A Clustered Hybrid Honeypot Architecture

Eva Chovancová, Norbert Ádám

Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice
Letná 9, 042 00 Košice, Slovak Republic

eva.chovancova@tuke.sk, norbert.adam@tuke.sk

Abstract: Nowadays, when computers and computer systems are almost omnipresent and are part of most households and most people have access to them, they are more vulnerable than ever. Today, protection and security of computer networks and systems using passive protection does not suffice anymore. One has to anticipate attacks and be a step ahead of the attackers. To achieve this type of security, one has to employ active protection techniques, such as digital baits – honeypots. The primary goal of using honeypots is to divert the interest of potential attackers from other really important targets within a particular computer network. The secondary goal is to acquire information about the attackers' activities and methods. Subsequently, these data are thoroughly analysed. By analysing the attacks, security improvement measures aimed at the particular network and/or computer system may be proposed in order to prevent threats. The goal of this paper is to contribute to computer security by proposing a clustered, high-interaction-honeypot-based security system.

Keywords: security; honeypot; intrusion detection system; malware; clustered honeypot

1 Introduction

The world full of information and information resources means an ever growing amount of data stored in computer systems. The risk of threats is also increased by improving connectivity and access to data and computer system services from various locations within the computer network. Any system providing connectivity to a computer network may be at risk. Currently, security by authentication [1] [2] [3] and authorisation of production systems is considered to be insufficient since a potential attacker may steal identity data of another user or bypass authentication, violating system security [4] [5] [6]. With the development of information technology, our knowledge of computer systems advances, too – attackers can analyse weaknesses of the respective systems and use them during the attack itself. The task of any security technology operator is to detect and subsequently identify any attack; however, in heavily used systems, this may be

very hard, in some cases even impossible. After performing the attack, the attacker often infects the compromised system with malware [7] [8] [9] [10] [11]. The goal of this malware is to fulfil the attackers' goals; often, malware is capable to replicate and spread in the compromised computer network. This makes them very dangerous since all the attacker has to do is execute them, without the need of any later surveillance. As a result, malware may then spread within the network freely – if computers from other networks trust computers from such a compromised network, these computers are also at risk [12] [13].

An intrusion detection system (IDS) is a passive protection mechanism monitoring activity within the computer network/system [14] [15] [16] [17] [18] [19] [20]. It monitors and gathers data on the activities performed within the network/system, to detect intrusions. An intrusion detection system does not only detect intrusions, but it may also be used as a monitoring tool to identify suspicious activities aimed at compromising the particular network/system. The gathered data may then be used to increase the security of the network/system, following a thorough analysis. However, today, protection and security of computer networks and systems using passive protection does not suffice anymore. One has to anticipate attacks and be a step ahead of the attackers. Therefore, not only passive, but also active protection techniques – such as digital baits (honeypots) – have to be used.

This paper proposes a clustered intrusion detection system architecture, based on high-interaction hybrid honeypots [21], eliminating the disadvantages of intrusion detection systems using honeypot technology.

2 Honeypots

Digital baits – honeypots – represent flexible, constantly developing technology with numerous use cases [21] [22] [23] [24]. Honeypots may be used secure systems, but also to gather information for subsequent analysis. The analyses of the gathered data may result in the detection of suspicious activities, proposal and eventual implementation of countermeasures improving system security and preventing further similar attacks.

Essentially, honeypots are devices with purposely constructed security holes, while pretending a false identity. The goal of using honeypots is to divert the interest of the potential attackers from real devices. If an attacker focuses on the honeypot, one may register, identify and analyse his/her activities. To increase efficiency, intrusion detection tools may be added to the honeypot itself, to monitor the use of specific, monitoring and/or malicious applications.

Assuming that any interaction with the honeypot is a potential attack, the need to classify the gathered data falls to the minimum, increasing the speed of the analytic process and evaluation of the data gathered by the bait itself. To increase

computer system security and improve the efficiency of fending off the attack, it is wise to use honeypots with other standard security tools and methods.

One of the most important benefits of using honeypots is related to detection since it minimises the negative aspects of common intrusion detection systems. These include the following:

- evaluating an attack, which did not happen (false negative) or failure to detect an attack (false positive);
- elimination of errors in the configuration of intrusion detection systems.

Another advantage of this security solution is that they do not influence system operation during the interaction with the attacker.

2.1 Fundamental Functionality

Honeypots have numerous functions allowing multifaceted use. The basic functions are the following:

- to divert the interest of the potential attacker from production equipment.
- to identify security holes in the OS.
- to analyse the attackers' behaviour during the interaction with them.
- logging attacker behaviour.

2.2 Interaction Level

Honeypots may be classified also by the level of attacker interaction; in this case, "interaction" refers to the possibilities of the attackers' communication with the honeypot [25]. There are low-interaction and high-interaction honeypots. Low-interaction honeypots have no operating system; therefore, the attacker cannot initiate interaction with the OS. A disadvantage of these honeypots is that services and processes are emulated. The attacker may detect them easier and since it is impossible to gain full access to the honeypot, the amount of data that may be gathered about the attackers' activities, is also limited.

High-interaction honeypots are more sophisticated, having a more complex design since they use real operating systems. In this case, one may gather more information about the attackers. High-interaction honeypots record all attacker activities; the records may be then analysed to improve system security.

3 Clustered Hybrid Honeypots

Honeypots are frequently used means of system security [23] [24] [26] [27] [28]. When using honeypots, their level of security has to be taken into account. In case of week honeypot security, the attacker may easily discover that the honeypot is not a production device. On the other hand, if the honeypot security level is too high, it may discourage the attacker.

Most existing honeypot systems focus on the configuration of the equipment running the honeypots. Configuring such equipment means mostly configuring these devices to allow them to find and allocate unused IP addresses automatically, after connecting them to the network and, eventually, to allow them to adapt to the changing environment of the computer network.

The architecture proposed herein is a clustered hybrid honeypot [22]. The proposed hybrid honeypot architecture combines two types of baits with different interaction levels, focusing on high-interaction honeypots since, in terms of security, to the attacker; it is an interesting target with an IP address. An ideal tool meeting the requirements of a low-interaction honeypot is Honeyd, an open-source program. With its help, the load of the high-interaction honeypot may be relieved, allowing focus mainly on the initial attack analysis. Attacks are detected by the architecture itself. The proposed solution differs from the security solution, in which any anomaly triggers diversion of the interaction to a "shadow honeypot" during the operation of the computer system. In this system [29], analysis of the incoming attack is performed by the shadow honeypot. However, such a solution increases the operating costs of the network/system.

3.1 Clustered Hybrid Honeypot Architecture

The proposed architecture consists of these 4 main parts:

- Internet access management
- clustered honeypots,
- auditing and data repository cluster, and
- correlation.

The architecture of the proposed clustered high-interaction honeypot is depicted in Fig. 1. The proposed architecture uses both low-interaction and high-interaction honeypots, increasing the efficiency of such a system in terms of attack discovery. The proposed architecture type requires fewer interventions into the configuration of the security computer system, related to the reinstallation of successfully compromised honeypots.

3.1.1 Access Management

This part of the proposed architecture consists of honeypots acting as computers connected directly to the Internet, having public IP addresses. A further part of the management is Honeywall – a tool limiting the speed of the Internet connection to 10 MB per hour, the number of TCP sessions to 100 sessions per hour and thus limiting bandwidth.

The network operation analyser and Honeywall are mutually separated. The reason for this separation is to minimise the threat to the protected computers if the network analyser is compromised. Should the network analyser get compromised and should it be used for an attack beyond the system, the chances of such an endeavour are minimal since Honeywall is configured to limit bandwidth.

3.1.2 Honeypot Cluster

The honeypot cluster consists of high-interaction honeypots, which are again clustered. In the proposed architecture (Fig. 1), there are three types of honeypots: a GNU/Linux OS with MAC access management, a GNU/Linux OS with DAC access management and representing MS-Windows operating systems. Every computer used as a honeypot has two network interfaces configured. One of these has a public IP address, while the other has a local one, such as 172.30.3.X.

By adding a private IP address to the second network interface, all honeypots are interconnected in a separate local network. By creating a local honeypot network, we limit the attackers' attempts only to local devices. This local network consists of 14 honeypots without any firewall installed.

Every honeypot using GNU/LINUX OS contains a modified OpenSSH service, which simplifies opening new relations for the attacker. As soon as the attacker tries to log in to one of the honeypots with the aforementioned configuration using an SSH connection and a brute force attack, the modified OpenSSH service shall generate a separate account for the attacker.

This part of the proposed architecture creates an account in the system for the attacker, along with a regular home directory. The created account is persistent, which allows authorisation of the attacker if he decides to return later, to continue the attack. To allow capturing all attacker activities, there are multiple IDS systems and security tools available for GNU/LINUX distributions, as well as Windows operating systems.

These intrusion detection systems monitor four types of information resources:

- system activities (system calls and processes);
- system file integrity;
- the kernel and the logging daemons;
- Bash relations.

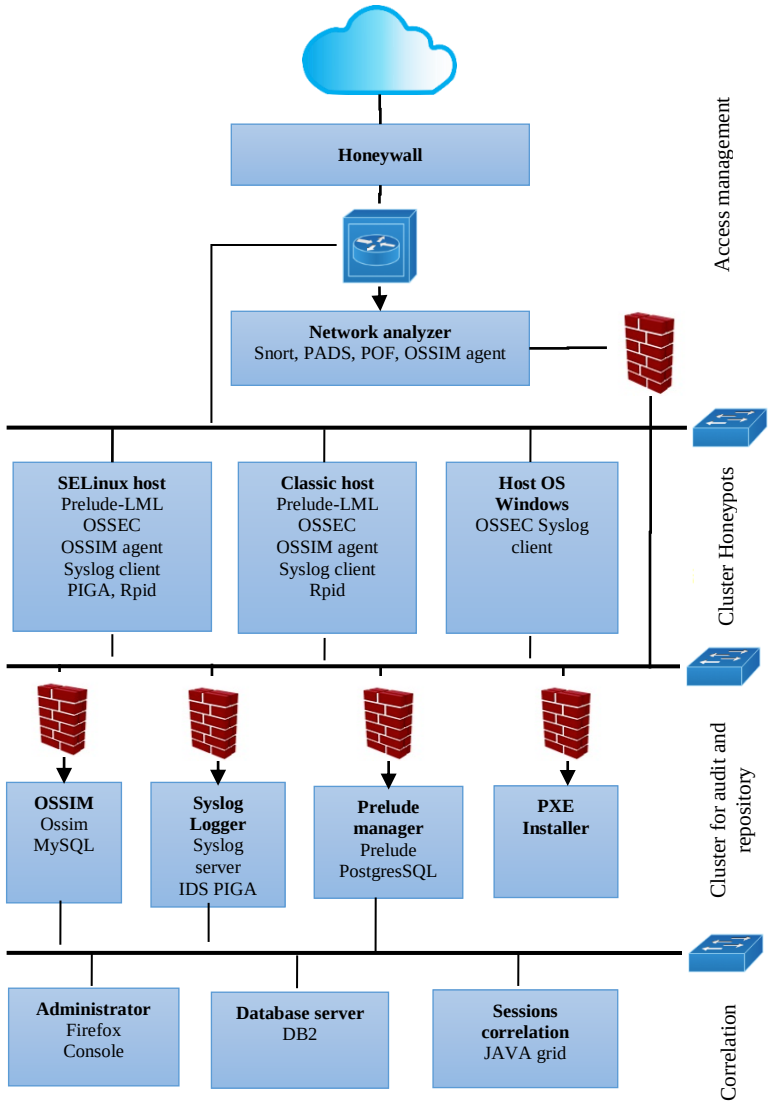

Figure 1: Clustered high-interaction honeypot architecture

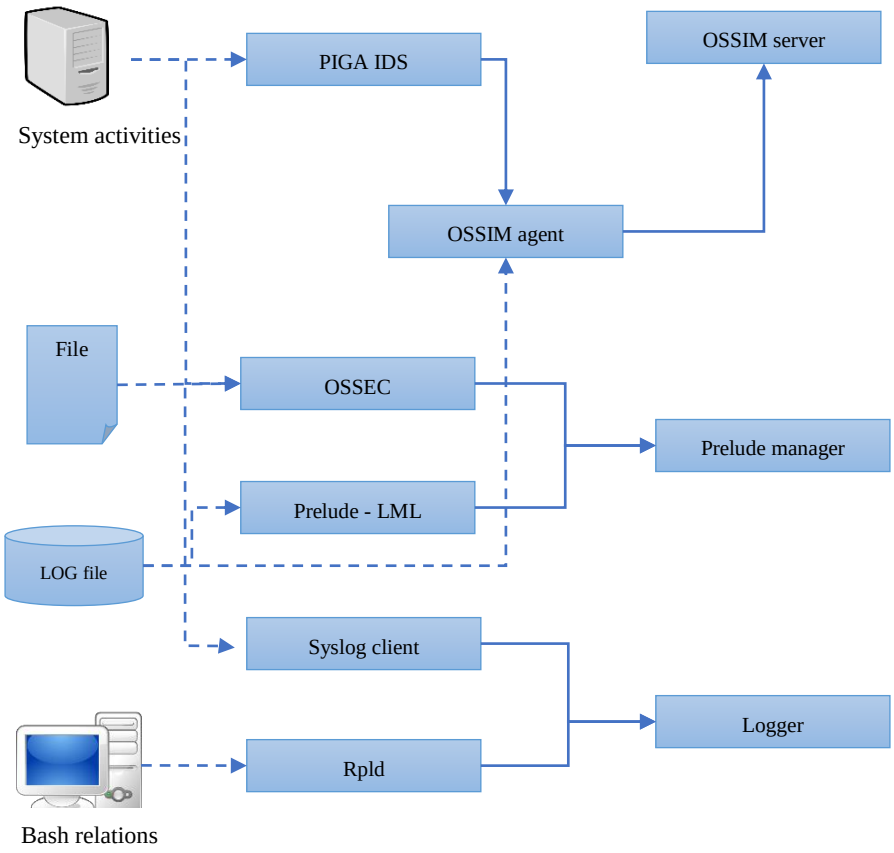

Figure 2
Tools and intrusion detection systems monitoring activity on the hosts

The gathered information shall then be used to discover the correlation between the host processes and network activity. Figure 2 is a summary of the data used by the respective security agents (using a dashed line). All registered alarms, logs, and relations are sent to another cluster of the proposed architecture – the cluster aimed at auditing and storage of the recorded data.

3.1.3 Auditing and Storage Cluster

The third part of the proposed architecture is dedicated to the storage and subsequent analysis of the registered alarms and logging records. Data storage equipment is connected to the network, e.g. using the 172.30.3.0 prefix. The whole auditing and storage cluster is protected by a firewall. This is configured to allow only incoming network traffic, the only allowed and opened ports are those used by the security tools OSSIM (gathers alarms generated by the Piga IDS and

analyses system log files), Prelude-LML (system log analyser reporting system activities) and the Syslog tool, dedicated to the servers (transfers kernel and system log files to the logger).

To gather all events and report alarms in a human-readable form, three analytic frameworks are used in the proposed solution:

- OSSIM – provides a security information management framework. The framework itself generates reports, aggregate alarms, and triggers incidents. It writes the gathered data into a MySQL database. If necessary, the framework may be configured to store data in a MySQL cluster server.
- Prelude manager – it aggregates the gathered data, it also provides means to visualise the data on websites. All registered events are stored using the Intrusion Detection Message Exchange (IDMEF) standard and a PostgreSQL server. If necessary, the framework may be configured to store data in a cluster.
- System logger – stores all syslog messages generated by any honeypot in a Lustre distributed file system.

All network and system alarms are stored by the system manager, which also allows their visualisation. Since these three frameworks do not use the same data recording standard, they have to be connected to the correlation part of the proposed clustered honeypot. One of the main advantages of using three different standards on the recording servers is that by this approach, it is harder to remove any attack analysis files generated by the honeypots. Such files include logging files, network communication fingerprints, and/or other activities.

3.1.4 Correlation

To visualise the recorded alarms, we shall use the last part of the proposed architecture, the correlation part. As a process, the task of correlation is to describe attacks by means of network and host IDS alarms. The algorithms used in the correlation part are open source, with minimum modifications performed to them. These algorithms use all three databases of the respective frameworks (OSSIM, Prelude manager and Syslog logger), using a private network, such as the one having the prefix 10.0.0.0. To optimise the execution times of the aforementioned algorithms, an available Java application was used.

3.2 Security Elements of the Proposed Architecture

A disadvantage of the proposed architecture is that they make real computer systems available for interaction with the attackers and thus the freight that the attacker manages to gain root privileges and conquer the honeypot is a real. Therefore, additional security elements have been implemented in the proposed

clustered hybrid honeypot, demonstrably increasing the security level of both the honeypot and the computer system, in which the clustered honeypot shall be operated.

3.2.1 Verification Module

During the interaction with the attacker, the verification module used in the clustered hybrid honeypot implementation manages all outgoing DNS requests to the Snort network intrusion detection tool, which stores them in log files for further analysis. Snort manages all outgoing traffic – it allows only using the attacker's IP address. Should Honeywall be compromised, any connection attempts will result in initiating the stability check process, or, eventually, restoring the original settings of the whole security system.

3.2.2 MAC Access Control

This access type makes sure root users cannot achieve super administrator privileges. The only way to gain super administrator privileges is to exploit a kernel bug or to successfully attack the MAC mechanism, i.e. to surpass SE Linux. If the attacker succeeds, the system shall be considered compromised and shall be reinstalled using the Pre eXecution Environment (PXE) server. The main advantage of using MAC access control is that honeypots using this access control type are persistent in time. if the same attacker performs a repeated attack, this access control type prevents the detection of system re-installation. In addition to this, all shell activities shall be recorded using an Rpld server.

3.2.3 DAC Access Control

This traditional access control system may be easily compromised by attackers. If the attacker manages to gain administrator privileges, the particular honeypot must be reinstalled using the PXE server. The main disadvantage of honeypots using this access control type is the associated higher administration costs. The registered alarms have to be inspected – it is up to the administrator to decide about the extent, to which the system is compromised and whether the honeypot has to be reinstalled or not.

To simplify the analysis in case of this honeypot type, a cron time-based job scheduler was implemented under Linux to monitor the changes to the file system of the Honeypot during its operation. If any file of the honeypot differs from the corresponding file stored at the PXE server, the system automatically generates and sends a warning to the administrator using the OSSEC integrity analyser, while simultaneously storing the specific difference at the PXE server for further analysis.

3.2.4 Automatic Installation of the Clustered High Interaction Honeypot

The PXE server shall use a TFTP server dedicated to the re-installation process if any of the high-interaction honeypots get compromised. Currently, the PXE server allows re-installation of the following honeypot types:

- SELinux with MAC based on Debian, Gentoo, Fedora, Ubuntu and RedHat GNU/Linux distributions.
- SELinux with DAC based on Debian, Gentoo and Ubuntu GNU/Linux distributions.
- MS Windows XP, Vista, 7 and 8.1 systems.

The PXE server is primarily used to re-install compromised honeypots using the PXE protocol in the private network, with a prefix of 172.30.3.0, for example.

4 Experimental Proof

To provide an experimental proof of the functionality of the proposed architecture, we selected a heterogenous computer network, following the concept model of the proposed clustered hybrid high-interaction honeypot security architecture (Fig. 1). The testing environment consisted of the following:

- The network of the student hostels of the Technical University in Košice, at Jedlíkova Street (the ŠD TUKE heterogeneous network with approximately 1,600 active users). In this environment, we verified the functionality of the clustered hybrid honeypot in terms of honeypot recovery.
- A closed network of the Institute of Computer Technology (ICT) at the Technical University of Košice (the ICT TUKE simulated and partly emulated heterogeneous network). We also verified the functionality of the clustered hybrid honeypot in terms of monitoring its behaviour during specific types of attacks, in a secure environment.

The verification processes were based on the execution of multiple experiments. The following chapters present the results of the experimental proof of functionality of the proposed clustered honeypot architecture and its respective parts.

4.1 Attacks Registered by the Honeypots

The first experiment was aimed at the influence of the applied access control policy – MAC and DAC, respectively – on the amount of alarms generated by the respective honeypots. Most generated alarms were issued by GNU/Linux

distributions with MAC access control policy (Fig. 3). Honeypots using DAC access control generated fewer alarms. The lower alarm count of honeypots using DAC access control can be explained in two ways. The DAC honeypots were assigned public IP addresses significantly later, while until then, the specific honeypots were subject to attacks only from the closed network. In this case, the Snort tool did not report any activity from the DAC honeypots. The second cause of the reported values was the impossibility of using the Piga IDS without using the MAC access policy, which also significantly impacted the number of generated alarms. Figure 3 shows an attack aimed at the logging server itself, in spite of the fact that no public IP address was allocated to it and it was protected by a firewall.

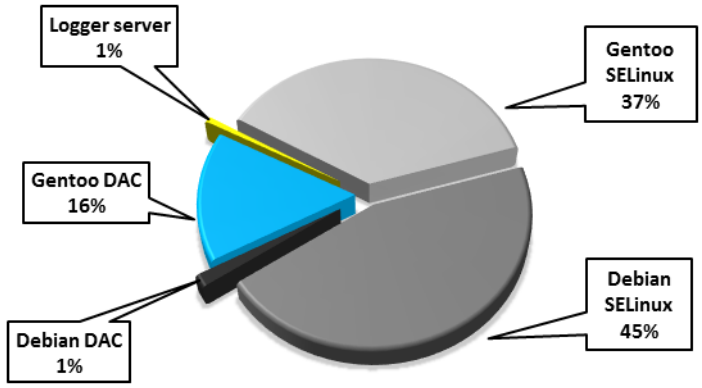

Figure 3
Generated alarms, by the respective hosts

During the experiment, we have not experienced any targeted attacks against specific Linux distributions.

4.2 Statistical Results of Intrusion Detection Tools

During the proof of functionality of the proposed architecture, a total of 2,469,840 events and 97,116 alarms were recorded in the respective server databases. That amounts to 1140 events and 45 alarms per hour. The events included also user login events. Alarms included also suspicious events, such as the discovery of bad packets, which also indicate attacks. During the experiment, we used also Snort. In the phase of detailed analysis, most alarms generated by Snort were evaluated subsequently as false alarms. Figure 4 shows the rate of alarms generated by the honeypots and the respective security tools.

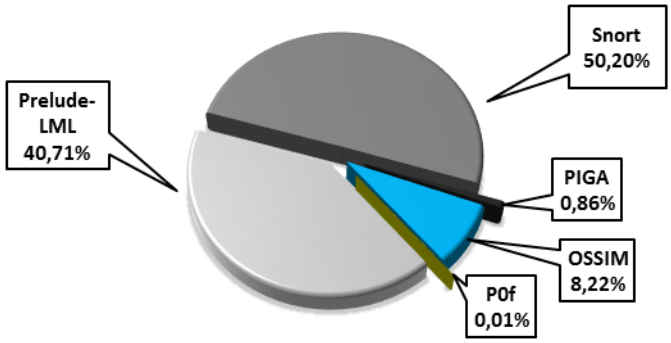

Figure 4
Generated alarms, by the respective IDS sensors

Snort generates a number of false alarms that are caused by the way, when detecting attacks. Snort detects attacks by using a database of signatures and attack patterns, which it compares to ongoing network traffic. The tool itself cannot correctly tell, whether the attack is real or successful. Due to this, in the proposed architecture, we included the Piga IDS to eliminate false alarms generated by the Snort NIDS. During the experiment, Piga recorded 13,965 open relations due to automated network environment scanning and only 779 relations performing any activity on the monitored honeypots.

Table 1
The main types of registered alarms

Detection sensor	Description	Count
Prelude-LML	SSHd: Root login refused	124 617
Snort	Destination udp port not reachable	113 002
Prelude-LML	SSHd: Bad password	27 305
OSSIM	SSHd: Possible brute force tentative	13 361
Prelude-LML	SSHd: Invalid user	10 827
PIGA	Integrity: system file modification	10 265
Prelude-LML	FTP bad login	5 341
Snort	Potential outbound SSH scan	4 995
PIGA	Confidentiality: information flow	4 047

The data in table 1 show the most frequent alarms registered by the intrusion detection system sensors. The most frequent were brute force attacks occurring with SSH connections, trying to break the password of the user and gain access to the system – in this case, a honeypot. The use of brute force resulted in the generation of vast numbers of alarms. Please note that the targets of brute force attacks were also FTP accounts.

As far as outgoing traffic is concerned, Snort found not less than 4995 SSH scanning procedures, trying to use user accounts to attack other equipment in the network. Since all honeypots were connected to a single local network, in the proposed architecture, this resulted only in further interaction with honeypots.

Piga registered some cases of modifying configuration files. In addition to the modification of files, it also registered information streaming mostly from these files to the system file folders of other users. On GNU/Linux distributions, these folders included /etc/shadow, /etc/apache and /httpd.conf, respectively.

The total share of alarms generated by Snort amounted to 50.20%. These alarms included UDP port scanning (160,502 activities), as well as a number of lower severity alarms, as far as the protected system was concerned. The analysis of the generated alarms in terms of the respective ports, as shown in Figure 5, corresponding to the data of table 1, clearly shows the dominance of activities aimed at SSH port 22/TCP. The next most popular was HTTP port 80/TCP, attacked in most cases from the outside, attempting the installation of phishing websites. Harmful ICMP packets amounted to a significant 10%. Other standard ports were used to exploit vulnerabilities of MS Windows, mainly by worms attempting to spread. The ports of the IRC protocol were mostly used by bots installed in the honeypots with the aim to create connections from the infected system to the Internet using different IRC channels. Using these newly created IRC channels the attacker may remotely control any of the installed bots.

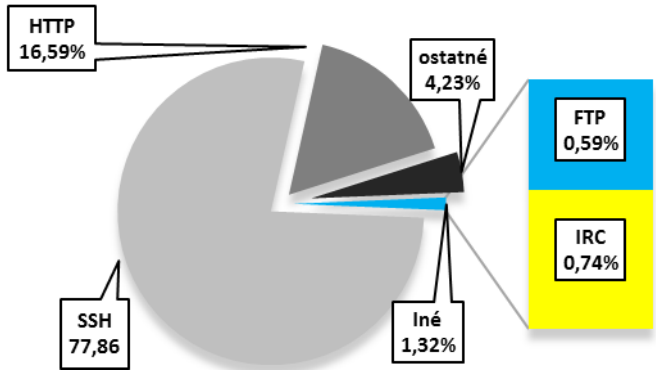

Figure 5
Generated alarms, by the respective ports

4.3 Malware Activity

A further experiment showed that the attackers used mostly i386 binary files to automate the attacks; these were most probably programs are written in C. Often, the attackers resorted to recompiling programs written in C directly on compromised honeypots. The experiment showed unusual attack types – programs

written in Pascal, C++ and/or shell scripts. The aim of these attacks was to perform automated operations to execute or compile malware source code. We also detected files containing malware documentation. The following table represents data related to the detected malware, installed by attackers into the corresponding testing environment.

Table 2

Main types of detected malware

Presence in the home folder	Presence in the /tmp folder	Malware type
46	48	Dynamically linked ELF 32-bit LSB executable (uses shared libs)
35	25	C source code (ASCII text file)
9	11	Shell script
4	3	Pascal source code (ASCII text file)
2	5	Statically linked ELF 32-bit LSB executable
1	0	Mach-O binaries / PPC binaries
1	3	C++ source code documentation

Conclusions

The proposed architecture allows detection of attacks and identification of tools and methods used by attackers and thus to improve the security of the computer network, in which the aforementioned security architecture is being used. An advantage of this solution is that if the specific honeypot is configured correctly, it can actively attract the attention of the attacker, immediately after being started and thus minimise the risk of compromising the protected system within the particular network.

The functionality of the proposed solution, i.e. the threat-detection capability of the presented architecture has been proven experimentally.

The results of the experiments showed that the proposed clustered high interaction hybrid honeypot security architecture can efficiently emulate typical services of low-interaction honeypots cooperating with high-interaction honeypots, without the necessity to create complicated low-interaction honeypot scripts manually. By using multiple IP addresses allocated to low-interaction honeypots and the Honeyd tool, the proposed clustered honeypot architecture is capable of filtering and analysing system operation, pursuant to the predefined requirements. One of the main advantages of Honeyd is the capability to simultaneously simulate virtual low-interaction honeypots. It has been experimentally proven that during the simulation of a LAN network, Honeyd could successfully simulate up to 65,536 heterogeneous devices of various kinds and functions. Honeyd requires an

appropriate configuration of the host hardware server/workstation configuration to correctly operate and run.

The analysis of the events registered by the intrusion detection systems showed that in almost every case, the host environment is being scanned before the attack to gather the most detailed information about the target computer. Following the acquisition of such information and their analysis, the attacker can identify the weaknesses of the protected system.

Acknowledgements

This work was supported by the Faculty of Electrical Engineering and Informatics, Technical University of Košice under the contract No. FEI-2018-59: Semantic Machine of Source-Oriented Transparent Intensional Logic and by KEGA Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under Grant No. 003TUKE-4/2017 „Implementation of Modern Methods and Education Forms in the Area of Security of Information and Communication Technologies towards Requirements of Labour Market“.

References

[1] S. Y. Lim, M. L. M. Kiah, and T. F. Ang, “Security Issues and Future Challenges of Cloud Service Authentication,” Acta Polytechnica Hungarica, Vol. 14, No. 2, pp. 69-89, 2017

[2] L. Vokorokos, A. Pekár, N. Ádám, and P. Darányi, “Yet Another Attempt in User Authentication,” Acta Polytechnica Hungarica, Vol. 10, No. 3, pp. 37-50, 2013

[3] M. Uchnár and J. Hurtuk, “Safe user authentication in a network environment,” in 2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI), 2017, pp. 000451–000454

[4] Prajitha M V, Rekha P, and Amrutha George A, “A secured authentication protocol which resist password reuse attack,” in 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015, pp. 1-5

[5] A. Balaz and R. Hlinka, “Forensic Analysis of Compromised Systems,” in 10th IEEE International Conference on Emerging Elearning Technologies and Applications (ICETA 2012), New York, USA, 2012, pp. 27-30

[6] L. Vokorokos, A. Baláž, and N. Ádám, “Secure web server system resources utilization,” Acta Polytechnica Hungarica, Vol. 12, No. 2, pp. 5-19, 2015

[7] M. K. A, Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware. Packt Publishing, 2018

[8] W. A. Conklin, G. White, C. Cothren, R. L. Davis, and D. Williams, Principles of Computer Security: CompTIA Security+ and Beyond, Fifth Edition. McGraw-Hill Education, 2018

[9] L. Vokorokos, J. Hurtuk, and B. Madoš, "Malware categorization and recognition problem," in IEEE 18th International Conference on Intelligent Engineering Systems INES 2014, 2014, pp. 105-108

[10] J. Hurtuk, B. Madoš, Š. Halčín, "Sound-based communication in the process of malware distribution", in Acta Electrotechnica et Informatica, Vol. 15, No. 2, 2015, pp. 62-65

[11] L. Vokorokos, B. Madoš, M. Čajkovský, J. Hurtuk, and K. Moravčík, "Analysis of the Software Behaviour Using Forensic Methods for Computer Security Purposes", in Acta Electrotechnica et Informatica, Vol. 14, No. 2, 2014, pp. 36-40

[12] J. A. P. Marpaung, M. Sain, and Hoon-Jae Lee, "Survey on malware evasion techniques: State of the art and challenges," in 2012 14th International Conference on Advanced Communication Technology (ICACT), 2012, pp. 744-749

[13] L. Vokorokos, A. Baláž, and B. Madoš, "Application Security through Sandbox Virtualization", Acta Polytechnica Hungarica, Vol. 12, No. 1, pp. 83-101, 2015

[14] A. Borkar, A. Donode, and A. Kumari, "A survey on Intrusion Detection System (IDS) and Internal Intrusion Detection and protection system (IIDPS)," in 2017 International Conference on Inventive Computing and Informatics (ICICI), 2017, pp. 949-953

[15] L. Vokorokos, A. Baláž, and M. Chovanec, "Intrusion detection system using self organizing map", in Acta Electrotechnica et Informatica, Vol. 6, No. 1, 2006, pp. 81-86

[16] L. Dali et al., "A survey of intrusion detection system," in 2015 2nd World Symposium on Web Applications and Networking, 2015, pp. 1-6

[17] L. Vokorokos, M. Ennert, M. Čajkovský, and J. Radušovský, "A Survey of parallel intrusion detection on graphical processors," Central European Journal of Computer Science, Vol. 4, No. 4, pp. 222-230, Dec. 2014

[18] L. Vokorokos, A. Baláž, and M. Chovanec, "Distributed Detection System of Security Intrusions Based on Partially Ordered Events and Patterns," in Towards Intelligent Engineering and Information Technology, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 389-403

[19] F. Sabahi, and A. Movaghar, "Intrusion Detection: A Survey," in 2008 Third International Conference on Systems and Networks Communications, 2008, pp. 23-26

[20] L. Vokorokos, A. Baláž, and B. Madoš, "Anomaly and Misuse Intrusions Variability Detection", in Acta Electrotechnica et Informatica, Vol. 10, No. 4, 2010, pp. 5-9

[21] E. Chovancová, and N. Ádám, "The Security of Heterogeneous Systems based on Cluster High-interaction Hybrid Honeypot," in 2019 IEEE 23rd International Conference on Intelligent Engineering Systems (INES), 2019, pp. 81-85

[22] J. Briffaut, J. Rouzaud-Cornabas, C. Toinard, and Y. Zemali, "A new approach to enforce the security properties of a clustered high-interaction honeypot," in 2009 International Conference on High Performance Computing Simulation, 2009, pp. 184-192

[23] S. Morishita et al., "Detect Me If You… Oh Wait. An Internet-Wide View of Self-Revealing Honeypots," in 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2019, pp. 134-143

[24] D. Fraunholz, M. Zimmermann, and H. D. Schotten, "An adaptive honeypot configuration, deployment and maintenance strategy," in 2017 19th International Conference on Advanced Communication Technology (ICACT), 2017, pp. 53-57

[25] R. M. Campbell, K. Padayachee, and T. Masombuka, "A survey of honeypot research: Trends and opportunities," in 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), 2015, pp. 208-212

[26] P. Fanfara, M. Dufala, and J. Radušovský, "Autonomous Hybrid Honeypot as the Future of Distributed Computer Systems Security," Acta Polytechnica Hungarica, Vol. 10, pp. 25-42, 2013

[27] L. Vokorokos, P. Fanfara, J. Radušovský, and P. Poór, "Sophisticated Honeypot mechanism – the autonomous hybrid solution for enhancing computer system security," in 2013 IEEE 11th International Symposium on Applied Machine Intelligence and Informatics (SAMI), 2013, pp. 41-46

[28] E. Chovancova et al., "Securing Distributed Computer Systems Using an Advanced Sophisticated Hybrid Honeypot Technology," Computing and Informatics, Vol. 36, No. 1, pp. 113-139, 2017

[29] K. G. Anagnostakis, "Shadow Honeypots," International Journal of Computer and Network Security, Vol. 2, No. 9, September 2010