14. Medical Devices and Human Engineering Four Volume Set [Text] / E. J. D. Bronzino, D. R. Peterson (Eds.). – CRC Press, 2014.
15. Becchetti, C. Medical instrument design and development: from requirements to market placements [Text] / C. Becchetti, A. Neri. – JohnWiley & Sons Ltd, 2013. – 891.
16. Рыков, С. А. Медико-социальный мониторинг в системе охраны зрения школьников [Текст] / С. А. Рыков, Н. М. Орлова, А. А. Костецкая // Офтальмология. Восточная Европа. – 2013. – № 2. – С. 105–111.
17. Литвиненко, М. В. Принципи національної системи охорони здоров'я в Україні [Текст] / М. В. Литвиненко // Теорія та практика державного управління. – 2015. – Вип. 2 (49). – С. 198–205.
18. Фірсова, О. Д. Система охорони здоров'я Норвегії, особливості її організації на муніципальному рівні: досвід для України [Текст] / О. Д. Фірсова // Економіка та держава. – 2011. – № 1. – С. 100–104.
19. Высоцкая, Е. В. Информационная система ранней диагностики первичной открытоугольной глаукомы [Текст] / Е. В. Высоцкая, А. Н. Страшненко, С. А. Синенко, Ю. А. Демин // Радіоелектронні і комп'ютерні системи. – 2012. – Вип. 1 (53). – С. 105–109.
20. Kusek, J. Z. Ten steps to a results-based monitoring and evaluation system : a handbook for development practitioners [Text] / J. Z. Kusek, R. C. Rist. – Washington, DC: The World Bank, 2004. – 248 p.
21. Cherednichenko, O. Models of Research Activity Measurement: Web-Based Monitoring Implementation [Text] / O. Cherednichenko, O. Yanholenko, O. Iakovleva, O. Kustov // Lecture Notes in Business Information Processing. – 2014. – Vol. 193. – P. 75–87. doi: 10.1007/978-3-319-11373-9_7
22. Liu, B. Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data. 2nd edition [Text] / B. Liu. – Springer, 2011. – 622 p. doi: 10.1007/978-3-642-19460-3
23. e-Health Sensor Platform V2.0 for Arduino and Raspberry Pi. Biometric/Medical Applications [Electronic resource]. – Available at: https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical
24. Агаханян, Т. М. Электронные устройства в медицинских приборах [Текст] / Т. М. Агаханян, В. Г. Никитаев. – М.: НИЯУ МИФИ, 2010. – 480 с.

UDK 004.738.5.057.4

DOI: 10.15587/1729-4061.2015.51352

ANALYSIS OF AN ACTIVE FINGERPRINTING APPLICATION OF THE TRANSPORT LAYER OF TCP/IP STACK FOR REMOTE OS DETECTION

V. Mosorov
Doctor of Technical Science
Department of Computer Science in Economics
Narutowicha str., 65, Lodz, Poland, 90-131
E-mail: wmosorow@uni.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

Активний фінгерпрінтинг є процесом передачі модифікованих чи дивно відформатованих пакетів на цільову операційну систему та аналізу її відповіді для пошуку вразливих місць. Здійснено порівняльний огляд методів активного фінгерпрінтингу, які використовуються в транспортному (четвертому) рівні в стеку Інтернет протоколу TCP/IP. Також продемонстровано різні реакції операційних систем на проведені тести

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

Активный фингерпринтинг является процессом передачи модифицированных или странно отформатированных пакетов на целевую операционную систему и анализа ее ответа для поиска уязвимых мест. Осуществлен сравнительный обзор методов активного фингерпринтинга, которые используются в транспортном (четвертом) уровне в стеке Интернет протокола TCP/IP. Также продемонстрированы различные реакции операционных систем на проведенные тесты

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

1. Introduction

Today's cyber world is more than programs, computer games, or even the Internet. This is interconnected networks, containing a telecommunications networks, embedded systems and critically important objects of infrastructure, which are closely linked with each other and with the user. Malicious attacks on critical infrastructure are a serious threat for a physical person, firms, business and even government operations. Today, the cyber criminal can become anyone with desire and with the necessary training. Many people want with the minimal technical means and with a sense of impunity become rich or get valuable information.

Cyber-actions are divided into cyber-attacks, cyber-crime and cyber-warfare. A cyber-attack is one of the cyber actions and it consists of any action taken to undermine the functions of a computer network for a political or national security purpose [1]. Cyber-attacks objective is to undermine the function of a computer network and it must have a political or national security purpose. Cyber-warfare is a same as cyber-attack but it is narrower term, when the effects are equivalent to an "armed attack," or activity must occur in the context of armed conflict. Cyber-crime involves only non-state actors.

By observing the normal operations of target useful information can be collected and analyzed for the future of cyber crime. Reconnaissance phase is used for the gathering the weak points of the targets system: network information (IP addresses, network topology etc.), host information (user names, versions of TCP services etc.), human data (telephone numbers, personal information, dark secrets), security policies (detection systems, firewall, password complexity, password change frequency) [2].

As one of the risks in the context of cyber crime is the remote identification operating system on the computer network. This paper aims to bring aspects of an active fingerprinting application of the transport layer in the network TCP/IP stack and the consequences of different operating system vulnerability.

2. Analysis of published data and problem statement

Active fingerprinting functions by sending oddly formatted and slightly modified TCP packets. The result is that each target responds differently to these malformed packets. Remote OS detection boils down to identifying the operating system or applications running on the scanned device, which are determined by certain methods using the slight differences between implementations of the protocol TCP/IP. Thanks to these seemingly insignificant errors, we can increase our chance to get to know important information about the device and the software used by scanned user. Among the many methods of fingerprinting one of the most interesting are those that use the fourth (transport) layer protocols of TCP/IP stack. Used methods such as Flag probing [3], Window size probing [4] Time of retransmission, where the TCP is used gives huge opportunities because of the many options that comes with the named protocol. A tool that carries out the most tests using these methods is Nmap. The result of its performance is a list of scanned addresses with additional information dependent on used options. One of the main information is a "list of interesting ports". It contains the port numbers and protocols, the names of the service and the detected condition. The condition can be described as an open [5], filtered, closed, or unfiltered. Open means that the application in the analyzed address waits for connections/packets arriving at this port. Filtered means that the system is prohibitive, or other device that block the network traffic does not allow the communication to that port and for this reason Nmap [6] is not able to determine whether the test port is open or closed. Closed port has no application that supports network communication. Ports classified as unfiltered responded to Nmap requests, but it was not possible to determine whether they were open or closed [7, 8]. Apart from the Nmap there are many different currently available programs for fingerprinting using transport layer for scanning which in greater or lesser extent are effective [9]. It should always reckon with the fact that it can never be one hundred percent [10, 11] specified a system/application of a scanned the device and the test should always take this into account.

In this article the basic methods of active fingerprinting of the transport 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 Transport TCP/IP stack.

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

1. Analysis of active fingerprinting methods, namely the transport layer.

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

4. The transport layer and its protocols

The purpose of this layer is to provide the data to the particular application. This type of transmission of the data stream to a location is based on the identification using port numbers. To each process that needs to communicate with the network is assigned a unique port. It is a "navigator" for informing the transport layer to which applications have to be sent to part of the data. Port redirects the request to a particular service which is under a given IP address. The port number is specified when connection is created. Due to the fact that different programs have different requirements, the transport layer must have specialized protocols. Some applications require delivery reliability, while for others matters speed at the cost of losing some data.

Ports are divided into:

– 0–1023 – generally known – they are registered for services and applications such as http(80), FTP(20, 21), SSH(22), Telnet(23), SMTP (25), DNS(53), Gopher(70), POP2(109), POP3(110), NETBIOS(137, 138, 139), IMAP(143), SNMP(161, 162), BGP(179), HTTPS(443) etc;

– 1024–49151 – registered ports - they are assigned to programs launched by users;

– 49152–65535 – private ports – are usually allocated dynamically to the client when it initiates the connection.

In the transport layer two protocols are distinguished:

– TCP (Transmission Control Protocol) is one of the most important protocols in TCP/IP stack. It has been entirely described in the standardization document RFC 793 as a connection protocol enabling initiate first connection wherein the data is efficiently and reliably sent on. This connection can control the flow, send an acknowledgment of receipt, divide larger data on parts and correctly submit them in the order on the destination host and perform the retransmission. A characteristic feature of this protocol is a way to connect, so-called three-way handshake. Host starting transmission sends a packet containing a TCP segment with the SYN flag set. The call recipient sends a package with the set SYN and ACK. flags. The initiating of the communication should now send the first portion of the data by setting only ACK flag. If the destination node does not want to connect or cannot pick him up, then it should respond by the packet with a set RST flag. Successful completion of the communication is based on sending FIN flag. The above described three-way handshake procedure is shown in a Fig. 1.

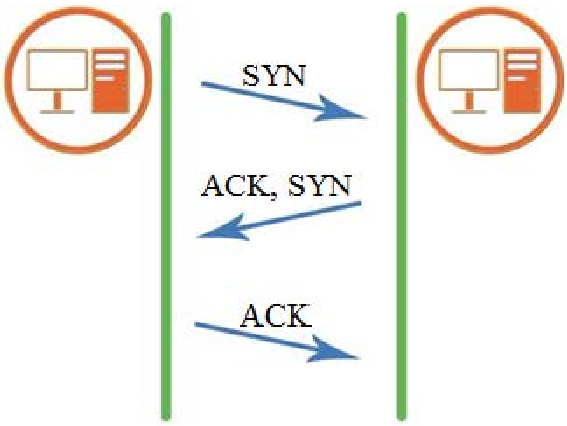

Fig. 1. Three-way handshake

Reliability in providing data provides a method called positive acknowledgment with retransmission. It requires that a recipient communicate with the broadcaster, indicating it using the sequential number field and confirmation of whether the data has been correctly delivered. The broadcaster waits for confirmation, and then depending on the situation accedes to the retransmission of damaged, lost data or continues the transmission of others.

– UDP (User Datagram Protocol) is a protocol TCP/IP standard defined in RFC 768. Its characteristic feature is the lack of establish connection before transmission and the lack of data retransmission mechanisms in a place of damaged or lost data. This design of datagram makes it much less of a TCP packet, and so the number of outgoing control information is reduced. This type of construction enables faster transmissions, which allow applications to directly benefit from IP services.

5. Fingerprinting of the transport layer

As it is described before the transport layer consists of two of two major protocols TCP and UDP. In the experiments [12] whose purpose is to decrypt the type of system on a scanned machine the first of these protocols will be used. As it turns out a huge amount of options offered those used, as well as those who currently have no application, it is used in various types of fingerprinting tests. In the case of TCP it gives us the quite a lot of serious discrepancies between systems, which include:

– Flag probing – to carry out this test we use the “flags” in the TCP protocol. We distinguish the following tests:

○ FIN probe – this test is based on sending a segment without the set of SYN and ACK flag to an open port of the scanned machine. According to the standardization document RFC 793, the system should react sending a response with a set of the RST flag. Up to this point everything is clear and understandable, there should not be any problems with implementation. However, further part of the message sounds: RST should not be sent if the system is not sure that the segment does not belong to the communication conducted by this port. This inaccuracy in the document meant that many systems rejects quietly modified by us segment, while others send back a message with the RST flag for example Cisco IOS, Microsoft products.

○ The Bogus flag – is based on sending an undefined flag with SYN flag set in the TCP segment. Versions 2.0.35 of Linux system not only return an answer, but also post the copy of the values of our manipulated flag in it. According to Gordon Lyon, “Nmap” developer that test identifies the above-mentioned system, although other sources indicate that there are groups other OS’s that respond to such a segment of RST flag.

○ ACK Values – that test draws attention to certain exceptions in the implementation behavior of TCP/IP stack. Our attention should attract the irregularity in the values of sequence numbers. If we send a segment FIN | PSH | URG to a closed port of the scanned machine, then most systems will answer with ACK segment for the same sequence number that was in our message. However, there will also be a group of OS, which increases the value by one, or insert pseudo-random value. The situation is similar with sending SYN | FIN | PSH | URG segment to an open port. Both experiments using irregularity in the ACK responses are implemented in the “Nmap” as a test T7, T3.

– Window size probing [13] – this test relies on observations the value of the “window” in packages that come back. As it turns out, their size may vary considerably, depending on the used data link layer technology and the type of incoming message. This can be seen by the following example in Table 1.

Table 1

Used technologies

Used technology	Maximum Transfer Unit (MTU)
Hyperchannel	65535
16 Mbits/sec token ring (IBM)	17914
4 Mbits/sec token ring (IEEE 802.5)	4464
FDDI	4352
Ethernet	1500
IEEE 802.3/802.2	1492
X.25	576
Point-to-Point (low delay)	296

Before analyzing the table with the tests should be remembered [14]: MSS is the field that informs us of the maximum segment size that cannot be divided into parts. The most common value set by the system can be seen in Table 2.

Table 2

Database of window field

System	Window	
	RST/ACK	SYN/ACK
BEOS 5	0	12288
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	0	12*MSS
FreeBSD 2.2.9, 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	0	12*MSS
FreeBSD 4.5, 4.6, 4.6.2, 4.7, 4.8	0	65535
FreeBSD 4.6, 4.6.2, 4.7, 4.8	0	57344
FreeBSD 5.0, 5.1	0	65535
Linux 2.0.29 (Debian)	0	15360
Linux 2.0.30 (RedHat)	0	31744
Linux 2.0.32, 2.0.36 (RedHat)	0	32736
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, 2.2.20, 2.2.21, 2.2.22, 2.2.23, 2.2.24	0	22*MSS
Linux 2.2.19, 2.2.20-idepci (Debian)	0	11*MSS
Linux 2.4.0, 2.4.1, 2.4.2, 2.4.2-2, 2.4.3, 2.4.4, 2.4.4-4GB	0	4*MSS
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, 2.4.18-14, 2.4.19, 2.4.19-4GB, 2.4.20, 2.4.20-8, 2.4.21-0.13mdk	0	4*MSS
MacOS 7.5.3, 7.5.5	0	12*MSS
MacOS 7.6, 7.6.1, 8.0, 8.1	0	12*MSS / 44*MSS
MacOS 9 9.0, 9 9.1, 9.2.1, 9.2.2	0	32768 / 65535
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	0	23*MSS
NetBSD 1.1, 1.2, 1.2.1	0	12*MSS
NetBSD 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	0	16384
Netware 4.11	0	2000 / 32768 / 65535
Netware 4.11 sp9	0	6144
Netware 5	0	8191 / 32768 / 65535
Netware 5 sp6a	0	6144
Netware 5.1	0	8191 / 65535
Netware 5.1 sp6	0	6144
Netware 6, 6 sp3	0	6144
OpenBSD 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7	0	12*MSS
OpenBSD 2.8, 2.9, 3.0, 3.1, 3.2, 3.3	0	12*MSS / 17366
QNX RTP 4, 6.0	Duplicates the value	8192
QNX RTP 6.1, 6.2, 6.2.1	0	16384
SunOS 5.5, 5.5.1	0	6*MSS
5.6	0	44*MSS /6*MSS
5.7	0	6*MSS
5.8	0	17*MSS
5.9	0	34*MSS
SunOS (Intel) 5.8	0	44*MSS
Windows 95	0	6*MSS
Windows NT 3.51 standard	0	6*MSS
Windows 98, 98 SE	0	6*MSS
Windows NT 4 standard, sp3, sp4, sp6	0	12*MSS
Windows Millennium standard	0	12*MSS
Windows 2000 standard, sp2, sp3, sp4	0	12*MSS
Windows XP Home, Professional	0	12*MSS
Windows 7	0	12*MMS
Windows Server 2008	0	12*MMS
Windows 8, Windows 8.1	0	12*MMS

– Time of Retransmission – this scanning uses the mileage of start communicating using the "three-way handshake" method. The creators of the fluctuations are Veysset, Courtay, Heen, who in 2002 presented this method of fingerprinting. This test does not require any effort from the striker. It is based on a sending of a packet with SYN flag set as information for the other machine of its wish to start "dialogue". A further part of the experience relies on listening. The scanned operating system will react as it is described in the standardization document RFC 793 by sending a SYN + ACK response [15]. The user receives a scanning packet, but still remains in hiding, without giving any sign of life. The observed OS after a specified period comes to the conclusion that package with its response was lost during transmission, so it sends the next. This situation is repeated several times, in ever larger intervals (increasing intervals is intended to prevent the network overloading, if there are any technical problems indeed). The system then comes to the conclusion that, no one wants to communicate with it on a particular port. At this time another irregularity in the systems behavior may appear. Some of them, when it realize that communication is not going to happen send a packet with a RST flag, while others simply interrupt broadcast on the last sent reply that can be seen in Fig. 2.

Microsoft Windows Vista, Windows 7 or Windows 8 and newer OS use a new algorithm protecting against attacks and scanning with the SYN packets. It implies that the use of the "Time of Retransmission" method becomes ineffective in this case.

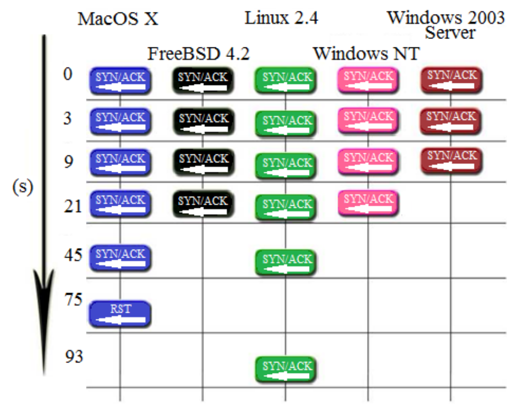

Fig. 2. Time of Retransmission

– Options sequence – this test is performed by sending a packet with SYN flag to an open or close port with a set of particular additional options in the TCP header. Requests of additional parameters from the scanned machine is best to set for achieving transparency in alphabetical order (L – "End of list options" M – "Maximum segment size" N – "Do not execute" S – "Selective confirmation" T – "Date Stamp" W – "Scaling windows "), omitting the" Do not execute". As it turns out operating systems react in different ways, responding in their reply packets SYN / ACK or RST / ACK (depending on the state of the port, to which was sent our SYN) addressing options in different sequences, adding a few of its parameters, or failing to respond to some of them that we see in Table 3.

Table 3

Database of imprints – the sequence of elements of the "Option" field

System	TCP OPTIONS	
	Sent in the SYN package	Received in the SYN/ACK package
1	2	3
BEOS 5	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460
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	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NW@0NNT
FreeBSD 2.2.9, 3.0, 3.1, 3.2, 3.3, 3.4, 3.5.1, 4.0, 4.1, 4.1.1, 4.2, 4.3	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460
FreeBSD 4.4	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NW @ 1NNT
FreeBSD 4.5	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NW @ 1 NNT
FreeBSD 4.6, 4.6.2, 4.7, 4.8	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NW @0NNT
FreeBSD 5.0, 5.1	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NW @ 1 NNT

Continuation of Table 3

1	2	3
Linux 2.0.29 (Debian)	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460
Linux 2.0.30 (RedHat)	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460
Linux 2.0.32, 2.0.36 (RedHat)	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460
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, 2.2.20, 2.2.21, 2.2.22, 2.2.23, 2.2.24	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460STNW@0
Linux 2.2.19, 2.2.20-idepci (Debian)	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460STNW@0
Linux 2.4.0, 2.4.1, 2.4.2, 2.4.2-2, 2.4.3, 2.4.4, 2.4.4-4GB	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460STNW@0
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, 2.4.18-14, 2.4.19, 2.4.19-4GB, 2.4.20, 2.4.20-8, 2.4.21-0.13mdk	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460STNW@0
MacOS 7.5.3, 7.5.5	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460 / M@ 1460W @0L
MacOS 7.6, 7.6.1, 8.0, 8.1	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460 / M@ 1460W @0L
MacOS 9 9.0, 9 9.1, 9.2.1, 9.2.2	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460W@0NNNT / M@1460W@2NNNT
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	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NW @0NNT
NetBSD 1.1, 1.2, 1.2.1	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NW @0NNT
NetBSD 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	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NW @0NNT
NetBSD 1.6, 1.6.1	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@ 1460NW @0NNT @0
Netware 4.11, 4.11 sp9, 5, 5 sp6a, 5.1	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460
Netware 5.1 sp6, 6, 6 sp3	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460W@0NSNN
OpenBSD 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2, 3.3	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NW @0NNT

Continuation of Table 3

1	2	3
QNX RTP 4	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460
QNX RTP 6.0	M@1459	M@1460
	M@1460	M@1459
	M@1460STW	M@1459 / M@1460
QNX RTP 6.1, 6.2, 6.2.1	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NW @0NNT
SunOS 5.5, 5.5.1	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	M@1460
SunOS 5.6	M@1459	M@1459
	M@1460	M@1460
	M@1460STW	NNTNW@0M@ 1460 / NNT-NW @1M@1460
SunOS 5.7	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	NNTNW @0M@ 1460
SunOS 5.8	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	NNTNW @0NN SM@ 1460
SunOS 5.9	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	NNTM@ 1460NW @0NN S
SunOS (Intel) 5.8	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	NNTNW@ 1NNSM@1460
Windows 95	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460
Windows NT 3.51 standard	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460
Windows 98, 98 SE	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460NNS
Windows NT 4 standard, sp3, sp4, sp6	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@1460
Windows Millennium standard	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@ 1460NW @0NNT @0NNS
Windows 2000 standard, sp2, sp3, sp4	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@ 1460NW @0NNT @0NNS
Windows XP Home, Professional	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@ 1460NW @0NNT @0NNS
Windows Vista, Windows 7	M@1459	M@1460
	M@1460	M@1460
	M@1460STW	M@ 1460NW @0NNT @0NNS

As it can be seen from the Table 3, there are systems and devices that react quite irregularly on the performed test, significantly allowing its identification in the network.

– TCP Timestamp – this test uses TCP additional option called "date stamp". As previously mentioned, this feature is intended to help in the diagnosis of calculating the time it takes for messages traveling from one place to another. This option consists of four fields, two of which are worthy of interest. These are the "Timestamp Value" or in short TSval [16], which informs us about the exact time of sending by a given TCP/IP communication, and the field "Timestamp Echo Reply Field" in short Tsecr, which contains the time the message, was received by the second system. Both of the data are taken from the so-called timestamp clock. Each of the clocks is periodically incremented by the operating system in order to give the most current time. The important factor [17] is the frequency at which the clock timestamp gets updated. As it turns out not all systems increase in the same period of time, the value of the clock by the same time unit [18] what can be seen in Table 4.

– TCP ISN – this test is based on the relationship [19] of the TCP sequence numbers. This experience uses the value of the consecutive numbers which are allocated to this field in the response of our sent packages. In practice, this consists in "filling up" on a variety of scanned system open ports of SYN packages. The OS we are interested at sends the messages in response to SYN/ACK with further sequence numbers set by itself. On this basis, we are able to determine what their assignment algorithm was used, and hence the name of the system being scanned or the group to which it belongs. When performing this test, it must be remembered that it can be easily mistaken. The scanned system is capable during our experience make other connections, which resulted in significant variance of the sequence numbers [20]. here are the following ways to generate successive sequence numbers:

○ Constant (Const) – number in the return SYN/ACK package is the same which in sent in.

○ 64k – the numbers in the return SYN/ACK package are a multiple of the value 64000.

○ 800x – the numbers in the return SYN/ACK package are a multiple of the value 800.

○ RPI (Random positive increments) – the numbers in the return SYN/ACK package are random values greater than the previous value of SEQ.

○ RI (Random increments) – the numbers in the return SYN/ACK package are random values.

○ TR (True random) – the numbers in the return SYN/ACK package are the truly random values (probability of selecting the numbers are the same).

○ TDI (Time dependent increments) – the numbers in the return SYN/ACK package are dependent on the time value.

In Table 5 it could be distinguished a several different anomalies observed in operating systems.

Table 4

Database of "timestamp" option

System	Timestamp clock update
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	2 times per second
FreeBSD 4.4, 4.5, 4.6, 4.6.2, 4.7, 4.8, 5.0, 5.1	100 times per second
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	100 times per second
Linux 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	100 times per second
Linux 2.2.20, 2.2.20-idepci, 2.2.21, 2.2.22, 2.2.23, 2.2.24	100 times per second
Linux 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	100 times per second
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	100 times per second
Linux 2.4.18-14	500 times per second
Linux 2.4.19, 2.4.19-4GB, 2.4.20, 2.4.20-8, 2.4.21-0.13mdk	100 times per second
MacOS 9.0, 9.1, 9.2.1, 9.2.2	1000 times per second
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	2 times per second
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	2 times per second
NetBSD 1.6, 1.6.1	Timestamp Value = 0 until establish a connection
OpenBSD 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2, 3.3	2 times per second
QNX RTP 6.2, 6.2.1	2 times per second
SunOS 5.5, 5.5.1, 5.6, 5.7, 5.8, 5.9	100 times per second
SunOS (Intel) 5.8	100 times per second
Windows 95	Timestamp Value = 0 until establish a connection
Windows NT 3.51 standard	Timestamp Value = 0 until establish a connection
Windows 98, 98 SE	Timestamp Value = 0 until establish a connection
Windows NT 4 standard, sp3, sp4, sp6	Timestamp Value = 0 until establish a connection
Windows Millennium standard	Timestamp Value = 0 until establish a connection
Windows 2000 standard, sp2, sp3, sp4	Timestamp Value = 0 until establish a connection
Windows XP Home, Professional	Timestamp Value = 0 until establish a connection
Windows Net standard	Timestamp Value = 0 until establish a connection
Windows 2003 Server standard	Timestamp Value = 0 until establish a connection
Windows Vista	Timestamp Value = 0 until establish a connection
Windows 7	Timestamp Value = 0 until establish a connection
Windows Server 2008	Timestamp Value = 0 until establish a connection

Table 5

Database – methods of generating the sequence numbers

System	ISN
BEOS 5	RI
Commodore 64	Const
FreeBSD 2.0.5, 2.1.0, 2.1.5, 2.1.6, 2.1.7.1	64k
FreeBSD 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	RI
FreeBSD 4.4, 4.5, 4.6, 4.6.2, 4.7, 4.8, 5.0, 5.1	TR
IBM OS/2	800x
Linux 2.0.29, 2.0.30, 2.0.32, 2.0.34, 2.0.36	TR
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	RI
Linux 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	RI
Linux 2.2.20, 2.2.20-idepci, 2.2.21, 2.2.22, 2.2.23, 2.2.24	RI
Linux 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	RI
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	RI
Linux 2.4.18-14	RI
Linux 2.4.19, 2.4.19-4GB, 2.4.20, 2.4.20-8, 2.4.21-0.13mdk	RI
MacOS 7.6, 7.6.1, 8.0, 8.1	64k
MacOS 9.0, 9.1, 9.2.1, 9.2.2	RI
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	TR
NetBSD 1.1, 1.2.1	64k
NetBSD 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	RI
NetBSD 1.6, 1.6.1	RI
Netware 4.11	TDI
Netware 4.11 sp9	RI
Netware 5	TDI
Netware 5 sp6a	RI
Netware 5.1	RI
Netware 5.1 sp6	TR
Netware 6	RI
Netware 6 sp3	TR
OpenBSD 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8	RI
OpenBSD 2.9, 3.0, 3.1, 3.2, 3.3	TR
QNX RTP 6.2, 6.2.1	RI
SunOS 5.5, 5.5.1, 5.6, 5.7, 5.8, 5.9	RI
SunOS (Intel) 5.8	RI
Windows 95	TDI
Windows NT 3.51 standard	TDI
Windows 98, 98 SE	TDI
Windows NT 4 standard, sp3, sp4, sp6	TDI
Windows NT 4 sp6	RI
Windows Millennium standard	RI
Windows 2000 standard, sp2, sp3, sp4	RI
Windows XP Home, Professional	RI
Windows Net standard	RI
Windows 2003 Server standard	TR
Windows Vista	RI
Windows 7	RI

On the basis of the above table, even those systems from the same family can generate different behaviors.

6. Conclusions

Looking at such a growing market for operating systems and the growing group of regular users who use the computer only as a tool for surfing the Internet it can be mentioned that the number of incidents involving theft, destruction of confidential data will increase a lot.

Summarizing, based on the comparative active fingerprinting tests and analysis of obtained results using this publication the interest of Internet users in active Fingerprinting is expected to grow. Transport layer protocols provide process-to-process connectivity across the Internet. Due to the fact that layer is fundamental to Internet connectivity and that many attack tools probe TCP ports in an attempt to discover vulnerabilities.

Also in this article were submitted a series of tests that help to identify the system on the other side, namely: Flag probing, Window size probing, Time of Retransmission, Options sequence, TCP Timestamp, TCP ISN. The reaction of operating systems on carried out tests was ambiguous, in some cases operating systems from even one family react differently to the same test.

A general overview of the main operating system reactions is intended to help users protect their computers, or at worst to own some basic information about who and how can illegally make a crime.

References

1. Hathaway, O. A. The law of cyber-attack [Text] / O. A. Hathaway, R. Crootof, P. Levitz, H. Nix, A. Nowlan, W. Perdue, J. Spiegel. – Yale Law School Legal Scholarship Repository, 2011. – 76 p.
2. Sanghvi, H. P. Cyber Reconnaissance: An Alarm before Cyber Attack [Text] / H. P. Sanghvi, M. S. Dahiya // International Journal of Computer Applications. – 2013. – Vol. 63, Issue 6. – P. 36–38. doi: 10.5120/10472-5202
3. Schiffman, M. Building Open Source Network Security Tools: Components and Techniques [Text] / M. Schiffman. – Wiley, 2002. – 416 p.
4. Lyon, G. Nmap Network Scanning [Text] / G. Lyon. – 2013. – 467 p.
5. Hills, R. NTA Monitor UDP Backoff Pattern Fingerprinting White Paper [Electronic resource] / R. Hills. – 2003. – 7 p. – Available at: http://www.nta-monitor.com/files/udp-backoff-whitepaper.pdf
6. Allen, J. M. OS and Application Fingerprinting Techniques [Text] / J. M. Allen. – SANS Institute InfoSec Reading Room, 2007. – 49 p.
7. Lyon, G. Nmap Scripting Engine Documentation [Text] / G. Lyon. – 2013. – 42 p.
8. Bennieston, A. J. NMAP-A Stealth Port Scanner [Text] / A. J. Bennieston. – 2006. – 20 p. – Available at: http://www.csc.villanova.edu/~nadi/csc8580/S11/nmap-tutorial.pdf
9. Spangler, R. Analysis of Remote Active Operating System Fingerprinting Tools [Text] / R. Spangler. – University of Wisconsin – Whitewater, 2003. – 36 p.
10. Schwartzenberg, J. Using Machine Learning Techniques for Advanced Passive Operating System Fingerprinting [Text] / J. Schwartzenberg. – Essay (Master), 2010.

11. Jirsik, T. Identifying Operating System Using Flow-based Traffic Fingerprinting [Text] / T. Jirsik, P. Celeda // 20th EUNICE/IFIP EG 6.2, 6.6 International Workshop, 2014. – P. 70–73. doi: 10.1007/978-3-319-13488-8_7
12. Arkin, O. Xprobe - Remote ICMP Based OS Fingerprinting Techniques [Text] / O. Arkin. – 2001
13. Zalewski, M. Cisza w sieci [Text] / M. Zalewski. – Helion, 2005. – 304 p.
14. Montigny-Leboeuf, A. De. A Multi-Packet Signature Approach to Passive Operating System Detection [Text] / A. De Montigny-Leboeuf. – Communications Research Centre Canada, 2005.
15. Gutkowski, M. Kilka ciekawych metod rozpoznawania systemu operacyjnego [Text] / M. Gutkowski // Hakin9. – 2004. – Vol.2
16. Allen, J. M. OS and Application Fingerprinting Techniques [Text] / J. M. Allen. – SANS Institute InfoSec Reading Room, 2007. – 49 p.
17. 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.
18. 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
19. Lippmann, R. Passive Operating System Identification From TCP/IP Packet Headers [Text] / R. Lippmann, D. Fried, K. Piwowarski, W. Streilein. – Springer, 2005.
20. Nostromo, Techniques in OS-Fingerprinting [Text] / Nostromo. – Hagenberg, 2005. – 24 p.

УДК 004.925.8: 624.953: 004.03
DOI: 10.15587/1729-4061.2015.50980

ИССЛЕДОВАНИЕ МАКЕТА КАНАЛА ИЗМЕРЕНИЯ ВИБРАЦИИ КОМПЛЕКСНОЙ СИСТЕМЫ МОНИТОРИНГА СТАЛЬНЫХ РЕЗЕРВУАРОВ

Н. И. Бурау
Доктор технических наук,
профессор, заведующий кафедрой*
E-mail: burau@pson.ntu-kpi.kiev.ua
С. А. Цыбульник
Аспирант*
E-mail: Tsybulnik.s.a@gmail.com
Д. В. Шевчук
Аспирант*
E-mail: 00012066@ukr.net
*Кафедра приборов и систем ориентации и навигации
Национальный технический университет Украины
«Киевский политехнический институт»
пр. Победы, 37, г. Киев, Украина, 03056

Розроблено програму обробки сигналів в математичному пакеті MATLAB і віртуальний прилад з підтримкою до семи вимірювальних каналів. Проведено оцінку корисних сигналів. У графічному пакеті CATIA створено тривимірну модель резервуара об'ємом 0,04 м^3. Проведено модальний аналіз конструкції з використанням програмного комплексу ANSYS. Показано, що ефективність макета каналу вимірювання вібрації становить понад 90 %

Ключові слова: вібраційна діагностика, вертикальний сталевий резервуар, LabVIEW, діагностичний комплекс, ANSYS

Разработана программа обработки сигналов в математическом пакете MATLAB и виртуальный прибор с поддержкой до семи измерительных каналов. Проведена оценка полезных сигналов. В графическом пакете CATIA создана трехмерная модель резервуара объемом 0,04 м^3. Проведен модальный анализ конструкции с использованием программного комплекса ANSYS. Показано, что эффективность макета канала измерения вибрации составляет свыше 90 %

Ключевые слова: вибрационная диагностика, вертикальный стальной резервуар, LabVIEW, диагностический комплекс, ANSYS

1. Введение

В последние десятилетия во многих развитых странах мира большое внимание уделяется обеспечению безопасной эксплуатации сложных инженерных сооружений и конструкций различного назначения, к которым относятся такие ответственные объекты, как: мосты, гидротехнические сооружения, хранилища опасных веществ, электростанции, объекты газо- и нефтетранспортной области, а также другие. На сегодняшний день проектирование большинства ответственных конструкций основывается на принципе