Фінгерпрінтинг, як правило, використовується, щоб уникнути порівняння та передачі громіздких даних. Фінгерпрінтинг включає в себе два методи: активний та пасивний. В активному фінгепрінтингу ми працюємо із певними інструментами, які за допомогою переданих пакетів дозволяють дізнатись, що система знаходиться в іншому кінці. У цій статті показано та проаналізовано основні методи активного фінгерпрінтингу, які використовуються в канальному рівні передачі даних та в стеку Інтернет протоколу TCP/IP

Ключові слова: активний фінгерпрінтинг, рівень передачі даних, стек рівня TCP/IP

Фингерпринтинг, как правило, используется во избежание сравнения и передачи громоздких данных. Фингерпринтинг включает в себя два метода: активный и пассивный. В активном фингерпринтинге определенными инструментами осуществляется передача пакетов, которые позволяют узнать, что система находится в другом конце. В этой статье показаны и проанализированы основные методы активного фингерпринтинга, которые используются в канальном уровне передачи данных и в стеке Интернет протокола TCP/IP

Ключевые слова: активный фингерпринтинг, уровень передачи данных, стек уровня TCP/IP

UDK 004.738.5.057.4
DOI: 10.15587/1729-4061.2015.50983

COMPARISON OVERVIEW OF AN ACTIVE FINGERPRINTING APPLICATION OF THE SECOND AND THE THIRD LAYER OF TCP/IP STACK

V. Mosorov
Doctor of Technical Science*
E-mail: volodymyr.mosorov@p.lodz.pl
S. Biedron
Postgraduate student*
E-mail: SBiedron@wpia.uni.lodz.pl
T. Panskyi
Postgraduate student*
E-mail: panskyy@gmail.com
*Institute of Applied Computer Science
Lodz University of Technology
Stefanowskiego str., 18/22, Lodz, Poland, 90-924

1. Introduction

Fingerprinting is an extremely vast issue, considering its origins. Remote operating systems detection is reduced to identifying the operating system or applications running on the scanned device which are identified applying methods that uses small differences between implementations of the TCP/IP stack protocols. With these seemingly insignificant trifles we can successfully gain some very important information from another user without his knowledge.

The objective of this method depends entirely on testing and a number of intentions. With the appropriate knowledge and tools, as the administrator we can check the security of our systems or, if we have not done this before, learn the topology and operation of the network we manage. Remote detection of network computers is also used to carry out reconnaissance without which Internet break-ins and thefts would be almost impossible. Before they set to break computer security, computer criminals diagnose the protections they are dealing with and they use precisely the methods described herein below. Uniform systems would reduce the chance of intrusions, which would dramatically reduce the number of attacks, and the time it takes to break the blockades would be much longer [1].

Fingerprinting involves two methods: active and passive. This division and classification depends on how the desired information is obtained. In active fingerprinting we deal with certain tools which via crafted and transmitted packets examine what system is at the other end. Passive fingerprinting is based on monitoring, capturing and analyzing the data transmitted by the victim. This method is mainly based on the principle of sniffing, which is why such activities are almost completely undetectable. When the prospective "victim" is trying to establish a connection, they may not be fully aware that the host or party they are connecting with is equipped with spyware that monitors and collects information on their computer. Basically, passive fingerprinting involves all methods designed to intercept packets/datagrams, and subsequent analysis of such packets/datagrams applying the methods employed by active fingerprinting. The ways to intercept data are legion, starting from the use of physical devices within the network, such as hotspot WiFi that would purport to be the router of an establishment, e.g. a café or office. This option is used by a cybercriminals for capturing or taking control over a legally operating devices during its owner's inattention or monitoring traffic on the Internet sites via malicious software, e. g. viruses, Trojans, as well as ordinary readable fluctuations in the received packets.

2. Analysis of published data and problem statement

Nowadays we are witnessing a dynamic development of computer networks. Unfortunately, rapid development often means that some products – in this case operating systems – are not fully worked out [2]. They feature loopholes which, while the given system exists, are tried to be "patched up" by developers. This feature is frequently used as the

advantage by the intruders, which are a consequence of dynamic growth of cybercrime and ever more, appearance of new ways and methods of criminal operation [3]. Each omission by developers during the production process of the operating system provides cybercriminals the opportunity to exploit it. Fingerprinting [4] is just that set of principles and methods that utilize such inaccuracies to determine the operating system of the potential victim, which then makes it much easier to carry out the attack [5, 6].

Over the years more and more complex methods for sampling the operating systems taking advantage of the protocols of the second [7] and third [8] TCP/IP layer have been invented. A new programming tools for active fingerprinting have created, among others SinFP [9], Xprobe2 [10], Nmap [11]. In recent years, due to the wireless network popularization, new fingerprinting method that uses active faults in the implementation of the protocols associated with WiFi had been appeared [12].

In this article the basic methods of active fingerprinting of second and the third layer of TCP/IT stack have been mentioned. Also the various reactions of certain systems for carried out scanning have been presented.

3. Purpose and objectives of the study

The main objective of this publication is to present basic methods and functioning of an active fingerprinting data link layer and the Internet TCP / IP stack.

In accordance with the set goal the following research objectives are identified:

– Increase the public interest in Active Fingerprinting;

– Analysis of active methods of Fingerprinting;

– Present the potential threats that fingerprinting could pose to ordinary users.

4. Reasons for the differences in the implementation of the TCP/IP stack

In 1969, as part of the ARPANET, the idea of relation of technical and organizational documents with the communications between computers had been born. Some of these documents eventually began to form official standards to which network protocols should be implemented. Unfortunately, in the course of the implementation of TCP/IP stacks in different systems there appeared some slight differences between them. In the available literature we can find information that the existing differences were caused by omission or mistake of the developers during the implementation of the stack.

Another more glaring issue is that the interpretation of documents is often too superficial and some of the recommendations are ignored. One example might be a minor "improvement" introduced by Microsoft into their products. Older Windows, inclusive of the Win 98, were interpreted by TTL=64 (as in Linux), but in later systems this number was changed to 128 [13], thus exposing the system operated by a potential client of the company to the risk of it being identified by an intruder. It was this deliberate, though not carefully considered, decision that has reduced the system security. A user with basic knowledge how to operate this software can easily identify which family the system comes from, as no other system is identified by this TTL number.

5. Active fingerprinting of a data link layer

To carry out remote diagnosis of the operating system on the level of the data link layer it takes to apply the ARP protocol and one of the most popular NTA open-source programs called arp-scans. This is a tool activated from the command-line which constructs and transmits an ARP request to the IP address and concurrently monitors the reaction caused by the given data bits. This application enables transmission of ARP packets to all the devices, helps determine the structure and number of hosts within the network, including those behind the firewall, and makes it possible to run tests without an appointed IP address. In this case, the tool will use the IP 0.0.0.0 as the address of the sender. In a nutshell, arp-scans provide the opportunity to easily create your own ARP packets. They monitor and decode the received ARP packets and include a fingerprinting utility, "arp-fingerprint".

This software, however, has significant drawbacks, first and foremost of which is that it generates network traffic based on custom-made and easy to capture packets, uses an unroutable protocol, and enables work on a limited number of selected operating systems.

Using ARP-scans it is possible to conduct the following tests:

– **Mac Vendor Decoding** – it is possible to guess the scanned system just by applying the ARP-scan without using any additional options.

The system's identity is partly betrayed by the feedback to the previously sent ARP request in the following form:

<IP address> <MAC address> <equipment manufacturer>.

The information obtained in the third column is the result of the Arp-scan decoding the first 24-bit MAC address which contains information indicating the manufacturer.

Example:

00:04:27:6d:3a:a1 Cisco Systems, Inc.

For decoding purposes the ARP-scan uses three files that must be updated from time to time in order to obtain correct results. These include:

1. ***ieee-oui.txt*** – OUI IEEE list (Institute of Electrical and Electronics Engineers Organisationally Unique Identifier) which can be downloaded from the IEEE website and updated using the get-oui script contained in the Arp-scan.

2. ***ieee-iab.txt*** – IEEE IAB list (Institute of Electrical and Electronics Engineers Individual Address Block) which can be downloaded from the IEEE website and updated using the get-IAB script contained in the Arp-scan.

3. ***mac-vendor.txt*** – a list identifying the other producers. It is maintained by the user and is usually much smaller than the OUI and IAB.

When the Arp-scan starts working it reads the contents of all three files into the hash table and checks with its contents the MAC address of each received ARP feedback in order to identify the manufacturer. Having such data it is possible, e. g. to determine that a Cisco Systems, Inc. product will run the IOS operating system. This method does not help to accurately determine the version of the system; nevertheless it reveals the devices that are bound to run a specific operating system (e. g. Cisco).

– **Non-standard ARP packets** – the Arp-scan contains a fingerprinting utility called ARP-fingerprint. It is a Perl script used by the Arp-scan to transmit and receive ARP

packets. It performs eleven tests which, by exploiting the differences in the implementation of the TCP/IP stack, will determine what system we are dealing with.

The tests that were carried out using the program ARP-scan are presented at the Table 1.

Table 1

Tests carried out by ARP-fingerprint

№	Description	Arp-scan parameters	Information
1	source address=localhost	–arpspa=127.0.0.1	127.0.0.1 should never appear in LAN as the source address
2	source address=0.0.0.0	–arpspa=0.0.0.0	Some systems respond only if the source address of the sender is „correct”
3	source address=broadcast	–arpspa= =255.255.255.255	
4	source address=non-local	–arpspa=1.0.0.1	The IP 1.0.0.0/8 network is re3served for IANA and should not appear in LAN
5	Incorrect operation code	–arpop=255	255 operation code has not been defined
6	Hardware Type=IEEE_802.2	–arphrd=6	Most systems respond to this in the same way as to the type of equipment=1
7	Wrong type of equipment	–arphrd=255	255 hardware has not been defined. (e. g. Ethernet=1)
8	Wrong protocol type	–arppro=0xffff	0xffff protocol type has not been defined
9	Protocol type=IPX	–arppro=0x8137	Selected IPX protocol
10	Incorrect length of protocol address	–arppln=6	The length of the protocol address is normally 4 bytes
11	Incorrect length of hardware address	–arphln=8	The length of the hardware address is normally 4 bytes

The result of each test is the answer yes (1) or no (0), depending on whether the given host did respond or not. From these eleven numbers the ARP-fingerprint builds a sequence and then compares them with the list of known “fingerprints”.

Example:

$ arp-fingerprint -o M–interface=eth0 –numeric” 172.128.1.2

01000100000 Linux 2.2, 2.4, 2.6

$ arp-fingerprint -o “–interface=eth0 –numeric” 172.128.1.3

11110100000 FreeBSD 5.3, Win98, WinME, NT4, 2000, XP, 2003 $ arp-fingerprint -o “–interface=eth0 –numeric” 172.128.1.4

00000100000 Cisco IOS 11.2, 11.3, 12.0, 12.1, 12.2, 12.3, 12.4

where: $ arp-fingerprint is the arp-fingerprint option that transmitted the sequence --interface=eth0 –numeric to the arp-scan

–interface=eth0 selects the interface to be used (in this case eth0)

–numeric is used in order to avoid the DNS lookup

Table 2 contains an example of the results of the scan performed by the program ARP-scan.

Table 2

ARP-fingerprint fingerprint base

Result	Sample systems
11110100000	FreeBSD 5.3, Win98, WinME, NT4, 2000, XP, Win 2003
01000100000	Linux 2.2, 2.4, 2.6
01010100000	Linux 2.2, 2.4, 2.6, Vista, Windows 7, Windows 8
0000100000	Cisco IOS 11.2, 11.3, 12.0, 12.1, 12.2, 12.3, 12.4
11110110000	Solaris 2.5.1, 2.6, 7, 8, 9, 10, HP-UX 11
10010100011	SCO OS 5.0.7
10110100000	Win 3.11, 95, NT 3.51
11110000011	BSD 4.3, OpenBSD 3.1, OpenBSD 3.9, Nortel Contivity 6.00, 6.05
10110110000	NetBSD 2.0.2
00000110000	Netware 6.5

As can be seen, many fingerprints concurrently identify several systems, so it is not always possible to say with absolute certainty that we are dealing with this or that system. Sometimes even a different configuration may produce a different result, as is the case with, e. g. Linux which responds to non-local source IP address when the IP is routed through the interface on which the experiments are conducted.

– **Ethernet Frame Padding** – ARP packets only have the length of 28 bytes, which is much less than the minimum size of the Ethernet frame (46 bytes). For this reason, at least an 18-byte filling must be placed before each ARP packet. If the ARP implementation does not do it, the Ethernet controller must deal with it. The arp-scan can add the filling to the outgoing ARP-request with the -padding option. If the -verbose option is chosen, then the arp-scan application will inform us of any non-zero fillings received in the ARP-response. Most implementations will reach the minimum length required by the addition of the filling comprised of zero bytes. However, there exist examples that differ significantly from the established norms (Table 3).

Table 3

Differences in the fillings of the frames containing ARP messages

IP address	System	Information
172.128.2.2	MacOS 10.4	Interval of 18 bytes 0x55 (binary 01010101)
172.128.2.3	NetScreen ScreenOS 5.0.0	Interval of 18 bytes 0x88 (binary 10001000)
172.128.2.4	Solaris 8/SPARC	Interval of 18 bytes 0x55 (binary 01010101)
172.128.2.5	Xerox Phaser 6200	Uses the interval from the ARP-request
172.128.2.6	HP JetDirect	Uses the interval from the ARP-request
172.128.2.7	Linux with eepro100 controller	Adds 4 non-zero bytes at the end of the interval
172.128.2.8	Cisco Catalyst IOS 12.0	Adds a part of the interval from the ARP-request frame

Based on the results, there are some systems and devices that react quite irregularly on the performed test, which significantly allow for its identification in the network.

6. Internet layer

At the level of this layer operate two very important protocols. One of them takes active part in the transport of data, while the second is an information protocol.

At the level of this layer the following methods of remote detection of network devices are available:

– **Don't Fragment flag** – the "flags" field [14, 15] of the IP protocol consists of 3 bits. The first and the third bit are not essential in this method. The second bit informs whether a piece of information is to be divided into parts or not. Its setting causes a certain anomaly. The Don't Fragment flag, despite its intended function, is currently used by modern operating systems to detect the optimum packet on the route between the two devices involved in the exchange of data so as to avoid fragmentation. The difference in the implementation of the TCP/IP stack is that some of the systems respond to the transmitted packet/datagram with the set DF flag by setting this flag in the reply message, while others ignore it altogether, returning 0 value, or duplicate this value. The systems' responses to the transmitted datagrams with the set DF flag generate the following ICMP messages:

Destination Unreach, Echo Response, Timestamp Response, are presented in the Table 4.

Table 4

DF fingerprints base for ICMP messages

Sample systems	DF		
	Unreach	Echo	Time-stamp
FreeBSD 2.0.5, 2.1.0, 2.1.5, 2.1.6, 2.2.0, 2.2.1, 2.2.2, 2.2.5, 2.2.6, 2.2.7, 2.2.8, 2.2.9, 3.0, 3.1, 3.2, 3.3, 3.4, 4.0, 4.1, 4.1.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.6.2, 4.7, 4.8, 5.0, 5.1	Duplicates the value	Duplicates	Duplicates the value
Linux 2.0.29, 2.0.30, 2.0.34, 2.2.0, 2.2.1, 2.2.2, 2.2.3, 2.2.4, 2.2.5, 2.2.515, 2.2.6, 2.2.7, 2.2.8, 2.2.9, 2.2.10, 2.2.11, 2.2.12, 2.2.12-20, 2.2.13, 2.2.14, 2.2.14-5, 2.2.15, 2.2.16, 2.2.1622, 2.2.17, 2.2.18, 2.2.19, 2.2.20, 2.2.20-idepci, 2.2.21, 2.2.22, 2.2.23, 2.2.24	Does not set	Sets	Does not set
Linux 2.0.32, 2.0.36	Does not set	Sets	Does not respond
Linux 2.4.0, 2.4.1, 2.4.2, 2.4.2-2, 2.4.3, 2.4.4, 2.4.4-4GB	Sets	Sets	Sets
Linux 2.4.5. 2.4.6, 2.4.7, 2.4.8, 2.4.9, 2.4.10, 2.4.10-4GB, 2.4.11, 2.4.12, 2.4.13, 2.4.14, 2.4.15, 2.4.16, 2.4.17, 2.4.18, 2.4.18-3, 2.4.18-4GB, 2.4.1814, 2.4.19, 2.4.19-4GB, 2.4.20, 2.4.208, 2.4.21-0.13mdk	Does not set	Sets	Does not set
MacOS 7.5.3, 7.5.5, 7.6, 7.6.1, 8.0, 8.1	Sets	Sets	Does not respond

IPID sampling – This method uses another loophole in the implementation of the IP protocol's TCP/IP stack. It brings to attention the "Identification" field. It is used in the same way as during transmission of large size data, i.e. such data that cannot be processed by intermediate devices at a time (obviously provided that the "Do not Fragment" flag is not set) [16]. Without this field it would be impossible to distinguish which element belongs to which packet/datagram. Any system with TCP/IP stack has a way of assigning subsequent ID values (0–65536) to subsequent split packets/datagrams for each protocol. However, the methods of numbering, as it turns out, are not the same for all systems, which is why they are divided into groups (Table 5).

Table 5

Fingerprint base of IPID fields depending on the layer 4 protocol used

Sample systems	Value	Session	Protocol
Microsoft Windows 95 Retail, SP 1	+256	Global	TCP, UDP, ICMP
Microsoft Windows 95 OEM Service Release 2, 2.1, 2.5	+256	Global	TCP, UDP, ICMP
Microsoft Windows 98	+256	Global	TCP, UDP, ICMP
Microsoft Windows 98 Second Edition	+256	Global	TCP, UDP, ICMP
Microsoft Windows NT 3.1–4.0	+256	Global	TCP, UDP, ICMP
Microsoft Windows NT 5.0–6.0	+1	Global	TCP, UDP, ICMP
Microsoft Windows Millennium Edition	+1	Global	TCP, UDP, ICMP
Microsoft Windows 2000, SP 2, SP 3, SP 4	+1	Global	TCP, UDP, ICMP
Microsoft Windows XP	+1	Global	TCP, UDP, ICMP
Microsoft Windows VISTA	+1	Global	TCP, UDP, ICMP
Microsoft Windows 7	+1	Global	TCP, UDP, ICMP

Time To Life Identification – to perform this test, use the TTL field contained in the ICMP in the IP header. According to commercially available books in which the ICMP, IP protocol is discussed at least on a basic level, you can learn that many systems set different TTL values. At this point the mention of this field most frequently ends and we are referred to the RFC standards [17]. Unfortunately, these documents represent a dead end, as they only tell us what TTL is and how it should be implemented. The difference in the values of the TTL field in some older systems frequently varies depending on the selected protocol (Table 6) or ICMP message (Table 7).

ICMP echo request, echo reply – the systems set different TTL field values depending on the ICMP message.

Impact of higher layer protocols – UDP, TCP.

Table 6

Fingerprint base of TTL fields depending on the type of message

System	Echo request	Echo reply
RedHat 5.0, 5.2	64	64
RedHat 6.1	64	255
Mandrake 7.0	64	255
FreeBSD 7.1	64	64
Windows95	32	32
Windows NT4 Workstation (SP4, SP5)	32	128
Windows Milenium	128	128
Windows XP	128	128
Windows Vista 64 bit home premium	128	128
Windows 7	128	128
Windows Server 2008	128	128
OpenBSD 2.6, 2.7	255	255
NetBSD	255	255
HP UX 10.20	255	255

Table 7

Fingerprint base of TTL fields depending on the layer 4 protocol used 4

System	IP +UDP	IP + TCP
AIX	30	60
FreeBSD 2.1R	65	64
HP-UX9.0x	30	30
HP-UX10.01	64	64
Irix5.3	60	60
Windows95	32	32
MS Windows 98	128	128
MS Windows NT 3.51	32	32
MS Windows NT 4.0	128	128
MS Windows 2000	128	128
MS Windows XP	128	128
MS Windows VISTA	128	128
MS Windows 7	128	128
MS Windows Server 2008	128	128

ICMP Error Message Quenching [18] – this is a method most commonly employed in the course of scanning the ports. It utilizes more simply built datagrams (UDP instead of TCP) transmitting them to the sampled port. The reaction occurs when we find the port closed. The scanned system responds with the ICMP message "port unreachable" ("port Unreach"). On this basis it is possible to tell the closed UDP ports from the open ones. Keep in mind that neither protocol is sure to be reliable. Sending about 100 such datagrams to a closed port, we can come across some surprising behavior. This discrepancy has its roots in the standardization document RFC 1812 which proposed three approaches to reducing the number of ICMP messages sent in order to ensure reduced network traffic:

– Count-base – for each rejected datagram one ICMP message is transmitted;

– Time-base – one ICMP message can be sent at T milliseconds;

– Bandwidth-base – an ICMP message can be transmitted only at the maximum speed established for this type of information.

Most of the operating systems have an implemented method of reducing transmission of an excessive number of ICMP messages, but, as one might expect, not all of them. Such systems include those produced before 1995. Further distinguishing parameters are the values used in setting the limiter.

Example:

Linux limits the number of "unattainable goal" ("destination unreachable") messages to 80 per 4 seconds. If this number is exceeded, subsequent messages are transmitted at intervals of 0.25 seconds. As a standard, FreeBSD 7.1 supports up to 200 messages per 1 second, and after an overrun it will automatically reduce the number of incoming datagrams per second.

– **ICMP Message Quoting** - operating systems with implemented ICMP protocol react to the error resulting from the failure to transmit a received packet to the specified port number that is closed at the time by sending the feedback message "port unreachable" ("port unreachable"). According to the RFC 1122 standard this message should have at least 8 bytes replicated from the original message that caused this error. The vast majority of software vendors have implemented in their TCP/IP stacks a return "cargo" equal to 8 bytes. Many systems return more bytes, sometimes even a whole datagram delivered with the ICMP message. This method not only distinguishes a group of systems, for example. 7.x Sun Solaris, HP-UX 11.x, MacOS 7.55, 8.0, 8.1, 9.04 or devices from 3Com, Foundry, Alcatel, but it can identify Linux 2.2.x / 2.4 tx that relies on Kernel.

– **ICMP Error message echoing integrity** – like the previous one, this test is based on the RFC 1122 standardization document which provides that the return ICMP message "port unreachable" should contain at least 8 bytes of the original packet. In this case, however, we will look not at the quantity of the "cited" datagram, but the quality of the data returned. Deficiencies in the implementation of the TCP/IP stack have caused certain groups of operating systems to incorrectly cite the original datagram received [19]. The fields in which they can be distorted are as follows:

– IP – Overall Length – some systems, such as, e. g. AIX, BSDI wrongly cite the length of the field complete IP protocol in the returned datagram for adding to its value 20 bytes. Other OSs can, therefore, react differently and go in the opposite direction by reducing the original value by the said 20 bytes.

– IP – Identification – for certain versions of BSDI, FreeBSD, OpenBSD, Ultrix, VAX it is a huge problem to correctly cite the "Identification" field.

– IP – flags – another relationship is the wrong citation of the flags field, whose bits will be transmitted in a different order than that in which they were received.

– IP – Type of service – most systems return 0 value in the return ICMP message, however, some of them will change the value of this field, e. g. Linux returns 0xC0 value. We can observe this in the preceding example above. The difference is highlighted in blue.

– IP – Checksum – the checksum is another determinant of correct identification of the system. A group of OSs incorrectly calculates the value of the field, or returns 0 value in the returned datagram.

– UDP –checksum – as in the case of the IP the checksum is either wrongly calculated or returns 0 value.

– **ICMP Echo Requests Field Code Values** – this is one of the simplest fingerprinting tests that utilize a slightly modified ICMP Echo Request/Reply communication method. This small modification consists in transmitting an ICMP Echo Request datagram with modified "Code" field. If you set it to a value other than zero, in keeping with the RFC 792 standardization document this field should be replicated without any change in the return ICMP Reply message. However, Microsoft operating systems react quite differently. In addition to changing the "type" value and converting the checksum they reset the "Code" field in the ICMP Reply message.

– **ICMP Responses** – another very interesting relationship is the handling or non-handling of the various ICMP messages [20]. Non-application or less and less frequent use of certain options has led to a situation in which many OS manufacturers resigned from their implementation. This has resulted in some confusion, because not all manufacturers introduced changes, which is why on this basis many systems can be identified or classified in the given group. Such irregularities occur during the exchange of messages:

All of the fingerprinting tests are shown at Table 8 (timestamp request), Table 9 (address mask request), and Table 10 (ICMP information).

Table 8

Fingerprint base of ICMP Timestamp replies

Sample systems	Timestamp response
BEOS 5	Reply not sent
FreeBSD 2.0.5, 2.1.0, 2.1.5, 2.1.6, 2.1.7.1, 2.2.0, 2.2.1, 2.2.2, 2.2.5, 2.2.6, 2.2.7, 2.2.8, 3.0, 3.1, 3.2, 3.3, 3.4, 3.5.1, 4.0, 4.1, 4.1.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.6.2, 4.7, 4.8, 5.0, 5.1	Reply sent
Linux 2.0.30, 2.32, 2.0.36 (all Red Hat)	Reply not sent
Linux 2.0.29, 2.0.34, 2.0.36 (all Debian)	Reply sent
Linux 2.2.0, 2.2.1, 2.2.2, 2.2.3, 2.2.4, 2.2.5, 2.2.5-15, 2.2.6, 2.2.7, 2.2.8, 2.2.9, 2.2.10, 2.2.11, 2.2.12, 2.2.12-20, 2.2.13, 2.2.14, 2.2.14-5, 2.2.15, 2.2.16, 2.2.16-22, 2.2.17, 2.2.18, 2.2.19, 2.2.20, 2.2.20-idepci, 2.2.21, 2.2.22, 2.2.23, 2.2.24, 2.4.0, 2.4.1, 2.4.2, 2.4.2-2, 2.4.3, 2.4.4, 2.4.4-4GB, 2.4.5, 2.4.6, 2.4.7, 2.4.8, 2.4.9, 2.4.10, 2.4.10-4GB, 2.4.11, 2.4.12, 2.4.13, 2.4.14, 2.4.15, 2.4.16, 2.4.17, 2.4.18, 2.4.18-3, 2.4.18-4GB, 2.4.18-14, 2.4.19, 2.4.19-4GB, 2.4.20, 2.4.20-8, 2.4.21-0.13mdk	Reply sent
MacOS 7.5.3, 7.5.5, 7.6, 7.6.1, 8.0, 8.1, 9.0, 9.1, 9.2.1, 9.2.2	Reply not sent
Windows 98, 98 SE	Reply sent
Windows NT 4 standard, sp3, sp4, sp6	Reply not sent
Windows Millennium standard	Reply sent
Windows 2000 standard, sp2, sp3, sp4	Reply sent
Windows XP Home, Professional	Reply sent
Windows Net standard	Reply sent
Windows 2003 Server standard	Reply sent
Windows 7	Reply sent
Windows Server 2008	Reply sent

Table 9

Fingerprint base of ICMP Address Mask Request replies

System	Timestamp response
Linux 2.4.10, 2.4.10-4GB, 2.4.11, 2.4.12, 2.4.13, 2.4.14, 2.4.15, 2.4.16, 2.4.17, 2.4.18, 2.4.18-3, 2.4.18-4GB, 2.4.1814, 2.4.19, 2.4.19-4GB, 2.4.20, 2.4.20-8, 2.4.21-0.13mdk	Reply not sent
MacOS 10.1.0, 10.1.1, 10.1.2, 10.1.3, 10.1.4, 10.1.5, 10.2.1, 10.2.2, 10.2.3, 10.2.4, 10.2.5, 10.2.6	Reply not sent
NetBSD 1.1, 1.2, 1.2.1, 1.3, 1.3.1, 1.3.2, 1.3.3, 1.4, 1.4.1, 1.4.2, 1.4.3, 1.5, 1.5.1, 1.5.2, 1.5.3, 1.6, 1.6.1	Reply not sent
Netware 4.11, 4.11 sp9	Reply sent
Netware 5, 5 sp6a	Reply sent

Table 10

Fingerprint base of ICMP Information replies

System	Timestamp response
BEOS 5	Reply not sent
HP UX 10.0, 10.2, 10.24, 10.30 [28]	Reply sent
FreeBSD 2.0.5, 2.1.0, 2.1.5, 2.1.6, 2.1.7.1, 2.2.0, 2.2.1, 2.2.2, 2.2.5, 2.2.6, 2.2.7, 2.2.8, 3.0, 3.1, 3.2, 3.3, 3.4, 3.5.1, 4.0, 4.1, 4.1.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.6.2, 4.7, 4.8, 5.0, 5.1	Reply not sent
Linux 2.0.29, 2.0.30, 2.0.32, 2.0.34, 2.0.36, 2.2.0, 2.2.1, 2.2.2, 2.2.3, 2.2.4, 2.2.5, 2.2.5-15, 2.2.6, 2.2.7, 2.2.8, 2.2.9, 2.2.10, 2.2.12, 2.2.12-20, 2.2.13, 2.2.14, 2.2.14-5, 2.2.15, 2.2.16, 2.2.16-22, 2.2.17, 2.2.18, 2.2.19, 2.2.20, 2.2.20-idepci, 2.2.21, 2.2.22, 2.2.23, 2.2.24, 2.4.0, 2.4.1, 2.4.2, 2.4.2-2, 2.4.3, 2.4.4, 2.4.4-4GB, 2.4.5, 2.4.6, 2.4.7, 2.4.8, 2.4.9, 2.4.10, 2.4.10-4GB, 2.4.12, 2.4.13, 2.4.14, 2.4.15, 2.4.16, 2.4.17, 2.4.18, 2.4.18-3, 2.4.18-4GB, 2.4.18-14, 2.4.19, 2.4.19-4GB, 2.4.20, 2.4.20-8, 2.4.21-0.13mdk	Reply not sent
MacOS 7.5.3, 7.5.5, 7.6, 7.6.1, 8.0, 8.1, 9.0, 9.1, 9.2.1, 9.2.2, 10.1.0, 10.1.1, 10.1.2, 10.1.3, 10.1.4, 10.1.5, 10.2.1, 10.2.2, 10.2.3, 10.2.4, 10.2.5, 10.2.6	Reply not sent
NetBSD 1.1, 1.2, 1.2.1, 1.3, 1.3.1, 1.3.2, 1.3.3, 1.4, 1.4.1, 1.4.2, 1.4.3, 1.5, 1.5.1, 1.5.2, 1.5.3, 1.6, 1.6.1	Reply not sent
OpenVMS 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 6.1, 7.0, 7.1 [29]	Reply sent
Netware 4.11, 4.11 sp9	Reply not sent
Netware 5, 5 sp6a	Reply not sent

With the information gathered from the test results presented in Table 6–8 users could quite clearly increase their chances to identify the scanned computer or device.

7. Summary

The future of fingerprinting is very clear at the present time. To date, only a negligible part of the "faulty" implementation of TCP/IP stack has been corrected, while the overlays frequently installed to ensure computer security open further loopholes, thus exposing both the system and our confidential data. First and foremost, this article helps to draw attention to the threat posed by non-observance of the principles of standardization in the implementation of the TCP/IP stack in operating systems. Subsequent versions of the same operating system usually contain the same irregu-

larities, and not infrequently even new unexpected reactions do appear. Looking at how the market for operating systems is developing and how the group of regular users who use the computer only as a tool for "surfing" on the Internet is growing, the era of illegal harvesting of information is only just beginning. Lack of proper knowledge and negligence in the implementation of the TCP/IP stack make up a marriage that will always facilitate fingerprinting and dynamic development of cybercrime that employs more and more new ways and methods of its operation.

Major breakthroughs and further discoveries can be expected in fingerprinting of the application layer. In contrast to the other layers of the TCP/IP model, it is exactly at this level that tens of new programs are created with newly implemented protocols. It can be concluded that the greater the number of new applications, the greater the chances of finding a mistake, and the factors that facilitate its occurrence go beyond big corporations that develop operating systems.

To sum up, the aim of the publication was to draw attention to a safety problem associated with different approaches to program developers at the implementation of TCP/IP stack and motivate developers for better protection of their products. At the same time to warn against the dangers of a regular user and ease of how you can become the object of attack or scan carried out by an unauthorized person. The development of Fingerprinting definitely will occur in the coming years as well as we use computer networks as a preventive or diagnostic tool, or as a tool for obtaining information unlawfully. This situation may change, if the developers will pay more attention to customer data security and documents standardization.

References

1. Montigny-Leboeuf, A. D. A Multi-Packet Signature Approach to Passive Operating System Detection [Text] / A. D. Montigny-Leboeuf. – Communications Research Centre Canada, 2005
2. Arkin, O. Active & Passive Fingerprinting of Microsoft based Operating Systems using the ICMP Protocol [Text] / O. Arkin. – BlachHat Windows 2k Security, 2001.
3. Lloyd, G. G. Evaluating Tests used in Operating System Fingerprinting [Text] / G. G. Lloyd, J. T. Tavaris // LGS Bell Labs Innovations Technical Memorandum TM-071207. – Available at: http://alcatel-lucent.de/wps/DocumentStreamerServlet?LMSG_CABINET=LGS_Resources&LMSG_CONTENT_FILE=LGS_White_Papers/GreenwaldThomasTM-071207.pdf
4. Pallavi, M. TCP/IP STACK Fingerprinting [Text] / M. Pallavi // Raport. – 12 p.
5. João Paulo, S. M. A Qualitative Survey of Active TCP/IP Fingerprinting Tools and Techniques for Operating Systems Identification [Text] / S. M. João Paulo, B. J. Agostinho de Medeiros, P. Paulo S. Motta // 4th International Conference CISIS, 2011. – P. 68–75. doi: 10.1007/978-3-642-21323-6_9
6. Allen, J. M. OS and Application Fingerprinting Techniques [Text] / J. M. Allen. – SANS Institute InfoSec Reading Room, 2007. – 49 p.
7. Templeton, S. J. Detecting Spoofed Packets [Text] / S. J. Templeton, K. E. Levitt // In proceedings of the DARPA Information Survivability Conference and Exposition, 2003.
8. Beverly, R. Server Siblings: Identifying Shared IPv4/IPv6 Infrastructure via Active Fingerprinting [Text] / R. Beverly, A. Berger // 16th International Conference, PAM 2015, 2015. – P. 149–161. doi: 10.1007/978-3-319-15509-8_12
9. Auffret, P. SinFP, Unification Of Active And Passive Operating System Fingerprinting [Text] / P. Auffret // Journal in Computer Virology. – 2010. – Vol. 6, Issue 3. – P. 197–205. doi: 10.1007/s11416-008-0107-z
10. Arkin, O. Revolutionizing Active Operating System Fingerprinting The Present & Future of Xprobe2 [Text] / O. Arkin. – Sys-Security Group, 2003.
11. Fyodor. Nmap: Scanning the Internet [Text]. – Black Hat Briefings Defcon 16, 2008.
12. Maurice, C. Improving 802.11 Fingerprinting of Similar Devicesby Cooperative Fingerprinting [Text] / C. Maurice, S. Onno, C. Neumann, O. Heen, A. Francillon // SECRYPT 2013, 10th International Conference on Security and Cryptography, 2013.
13. Tomaszewski, M. Haking – Sprzątanie pajęczyn -detekcja nielegalnego współdzielenia łącza [Text] / M. Tomaszewski, M. Szmit, M. Gusta // Hakin9. – 2005. – Vol. 2. – P. 10
14. Sanders, C. Operating System Fingerprinting with Packets [Electronic resource] / C. Sanders. – 2011. – Available at: http://www.windowsecurity.com/articles-tutorials/intrusion_detection/Operating-System-Fingerprinting-Packets-Part1.html
15. Shu, G. Network Protocol System Fingerprinting – A Formal Approach [Text] / G. Shu, D. Lee // Proceedings of 25th IEEE International Conference on Computer Communications, 2006. – P. 12. doi: 10.1109/infocom.2006.157
16. Sobolewski, P. Czy da się oszukać fingerprinting warstwy aplikacji [Text] / P. Sobolewski // Hakin9. – 2006. – Vol. 6. – P. 15
17. Zalewski, M. Cisza w sieci [Text] / M. Zalewski. – Helion, 2005. – 304 p.
18. Arkin, O. Xprobe - Remote ICMP Based OS Fingerprinting Techniques [Text] / O. Arkin. – Managing Security Architect, 2001.
19. Crowley, D. Advanced application-level OS fingerprinting: Practical approaches and examples [Text] / D. Crowley. – 2008
20. Prigent, G. IpMorph: Fingerprinting spoofing unification [Text] / G. Prigent, F. Vichot, F. Harrouet // Journal in Computer Virology. – 2010. – Vol. 6, Issue 4. – P. 329–342. doi: 10.1007/s11416-009-0134-4