A HYBRID IDS ARCHITECTURE

Beatrice-Nicoleta CHIRIAC[1], Florin-Daniel ANTON[2], Anca-Daniela IONIȚĂ[3]

In the recent years, computing systems and applications are ubiquitous, the security and privacy are components which cannot be ignored anymore. Even if security is not a functional component of applications the concept of security and privacy by design is a hot topic addressed by companies activating in the IT field. Security has many layers and components, one of them is the monitoring and reporting component, and part of this are the intrusion detection systems (IDS). The paper is presenting an architecture of an open-source, modular, hybrid IDS system which is combining the advantages of both host and network IDS.

Keywords: hybrid intrusion detection system, cybersecurity, cyberattack, network, host, traffic capture

1. Introduction

Nowadays, when online application interaction, cloud computing, containers and other technologies which require network connectivity, are adopted almost everywhere, the cybersecurity component has a significant impact. The first aspects when networking comes into discussion are related to the distributed control, sharing tasks between machines and how they communicate to manage all their common activities. In these circumstances, a software capable to ensure the protection of computing resources and infrastructures is a growing need and different protection solutions were the topic of studies during this time. These include firewalls, antivirus applications, network behavior analyst detection (NBAD), intrusion detection system (IDS) and intrusion prevention systems (IPS) [16]. An intrusion detection system represents a passive method of supervision. Its scope is to monitor the information resulted from a networking traffic capture, from the state of the files, from the state of the processes and log files from a host [5]. In short terms, an IDS must report any unauthorized access or actions conducted inside a monitored infrastructure. A superior system is the intrusion prevention system. The intrusion prevention system relies upon the IDS features, but more than that it can respond to an attack, not only to identify and report it [2]. Additionally, systems

[1] PhD student, Faculty of Automatic Control and Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: beatrice.chiriac@stud.acs.upb.ro
[2] Reader, Faculty of Automatic Control and Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: florin.anton@upb.ro
[3] Prof., Faculty of Automatic Control and Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: anca.ionita@upb.ro

which combine the specific of an IDS with the one of an IPS are suggestively called IDPS.

The paper is proposing a hybrid IDS solution. The aspire of this objective is to combine the benefits offered by different type of intrusion detection systems (network and host IDS) in the scope of creating a generic architecture capable of monitoring a simple host or an entire network.

2. Related work

Most of the common intrusion detection systems can be divided into two main categories: network-based intrusion detection systems (NIDS) and host-based intrusion detection system (HIDS) [15]. A network-based IDS has the capacity to observe the behavior of an entire subnet or of a segment of it [7]. Specifically, a NIDS captures the traffic exchanged between a source and a destination port by making a copy of the transferred packets. The NIDS is integrated and properly configured between two network interfaces (before or after the firewall), only if it will be able to create a traffic capture. Furthermore, this essential requirement highlights the greatest advantage of this category of IDS, the fact that it adds no load to the checked systems (the systems behind the firewall) [7]. Without any observable impact to the network, the ability of an attacker to realize that his target is a supervised infrastructure is reduced. From the developing point of view, a NIDS is implemented and chosen according to the specification of the protected subnet. It could be a specialized device implying that a hardware IDS will be used, or it could be a software application running on a host from the monitored network. On the contrary, the HIDS can protect only the device where it is installed, therefore this type of intrusion detection system is specialized in detection of the internal malicious issues. The most common elements monitored by an HIDS are the log files generated from the operating system, the files which store critical information (credentials, rights, attributes, authorized users) and the system calls and processes [13]. Even though it is possible to adopt a hardware solution for creating an HIDS, software solutions are more often embraced, because hardware integration problems could arise. All of these also imply the fact that the resources of the protected equipment are used (CPU, storage volumes).

A common task for both types of intrusion detection systems is the amount of data that is constantly required to be stored and verified even if the analysis process is made online or offline. From this perspective, the intrusion detection systems can take a decision by using a signature-based, an anomaly-based or a specification/protocol-based technique [7]. Their names are representative for the mode that they work on. The signature-based intrusion detection system uses predefined patterns for detecting a malicious activity [6]. Practically, one of its components is a database where every known attack is defined by a rule written in

a generic format. Taking this into consideration, it is easy to conclude that the main disadvantage of a signature-based intrusion detection system regards the fact that it cannot identify the malicious activities that are not defined in the signature database. With a solution for this problem come the other two detection methods. In case of an anomaly-based or a protocol-based IDS, even the unknown attacks can be recognized. Their functioning relies on a set of policies defined by an administrator, which describe what type of behavior is normal or not for the monitored system or network [12]. More specific, it could be affirmed that the intrusion detection system is equipped with a template describing what a regular activity looks like. When the supervised infrastructure is running, its current state will be compared with this template and based on the result of this comparison the IDS can report the occurrence of an intrusion.

Furthermore, to reduce the probability of the false negative results, it was adopted as detection method a combination of signature-based with anomaly-based. This is the first stage of a more complex research; therefore, the signatures and the behavior templates are static information. In addition, basic detection algorithms relied upon matching functions were used. In further development it is considered to integrate the artificial intelligence technics, according with the state of the art. The reason why this kind of algorithms started to be used in the process of the intrusion detection is associated with the increased number of cybersecurity attacks. Moreover, conventional deep learning or machine learning algorithms are suitable for implementing an IDS because the dataset (attacks signatures, rule sets, protocols) can be used as a training information. In [14] benchmarking, it is applied for an anomaly-based IDS where 98% of the dataset represents the normal behavior while 2% is considered as being attacks. Other example of IDSs where neural networks are used for developing a classification algorithm is suggested in [10], where scientists from Bhabha University affirm that their proposal has a strong exactness and considerable false-positive proportions. As a deduction, a recurrent neural network or a machine learning algorithm could be applied for realizing binary classifications (normal behavior vs. anomalous behavior which is classified as an intrusion). Because of these, the most used techniques are CNN, KDDCUP99 and NSLKKD [3], but considerable research is about the GAN framework. This revolutionary idea was proposed by Ian Goodfellow in [9]. Their framework is implemented for two networks which are in competition and in this way, they train each other. The output of a network is an input for the other ones and the first tries to choose an input that is unknown for the second network. It is exactly like in a game played between two people where the move made by one will determine the next move of his competitor and an IDS or an IPS could be efficiently trained with the help of this technology. In the end, the result will be a system capable of detecting and taking decisions in real time even though the intrusion is represented by an unknown attack.

3. The proposed architecture and functions

The main purpose of this paper is to create an Intrusion Detection System (IDS) capable of offering a higher level of security by combining the role of a host-based IDS with the role of a network-based IDS. In this context, the main goal of the current study is to obtain a generic solution which can assure the security of data from a host but also from a network. Hence, the understanding of both types of architecture (HIDS and NIDS) is necessary, because it is ultimately desired to have a hybrid solution. The architecture of these generic intrusion detection systems and the main features necessary for their implementation were used as a reference point. Also, aspects like the differences between a NIDS and a HIDS and their specific functionalities were taken into consideration and put together. Therefore, the current system comprises two principal modules (NIDS and HIDS), each of them with its specific functions, modules which also incorporate other specific sub-components. Furthermore, this idea of a modular architecture helps reducing the risk of any design error.

Starting with this set of theories, the most important subsystems of the IDS presented in this paper is the data collector, the analyzer and the alarm component. The log files (e.g. files which contains a copy of the network traffic) and the critical files (files used to store critical information from the computer where the application runs) form the input for the data collector. After collecting, these dates are transferred to the analysis unit. As a results, the network traffic consisting of IP datagrams is investigated by verifying the header of packets, field by field and their content in real time. An offline type of analysis was chosen for the data used as an input for the HIDS component. Every 24h, the system investigates the content and the properties of the monitored files. If noticed that, there are some signs of a possible cyber security attack, then the alarm subsystem would have the role of alerting the administrator about this problem and of recording the problem. In additions, the system generates a real-time notification sent via email. Fig.1. presents the process previously described in a succinct form

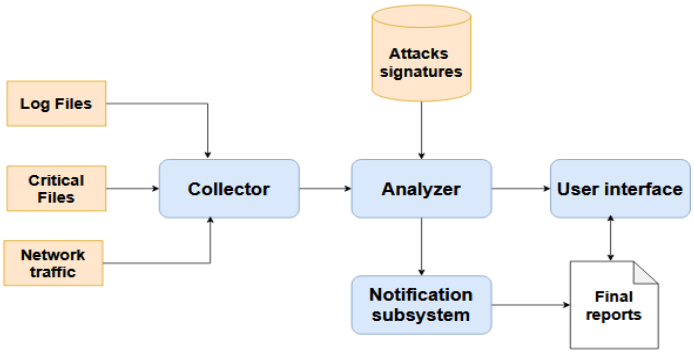


Fig. 1. Schema of the IDS's process

Meanwhile Fig.2. exposes the main modules of the application and the logic according to which they are interconnected.

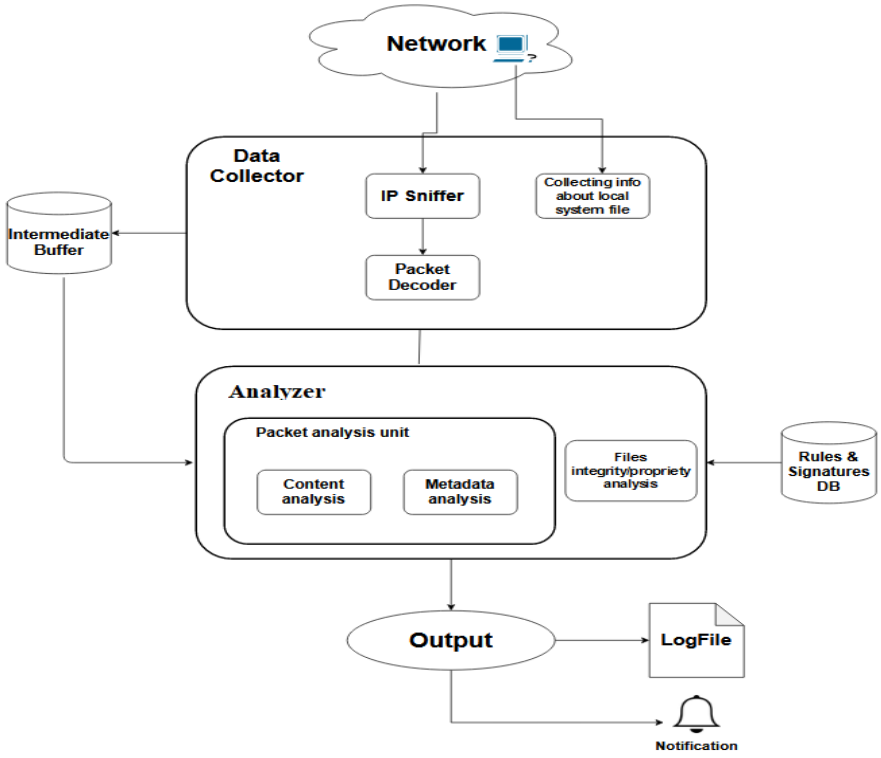


Fig.2. The proposed hybrid architecture of the IDS

The architecture proposed in Fig.2 is developed with the scope of making a system capable to encapsulate the features offered by two independent software applications. In this manner, another advantage can be obtained, namely the reduction of volume of the computing resources. This is an important improvement because in the state of the art, the majority IDS systems used for networking monitoring purpose need additional hardware. Moreover, as it was already mentioned, this architecture is modular, so the level of portability is increased and finally, this proposal offers not only the opportunity to be integrated in different infrastructures, but also to be adapted to their specific needs. By looking the solution from Fig.1. in detail, it is noticed that the first layer is represented by the network comprising the component computers running this IDS system. The main role of the application is to capture the traffic transferred between these hosts; aspect possible only if these computers have available network interfaces. This happens because the component which represents the packet sniffer is positioned

between two network interfaces. After the packets are copied and the information is collected, they are saved for a short period of time in an intermediate buffer. From this buffer, the information goes to the decoder for the analysis process where it is decided if there is a normal behavior or a threat. This process implies the fragmentation of every captured IP datagram and the comparison of the current state of a critical file with its initial state. The way the analysis is performed is using also a hybrid solution, being a combination of multiple intrusion detection methods. It was created as a combination between the technique used for signature-based IDS and the one used for anomaly-based IDS. More specifically, the system has a database formed by files which store the normal operating parameters for the monitored system (used for the HIDS component) and a set of rules created according to a generic pattern (used for NIDS component). The role of these rules is to define de abnormal behavior of the monitored system. Therefore, the actual parameters of the host, the traffic captured from the network and the database represent the input for the analyzer module. Inside of the analyzer these dates are processed. After that, a comparison process follows and thus the output is obtained. In the end this output will be an input for the last level of the application, the user interface and the notification module.

From the software architecture's point of view, every component is created based on a specific algorithm. As it was already mentioned, the scope of the host-based component is only to verify the integrity of some specific files. The processes which run on the monitored systems are not verified by this hybrid IDS. For portability, the IDS will have a configuration file for this module. The information saved in this file actually represents the input for the host-based algorithm. Aspects like the path of the critical files, the type of the desired hash functions that want to be used for the detection process (e.g., MD5, SHA, etc.) or the system administrator's email, are kept in the configuration file. The correct state of the monitored files together with the information regarding the rights of a user or the dates of the last modification of a file and an associated hash code are stored in the databased detailed in the previous paragraph. Actually, in this way the verification of the integrity is made, and it represents an offline process which uses matching functions. This action is reevaluated at a fix time interval, in the case presented in the current study – daily, but this time interval could be also configured. The idea behind this feature is having the stable states of the monitored files saved in the database and the current state of them also, because those two will be compared. If there is not a perfect match, an alarm will be generated. At the end of every run of this functionality an email with a complete report regarding a specific host will be sent. The important aspect about the database is the fact that it could be created by the administrator or if it does not exist, the states which the monitored files have at the first run will be considered to be correct and the database will be automatically generated using this default information. Moreover, it offers the opportunity to be

regularly updated by the administrator and for a better security level the database is encrypted because it is compulsory to make sure that it is updated only by authorized users. In conclusion, in a specific time frame, at the end at each run, the HIDS component determines if some specific files were added or if the exited ones were modified (their content or their attributes) or deleted.

On the other hand, a more complex analyzing mechanism was built for the network-based component. The first step in the development of this component was the implementation of the traffic collector, represented by an IP sniffer capable to capture IP datagrams by listening the available network interfaces. Each packet will be copied and saved in an intermediate buffer. The metadata which can be found in the header of the packet are very useful for detecting the network attacks. The implementation gives the user the opportunity to capture the entire traffic or to use filters for making a partial capture. This functionality was created based on libPcap library [8]. In the analysis mechanism are two important elements. Firstly, it is the information, which is extracted from the packet's fields and secondly, we have the rules saved in the same database used by the HIDS component. As it was already mentioned before, the rules have a generic template based on attributes which role is to describe the behavior in case of a specific software attack. For instance, in case of a DoS attack, the number of packets received in a specific time is relevant. In case of other attacks like messages sent by an unknown sender, the source IP is important. In this scenario the IP extracted from the header will be compared with the IP specified by the rule. The advantage of this kind of software architecture for the NIDS components is that the administrator might improve the efficiency of the system. If a new attack is discovered, it can be described using this generic profile of a rule. The same kind of principle is used by other open-source IDS as for example Snort [1],[4]. In short terms, the idea behind this component can be described by a scenario in which a packet is captured, and it is broken into fragments (content and header). Therefore, the algorithm behind the entire process goes through the list of the defined rules. At the same time the relevant packet's fields are extracted. Every field of the packet will be associated with the corresponding one from a rule based on matching algorithm and the process is repeated for every rule. (These algorithms described before could be observed from Fig.3. to Fig.6.).

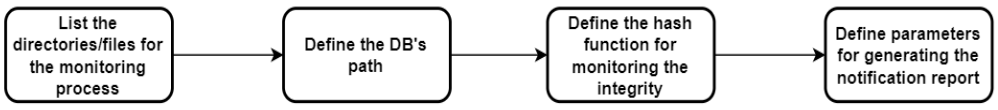


Fig.3. The necessary steps for HIDS component's configuring process

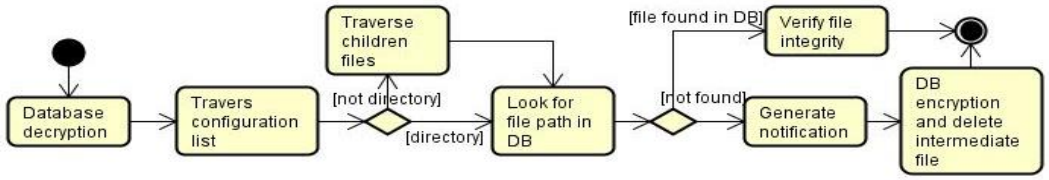


Fig.4. The algorithm for file integrity

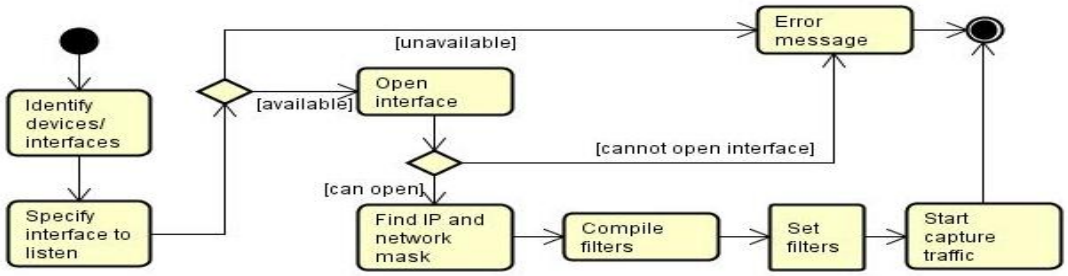


Fig.5. The algorithm for the sniffer component

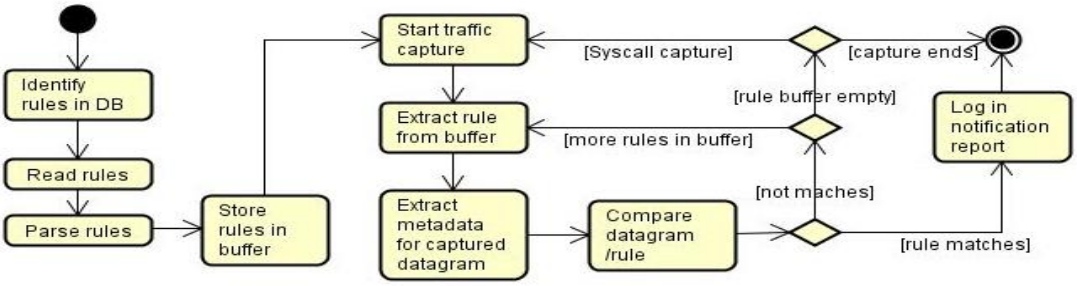


Fig.6. The algorithms for the NIDS component

4. Experimental results

Because it is wanted to have a valid point of view which proves the efficiency of this hybrid intrusion detection system, it was decided to make a battery of functional tests in the last phase of the developing process when a stable version of the application was obtained. For testing the application in a safe way, a special test environment, based on virtual machines, was implemented and used. Hence, these virtual machines play the role of the attackers or the role of the victims according to their scope. The virtual machines operate on Linux distributions like Kali Linux and Ubuntu. As it can be deduced, the hybrid IDS runs on the victim and the scope of this testing process is to assess the correctness of its capacity of identifying different types of attacks, even if they are network oriented or host oriented. What is necessary to be mentioned is the fact that the application will automatically start

to run when the monitored system boots. In this way, the level of security is increased, because the monitored system is under protection from the first moment that it starts to function until it stops its activity or until an authorized user decide that the IDS need to be stopped.

In the first part of this testing process, the capacities of the HIDS component were tested. The most important aspects were the verification of the database generation and the identification of an unexpected or unwanted behavior. The input, as it can be deducted from the previous paragraphs is the configuration file. For the test scenarios of this component, we used a directory called *testDir* with 3 child files: *testFileHIDS1*, *testFileHIDS2*, *testFileHIDS3* and a directory called *testDirA* with a child file *testFileHI*. The required files will be listed in the configuration file and at the first run the data base will be generated containing the default state of these files (file permissions and attributes, owner, group, and hash). The test scenarios include: the change of a file's rights, the change of their content, the case when a monitored file is deleted and the opposite case, when a new file is added. All of them worked properly in the requested parameters and an example of the obtained results were illustrated in Fig.7.

In the testing process of the NIDS component, using an environment based on virtual machines is necessary. The first step was to identify the interfaces IPs and to create the rules set. Another important point is to verify if the rules are read and parse in the correct way. This aspect can be checked in the command line after the application starts but also can be sent to a log file when the application is started automatically. The next phase in this context is to identify the capacities of the system and its limits. In other words, a set of relevant rules describing the unwanted behavior of the network needs to be defined and to see their efficiency in the process of the attack detection. Because the implementation of an attack requires special condition and it is necessary to be done in a secure environment, the best known of them were tested.

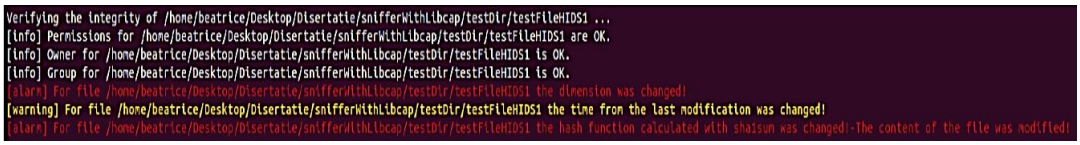
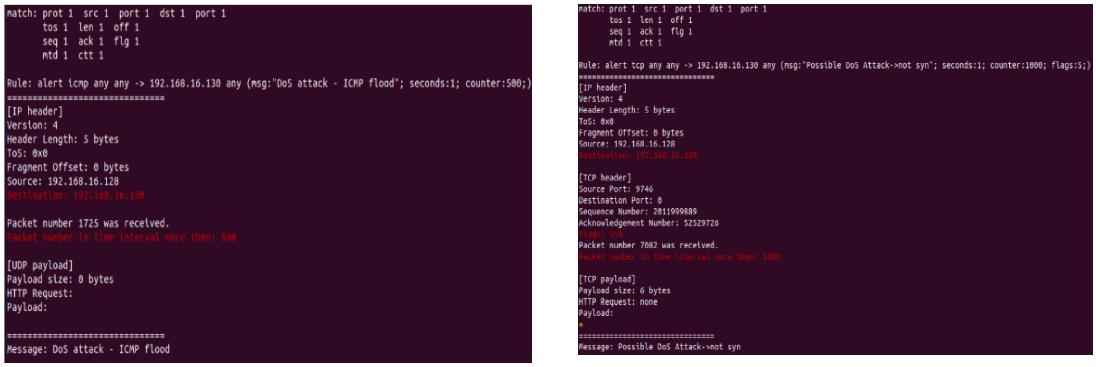


Fig.7. The output for the detection of a host base anomaly

The first one was the Denial-of-Service attack, and it was implemented by creating an ICMP flood attack. Sending multiple ICMP packets in a very short period of time, the host which received this kind of packets will try to answer and, in the end, its resources will be depleted. The rule which shall identify this kind of attack is the following: **alert icmp any any -> 192.168.16.130 any (msg:"DoS**

attack - ICMP flood"; seconds:1; counter:500;) and the explanation behind it is the fact that a notification will need to be sent to the administrator if more than 500 ICMP packets are received by the host with IP 192.168.16.130 (the victim from our virtual network) in less than a second for a specific period of time, in a recurrent way. The flood attack was implemented using another host from the network with the help of the hping3 tool. When the attack started, it was noticed that immediately after the number of packets have been reached, an email with a general format which contains the report has been sent at the specified address. At the same time, if the terminal is open, the output can be seen in real time by the administrator. The same type of attack can be implemented using the Three-Way Handshaking procedure which is used by the TCP/IP protocol. In this situation, the DoS attack can be implemented if multiple synchronization packets are sent, so the receiver will answer with SYN-ACK packets. In a similar way as the previous example, the rule is developed based on the required template. **alert tcp any any -> 192.168.16.130 any (msg:"Possible DoS Attack->not syn"; seconds:1; counter:1000; flags:S;)**. The fact that the system is capable to identify a DoS attack could be seen in the results of Fig.8.

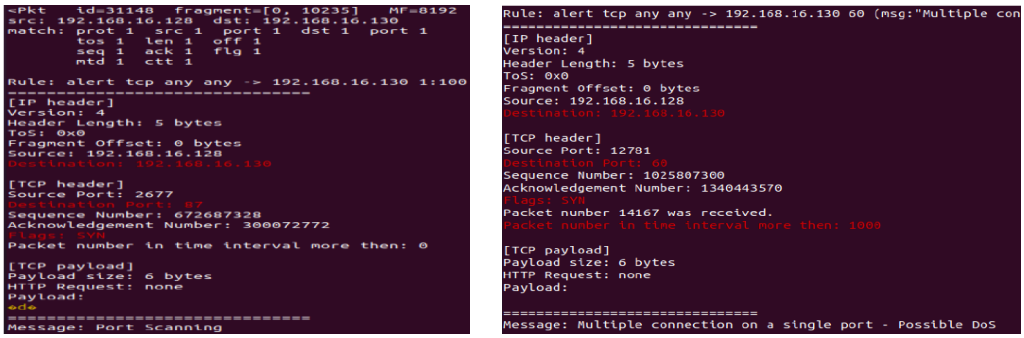


a) b)

Fig.8. a) The identification of an ICMP flood attack; b) The identification of a SYN-ACK attack

Other attacks which were met very often are the port scanning and the situation when an entity tries to create multiple connection to a single port. The port scanning attack is in most of the cases used for discovering and exploring different vulnerabilities of a specific host and for it, it was important to mention in the rule's body the number or the interval of values where the port is situated, so the form of the obtained rule was: **alert tcp any any -> 192.168.16.130 1:100 (msg:"Port Scanning"; seconds:3; flags:S;)**. Following the same perspective, the attack where a multiple connection is tried to be established had a similar rule definition **alert tcp any any -> 192.168.16.130 60 (msg:"Multiple connection on a single port - Possible DoS"; seconds:1; counter:1000; flags:S;)**, but the main difference

between these two attacks was the fact that the last one could be also integrated in the category of DoS attacks. As the scenarios presented in the previous paragraphs, these two were simulated with the help of hping3 tool and them results were pictured in Fig.9.



a) b)

Fig.9. a) The identification of a port scanning attack; b) The identification of a multiple connections to a single port attack

5. Performance of the proposed solution

Regarding the HIDS component, its performance is directly influenced by the host system's resources and application parameters stored in the configuration file. An offline type of analysis is used for host monitoring, therefore the level of accuracy was noticed to be affected by the configuration established by the system's administrator. In consequence, aspects like the number of the monitored directories, the way that the normal behavior of the host is described or the time frame when the analysis is preformed play important roles in the efficiency. With a configuration corresponding with the supervised system's needs, the application did not report any situation of false positive or false negative results during the testing process.

On the other hand, the NIDS component is based on libPcap library, aspect which makes this hybrid IDS similar with others open-sours NIDS solutions like Snort or Suricata. It integrates three major operational modes, more specific it could be used as a sniffer, as a logger or as a hybrid intrusion detection system. During the network monitoring, all these three functions are working together to perform a real-time supervision. This kind of supervision implies a real-time traffic capture and analysis (the entire process was described in the previous sections). The testing environment was represented by a network of virtual machines, while the IDS application was running only on a part of them. Between virtual machines was an Internet connection with an average speed of 144Mbps. Also, the average dimension of a packet exchanged between two hosts is around 60 bytes. In

conclusion, there is an amount of approximately 300000 packets sent from a source to a destination in just one second on a full duplex connection. Depending on what processes are running on the virtual machines, this number could be smaller. At this rate, the IDS was capable to make a full capture, to identify flood attacks and to response immediately to them. Hence, it could be concluded that the hybrid IDS's performance is not influenced by the number of packets because it was able to function in proper parameters on a high internet speed. The only aspects that can influence its efficiency and can provoke false negative situations are the rule's signatures. As it was mentioned in the previous chapter, a rule is defined based on a specific template. If the parameters of the rules are not set according to the specification of the monitored architecture, then problematic situations could occur. For example, one of the important parameters is represented by the tolerated number of packets of a certain communication protocol type sent in a specific time frame. This number varies according to the monitored system's functionality and resources, therefore the system's administrator needs to properly define the range of acceptance.

6. Conclusions

This paper presents an experimental hybrid intrusion detection system capable of being integrated in different infrastructures and to offer the possibility to be extended in order to offer a higher level of cyber security protection. The final product is a pure software solution based on open-source technologies and which can be easily installed into any kind of system that is desired to be monitored. The application has a modular architecture with two principal components. According with their roles and maintaining the balance between the computation resources used by the application and its efficiency in intrusion detection procedure, every component made the decision in the way that is more suitable for its own scope. Specifically, the HIDS component realizes an offline analysis, while the NIDS makes an online verification. Both components are integrated in the same product which started to run during the booting procedure of the system where it is installed. They use a hybrid between anomaly-based and signature-based methods for neutralize the impact of the disadvantages owned by each technique. Another important reason which conducted to a modular architecture is about the portability of the application. In this moment, this intrusion detection system is available for the Unix platform only, but behind it (exactly like SNORT or Wireshark [11]) is the libPcap library. The advantage of libPcap is the fact that it can be updated for other platforms also. Moreover, the capacities of this hybrid IDS were tested in a controlled environment. At the end of the testing phase the conclusions traced were that the system is working properly and according with the initial objectives.

Furthermore, these results will be used in the further research which scope is to apply the GAN technique for the detection procedure of a hybrid IDS, to identify if the system can be fooled and if the deep learning algorithms represent a better option in cyber security.

REFERENCES

[1]. *F., Alam.* Intrusion Detection & SNORT. In: APRICOT 2015, [Online]. Available: https://nsrc. org/workshops. 2015.

[2]. *A. S., Ashoor, et S., Gore*. Difference between intrusion detection system (IDS) and intrusion prevention system (IPS). In : International Conference on Network Security and Applications. Springer, Berlin, Heidelberg, July, 2011. p. 497-501.

[3]. *K., Atefi, H., Hashim, et T., Khodadadi*. A hybrid anomaly classification with deep learning (DL) and binary algorithms (BA) as optimizer in the intrusion detection system (IDS). In: 2020 16th IEEE international colloquium on signal processing & its applications (CSPA). IEEE, 2020. p. 29-34.

[4]. *B., Caswell et J., Beale*. Snort 2.1 intrusion detection. Elsevier, 2004.

[5]. *Y., Chen, T., Chen, and W.H., Liu.* Network intrusion protection system. US Patent App., September, **vol. 12**/049,890., 2009.

[6]. *J., Díaz-Verdejo, J., Muñoz-Calle, A., Estepa Alonso, et al.* On the Detection Capabilities of Signature-Based Intrusion Detection Systems in the Context of Web Attacks. *Applied Sciences*, no 2, **vol. 12**, 2022, p. 852.

[7]. N., *Dutta, N., Jadav, S., Tanwar, et al.* Intrusion Detection Systems Fundamentals. In : Cyber Security: Issues and Current Trends. Springer, Singapore, 2022. p. 101-127.

[8]. L. M., Garcia. Programming with libpcap-sniffing the network from our own application. Hakin9-Computer Security Magazine, **vol.2**, 2008.

[9]. I. J., *Goodfellow, J., Pouget-Abadie, M., Mirza, et al.* Generative adversarial nets. Advances in neural information processing systems, **vol. 27**, 2014.

[10]. *K. P., Jadhav, et M., Gangwar.* Intrusion Detection System and Vulnerability Identification Using Various Machine Learning Algorithms. In: Proceedings of International Conference on Recent Trends in Computing. Springer, Singapore, 2022. p. 527-536.

[11]. *V., Jain.* Introduction to Wireshark. In: Wireshark Fundamentals. Apress, Berkeley, CA, 2022. p. 1-34.

[12]. *V., Jyothsna, et K. M., Prasad*. Anomaly-based intrusion detection system. In: Computer and Network Security. IntechOpen, 2019. p. 35.

[13]. *M., Liu, Z., Xue, X., He, et al.* SCADS: A Scalable Approach Using Spark in Cloud for Host-based Intrusion Detection System with System Calls. arXiv preprint arXiv:2109.11821, 2021.

[14]. *Z. K., Maseer, R., Yusof, N., Bahaman et al.* Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. *IEEE access*, **vol. 9**, 2021, p. 22351-22370.

[15]. *D., Park, S., Kim, H., Kwon, et al.* Host-Based Intrusion Detection Model Using Siamese Network. IEEE Access, **vol. 9**, 2021, p. 76614-76623.

[16]. *A., Skorobogatjko, P., Dorogovs, et A., Romanovs*. The Use of Intrusion Detection Systems Based on the Network Behaviour Analysis in SCADA Networks. Information Technology and Management Science, no 1, **vol. 15**, 2012, p. 171-175.