly predicted results (True Positives (Tp)) by the total number of results (True Positives (Tp) + False Positives (Fp)), as shown in Fig. 14.

In the preprocessing stage, 1000 transmitted/received data (Windows 60%, Linux 30%, MAC 7%, and AIX 3%) were newly collected and applied to the learning model. Eventually, the different operating systems were identified with an accuracy of 95% and precision of 94%.

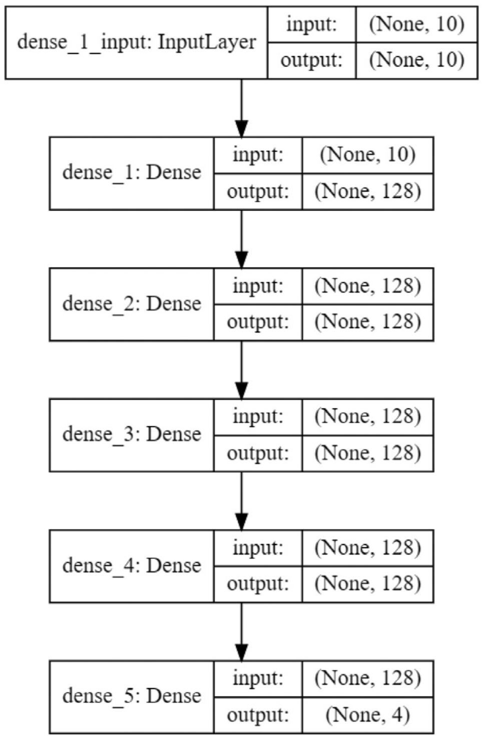


Fig. 11. Model configuration.

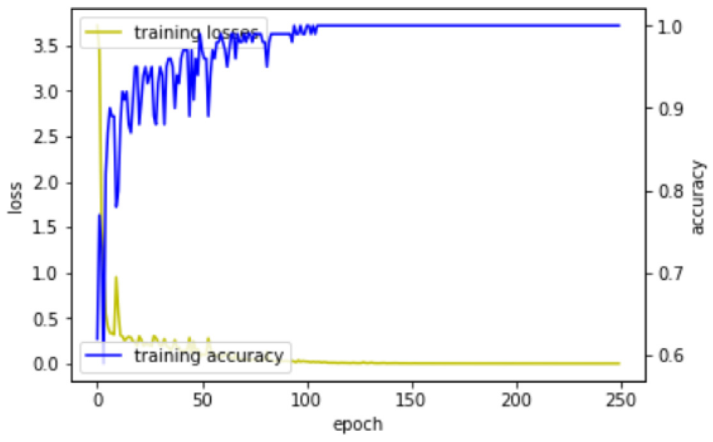


Fig. 12. Learning model evaluation.

$$Accuracy(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y}_i = y_i)$$

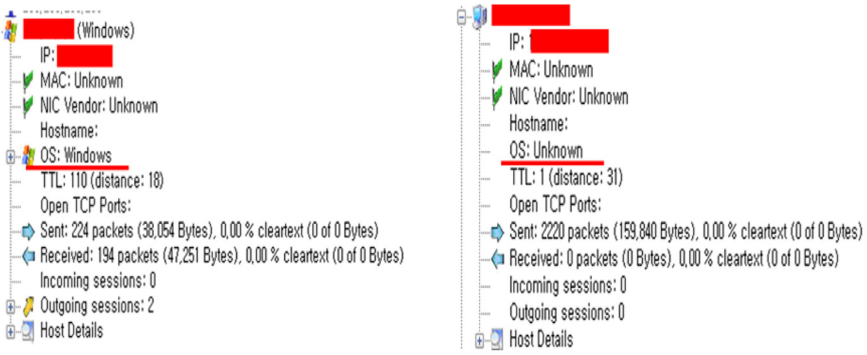
Fig. 13. Accuracy evaluation expression.

$$\mathrm{P} = \frac{T_p}{T_p + F_p}$$

Fig. 14. Precision evaluation expression’.

Table 7
Results from the various models.

Model	Learning accuracy	Test accuracy	Processing time (min)	Learning time (min)
K-NN	–	87%	1.68	–
Decision Tree	93%	90%	0.20	10
ANN	95%	94%	0.03	15



```
[redacted] (Windows)
IP: [redacted]
MAC: Unknown
NIC Vendor: Unknown
Hostname:
OS: Windows
TTL: 110 (distance: 18)
Open TCP Ports:
Sent: 224 packets (38,054 Bytes), 0,00 % cleartext (0 of 0 Bytes)
Received: 194 packets (47,251 Bytes), 0,00 % cleartext (0 of 0 Bytes)
Incoming sessions: 0
Outgoing sessions: 2
Host Details
```

```
[redacted]
IP: [redacted]
MAC: Unknown
NIC Vendor: Unknown
Hostname:
OS: Unknown
TTL: 1 (distance: 31)
Open TCP Ports:
Sent: 2220 packets (159,840 Bytes), 0,00 % cleartext (0 of 0 Bytes)
Received: 0 packets (0 Bytes), 0,00 % cleartext (0 of 0 Bytes)
Incoming sessions: 0
Outgoing sessions: 0
Host Details
```

Fig. 15. Unrecognized operating systems in NetworkMiner.

Table 8
Recognition result rate.

	Recognizable data	Unrecognizable data	Precision
NetworkMiner	–	–	41%
ANN Model	90%	68%	77%

4.4. Application of different algorithms

The learning process and evaluation for ANN were examined, and the K-NN and Decision Tree models, explained earlier, were implemented and compared with the ANN.

For the K-NN model, three separate learning models ($K = 10$, 20, 30) were implemented, and for the Decision Tree, two separate models (Depth = 4, 10) were implemented. The same preprocessing data applied to the ANN were used.

Table 7 confirms that the ANN had the highest accuracy among the models. Although the Decision Tree model had a high accuracy as well, it showed a large difference from that of the ANN. The K-NN had 87% accuracy with a long processing time. The Decision Tree had a shorter learning time than the ANN, but it was less accurate. The ANN had the longest learning time, but processing speed was faster and accuracy higher than the other models.

4.5. Comparison with the OS fingerprint tool

We compared the ANN, which had the highest accuracy, with the currently used commercial rule-based matching program NetworkMiner from among the programs that have been released in the market. For the comparison, Wireshark was used to extract the relevant data from the data collected in NetworkMiner, and preprocessing was performed to input them into the model. Fig. 15 shows the results of unrecognized and recognized operating systems in NetworkMiner. The unrecognized data also belong to the four types of operating systems (Windows, Linux, AIX, and MAC) used in this study. The data applied to the ANN learning model comprised 446 identified data samples and 620 unidentified samples from NetworkMiner. The unidentified data were collected using the OS identification tools Nmap, SinFP3, and Shodan.

Table 8 shows the comparison results. When the ANN model was applied, the accuracy was 90% probability, and in the case of unrecognizable OS, the OS was identified with 68% probability. Among the total of 1066, NetworkMiner showed 41% precision and the ANN model identified the OS with 77% probability.

5. Conclusions

In this study, using machine learning and OS attribute values (Port, IPID, TTL, Do not Fragment, MSS, No operation, Selective Acknowledgement, Window scale, and Window size), operating systems were correctly identified with 94% probability, which proved higher than that of a conventional rule-based method. However, because of the problem of individual OS versions using identical attribute values, the versions could not be classified specifically. Furthermore, in the case of relatively new operating systems, as with the rule-based method, the machine learning model could not identify the OS with a high level of accuracy, although it was configured in this study. To solve this problem, a follow-up study is required for classification of operating systems by finding a new attribute value or using a clustering algorithm. In addition, an additional study should be conducted 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 and Nmap, as well as NetworkMiner.

Acknowledgments

This work was supported by the research fund of Chungnam National University.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compeleceng.2019.06.012.

References

[1] Gartner, http://www.gartner.com/newsroom/id/3165317, Oct 2016 [accessed 26 February 2019].
[2] Alireza S, Rahil H. A state-of-the-art survey of malware detection approaches using data mining techniques. Human Centric Compute Inf Sci 2018;8(1):3.
[3] Jung WS, Sang JL. A study on efficient detection of network-based IP spoofing DDoS and malware-infected systems. Springerplus 2016;5(1):1878.
[4] Shulman H, Waidner M. DNSSEC for cyber forensics. EURASIP J Inf Secure 2014;2014(1):16 Springer.
[5] Tuvell G, Venugopal D, Hu G. Malware modeling detection system and method for mobile platforms. Juniper Networks Inc., April 2006. United States patent US 8,321,941. 2012 Nov 27.
[6] Taleck G. Ambiguity resolution via passive OS fingerprinting. In: Vigna G, Kruegel C, Jonsson E, editors. Recent advances in intrusion Detection. RAID 2003. Lecture notes in computer science. Berlin, Heidelberg: Springer; 2003 Sep 8. p. 192–206.
[7] Wei-hua J, Wei-hua L, Jun D. The application of ICMP protocol in network scanning. In: Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies. IEEE; 2003 Aug 27. p. 904–6. doi:10.1109/PDCAT.2003.1236446.
[8] Beverly R. A robust classifier for passive TCP/IP fingerprinting. In: International workshop on passive and active network measurement. Berlin, Heidelberg: Springer; 2004 Apr 19. p. 158–67.
[9] Alse BK, Oduwale MA. Remote operating system identification using artificial neural networks. J Multidiscip Eng Sci Technol 2017;4(10):8310–13.
[10] Al-Shehari T, Shahzad F. Improving operating system fingerprinting using machine learning techniques. Int J Comput Theory Eng 2014;6(1):57.
[11] Meidan Y, Bohadana M, Shabtai A, Guarnizo JD, Ochoa M, Tippenhauer NO, Elovici Y. ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis. In: Proceedings of the symposium on applied computing. ACM; 2017 Apr 3. p. 506–9.
[12] Steve H, Ashley M. OS Fingerprinting Techniques and Tools, Cryptography and Network Security. Keene State College, CS-455, November 2013
[13] Santini S, Jain R. Similarity matching. In: Asian conference on computer vision. Berlin, Heidelberg: Springer; 1995 Dec 5. p. 571–80.
[14] Blackwell M, Iacus S, King G, Porro G. cem: coarsened exact matching in stata. Stata J 2009;9(4):524–46.
[15] Leif EP. K-nearest neighbor. Scholarpedia 2009;4:1883.

Jinho Song is currently a master's student in the Department of Computer Science and Engineering at ChungNam University, Korea. His main research interest is network security using machine learning.

Chaeho Cho is a Ph.D. candidate in the Department of Computer Science and Engineering at ChungNam University, Korea. His main research interests are network security using machine learning and digital forensics.

Yoojae Won is a Professor in the Department of Computer Science and Engineering at ChungNam University, Korea. He obtained his Ph.D. degree from the Department of Computer Science and Engineering at ChungNam University, Korea, in 1998. His main research interests include cybersecurity, blockchain security, and Internet of Things (IoT) security.