© Chumachenko K., Chumachenko D. Study of snort performance in counteracting port scanning techniques // Ukrainian Scientific Journal of Information Security, 2017, vol. 23, issue 1, p. 15-18.

DOI: 10.18372/2225-5036.23.11546

STUDY OF SNORT PERFORMANCE IN COUNTERACTING PORT SCANNING TECHNIQUES

Kateryna Chumachenko[1], Dmytro Chumachenko[2]

[1]Kharkiv National University of Radioelectronics, Ukraine
[2]National Aerospace University «Kharkiv Aviation Insitute», Ukraine

CHUMACHENKO Kateryna

Year and place of birth: 1997 year, Kharkiv, Ukraine.
Education: Kharkiv National University of Radioelectronics, 2017 year.
Position: student of Software Engineering Department.
Scientific interests: information security, penetration testing.
Publications: about 10 publications in information security.
E-mail: kateryna.chumachenko@gmail.com

CHUMACHENKO Dmytro

Year and place of birth: 1989 year, Kharkiv, Ukraine.
Education: National Aerospace University, 2011 year.
Position: teaching assistant of Informatics Department.
Scientific interests: multiagent simulation, information security.
Publications: more than 60 publications in simulation and information security.
E-mail: dichumachenko@gmail.com

Abstract. *Snort Intrusion Detection System became the de-facto standard among the software-based Intrusion Detection Systems because of the high level of customization and the relative ease of use. However, it is essential for an Intrusion Detection System not only to prevent the known attacks, but also to detect zero-day attacks and their preceding steps, such as port scans. A lot of companies neglect the security measures, associated with the prevention of the steps, preceding the attack, such as port scans. This article analyzes the performance of Snort in relation to detecting various port scanning methods and common evasion techniques, as well as the configurations that lead to the best performance. Port scanning prevention is discussed in the context of the nmap service and all the scanning techniques associated with it. Moreover, a packet defragmentation technique is discussed as the evasion technique, as well as the ways of the evasion detection. The article includes the recommendations for configuration of the Snort Intrusion Detection System for effective detection of the port scanning attacks.*

Key words: *Snort, port scanning, attack detection, zero-day attack, evasion, information security.*

Introduction

Network Security plays a crucial role in the operation of the enterprise and various measures are used to protect the company's assets. In addition to firewalls, one of the used methods are Intrusion Detection and Intrusion Prevention Systems (IDS/IPS) [1]. They are explicitly used to detect zero-day attacks, for which signatures have not been released yet. They are also good in the detection of attacks, that employ some anomalies, e.g. anomalies of traffic amount in Denial of Service (DoS) attacks. *The aim* of this research is to evaluate the performance of the de-facto standard on the market of Network-based IDSs, Snort, in relation to port scanning attacks. In this paper, it will be described how and which port scans are identified well, which evasion techniques are successful and how Snort performance can be improved.

An important goal of any IDS is to prevent not only attacks, but also their preceding activities, such as port scanning. However, most studies of prevention of attacks, neglect these actions and explore the attack only. This paper offers a *new perspective* on the detection and prevention of attacks, focusing on Snort capabilities to detect and prevent the port scanning.

Snort IDS. Snort is an open-source and free Intrusion Detection System. The ease of rule creation and personalization for specific needs of business made it a de-facto standard in intrusion detection and prevention. Snort uses rule-based detection approach and can be installed in 3 ways: as a simple packet sniffer, packet

logger and as a Network-based Intrusion Detection System (NIDS) [2]. NIDS is the most complicated and configurable mode and this mode is described in this paper. When running as NIDS, Snort is capable to analyze the traffic and making decisions on whether it is malicious based on rules. Snort is also capable of taking proactive actions. Snort architecture can be defined in the following way: packet decoder, preprocessors, detection engine, logging and alerting system, output modules. Packet decoder is the first stage where traffic enters the IDS. Its goal is to prepare the packets to be preprocessed or to be sent to the detection engine. The second stage is the preprocessors They are responsible for packet defragmentation. Also, some simple preanalysis can be done here. It includes finding anomalies in packet headers, decoding HTTP URI, re-assembling TCP streams. Detection Engine compares the traffic to the rules and tries to find any matches. In the first releases of Snort there was a problem that when some rule was matched, the search for another matches was not performed. Because of that, there was a possibility to create an alert of lowest criticality, although there might be something more severe. In later versions of Snort, this problem was solved. Different rules have different priorities. If several matches are found, the highest priority rule is selected to generate the alert. After the Detection Engine has made the decision on whether to generate the alert, the logging and alerting system is responsible for sending alerts and logging packets. After all, the processing is done and alerts are generated, these modules define the types of output to be generated (simply save it to a file, MySQL database, Syslog, etc.).

Decisions are made upon Snort rules. Snort rules contain two logical parts [3]. The first part is called the rule header and the second one is the rule option. The header consists of the destination and source IP addresses, rule action, protocols and the ports. The rule option part determines the alert messages that will be sent as well as the information about the inspection ways – e.g. threshold values for the whole traffic or packet segments to be inspected to determine if the rule action should be invoked.

Port scanning. Port scanning is one of the first steps of any network-based attack. It can reveal the open ports, services running on them as well as operating system and other attributes of the scanned ports. The most common service used for port scanning is nmap scanner. It is an open source and free tool that will be used as well in this research. For better understanding of the process, it is important to take a look at the different port scans [4].

TCP SYNscan is the default scan for nmap. It is fast, relatively unobtrusive and stealthy. During this scan, a SYN packet is sent as if a real connection was going to be opened. The response is then checked. If it is a SYN/ACK packet – the port is open, while RST indicates that it is closed. If no response was received, the port is marked as filtered. Additionally, the port is marked filtered if an ICMP unreachable error was received. During this scan full connection is never established and sometimes it is referred to as half-open scanning. *TCP connect scan* is a good choice when *TCP SYNscan* is not possible, for example when the user does not have raw packet privileges. During this scan, nmap asks the operating system to establish a connection using the connect system call – it is the same system call that is used by web browsers and P2P applications. However, nmap has less control over high-level calls than with raw packets, so this type of port scan is usually less efficient than *TCP SYNscan*. During this type of scan a full connection is opened to the target port. This results in bigger delays and packet amounts to be sent as well as in high probability of detection.

UDP scan targets UDP services. Such scanning is generally slower and harder to implement than TCP, so it is often neglected. This is not a right thing to do as UDP services are also often exploitable. *UDP scan* works by sending a UDP packet for each port. If ICMP unreachable is received, the port is categorized as closed. Other errors mark the port as filtered. If a response UDP packet is received, the port is considered to be opened. If no response was received at all, theport is categorized as open or filtered, meaning that it can be either of them. UDP is very slow comparing to other port scans, since before the port can be identified as open, closed or filtered, nmap waits for the response packet, times out and has to retransmit the packet to that port. If a port is closed and an ICMP unreachable error is expected, huge delays are possible – a lot of systems put limitations on theamount of ICMP packets sent per amount of time, e.g. in Linux 2.4.20 this limitation is one per second. This results in the scan of all 65,536 packets taking more than 18 hours.

TCP NULL, TCP FIN, TCP Xmas scans exploit the RFC793 specification, that states that an incoming segment not containing a RST, causes a RST to be sent in response. So, any packet not containing SYN, RST or ACK will result in a returned RST if the port is closed and no response at all if the port is open. *Null scan* – does not set any bits at all. *FIN scan* – Sets only the TCP FIN bit. *Xmas scan* – Sets the FIN, PSH, URG flag. These scans result in a same behavior - if a RST packet is received, the port is considered closed, while no response means it is open or filtered. The port is marked filtered if an ICMP unreachable error is received. The main advantage of these scans is that they are very unlikely to be detected unless specific configurations are made. They are also a little bit stealthier. One of disadvantages is that it cannot distinguish open ports from open or filtered.

TCP ACK scan never determines open ports. It is used to map out firewall rulesets. The TCP ACK scan has only the ACK flag set. When scanning unfiltered systems, open and closed ports will both return a RST packet. Nmap then labels them as unfiltered, meaning that they are reachable by the ACK packet, but whether they are open or closed is undetermined. Ports that don't respond are labeled filtered.

TCP Maimon scan is named after Uriel Maimon, who developed it. The technique is the same as NULL/FIN/Xmas, except that the probe is FIN/ACK. It is already known that RST packet should be generated in response to such a probe whether the port is open or closed. However, in this scan, it was noticed that many BSD-derived systems simply drop the packet if the port is open.

IP protocol scan allows determining those supported by the machine. Technically, it is not a port scan. It works similarly to the UDP port scan – it sends IP packet headers and iterated through the 8-bit protocol field. Nmap sends packets and waits for ICMP protocol unreachable messages. Any response means that protocol is open. An ICMP protocol unreachable means that protocol is closed. Other ICMP unreachable mark the protocol as filtered.

Evasion techniques. Port scanning would be quite useless if it was easy to detect and prevent it. That's why there are certain anti-detection techniques. The most common and successful evasion technique is packet fragmentation. Using fragmented scans option results in using tiny fragmented IP packets. The idea behind this evasion technique is to split up the TCP header over multiple packets and, thus, make it harder for firewalls and IDSs to understand what is happening. Packets are split up into 8 bytes. The custom offset size can be specified by using the –MTU option (it should be a multiple of 8). This feature is obviously only supported with raw packets (not supported for TCP connect).

Another common evasion technique is using decoys. For this scan, several decoy hosts have to be identified. For the target machine, it will appear that scans are happening for several different IP addresses, but IDS won't know which IP was actually scanning. Obviously, all hosts that are specified as decoys have to be up at the moment of the scan, since if they are down, it is very easy to determine which IP address was actually performing the scan. This can be easily defeated but is generally an effective technique to hide your IP address. Decoys won't work with version detection or TCP connect scan. It is also important that decoys can slow down the scan.

The question of timing and performance is essential, especially in relation to the IDS systems. At first, knowing that UDP and IP protocol scans take a lot of time it should be ensured that they are processed quickly enough while not missing any ports and protocols. It should be confirmed that IDS detects it. There are various timing options, but this research only focuses on 6 timing templates. Using timing templates with nmap is a simple approach, offering 6 options: paranoid, sneaky, polite, normal, aggressive, insane. They are specified as an option after –T attribute (0-5). The paranoid and sneaky approach can be considered as another IDS evasion techniques. Polite mode is usually used to use fewer resources of the target machine. Normal is default. Aggressive and insane scans assume that your network is very fast and you want to sacrifice some accuracy for speed. In this research, it will be tested how different configurations affect the result of detection. But first, it is important to understand which modification to Snort can be made. sfPortscan is a package that is related to the detection of port scans. It is essential to enable it before putting Snort into real life operation. Before enabling this package, not a single port scan was detected. Frag3 preprocessor is another factor important in our research. It is responsible for packet defragmentation.

Experiments. During the experiments, several evasion techniques, preprocessor configurations and port scanning methods were used. Different timing options were also tested. The main results can be seen on the table 1.

Main results of the experiment Table 1

Port scan type	No fragmentation, without preprocessing (default)	Fragmentation, without preprocessing (default)	No fragmentation, preprocessing enabled	Fragmentation, preprocessing enabled
TCP SYN	+	+	+	+ (recognized as attempt of DoS)
TCP connect	+	n/a	+	n/a
UDP	+	+ (recognized as attempt of DoS)	+	+ (recognized as attempt of DoS)
TCP NULL	-	+ (recognized as attempt of DoS)	+	+ (recognized as attempt of DoS)
FIN	-	+ (recognized as attempt of DoS)	+	+ (recognized as attempt of DoS)
Xmas	-	+ (recognized as attempt of DoS)	+	+ (recognized as attempt of DoS)
TCP ACK	+/-	+ (recognized as attempt of DoS)	+	+ (recognized as attempt of DoS)
TCP Maimon	+/-	-	-	+ (recognized as attempt of DoS)
IP protocol	+	+	+	+ (recognized as attempt of DoS)

The first thing to do was testing the simple default scans without any additional preprocessors and preprocessor rules (frag3 is the most essential preprocessor at this point). Different kinds of tests performed differently, as it can be seen. More advanced and complicated scans like TCP NULL, FIN, XMAS were not detected. TCP ACK and TCP Maimon scans were detected several times, however, 70% of scans of TCP ACK were undetected, so they will be treated as undetected. Around 80% of TCP Maimon were detected, so they will be treated correspondingly. Next step was to test the detection of port scan using fragmentation. Fragmentation is one of the most common IDS and firewall evasion techniques, so it is important to test how IDS detects it. Surprisingly, it still detects SYN and UDP scans. Additionally, now it detects FIN, Xmas and ACK scan. However, port scans are not detected as port scans: tiny fragments make an IDS think that there is an attempt of DoS attack. It is important to understand that TCP connect scan cannot be fragmented since it relies on high-level system calls. Then the testing of how the defragmentation preprocessor affects the detection of the port scans should be performed. In this experiment, the unfragmented nmap scans are sent. The result is expected not to be changed from the default one since no fragmentation occurred and frag3 should not affect the detection in any way. Indeed, the result is the same. Probably, the only difference is related to the disappearance of uncertainties related to TCP ACK and TCP Maimon scans.

Finally, the Snort detection of fragmented packets when frag3 is used is tested. As it can be seen from the previous experiments, all successful scans were identified as an attempt of DoS attacks, while IP scan was

identified successfully and Maimon scan was not identified at all. Now the result is slightly different: any port scan is identified as being an attempt of a DoS attack. Of course, it can be said that port scans are detected, but since they are not classified correctly, a network administrator might be easily fooled, thus, this evasion techniques can be considered successful. Experiments with timing options in relation to port scanning detection resulted in quite interesting conclusions. As described above, nmap offers 6 timing options: paranoid, sneaky, polite, normal, aggressive, insane. They are specified as an option after –T attribute (0-5). –T 5 and –T 4 are the fastest options. They were always detected but were not always accurate. The results of the port scan were also insufficient in most cases. –T 0 and –T 1 were too slow and were also undetected with IDS – using these timing options is a common evasion technique. The best option in relation to the time of waiting and the performance was the – T 2 option. Such scans were detected according to my expectations and did not take a lot of time. Unexpectedly, using decoys resulted in certain mess in Snort analysis as well. Decoys allow hiding your host inside the other hosts IP addresses. It can be used as: decoy [IP1], [IP2], [IP3] etc. The target machine will see port scans coming from all the IP addresses that were specified as decoys. When port scans were performed with decoys, they were recognized randomly – it was hard for Snort to track if the port scanning actually occurred because of multiple IP addresses. Therefore, port scans were not always detected.

Conclusion

This research showed that the default configurations are not appropriate for enterprise solutions and some advanced configurations have to be made. The baselines for threshold values should be identified carefully and Snort modes should be also configured. Preprocessor requirements should be fulfilled depending on the requirements of your network. Rules should be carefully written. It is also advised to review some prewritten community packages, as they often contain rules against situations that you might have forgotten. In general, it can be concluded that Snort is a good solution if appropriate configurations are made. Its advantages are that it is the free and open source and so – highly customizable. As a future work, more research can be done in relation to evasion techniques and ways of detection of fragmented packets correctly.

References

[1] Stallings W. Computer Security: Principles and Practices / W. Stallings, L. Brown. – Harlow, UK: Pearson Education Limited, 2012. – 816 p.

[2] Lyon G.F. Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning / G.F. Lyon. – Nmap Project, 2009. – 468 p.

[3] Roesch M. Snort Users Manual 2.9.8.2. / M. Roesch. – Cisco, 2016. – 267 p.

[4] Rehman U.R. Intrusion Detection Systems with Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID / U.R. Rehman. – New Jersey, USA: Prentice Hall PTR, 2003. – 275 p.

UDC 004.457 (045)

Чумаченко К.І., Чумаченко Д.І. Дослідження ефективності Snort в протидії методам сканування портів

Анотація. *Система виявлення вторгнень Snort стала де-факто стандартом серед систем виявлення вторгнень на основі програмного забезпечення через високий рівень настроюваності і відносну простоту конфігурації. Тим не менш, вона є виключно важливою системою виявлення вторгнень не тільки для запобігання відомих атак, але і для виявлення атак нульового дня і попередніх їм дій, таких як сканування портів. Проте, як компанії, так і дослідження часто нехтують заходами безпеки, необхідними для запобігання попередніх дій, таких як сканування портів. У даній статті досліджуються ефективність Snort щодо виявлення різних методів сканування портів і популярних технік обходу, а також конфігурації, які призводять до кращої продуктивності. Запобігання сканування портів розглянуто в контексті стандартного сервісу nmap і всіх методів сканування, доступних в даному продукті. Так само розглянуто такий метод запобігання виявлення як дефрагментація пакета, а також шляхи блокування цього методу обходу виявлення. Стаття включає в себе рекомендації по конфігурації системи Snort для ефективного виявлення атак сканування портів.*

Ключові слова: *Snort, сканування порту, виявлення атаки, атака нульового дня, техніка обходу, інформаційна безпека.*

Чумаченко К.И., Чумаченко Д.И. Исследование эффективности Snort в противодействии методам сканирования портов

Аннотация. *Система обнаружения вторжений Snort стала де-факто стандартом среди систем обнаружения вторжений на основе программного обеспечения из-за высокого уровня настраиваемости и относительной простоты конфигурации. Тем не менее, она является исключительно важной системой обнаружения вторжений не только для предотвращения известных атак, но и для обнаружения атак нулевого дня и предшествующих им действий, таких как сканирование портов. Тем не менее, как компании, так и исследования часто пренебрегают мерами безопасности, необходимыми для предотвращения предшествующих действий, таких как сканирование портов. В данной статье исследуются эффективность Snort в отношении обнаружения различных методов сканирования портов и популярных техник обхода, а также конфигурации, которые приводят к лучшей производительности. Предотвращение сканирования портов рассмотрено в контексте стандартного сервиса nmap и всех методов сканирования, доступных в данном продукте. Также рассмотрен такой метод предотвращения обнаружения как дефрагментация пакета, а также пути блокировки данного метода обхода обнаружения. Статья включает в себя рекомендации по конфигурации системы Snort для эффективного обнаружения атак сканирования портов.*

Ключевые слова: *Snort, сканирование порта, выявление атаки, атака нулевого дня, техника обхода, информационная безопасность.*

Отримано *15 листопада 2016 року,* затверджено редколегією *1 березня 2017 року*