Remote Host Operation System Type Detection Based on Machine Learning Approach

LEONID KUPERSHTEIN

TATIANA MARTYNIUK

OLESIA VOITOVYCH

ARTUR BORUSEVYCH

VINNYTSIA NATIONAL TECHNICAL UNIVERSITY, VINNYTSIA, UKRAINE

Introduction

To conduct authorized attacks in the initial stages, penetration testing specialists and ethical hackers need to gather as much information about the object as possible in order:

- to form the most effective vector of attack;
- to identify potential vulnerabilities in the protection system;
- to identify potentially vulnerable operating system;
- to create more specific plan for security improvement.

Methods for operating system detection

Active

Pros:

- speed
- simplicity of implementation
- flexibility

Cons:

- visibility
- signature DB needed
- node response required

Passive

Pros:

- invisibility
- node response is not required

Cons:

- slow speed
- complexity of implementation

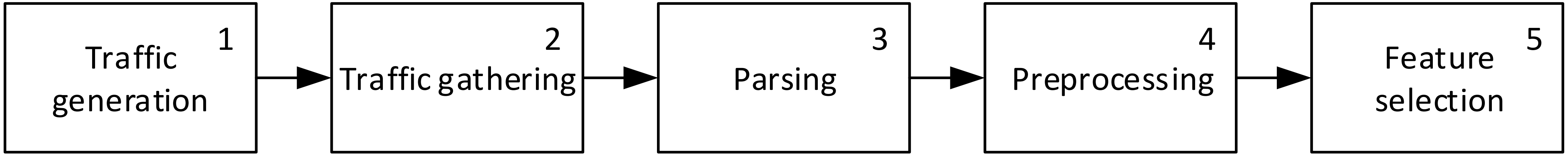
Tools for operating system detection

Software	Method	Protocols	Last update	OS family
Nmap	Active	TCP, UDP, ICMP	23-04-2021	FreeBSD, iOS, Mac OSX, OpenSolaris, Linux, Windows
p0f	Passive	TCP, HTTP	18-04-2016	FreeBSD, iOS, Mac OSX, OpenSolaris, Linux, Windows
NetScanTools Pro	Active	ICMP	02-09-2020	-
Xprobe	Active	ICMP, TCP	27-07-2005	FreeBSD, Mac OSX, Linux, Windows
Satori	Passive	DHCP, TCP, HTTP, SMB	04-05-2021	FreeBSD, iOS, Mac OSX, OpenSolaris, Linux, Windows
NetworkMiner	Passive	TCP, HTTP	23-09-2020	FreeBSD, iOS, Mac OSX, OpenSolaris, Linux, Windows
PRADS	Passive	TCP, UDP, DHCP, ICMP	19-09-2020 (app), 16-02-2010 (DB signature)	FreeBSD, iOS, Mac OSX, OpenSolaris, Linux, Windows
Ettercap	Passive	TCP	06-01-2021	FreeBSD, iOS, Mac OSX, OpenSolaris, Linux, Windows

Problem formulation

Analysis showed that all approaches are based on signature DB. But if signature is missing in the DB - this can lead to low credibility of the answer. A solution can be to use machine learning methods. This will allow to make a decision even if some data is missing. Also machine learning methods can discover dependencies, that are not clearly visible.

Experiment flow



The following OS versions were studied:
- **Linux** (version 5.4.0);
- **Mac OS** (version 10.12.4 and 11.4);
- **Windows 10** (Corporate 20h2, Home 20h2);
- **Windows 7** (Professional);
- **Windows XP** (Professional SP3).

The following OS versions were were studied using **virtualization technology** (**VirtualBox**):
- **Linux** (version 5.4.0);
- **Windows XP** (Professional SP3).

Other OS's were installed on real PC's.

Traffic generation

Such actions were performed to generate traffic:

- send 20 ICMP packets with the Type 8 Echo Request value to the studied OS;
- view video content on the YouTube web resource for at least 30 seconds from the studied OS;
- view different web pages from the studied OS;
- download images from the web to the studied OS.

Traffic gathering

Amount of packets collected:

— **MacOS** 11.4 - 67 949;

— **Windows XP** Professional SP3 – 21 294;

— **Windows 10** Home 20h2 – 19 291;

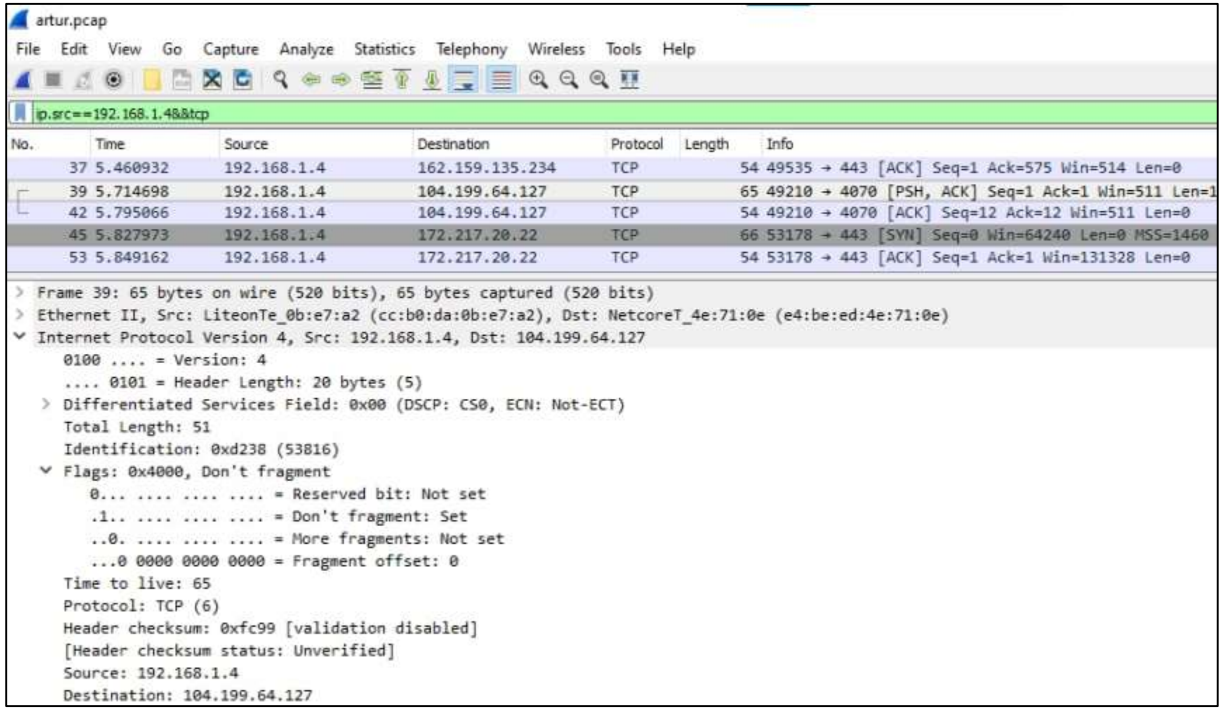
— **Windows 7** Professional – 17 741;

— **Mac OS X** 10.12.4 – 16 204;

— **Linux** 5.4.0 – 14 307;

— **Windows 10** Corporate 20h2 – 14 072.

Total: 170 858



Wireshark

Parsing

Parsing is performed using PyShark library. **IP** (16 fields), **ICMP** (10 fields), **TCP** (18 fields), **DNS** (16 fields) and **HTTP** (1 field) protocols were. Results were saved in CSV format (fields + OS name). As a result, a dataset containing 42,318 records was formed.

Full dataset			Small dataset		
OS	Samples	Percent, %	OS	Samples	percent, %
MacOS 11.4	16949	40.05	MacOS 11.4	999	14.67
Windows XP Pro SP3	5281	12.48	Windows XP Pro SP3	999	14.67
Windows 10 Home	4717	11.14	Windows 10 Home	999	14.67
Windows 7 Pro	4385	10.36	Windows 7 Pro	999	14.67
Mac OS X 10.12.4	4024	9.51	Mac OS X 10.12.4	999	14.67
Linux 5.4.0	3549	8.39	Linux 5.4.0	816	11.98
Windows 10 Corp 20h2	3413	8.07	Windows 10 Corp	999	14.67
Total:	42318	100	Total:	6810	100

Preprocessing

Two methods were used for converting categorical data to numbers:

- **label encoding** - encode values of operating systems (certain number-certain OS)
- **one hot encoding** - encode attribute value by creating column and setting value to 1 in corresponding column, and setting 0 in other columns.

Categorical data, that needed to be converted:

OS name;

IP: dsfield, flags;

TCP: flags, flags_str;

DNS: flags, qry_class.

For the study of some classifiers data normalization was performed, i.e. reduction of all values of each parameter with a mean of 0 to the standard deviation of 1.

Feature selection/importance

Protocol	Parameter name	Description	Value example	Importance (%)
IP	hdr_len	the length of the IP header	20	0.0001
IP	flags	list of set flags	0x00004000	0.0001
IP	flags_df	the value of the flag Don't Fragment	1	0.0001
IP	ttl	packet lifetime	64	24.8843
IP	proto	the protocol used below	17 (UDP)	0.0001
ICMP	ident	identification of parts of the protocol packet	1	8.71
ICMP	seq	packet number in the sequence	558	0.0001
ICMP	seq_le	the current length of the sequence	142848	0.0088
ICMP	data_data	the contents of the message in the packet	61:62:…:76:77:61:62:63:64:65:66:67:68:69	0.0001
ICMP	data_len	the length of the message in the packet	32	36.87
TCP	hdr_len	the length of the protocol header	32	0.0001
TCP	window_size_value	window size value	513	0.008
TCP	window_size	the calculated value of the window	131328	0.0001
TCP	window_size_scalefactor	the calculated window size modifier (2^n)	256	29.5181

Model selection and training

For each estimator **GridSearch** package from Scikit-learn library was used. This approach is quite costly, however it provides best results concerning hyperparameters selection.

Dataset structure:

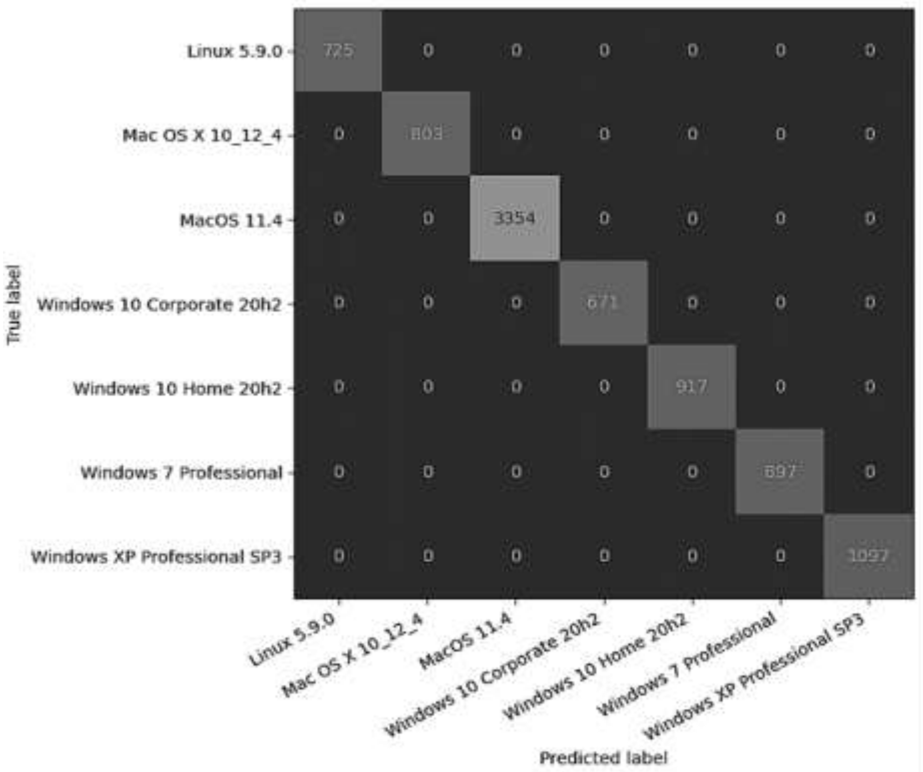
OS name, ip fields, icmp fields, tcp fields, dns fields, http field

Also, a small dataset was created in order to perform hyperparameters tuning.

Small dataset		
OS	Samples	percent, %
MacOS 11.4	999	14.67
Windows XP Pro SP3	999	14.67
Windows 10 Home	999	14.67
Windows 7 Pro	999	14.67
Mac OS X 10.12.4	999	14.67
Linux 5.4.0	816	11.98
Windows 10 Corp	999	14.67
Total:	6810	100

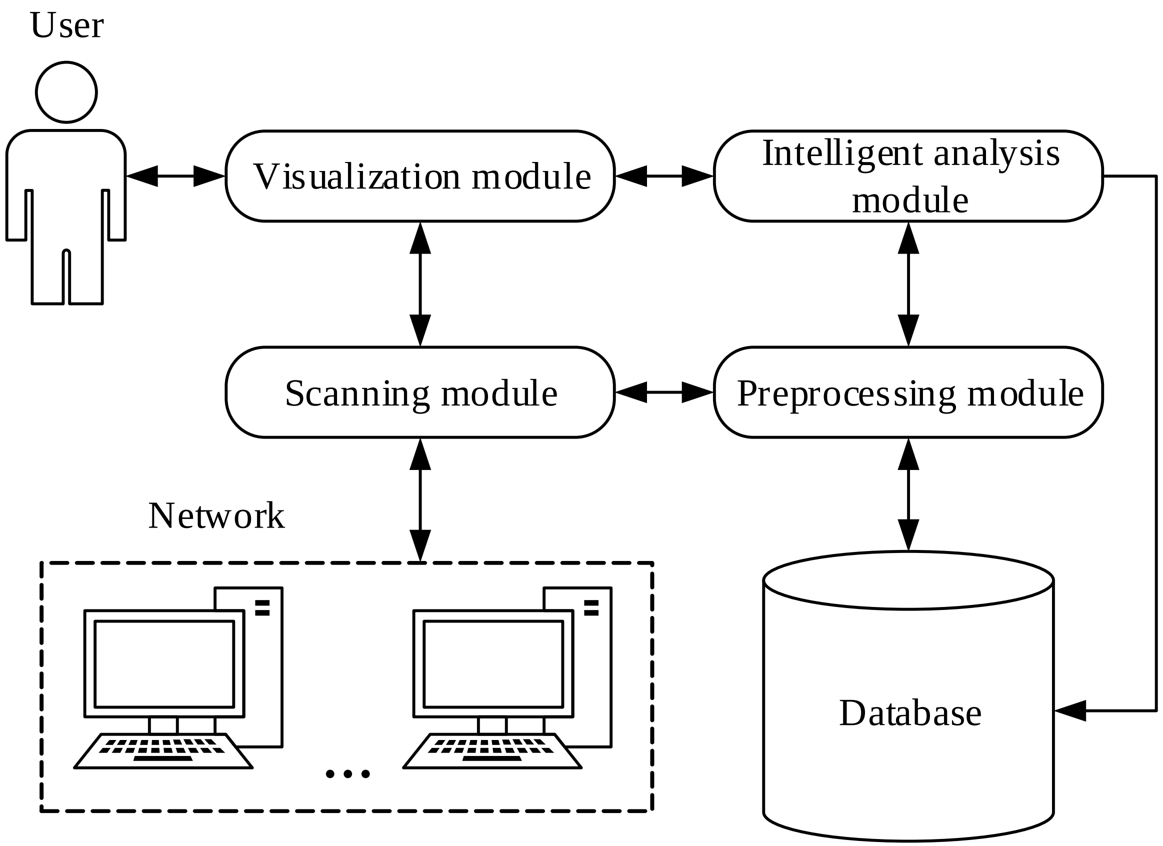
Model selection and training results

Classifier	Accuracy	Precision	Recall	F1	FP, %	FN, %	Average cross-validation score
DT	1.0	1.0	1.0	1.0	0	0	0.9999
MLP	0.99964	0.99952	0.9995	0.9995	0.002	0,004	0.9998
GNB	0.90548	0.91690	0.8736	0.8474	1.7	0,189	0.9031
KNN	0.99763	0.99682	0.9960	0.9964	0.016	0,031	0.9949
SVM	0.99952	0.99926	0.9994	0.9993	0.009	0	0.9987
LR	0.97861	0.97287	0.9880	0.9792	0.378	0,059	0.9784
RF	1.0	1.0	1.0	1.0	0	0	0.9999
Frequency analysis [23]	0.7-1	-	-	-	-	-	-
Euclidean distance [24]	0.955	-	-	-	4.5	-	-
SVM [25]	0.8663	-	-	-	-	13.36	-
Decision Table [26]	0.994	-	-	-	-	-	-
DT/J48 [27]	0.94	-	-	-	-	-	-
DT/C4.5 [28]	0.9086	-	-	-	-	-	-

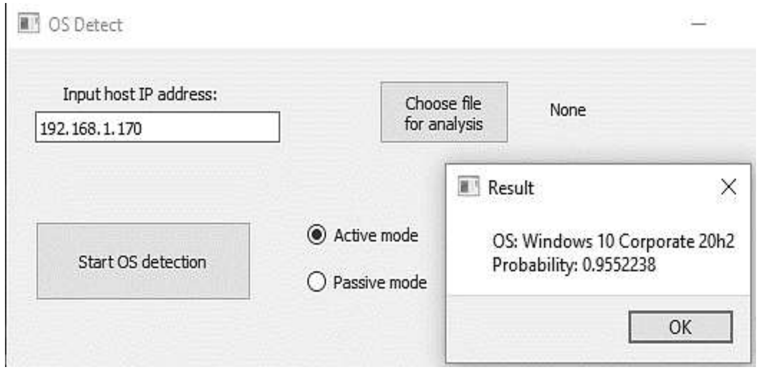


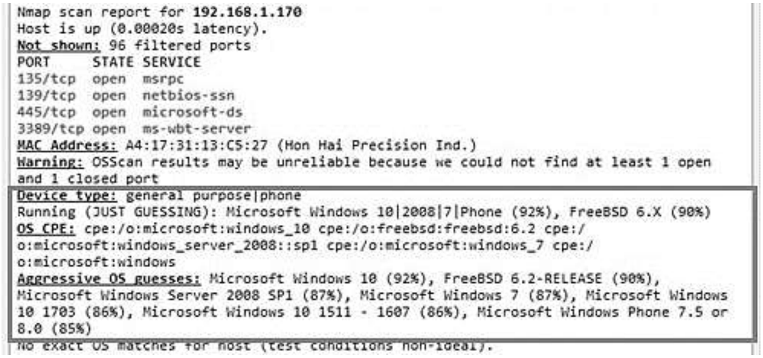
Confusion matrix for Decision Tree & Random Forest

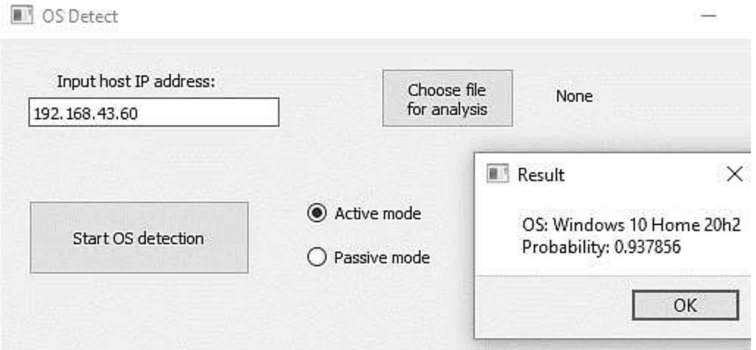
Application architecture

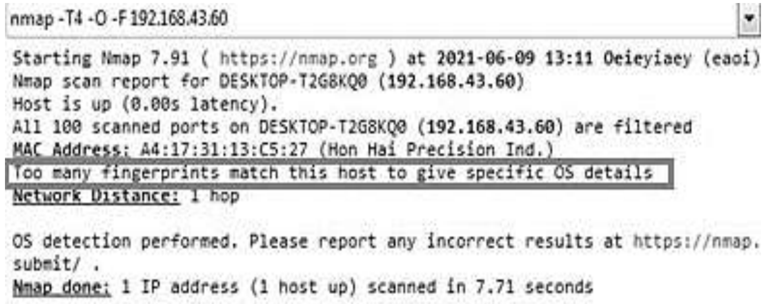


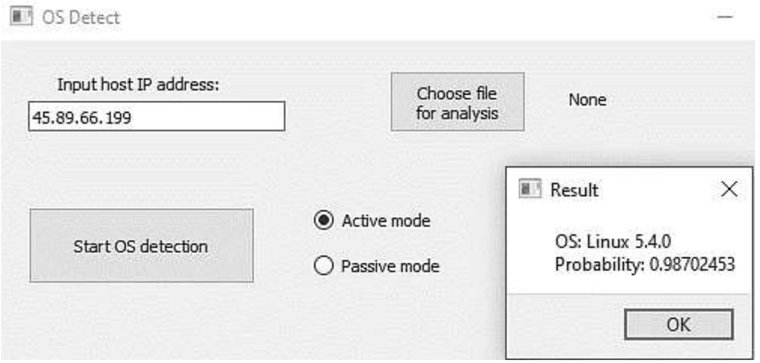
Experiment results

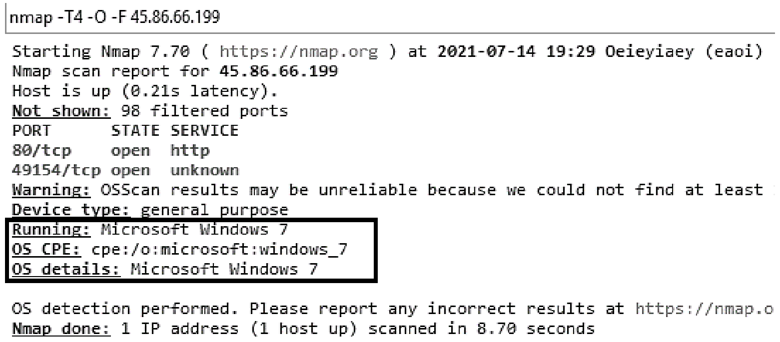












Conlusion

As a result of the experiment, the developed software was more effective compared to Nmap.

However, Nmap allows to define many more types of OS. Therefore, further research will be associated with scaling the model, as well as expanding the functionality of application. One such future function will be to implement the ability to provide a list of ranked vulnerabilities inherent in a particular OS. It provides faster vector attack creation or decision making for network protection.

Disadvantages of software:

– fewer OS's, that can be detected, compared to others;

Future work includes increasing number of OS's that can be detected, as well as providing information about know vulnerabilities of detected OS to end user.

Thank you for attention!