Computing and Informatics, Vol. 36, 2017, 113–139, doi: 10.4149/cai_2017_1_113

SECURING DISTRIBUTED COMPUTER SYSTEMS USING AN ADVANCED SOPHISTICATED HYBRID HONEYPOT TECHNOLOGY

Eva Chovancová, Norbert Ádám, Anton Baláž
Emília Pietriková

Department of Computers and Informatics
Technical University of Košice
Park Komenského 6, 042 00 Košice, Slovakia
e-mail: {eva.chovancova, norbert.adam, anton.balaz, emilia.pietrikova}@tuke.sk

Peter Feciľak, Slavomír Šimoňák

Department of Computers and Informatics
Technical University of Košice
Letná 9, 042 00 Košice, Slovakia
e-mail: {peter.fecilak, slavomir.simonak}@tuke.sk

Martin Chovanec

Institute of Computer Technology
Technical University of Košice
B. Němcovej 3, 042 00 Košice, Slovakia
e-mail: martin.chovanec@tuke.sk

Abstract. Computer system security is the fastest developing segment in information technology. The conventional approach to system security is mostly aimed at protecting the system, while current trends are focusing on more aggressive forms of protection against potential attackers and intruders. One of the forms of protection is also the application of advanced technology based on the principle of baits –

honeypots. Honeypots are specialized devices aimed at slowing down or diverting the attention of attackers from the critical system resources to allow future examination of the methods and tools used by the attackers. Currently, most honeypots are being configured and managed statically. This paper deals with the design of a sophisticated hybrid honeypot and its properties having in mind enhancing computer system security. The architecture of a sophisticated hybrid honeypot is represented by a single device capable of adapting to a constantly changing environment by using active and passive scanning techniques, which mitigate the disadvantages of low-interaction and high-interaction honeypots. The low-interaction honeypot serves as a proxy for multiple IP addresses and filters out traffic beyond concern, while the high-interaction honeypot provides an optimum level of interaction. The proposed architecture employing the prototype of a hybrid honeypot featuring autonomous operation should represent a security mechanism minimizing the disadvantages of intrusion detection systems and can be used as a solution to increase the security of a distributed computer system rapidly, both autonomously and in real-time.

Keywords: Honeypot, hybrid honeypot, virtual honeypots, malicious code, security of computer systems

Mathematics Subject Classification 2010: 68-U99

1 INTRODUCTION

Today, due to the rapid development of the Internet and the advancements in web technologies, people can search for various information sources easily and simply; however, the information management is getting still more complicated. Nevertheless, if we do not implement the measures of providing computer systems with at least a basic level of security, adequate to the fast paced development of the Internet, attackers might gain control over the system using malicious code or existing vulnerabilities and programming errors in the system. As a result of the potential threats there is an ever-increasing demand to improve information security and intrusion detection in computer systems.

The beginnings of intrusion detection led also to complications. Currently, there is still a gap between the theoretical and the practical side of intrusion detection. This situation resulted in all kinds of experiments related to the subject products for research and development in this problem domain. There have been attempts to define terms and develop adequate solutions cooperating with other security system elements or with the control infrastructure. The requirement to achieve a global solution to the problems with the preferred solution/approach (regardless of the validity of the statement) means another important milestone.

"Traditional/recommended" computer system security is based on employing the three basic pillars of protection: An anti-virus program, a firewall and an *in-*

trusion detection system (IDS). However, the last two systems have two disadvantages [16]:

- As soon as the attackers know that the firewall has an enabled security exception or one of its ports is open towards an external service, they can get access to the internal resources of the system through this service and perform another attack inside.
- An intrusion detection system cannot provide sufficient additional information to detect malicious attacks, nor can it mitigate the losses caused by such attacks.

If we could – right at the first attack – immediately identify a previously unknown vulnerability and a possible attack of the device in the system, analyze the unknown attack and forward the results of similar warnings to security specialists then chances of issuing warnings and implementing system security measures, finding the analyzed attack patterns along with the possible risks, methods and tools would multiply – thus we could prevent potential attacks and other damages. In this way we could successfully mitigate and decrease the risk of information security in advance.

The traditional approach to system security is aimed at the protection of the system, while current trends are focusing on more aggressive forms of protection to tackle the potential attackers and intruders more strongly. Such form is also an intrusion prevention system using baits, commonly referred to as *honeypots*.

This paper deals with problems related to the security of distributed computer systems ensured by intrusion detection systems with the addition of honeypot and the architecture of a sophisticated, autonomously functioning hybrid honeypot that is capable to adapt to environment changes in real-time, without the necessity of human intervention.

2 HONEYPOTS

The honeypots are still new and constantly evolving. Honeypots may be used in various aspects of system security, such as prevention, discovery and data acquisition. What makes honeypots unique is their general nature not their specificity – they do not solve any specific security problem. On the contrary: a honeypot is highly flexible and applicable in various domains such as: intrusion detection, slow-down of the attacker or forensic analysis of the network. All of these depend on the placement of the honeypot and on the targeted goals. Some honeypots may prevent attacks, others might be used to detect them, while next ones may be used to collect research information. As Lance Spitzner, the well-known expert on honeypots said [4, 17]:

> "A honeypot is a security resource who's value lies in being probed, attacked or compromised."

Honeypots may be used in open or private networks. In both cases they gather information related to the behavior of the attacker or perform other specific tasks. Since they use unused IP addresses of the system, it is highly probable that the

incoming data flow represents an irregular connection, thus – compared to an IDS – the probability of identifying the irregular connection is higher. Information security specialists may then use data mining techniques to analyze the motivation and purposes of the attack sporting unknown patterns – by analyzing the content of the network packets. They may use the results to identify methods and tools used in the attack, as well as the motivation of the attacker. The goal of honeypots is to acquire information about system data threats to prevent future infiltrations of the computer system data [4, 10].

2.1 State-of-the-Art

Honeypots are closely monitored network baits. They are available in various shapes and sizes, serving various purposes. They may be installed in a computer network having a firewall both in front of and behind the firewall, even in demilitarized zones – these are the most popular locations among the attackers trying to gain access to the system – this is where one can collect the most information about the attackers. A honeypot may be some fictitious database records, a low-interaction network device or a high-interaction workstation running a real operating system and providing real services [12].

Honeypots may serve many purposes. The most important functions of a honeypot are the following [17]:

- To divert the attention of the attackers from devices having a higher value in the system to ensure that the main information resources remain unexposed to threats.
- To identify new, previously unknown vulnerabilities and risks of operating systems, development environments and programs.
- To issue timely warnings about new attack types and the use of unknown tactics and trends.
- To allow a more detailed analysis of the attackers' activities during and after interacting with the honeypot.
- To provide information on new attacks and new techniques captured new viruses or worms for later analysis.

The benefit of honeypots [6, 15] lies mainly in their detection capabilities. Due to their simplicity, honeypots are capable of overcoming the disadvantages of intrusion detection systems – they minimize the amount of false alarms generated. There are some situations in which intrusion detection systems are incapable of detecting attacks: the attack is too short, the corresponding security rule results into many false alarms or it detects an overload in the system and therefore it discards packets. False alarms occur when the intrusion detection system is not set up correctly and it generates too many warnings during normal network operation. These warnings are then either ignored or their execution rules are soon modified, therefore resulting

in leaving a real attack unidentified. A possible solution to this issue is to use honeypots, since they do not influence system operation. Any interaction with the honeypot is almost surely unauthorized – i.e., no false alarms are generated and there are no large volume datasets to be analyzed. As soon as an intrusion is detected, the honeypot may switch to off-line mode and an analysis may be performed. This is difficult (if not impossible) in real systems [6, 15].

2.2 Honeyd

Honeyd is not exactly a honeypot, but rather a honeypot software tool allowing the creation and amendment of a requested solution. Honeyd is an open-source tool, thus probably the most potent, freely accessible low-interaction honeypot capable of creating hundreds of virtual hosts1 and simulating large network structures. The created virtual honeypots are then executed on the Honeyd server using a free IP address. It may be also implemented on a single device connected to the network using ARP spoofing. Host computers may be configured to execute any kind of service; they may be additionally personalized to mimic the execution of various operating systems and network services. One may modify the latency, loss rate and bandwidth of the network within the computing system [11, 14].

The most significant advantage of Honeyd is that it is not passive in the incoming and outgoing honeypot traffic. It creates traffic, which it sends through its configured subsystems. It provides a mechanism to identify and assess the level of threat. Honeyd emulates a device at an unused IP address and provides a "facade" the attacker may attack. Honeyd can emulate more than 400 operating systems at both the level of applications and IP-level. After receiving a signal or identifying a connection to the non-existent system, Honeyd suspects the connection which is an attack. After the identification of this kind of traffic it records the target IP address. Subsequently, it starts the emulation of the services for the port, where the connection to the system was identified. After the start of the emulated service, it starts the interaction with the attacker recording all of his activities. Service emulation ends immediately after the attacker terminates the connection. Normally, honeypots created by the Honeyd server are capable of identifying and recording any activity on any UDP or TCP port and also some ICMP activities. The IP stack itself is emulated in user land and the packets are delivered to the ICMP, TCP or UDP service programs. TCP and UDP service programs are components emulating the ICMP stack [5, 7].

In general, service programs transfer packets to the respective emulated services using scripts. The services emulated by Honeyd honeypots used for the interaction with the attackers include Telnet, FTP, HTTP, POP3 and SMTP servers. In case of viruses, backdoors are emulated. For example, in case of packets targeted at port 80 these are routed to a script emulating a Web server. Scripts might be external programs or proxies to real services. The personalization component is the one responsible for the correct setting of the emulated IP address behavior. The whole architecture is displayed in Figure 1.

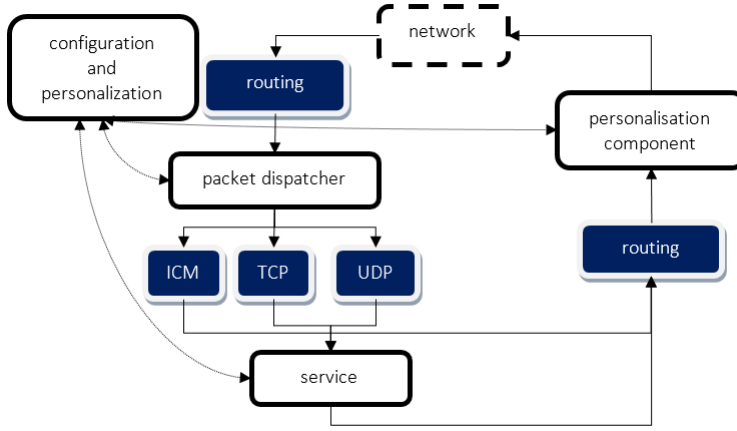


Figure 1. A simplified view of the Honeyd architecture [11]

In Honeyd, the virtual honeypots are configured using a template created by the configuration file, defining properties of the honeypot, including the operating system, ports where the system listens and the behavior of the emulated services [5, 7].

2.3 Honeynet

This honeypot type represents a network architecture containing one or more honeypots. Specifically, it is a high interaction honeypot designed with the goal to acquire the most data about the threats and to provide a real operating system, applications and services for the attacker to interact with [13]. With virtual solutions it is possible to implement various operating systems on a single physical device, while these will appear to the attacker as separate, independent devices; however, as far as topology and physical location is concerned, these are going to be at the same place. This flexible solution allows the interaction between the attacker and the system acting as a lure without letting the attacker harm the computers located beyond the Honeynet [1, 3].

2.4 Honeypot Advantages

Honeypots have several significant advantages compared to the most popular security mechanisms [4, 17]:

Fidelity – recording data interacting with the attacker, since honeypots monitor only traffic coming directly from them. Honeypot datasets may be small; nevertheless, they may contain high-value information.

Discovery of new tools and tactics – honeypots capture anything interacting with them, such as previously unknown tools and tactics of the attackers.

Minimal resources – since they capture malicious activity only in direct interaction, they need only minimal resources to operate correctly. Thus, one may use even decommissioned or low-end devices.

Simplicity – honeypots are very simple and flexible. They do not require complicated algorithms or costly administration to work properly in distributed computer systems.

All these benefits show how honeypots increase system security and how they can increase the overall security of distributed computer systems.

2.5 Honeypot Disadvantages

Honeypots have also some disadvantages and risks. Even though these are not numerous, they prevent honeypots from completely substituting the current security mechanisms [4, 17]:

Limited view – a honeypot may identify and monitor the activities of an attacker only if the attacker communicates with the honeypot directly. Attacks to other parts of the system are not recorded if the honeypot is subject to the attack.

Discovery and fingerprinting – a honeypot has certain expected features and behavior, which allows the attacker to discover its real identity [8]. A simple error, such as typographical mistake in the emulated service may be a sign of interacting with a honeypot.

Risk of takeover – if the attacker gains control over a honeypot, he may abuse it in an attack aimed at the devices within or beyond the system. Such a controlled honeypot may be used to store and distribute illegal data.

3 THE PROPOSED SOPHISTICATED HYBRID HONEYPOT

There are many complications to be taken into account when deploying a honeypot in a system. After all installation and configuration procedures are completed, there must be someone to maintain the honeypot deployed into the production environment. The speed of development and changes do not affect only the honeypot, but also the system it is deployed in. The system is a venue of constant changes, such as adding new and removing old hardware or various improvements. The software components of the systems are also subject to constant change – software is subject to constant amendments and development. All existing honeypots must adapt to these changes. The traditional/manual way of updating and modifying honeypots – aimed at increasing system security – costs time, money and is prone to errors.

The main problems are configuration and maintenance focused on the functionality and efficiency of honeypots. A further disadvantage of using honeypots is their low flexibility in reacting to changes made to the system and representing the risk of being discovered or – eventually – abused. The most significant disadvantage of the

majority of security technology/techniques, including honeypots, is the necessity of their configuration. All technology – from encryption keys to firewall rules – need a human expert to analyze a problem and come to, configure and implement a solution. Nevertheless, work does not end with implementation of the more complex technology. After deploying it, a honeypot requires every-day tedious maintenance. Even the slightest configuration error may tell the attacker about the presence of a honeypot – an experienced attacker may use this knowledge for his own benefit. Honeypot configuration problems include specifications of the operating system type, how many and which UDP and TCP ports should remain open, what network services to emulate, which IP addresses to monitor, how to behave in the interaction with the attacker – when to reply and when to be passive [8, 9].

3.1 Architecture

The honeypot is deployed as an independent device, physically connected to the computer system network. After deployment honeypot monitors and learns to know the system it is deployed in. It also analyses network traffic, determines the number of systems in use, the operating system types, the started and provided services, and identifies the subjects communicating with the specific system with and how often they communicate with the system. This information serves for mapping and collecting information about the network. As soon as the honeypot collects all required information, it may start with deploying virtual honeypots created for the purpose of mirroring the whole system.

Honeypots capable of mimicking the appearance and the behavior of the production environment seamlessly integrate in their environment. That makes their identification or discovery much more difficult for the attackers. However, information acquisition does not end at this point, on the contrary: it is a continuous process performed by monitoring the whole network system. This approach increases the flexibility, since any change is identified in real-time and implemented by the virtual honeypots deployed in the system as soon as possible. The proposed sophisticated honeypot reduces the configuration and administration workload necessary in a constantly changing and evolving environment.

3.2 Hybrid Honeypot

A honeypot is created and configured to make it an easy target for attackers. It may be used in various security scenarios as a detection device, a protection mechanism or a reaction mechanism. Moreover, it may be situated in a system to consume the system resources of the attackers or to divert their attention from important targets, i.e. to slow them down. Slowing the attackers down is one of the most popular honeypot functions, since the attacker "loses time" with the honeypot, instead of attacking systems and servers. A hybrid honeypot is a combination of two honeypots (Figure 2) with different interaction levels [12].

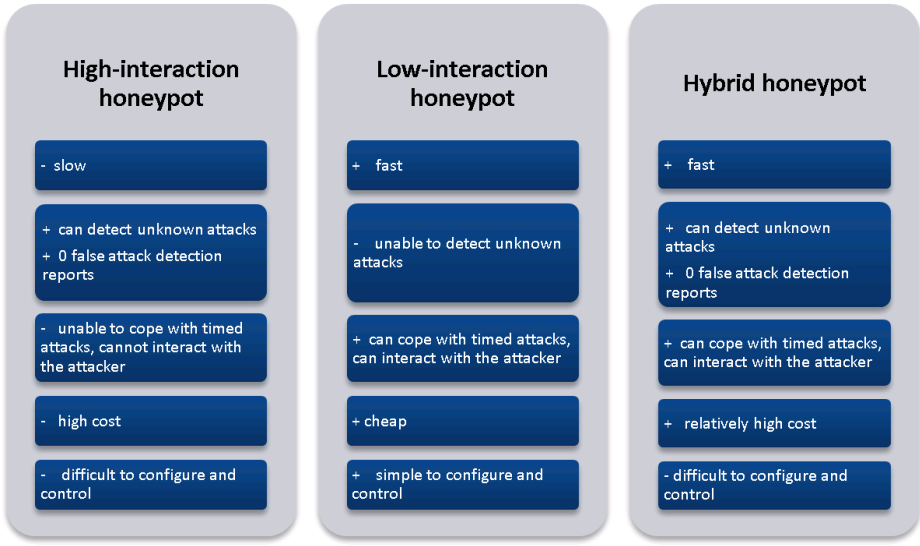


Figure 2. The principle of the hybrid honeypot

The first and most critical part of a sophisticated honeypot is the way it can obtain information about the network. After obtaining information, the sophisticated honeypot will smartly map and promptly react to the environment.

One possibility is by an active probing to determine the used systems, their type and used service. Active scanning and data collecting leads to still increasing load, so this method does not constitute the best approach. Honeypot is deployed as a standalone device that is physically connected to the network. Honeypot follows-up, learns and analyzes the network and based on this information determines the number of systems used, OS type, provided services and topology.

When honeypot collects all the necessary information, it can begin to place virtual honeypots that are designed to mirror the system. Gaining information does not end, but is continuing, so the entire network system is monitored. This approach is more flexible, because any change is identified in real time.

The proposed architecture of a sophisticated hybrid honeypot consists of four main modules (Figure 3). The external modules are Low-interaction honeypot and HoneyWall gateway. The first module called Low-interaction honeypot is aimed at identification of the scope of relevant data concerning the technology and topology used in the production network, where the honeypots are to be dynamically deployed. The first module focuses on finding data about the technology and topology used in a production network for deploying Honeypot in a dynamic way. This module also contains network scanning component, network scanning parser and system database which contains all configurations for specific operation systems or devices which can be placed in network.

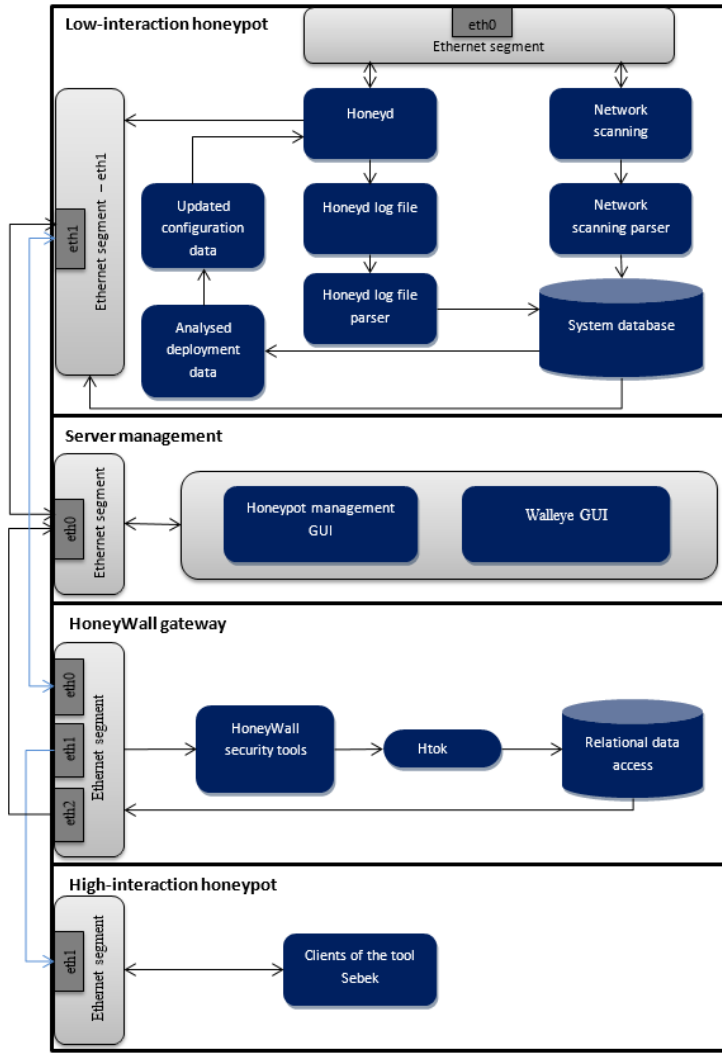


Figure 3. The logical structure of the proposed architecture

This module is aimed at covering a large IP address space and filtering irrelevant network operation out - it can divert traffic meeting the specified criteria to the high-interaction honeypot. The second module Server management is internal and is responsible for management. The third module HoneyWall gateway serves to prevent an attacker to jump from one device to another network device. The last module is a high-interaction honeypot, providing mimicking a real system to the attackers.

Low and high-interaction honeypot modules can be placed on the same device. It is not recommend to do that from the security point of view. Placing both honeypots on the same instance/device can lead to obtain system sensitive data located on high-interaction honeypot. Purpose of hybrid honeypot is to mitigate the majority of automated attack and lower the payload of high-interaction honeypot. Once the low-interaction honeypot evaluates attack as "made by human" (not by bot or other automated type of attacks) it will automatically forward this attack to high-interaction honeypot to gather data about the attacker. This behavior is also called as proxy mode.

3.2.1 Acquiring Information About the Deployment System

The first and most critical part of achieving the adaptive feature of the sophisticated hybrid honeypot is to select the appropriate data acquisition techniques in conjunction with the deployment system. The information acquisition module of the sophisticated hybrid honeypot – deciding which methods to choose in the selected architecture – uses both possible techniques: active and passive.

If the deployment system consisted only of routed networks, using the passive technique was inadequate, since the acquired fingerprint may not get beyond the specific routed network until the router is reconfigured. Should the deployment system consist of servers, workstations and other peripheral devices interconnected by routers operating on the second level of the OSI model, the so-called connection layer, using the active technique would result in a less adequate data acquisition approach (compared to the passive technique). To provide the highest possible precision rate in fingerprinting the remote device, the applied hybrid solution uses active fingerprinting combined with the passive technique.

The scanning process used during the initial deployment of the sophisticated hybrid honeypot initializes a ping scan of the Nmap tool – the active technique – to locate all devices and temporarily store their IP address in a list. The created list serves as an aid during scanning ports and/or operating systems within the specific group of IP addresses. The output is saved as an XML file, which is then analyzed after each scanning. After the scanning is finished, the analyzed object is initialized and executed in the thread extracting information from the XML file used to create the system profile; the information is then saved to the database.

3.2.2 Continuous Data Acquisition

After the initial deployment of the sophisticated hybrid honeypot and termination of the initial scanning process, the continuous data acquisition process of the Honeyd server serves as an aid during deployment the virtual honeypots. Its algorithm is depicted in Figure 4. The applied algorithm uses a combination of various search tools and techniques.

Passive fingerprinting captures network packets to identify network activities, analyze them and determine the IP address, operating system and services for the

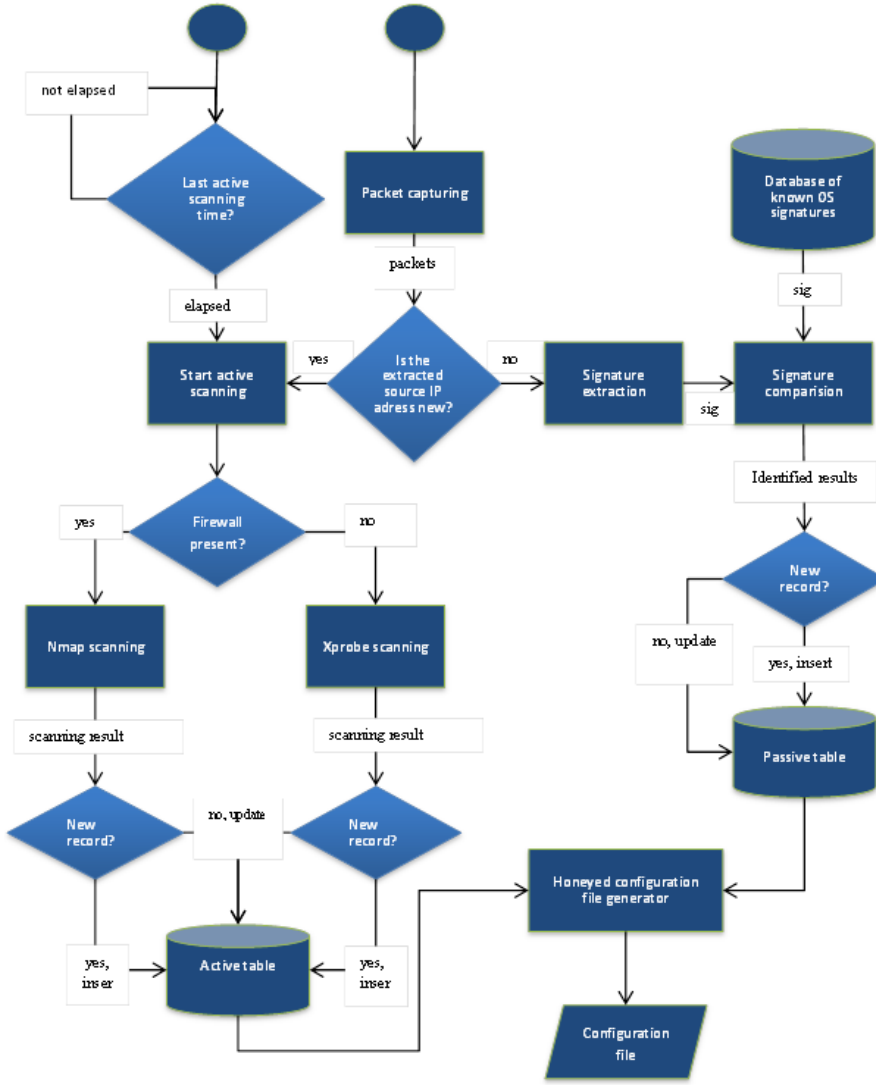


Figure 4. The data acquisition process used for the deployment of virtual honeypots

creation and deployment of virtual honeypots. This technique serves for the continuous acquisition of data ultimately needed to emulate and/or simulate devices in the deployment system without additionally increasing network traffic. Snort [2] is a data acquisition tool and an intrusion prevention system. To reach the highest efficiency in capturing the packets, Snort is placed near the gateway of the production system. After detecting match with any record in the knowledge-based database, IDS Snort can also raise an alarm.

As part of the continuous data acquisition process both active and passive techniques are used to update the initial configuration file created by the algorithm. During passive scanning, the p0f and Snort tools are being used to extract the operating system signatures, the state of ports and other information. After extracting signatures, the proposed system compares the data with the database of signatures of the known operating systems. Subsequently, the data are handed over to the network scanning passers and stored in the system database to create and/or update the configuration file. Since passive scanning captures only the interaction of the host, ports and services, it provides limited information. Therefore, the execution of active scanning is preset for a predefined timeframe – usually at times of lower loads and for a small IP address group; this minimizes the possibility of filling up bandwidth. To acquire new information, active scanning is running on all IP addresses in the specified timeframe. If a firewall is detected in the computer network, the proposed sophisticated honeypot initializes active scanning using Nmap. If any changes are identified in the deployment system during passive scanning, such as a new IP address of a device, the preset active scanning is automatically executed. If a firewall is present, Nmap is executed. If the identified change is new – either from the aspect of topology or technology – the sophisticated honeypot inserts all information about the deployment system into a MySQL database. Otherwise it updates the existing information modified compared to the previous state.

After fully scanning the production network of the deployment system using passive and active techniques, the proposed sophisticated hybrid honeypot can estimate the amount, type and kinds of services of the counterfeit systems (honeypots) relatively precisely, with the goal to create an appropriate configuration file to deploy the virtual honeypots in the deployment system.

3.2.3 Placing and Deploying Honeypots

The traditional solution to the problem of implementing honeypots in a system requires the physical placement of a new device to each monitored IP address. In addition to this, distributing physical honeypots takes also a significant time and incures labor cost. A simpler, more autonomous solution is to implement virtual honeypots instead of physical ones – these allow monitoring of sufficient amounts of unused IP addresses. The virtual honeypots monitor an IP address space identical to the system itself. All virtual honeypots were created, deployed and managed by a single physical device – the proposed sophisticated hybrid honeypot.

The basic idea behind the sophisticated hybrid honeypot is using free IP addresses. The main problem is how to distribute IP addresses among the existing operating systems and simultaneously mitigate the possibility of discovering the host stations in the network in case of an attack. We have selected the following approach to maintain a constant operating system distribution even after adding a virtual honeypot to the system and the exclusion of physical honeypots:

Let us define DD as the total number of deployed devices in the deployment honeypots, NDo_i as the number of production devices running a specific operating

system (the i^{th} operating system in the list) and let *NFIP* represent the number of free IP addresses available for the virtual honeypots. Then the number of virtual honeypots created by the Honeyd tool ($NVHo_i$), emulating the i^{th} operating system is as follows:

$$NVHo_i = \left(\frac{NFIP}{DD}\right) * NDo_i * (1 - PFIP). \tag{1}$$

PFIP is the percentage of free, unused IP addresses – these are available for other uses. Therefore, to the attacker it seems that the system expands the number of hosts running OS_i according to the value of $NVHo_i$.

3.3 Security Elements

The implementation of additional security elements of the sophisticated hybrid honeypot directly increases its security when facing experienced attackers and minimizes the direct risk the proposed honeypot faces.

The verification module implemented in the hybrid honeypot diverts the outbound DNS queries during the interaction with the attacker – for logging purposes – to the Snort tool, while it allows outbound connections only to the attacker. If the sophisticated hybrid honeypot does not capture any operation from the attacker in the high-interaction honeypot during the predefined timeframe of two minutes, identical measures will be taken – the honeypot starts the stability scan and reverts the configuration to its initial state.

To prevent a single attacker taking control of the high-interaction honeypot during a long time, the sophisticated hybrid honeypot contains an additional timer measuring the attack relation length. The preset value is 10 minutes. After the configured period, the attacker shall be routed to the low-interaction honeypot, while a scanning process is launched at the high-interaction honeypot to identify break-ins and revert it to its initial state.

4 EXPERIMENTAL PROOF

During the selection of the environment appropriate for the implementation and simulation, we took the availability of the environment and the expected specific events, eventually occurring during the implementation of the proposed architecture into account. During the selection process itself, we focused also on the possibility of an experimental proof of the proposed solution. Since we had to deal with a heterogeneous distributed computing system, we had to select an adequate environment – we have selected the following two:

- ANTIK Telecom, s. r. o. – functionality tests of the hybrid honeypot prototype (gross estimate: 50 000 active users).
- The network of the student hostels Jedlíkova 9 and Jedlíkova 5 of the Technical University in Košice (the TUKE13 heterogeneous network) – functionality tests of the hybrid honeypot prototype (approximately 2 000 active users).

- The topology proving the autonomous operation of the proposed sophisticated hybrid honeypot by simulating an evolving computer system – operating system variations, provided services and port configurations. The topology has been created at the Computer Architecture and Security Laboratory, Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovakia.

Hypothesis 1. The implemented architecture of the hybrid honeypot is capable of capturing more files containing malicious code than VirusTotal tool, consisting of 42 types of detection and heuristic techniques at the time of writing this paper.

Experiment 1. To expose the architecture of the hybrid honeypot to any malicious code or attacker and to test it we have deployed the prototype for a period of three weeks into a real environment of two heterogeneous networks – the network of ANTIK Telecom and TUKE student hostel network. Both computer networks contained a firewall – this allowed all inbound and outbound traffic. However, in the TUKE computer network, traffic coming from beyond the internal network TUKE was filtered by the university Internet service provider, which blocked some inbound connections containing malware. Nevertheless, some connections generated from beyond the TUKE network reached the deployed hybrid honeypot: during the specified test period we have registered a total of 1,762,289 TCP connections and 710 670 UDP connections. The details are stated in Table 1.

	TUKE student hostel network		ANTIK Telecom	
Protocol	TCP	UDP	TCP	UDP
Week 1	371 486	132 172	211 578	100 710
Week 2	365 635	136 253	223 349	103 357
Week 3	358 456	137 092	231 785	101 086
Total	1 095 577	405 517	666 712	305 153

Table 1. Connections recorded by the hybrid honeypot, aggregated by protocol

From Table 1 it is evident that TCP is the most popular Internet protocol (TCP/IP protocol) used by the attackers. This only underlines the fact that compared to other protocols, most services and applications use primarily the TCP protocol. The UDP protocol also had a significant impact on the overall measurements conducted during the experimental proof.

In the ANTIK Telecom network, the high-interaction honeypot was exposed to 7 683 connections in 2 358 relations during the test period. We have captured and extracted 1 861 malicious files or files containing malicious code (also with the help of the Dionaea tool) from these connections. 589 files of these have been verified using VirusTotal.

Malware captured most often at the low-interaction honeypot was the Win32.-All-aple.e14 worm. Similarly, some identical infections where registered also at the high-interaction honeypot: these represented 13 % of the total unique records and

52 % of all captured malware detected by VirusTotal. We have captured multiple variants of the Win32.Allaple worm attempting to connect to the default gateway operating at port 445 – most probably it was an attempt of self-propagation in a local network and contacting unknown web servers.

Figure 5 shows the details of unique detection records, as reported by VirusTotal, both from the low-interaction and high-interaction honeypots. We have also evaluated the number of infections attempting to create a connection to an IP address different from the IP address of the attacker.

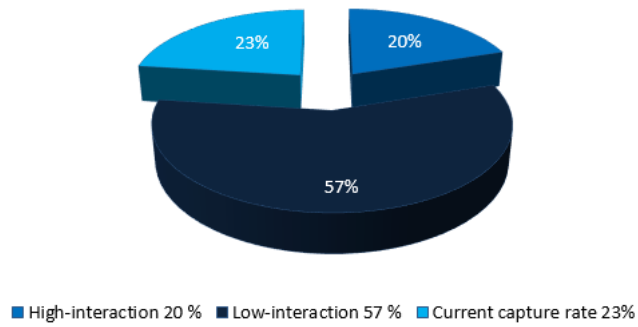


Figure 5. Unique records detected by the hybrid honeypot and the VirusTotal tool

The results depicted in Figure 6 show a relation15, which ended in an attempt of the high-interaction honeypot to create an outbound connection16 – it contained more captured files containing malware than 98 % of the files captured by VirusTotal. On the other hand, most relations – not attempting to create a final outbound connection – led to no modification of the file system. Moreover, VirusTotal verified only 26 % of the extracted files during these relations. Based on the above we may claim that an attempt to create an outbound connection is a better indicator of a successful abuse and/or infiltration. However, such a claim does not say much, if it is the only indicator examined.

The log file records showed significant amounts of initiated Telnet and SSH activities. The duration of these was very short – only a few seconds – in case of the virtual low-interaction honeypots – this indicates that these activities might be probing or have the effect of worms.

Hypothesis 2. Before attacking a potential target, the attacker scans the environment he is preparing to attack (to find the weaknesses and vulnerabilities). Subsequently, he attacks one of the eventually found weaknesses.

Experiment 2. In this section we describe the results of analyzing data captured by the honeypot in accordance with the warnings issued by the intrusion detection system during one week long operation in the TUKE student hostel computer network. To detect an issue IDS warnings we have selected and used the Snort tool. During the experiment aimed at the analysis we have first counted the total number

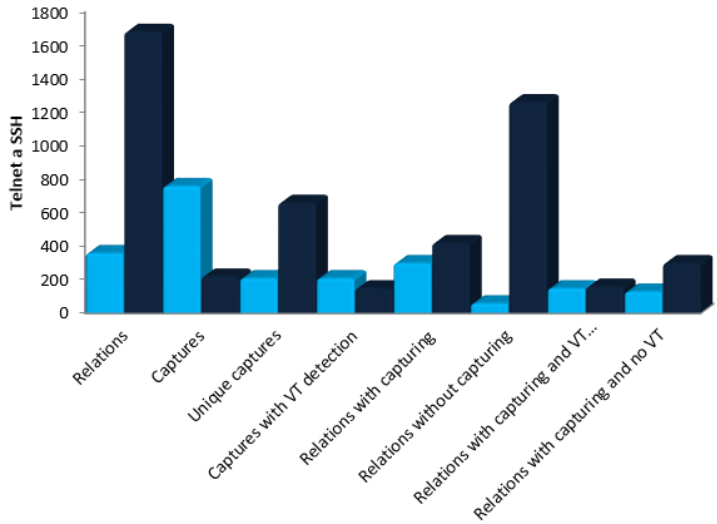


Figure 6. Statistics of the hybrid honeypot and VirusTotal in the heterogeneous TUKE student hostel network

of relations captured by the Snort IDS – we have recorded a total of 75 368 attack attempts. Snort has issued warnings in 815 relations of all these connections. 75.7 % of all attacks where denial of service (DoS) attacks during the specified period. The average relation count detected by Snort per day was 116.

By analyzing the hybrid honeypot (its low-interaction part) we have found that each attack was preceded by an environment scan. After the attacker received the data related to the target, he used them to perform an intrusion into the system. After the scanning process, the attacker is capable of identifying all vulnerabilities and open ports he may use for an eventual attack. Employing a high-interaction honeypot has multiple advantages compared to a low-interaction honeypot. First of all, the attacker may find out relatively quickly that the services provided by the low interaction honeypot are emulated. The services offered by the high interaction honeypot are not emulated at all, what prevents the attacker from using remote fingerprint acquisition. Since the high-interaction honeypot is fully virtualised, the risk of an error or mistake in the emulation and/or simulation is in fact equal to nil. In spite of this, the identity of the honeypot may be discovered as soon as the attacker is routed to a virtualised high interaction honeypot.

The measurement results may be affected by timed attacks, used by the attackers to scan the environment – these may try to measure the system memory load caused by the execution of the hybrid honeypot. However, the oscillation of the system memory load is not a unique fingerprint of the monitoring environment,

since executing multiple hosts may lead to similar symptoms. During the three-week experimental proof of the hybrid honeypot we met managed to capture and analyze 3 126 malware samples, while 68 % of these were not captured by antivirus software. This served as a proof of Hypothesis 1. It also proved Hypothesis 2, when the measurements performed in Experiment 2 showed that a scanning process preceded each attack in the vast majority of cases – this was to perform the discovery of existing weaknesses of the system, with the goal to use them during the implementation of the attack.

5 PROOF OF AUTONOMOUS OPERATION

We have separated the experimental proof of the autonomous operation of the sophisticated hybrid honeypot – simulating weaknesses and vulnerabilities of a specific device in the computer network – into two experiments. In the first experiment we tried to prove the acquisition of data concerning the deployment system, necessary to create the configuration file for the honeypot and ensure continuity in case of any changes in the deployment system. As the heterogeneous test network we have selected the TUKE student hostel computer network. The second experiment was aimed at proving the autonomous operation itself and the creation of virtual honeypots using a Honeyd server in the simulation environment of the Computer Architecture and Security Laboratory, Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovakia.

Experiment 3. (System data acquisition in a heterogeneous network.) After deploying the Honeyd server in the deployment system and after the initial scanning of the devices present in the system, the server started producing information on the IP addresses and the operating systems in use. Figure 7 shows the distribution of unused IP addresses and IP addresses allocated to active devices.

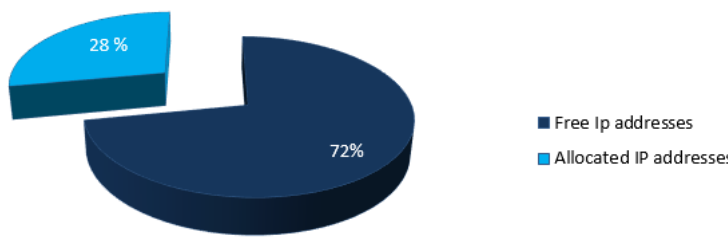


Figure 7. Distribution of IP addresses in a heterogeneous network

From the figure it is evident that the computer network should be able to cope with numerous virtual honeypots and it should most probably be able to divert

malicious operation from the active devices of the system to the created virtual honeypots.

Figure 8 shows the distribution of the detected operating systems – this proves that the TUKE student hostel computer network is a heterogeneous environment, mainly due to the number of operating systems in use. Therefore, the employed virtual honeypots should mirror the aforementioned operating systems at rates identical to those detected in the deployment system.

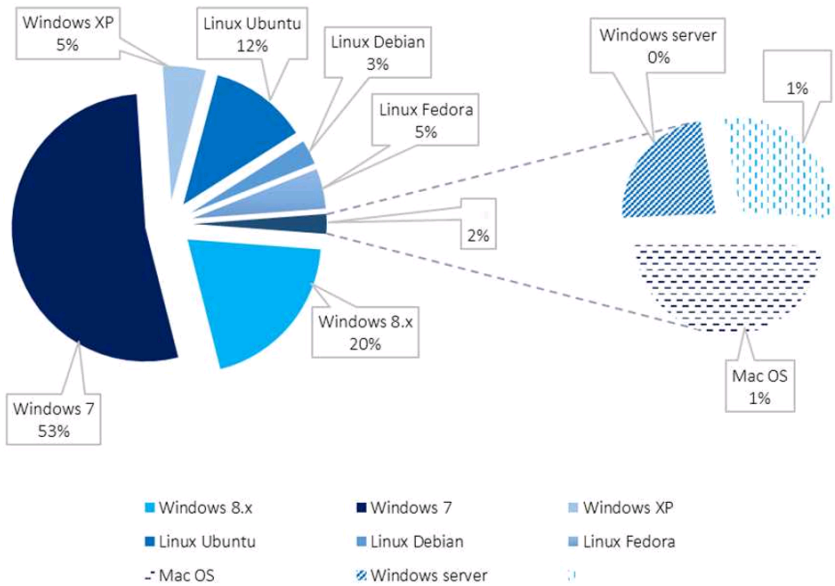


Figure 8. Distribution of specific operating systems within the heterogeneous network

As it is evident from Figure 8, devices running Windows server operating systems – 12 pcs – do not represent even 1 % of the deployment system, which (naturally) need not strictly correspond to other deployment systems.

During the three-month experiment, the prototype of the sophisticated hybrid honeypot specified not only data identifying the distribution of IP addresses and the counts of the respective operating systems, but also the distribution of packets passing through the deployed virtual honeypots, aggregated by the specified basic protocols – TCP, UDP and ICMP. After examining the chart in Figure 9 one may conclude that the traffic passing through the deployed low-interaction honeypots reflected the respective device type of the deployment system, which they mirrored. E.g., the most attractive honeypot was the one bearing ID 17, which emulated the DNS, while honeypots bearing the IDs 7, 15 and 25 emulated an HTTP server.

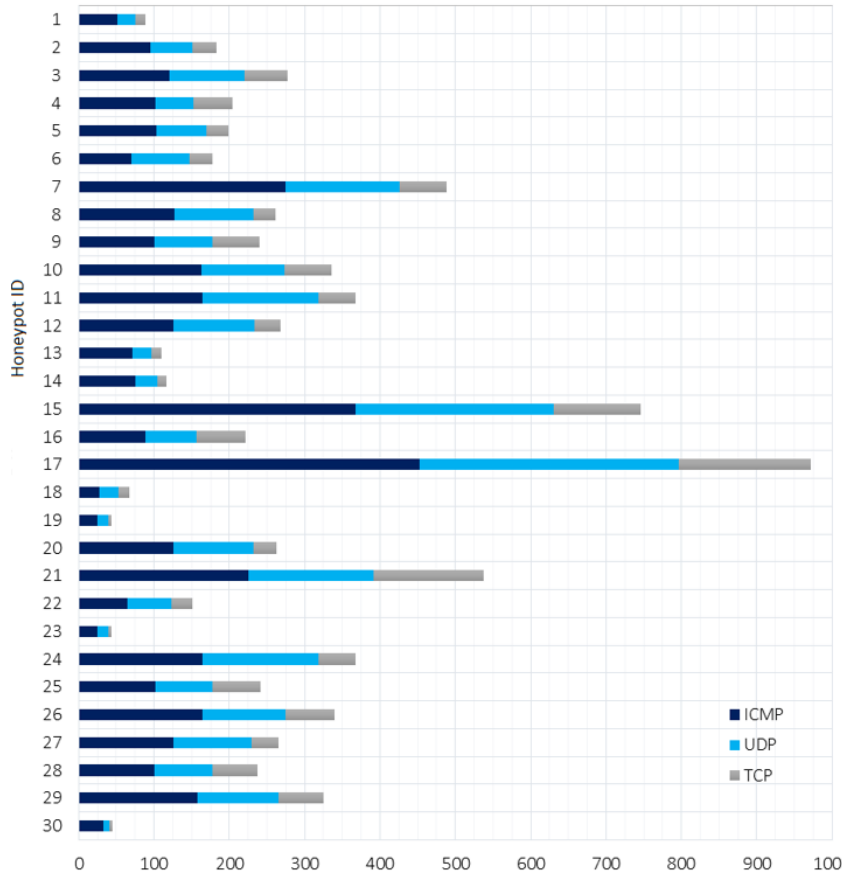


Figure 9. Duration of the attackers' interaction during the specified period in seconds

These charts show that the data collection algorithm was designed and implemented correctly, since the prototype of the sophisticated hybrid honeypot recorded any and all changes in the deployment system.

Experiment 4. (Proof of autonomous operation in the topology of the Computer Architecture and Security Laboratory)**.** After verifying the data acquisition concerning the deployment system and the partial proof of autonomous operation in Experiment 3, we performed an experiment in the environment consisting of 12 computers and a central server (being the Honeyd server), with the topology illustrated in Figure 10.

The proposed environment simulated various changes in the operating system versions running, the various service configurations and ports. For example, the ports open at the machine running Windows Server 2008 Enterprise 64-bit SP2

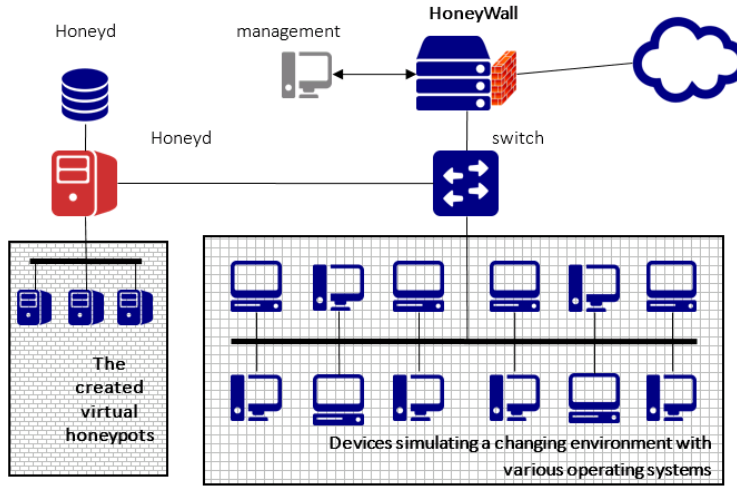


Figure 10. The topology of the network at the Computer Architecture and Security Laboratory, TUKE

included: HTTP (80), DNS (53), Telnet (23), FTP (21), SSH (22), etc. The other ports have been reset. To create the low-interaction honeypots we used the Honeyd tool, whereas for the high-interaction honeypots we used VMware software. During the experiment lasting one week we used various operating systems, as specified in Table 2. Our goal was to prove the autonomous operation of the proposed solution.

ID	*Operating System*
1	Ubuntu Server 10.04.4 LTD
2	Ubuntu 11.10 i386
3	Ubuntu 12.04.1 LTS 64-bit
4	Fedora 17 (Beefy Miracle)
5	Windows 7 Enterprise x64 SP1
6	Windows 7 Enterprise x86 SP1
7	Windows 8.1
8	Windows Server 2008 Enterprise 64-bit SP2
9	Windows XP Professional x64 SP3
10	Windows XP Professional x86 SP3
11	Sun Solaris 10
12	FreeBSD 9.1 RC3 i386
13	IMB OS/2 Warp 4.52
14	Windows Phone 7.x (experimentally)

Table 2. The operating systems used in the experimental proof of autonomous operation

During the one-week-long experiment we successfully proved the following features: system data acquisition, configuration file generation and creation of virtual

honeypots based on system changes in real-time; all of this even in spite of the tough simulations performed in the laboratory environment. During the experiment, we gradually added and removed operating systems. Similarly, their services and port configurations were gradually reconfigured to new values, while the reaction of the Honeyd server – from the implemented changes in the deployment system up to the update or creation/removal of the virtual honeypot – depended on the process of acquiring data about the deployment system, the efficiency of which directly depended on the network traffic in the system at the time. The average reaction time of the Honeyd server experienced with the specific, most popular operating systems is depicted in Figure 11.

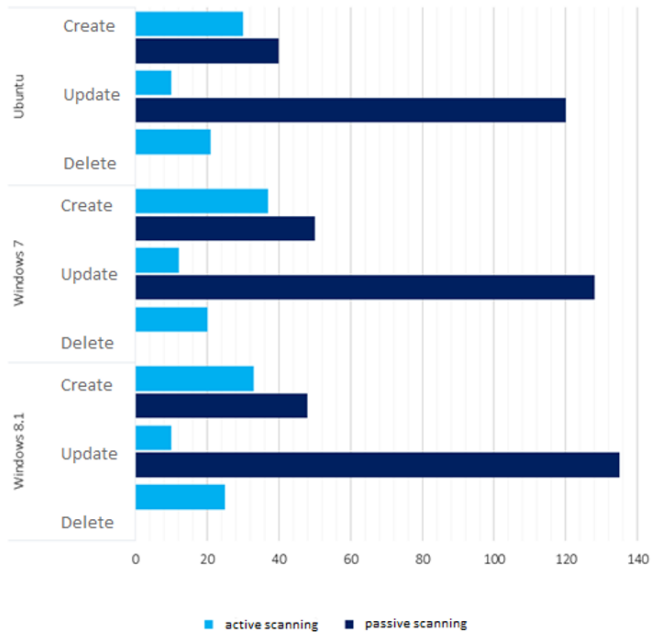


Figure 11. The mean reaction time following a change to the selected OS in the deployment system

Adding and/or creating a new device was detected in a short time, since this meant the registration of a new source IP address, which triggered active scanning. When updating the devices in the topology we simulated changes only in the port and service configuration. Removal of devices could not be exactly proven, since detection by passive data acquisition is not possible – it is impossible to determine whether a device is not present in the system or just not in use.

Based on the measurements of the sophisticated hybrid honeypot, Experiment 3 proves that the proposed algorithm of system data acquisition works correctly and reliably. Subsequently, in Experiment 4 we verified autonomous operation, which

showed that the prototype of the proposed solution managed to collect, remove or update honeypots in the deployment system in real-time, even in laboratory conditions created at the Computer Architecture and Security Laboratory, Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovakia.

6 CONCLUSION

The proposed detection system using a sophisticated hybrid honeypot consists of three main parts: the HoneyWall, a Honeyd server creating virtual honeypots and the Honeynet (A network of physical honeypots). The prototype itself may be considered proactive, while the capability of the system to divert intrusive traffic, write the data to log files and issue warnings makes it a reactive system, similarly to IDS systems.

The advantages of the proposed architecture are the following:

Minimal volume of false alarms: the generation rate of false alarms is suspected to the minimal, since honeypots do not contain real-world valuable data, therefore anybody entering into interaction with them poses a potential risk.

Scalability: the system may be enhanced at any time by creating additional virtual honeypots, and – under specific conditions – even physical honeypots; this directly contributes to an increase in the level of security, mitigates the probability of an attack to real devices containing sensitive data.

Adaptability: as to adaptability, scanning algorithms allow the proposed prototype to adapt to the overall configuration of virtual honeypots; this makes them capable of reflecting any change in the deployment system in real-time. A significant advantage of this solution is that no external help, training or requalification is necessary in abnormal network conditions.

Diminished system load: to decrease the load most efficiently, traffic is diverted to the high-interaction honeypots from the low-interaction honeypots only after meeting the defined conditions, what prevents automated attacks; the honeypots may reroute traffic only if they have the same operating system, provide the same services and have identical port configurations.

Platform independence: it is also evident that the proposed prototype does not strictly require the use of a specific platform or tool.

Diverting dangerous traffic from real systems: one of the basic features of the proposed architecture is its capability to divert dangerous traffic away from real devices in the system of deployment and therefore from the sensitive data they may contain. The level of protection is directly proportional to the available amount of free IP addresses necessary for the deployment of virtual honeypots. The increase in the level of security is exponential depending, on the number of free IP addresses, which may be very effective in reducing the number of malicious attacks led against the production system.

Timely intrusion detection: other systems are capable of detecting intrusions only after the corruption of one or more hosts. Since attackers are always on the run, the proposed architecture is aimed at attracting them and creating an illusion of interaction with a real device.

The proposed architecture of a sophisticated hybrid honeypot may be considered to be the fastest system capable of identification identifying intruders; however, only after they started interacting with these intruders. Therefore it may be considered a real system, applicable for detection of intrusions in real-time.

Acknowledgments

This paper is the output of the research supported by grant APPV-0008-10 entitled "Modeling, simulation and implementation of high-throughput network security tool architectures using GPGPU support". This article is a follow-up of the research conducted in the domain of distributed computers and information system security, conducted at the Computer Architecture and Security Laboratory, Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovakia.

This publication is also the result of the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF. We support research activities in Slovakia/This project is being co-financed by the European Union.

REFERENCES

[1] Honeynet Research Alliance: Project Honeynet Website (2003), `http://old.Honeynet.org/alliance/`.

[2] Snort Official Documentation (12 2013), `http://www.snort.org/docs`.

[3] Abhishek, M.—Debabrat, B.—Kanchan, V.—Debasish, J.: Honeypot in Network Security: A Survey. Proceedings of the 2011 International Conference on Communication, Computing and Security, February 12–14, 2011, pp. 600–605.

[4] Akkaya, D.—Thalgott, F.: Honeypots in Network Security: How to Monitor and Keep Track of the Newest Cyber-Attacks by Trapping Hackers. LAP Lambert Academic Publishing, Germany, 2012.

[5] Baumann, R.: A Low Involvement Honeypot in Action. `http://security.rbaumann.net/download/Honeyd.pdf`.

[6] Baumann, R.—Plattner, C.: Honeypots (White Paper). Technical Report, Swiss Federal Institute of Technology, Zürich, February 2002.

[7] Chandran, R.—Pakala, S.: Simulating Network with Honeyd. Technical Paper. December 2003.

[8] Fanfara, P.—Chovanec, M.: Influence of Sophisticated Hybrid Honeypot on Efficiency of Intrusion Detection System Architecture in Distributed Computer Systems. Acta Informatica Pragensia, Vol. 2, 2013, No. 1, pp. 39–56 (in Slovak).

[9] Fanfara, P.—Dufala, M.—Chovancová, E.: Usage of Proposed Autonomous Hybrid Honeypot for Distributed Heterogeneous Computer Systems in Education Process. 11th IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA 2013), Stary Smokovec, October 24–25, 2013, pp. 383–388, doi: 10.1109/iceta.2013.6674409.

[10] Fanfara, P.—Dufala, M.—Danková, E.: Security of Distributed Computer Systems Based on Intrusion Detection System and Advanced Technology of Hybrid Honeypots. International Conference on Applied Electrical Engineering and Informatics (AEI 2012), Germany, August 26–September 2, 2012, pp. 155–160.

[11] Khosravifar, B.—Bentahar, J.: Honeypot in Network Security: A Survey. Advanced Information Networking and Applications (AINA 2008), March 25–28, 2008.

[12] Kyaw, K. L.: Hybrid Honeypot System for Network Security. World Academy of Science, Engineering and Technology, Vol. 2, 2008, No. 12, pp. 232–236.

[13] Marchese, M.—Surlinelli, R.—Zappatore, S.: Monitoring Unauthorized Internet Accesses Through a 'Honeypot' System. Communication Systems, Vol. 24, 2011, No. 1, pp. 75–93, doi: 10.1002/dac.1141.

[14] Pekár, A.—Chovancová, E.—Fanfara, P.—Trelová, J.: Issues in the Passive Approach of Network Traffic Monitoring. IEEE 17th International Conference on Intelligent Engineering Systems (INES 2013), Costa Rica, June 19–21, 2013, pp. 327–332.

[15] Sutton Jr., R. E.: How to Build and Use a Honeypot. Section 1. Technical Report. DTEC 6873, Boston, 2008.

[16] Thuraisingham, B.: Data Mining for Malicious Code Detection and Security Applications. Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT '09), Vol. 02, September 15–18, 2009, pp. 6–7, doi: 10.1109/WI-IAT.2009.379.

[17] Watson, D.—Riden, J.: The Honeynet Project: Data Collection Tools, Infrastructure, Archives and Analysis. Information Security Threats Data Collection and Sharing (WISTDCS '08), April 21–22, 2008, doi: 10.1109/wistdcs.2008.11.

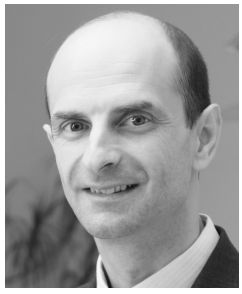
Eva Chovancová graduated (Ing.) at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice in 2009. She defended her Ph.D. thesis in the field of computers and computer systems in 2012; her thesis title was "Specialized Processor for Computing Acceleration in the Field of Computer Vision". Since 2012 she has been working as Assistant Professor at the Department of Computers and Informatics. Her scientific research is focused on the multicore computer architectures.

Norbert Ádám graduated (M.Sc.) with distinction at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice in 2003. He defended his Ph.D. thesis in the field of computers and computer systems in 2007; his thesis title was "Contribution to the Simulation of Feed-Forward Neural Networks on Parallel Computer Architectures". Since 2006, he has been working as a lecturer at the Department of Computers and Informatics. Since 2008, he is the Head of the Computer Architectures and Security Laboratory at the Department of Computers and Informatics. His scientific research is focused on parallel computers architectures.

Anton Baláž received his Master's degree in informatics in 2004 from the Faculty of Electrical Engineering and Informatics, Technical University of Košice. In 2008 he received Ph.D. in the area of computer security. Since 2007 he is working as Assistant Professor at the Technical University of Košice.

Emília Pietriková is Assistant Professor at the Department of Computers and Informatics, Technical University of Košice, Slovakia. She received her M.Sc. in 2010 and her Ph.D. in 2014 in informatics from Technical University of Košice. In 2010 she spent one month at the Department of Telematics at Norwegian University of Science and Technology, Norway. In 2011 she spent one semester at the Department of Computer Architecture at University of Málaga, Spain. The subject of her research is abstraction and generation in programming languages, and quality of education.

Peter Feciľak graduated (M.Sc.) at Department of Computers and Informatics at Faculty of Electrical Engineering and Informatics, Technical University of Košice in 2006. In 2009 he finished his Ph.D. studies at the same department with the focus on optimization of computer networks. Currently, he is working as employee of DCI, FEI, Technical University of Košice. His current teaching and research interests are computer networks, network monitoring, quality of services and smart energy systems.



Slavomír Šimoňák received his M.Sc. degree in computer science in 1998 and the Ph.D. degree in computer tools and systems in 2004, both from the Technical University of Košice, Slovakia. He is currently Assistant Professor at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice, Slovakia. His research interests include formal methods integration and application, communication protocols, algorithms, and data structures.

Martin Chovanec received his engineering degree in informatics in 2005 from Faculty of Electrical Engineering and Informatics, Technical University of Košice. In 2008 he received his Ph.D. degree at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice and his scientific research was focused on network security and encryption algorithms. Currently, he is Director of the Institute of Computer Technology of the Technical University of Košice.