Sep. 30
International Journal of advanced studies in Computer Science and Engineering
IJASCSE Volume 4, Issue 9, 2015

# Data control in virtual honeynets based on operating system-level virtualization

Pavol Sokol
Institute of Computer Science
Faculty of Science, UPJŠ in Kosice
Kosice, Slovakia

Ján Host
Dcore Slovakia, s.r.o.
Kosice, Slovakia

Michal Vaško
Dcore Slovakia, s.r.o.
Kosice, Slovakia

***Abstract*— A virtual honeynet plays a very important role in modern network security. Data control within should be able to ensure the honeypots cannot be used to attack other systems and computer networks. Also, the data control should be invisible for an attacker. In this paper we propose such framework. This framework is based on a set of legal and technical requirements and it represents whole new approach to data control.**

***Keywords - Honeypot; honeynet; virtual honeynet; data control; decision module***

## I. INTRODUCTION

To protect and secure communication between network services administrators are using limited set of tools. In past years these tools are less effective than they used to be because of advanced security threats. For that reason it is crucial for responsible individuals to improve methods that are used to protect networks from attackers. This is where the honeypot and the honeynet principle steps in. It represents a modern approach, which can defend given systems more adequately.

One definition of honeypot states that **a honeypot** is a "security resource whose value lies in being probed, attacked, or compromised" [1]. Primarily honeypots are categorized by the level of interaction, purpose, role and deployment. We in this paper consider only categorization by level of interaction, namely low-level interaction honeypot and high-level interaction honeypot. The **low-interaction honeypot** uses software emulation of network services and operating systems on the host operating system to detect an attacker. A **high-interaction honeypot** works differently; it permits the attacker all services on the given operating system and platform, nothing is restricted.

In the approach discussed in the following text, only the UNIX-like operating system can be used. According to W3Tech´s survey on the usage of operating systems for hosting websites [2] the UNIX-like operating system is used by 67.7\% of all the websites running on known operating systems.

**A honeynet** is a high-involvement honeypot. It shares the same problems and takes the same risks as are characteristic for many networks in different organizations today. It is "not a single system but a network of multiple systems" [3]. The primary value of honeynet lies in analysis of data on existing threats and zero-day knowledge threats. There are some modifications made to the honeypots to the extent of the attacker not knowing it. This gives him/her a full range of operating systems, applications and functionality within them to use [4].

There are several definitions on what is virtual honeynet. One of them states **virtual honeynet** can be defined as "a complete honeynet, running on a single computer in a virtual environment" [4]. A virtual honeynet can also be defined as "a technology that virtually implements many different operating systems in one hardware computer, and hence instead of having a honeynet of different physically separate honeypots, all the honeypots will be virtually set in one machine and still appear to the attacker as different separate machines" [5]. Virtual honeynets combine all the elements of a honeynet into a single physical system. Not only are all of the three requirements of data control, data capture, and data collection met, but also the actual honeypots themselves run on the single system [6].

There are different approaches to virtualization, namely full virtualization, paravirtualization and **operating system level virtualization**. In this paper we focus on the last mentioned approach. The kernel of an operating system allows multiple isolated userspace instances (containers). The advantage of this method lays mainly in performance due to little or even no overhead. Its disadvantage is kernel sharing of the host and guest. An operating system based on the Windows kernel cannot be run in a host operating system based on the UNIX kernel. Also, if the kernel crashes or is compromised, all containers crash or are compromised as well. In the design of the proposed system we use the implementation of the **Linux container (LXC)** [7]. LXC is a lightweight virtual system mechanism, which builds up from chroot system call in order to create a full-featured, reliable and secure mechanism for the separation of the processes.

A successful deployment of a virtual honeynet is a successful deployment of its architecture. There are some core elements of the virtual honeynet´s architecture [3].

- **Data control** is the first requirement whose purpose is to control and contain the activity of the attacker.
- **Data capture** monitors and logs all of the attacker's activities within the honeynet.

- **Data collection** – in case when the organization has more than one honeynet, all data has to be captured and stored in one central location.
- **Data analysis** is an ability to analyse the data collected from the honeynet.

The single most important purpose of data control is deploying a secure and safe honeynet, so that the attacker cannot compromise other, production networks. This is the most important function and it always has to be given the highest priority when implementing a honeynet [4]. Data control cannot prevent all attacks, but it tries to mitigate the risk of the honeypot being used. There are many data control techniques, such as counting the outgoing connections of the honeypot. Honeynet and data control within should be configurable remotely by a skilled administrator at any given time, so if a problem occurs, he or she can intervene immediately. The honeynet also has to have an automatic alerting system.

Abusing the honeynet is the main problem of deploying one. For example, if the attacker takes over the honeypot, he or she may attempt to launch exploits against a no-honeypot´s system (e.g. a web server). After several successful attempts, all further activity including any exploits, would be blocked. In such a case the attack is not carried out. Therefore, the concept of data control is an essential issue.

The first contribution of this paper lies in the proposed data control in a virtual honeynet. The concept of data control is not new, but our design is based on a set of both, technical and legal requirements. Joshi [6] outlines **technical requirements** for deployment and usage of data control in honeynets. Based on his paper we have outlined how the proposed system takes into account the eight requirements that data control needs to function properly and to reduce the risk to the minimum. Sokol [8] discusses the liability of honeynets´ administrators and outlines the honeynets´ data control, which meets the requirements of the EU law (**legal requirements**). According to him, the data control should contain five components including firewall, with restrict white list, dynamic connection redirection mechanism, emulated private virtual network, honeypots and control center. This control allows trained administrator of honeynet to monitor connections and quickly respond to incidents. The proposed system includes all these parts.

The second contribution is the fact that the proposed data control takes into account all types of honeypots according to their interaction.

This paper is organized as follows: In Section II, the paper discusses the related works in the field of data control. Section III focuses on proposed system, its design and discusses the specific parts of this system and shows how the proposed system meets technical and legal requirements. Section IV focuses on the decision algorithm, which is a sequence of control steps. Section V outlines implementation of four modules of proposed system. The last section contains conclusions and the authors´ vision of the future research.

## II. RELATED WORKS

The idea of data control is not novel. There are a lot of papers describing how to control honeypots and honeynets so as to prevent their abuse. Joshi [6] proposes a design of the deployment of a virtual honeynet with a unique data control solution. A number of researches close to this area have been made, we will mention just some of them. Tian-hua [9] deals with the general idea of the honeynet, its deployment and the necessity of using some kind of virtualization technology in order to solve the low hardware utilization, complexity of configuration and difficulty of the management of such a honeynet. They consult the proposal of Pakistan Honeynet Project. Sharma [10] discusses the need to use the Honeywall to secure the honeynet. The Honeywall is a powerful tool for capturing attacks in honeynet environment. The information gathered from the Honeywall can be used in designing an efficient model against malware in computer networks.

Yan [11] presents a new User-Mode Linux based virtual honeynet architecture. The honeypot controller is the new virtual honeynet component that assists in data control. Zhang [12] suggested a honeynet using XEN virtualization technology. The virtual honeynet system includes dynamic resource allocation, data control, data capture, data analysis. It is lightweight, but it has a high performance, which has been verified with extensive experiments. Mumtaz [13] introduces a mechanism of intrusion detection based on high interaction honeypots to assist efficiently in gathering information concerning attackers attacking an enterprise network via the Internet. The proposed system consists of five constituent modules: Honeypots, Sniffing, Tracing, Alert and Control.

All of the above mentioned virtual honeynets use other virtualization technologies than the operating-system level virtualization. Also, none of them have used advanced systems to control the data or data control with remote control and double data control, just a simple control or blocking of connections. Our proposed data control focuses on operating system level virtualization and much more sophisticated data control than ever before.

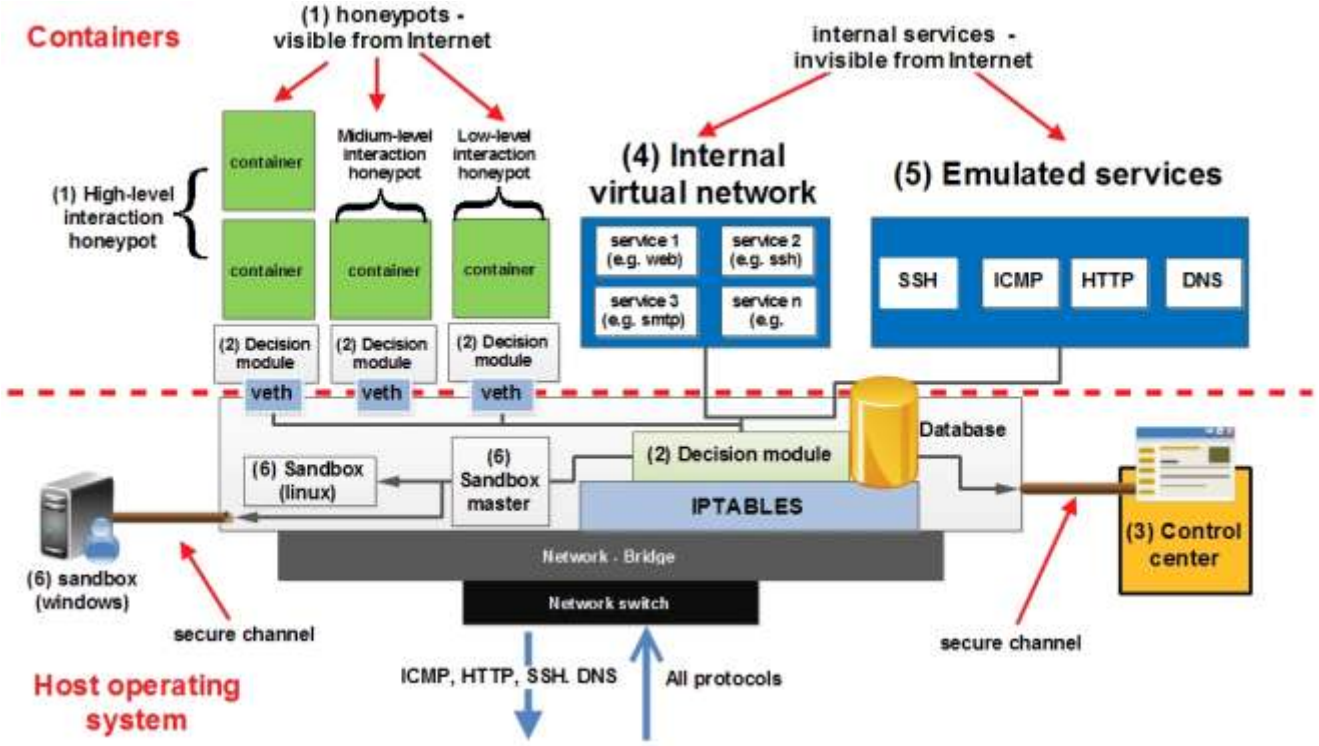


Figure 1. Proposed data control in virtual honeynet.

## III. PROPOSED DATA CONTROL IN VIRTUAL HONEYNET

As we have mentioned before, the main contribution of this paper is the proposed data control in a virtual honeynet, especially in a virtual honeynet based on operating system-level virtualization. The architecture of the proposed distributed system (Fig. 1) consists of six basic components, namely honeypots, decision module, control center, emulated services, internal virtual network and sandbox.

Connections to the virtual honeynet are allowed (all network protocols). On the other hand, connections from the honeypots are restricted. Based on decision modules with decision algorithm some connections are allowed to go out of the virtual honeynet (only ICMP, HTTP, SSH and DNS protocols), some connections are redirected to the emulated services. Other connections are dropped. A decision algorithm determines whether a connection is to be allowed, redirected or dropped. Since connections from the virtual honeynet are restricted, the proposed system contains an internal virtual network. Connections to the internal virtual network are allowed in both directions. The control center is one of the essential parts of the proposed system. It allows a trained administrator of the virtual honeynet to monitor connections and quickly respond to incidents.

The proposed system is based on technical requirements [6] and legal (EU law) requirements [8]. In the following subsections we focus on individual parts of the proposed data control in more detail and discuss its technical requirements.

### A. *Safety zones*

Before discussing individual parts of the proposed system and decision algorithm, it is necessary to outline the basic scheme of security zones. A zone can be seen as a network or virtual network with its own range of IP addresses connected to one virtual interface, such as virtual network (vlan) or virtual ethernet device (veth). The idea of dividing the honeynet into zones comes from the need to divide individual networks, in which groups of computers we want to filter a communication are located. The virtual honeynet is divided into several zones, which are shown in Fig. 2.

The zone of the Internet is the network out of the virtual honeynet. The zone of the honeypots is the zone of the virtual network of honeypots. The communication to the zone of honeypots is allowed, but every packet from the zone of honeypots is checked. The zone of the virtual network and the zone of the emulated services are zones, where the ingoing connection from the zone of honeypots is allowed. Connections from other zones are prohibited. The zone of the device is the special zone representing the virtual network (openvpn, ciphered tunnel, secondary connection), from which it will be possible to connect to the host operating system.

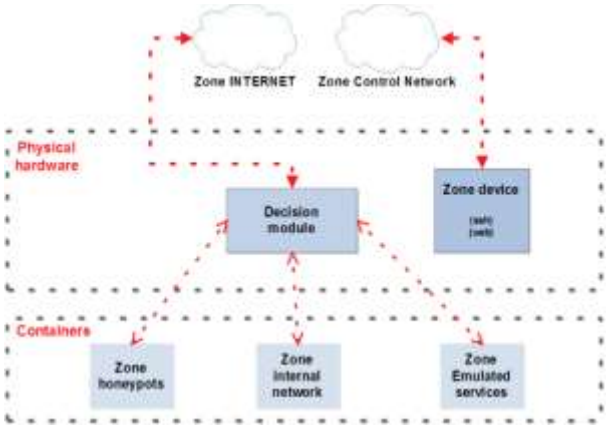


Figure 2. Zones in virtual honeynet.

### B. *Honeypots*

Honeypots form a substantial part of the virtual honeynet and they are partly included in the proposed system. Honeypots are shown in Fig. 1 (number 1). In the proposed system, the authors consider all types of honeypots according to interaction. Each low-interaction and medium-interaction honeypot consists of one virtual machine. The decision module is part of the virtual machine.The data control is very important in high-interaction honeypots. This type of honeypot consists of two virtual machines. The decision module is a part of one of them.

### C. *Decision modules*

**Decision modules** are the main part of data collection and they are shown in Fig. 1 (number 2). The proposed system has two types of decision modules. The first module is the main module for the virtual honeynet and is closely connected to the firewall. Second decision modules are placed in virtual machines. Each connection that goes through the firewall is controlled by the decision module. These modules deal with each connection according to a decision algorithm showed in the following section.

Usage of this part of the proposed system meets the following technical requirements: the automated control, control layers, control connections. Since decision modules are placed in the firewall and in the virtual honeypots, the technical requirement of at least two layers of data control to protect against failure is satisfied. All control mechanisms are located on the host operating system. Therefore, the attacker cannot access them. Some parts of the decision modules are located in kernelspace.

### D. *Control center*

**Data control center** is a remote part of the proposed system. It is shown in Fig. 1 (number 3). It allows a trained administrator to control decision modules at any time. Connections between the control center and decision modules are via a secure channel (e.g. s-tunnel). It includes a web interface. This control center allows to:

- **start, stop and freeze** honeypots – in case that decision modules allow harm connections,
- **show statistics** – statistics on the number of connections and the amount of data in the connections,
- **show alerts** – if the decision module drops or redirects the connection and
- **show and modify** the rules for decision centers.

Using the advantages of the operating system level virtualization, a trained administrator is able to perform the above mentioned activities without the knowledge of operating system level virtualization. It is due to the fact that the host operating system is able to access the honeypots and control all resources. Using this part of the proposed system satisfies the following technical requirements: manual data control, configurable at any time, remote administration and alerting.

### E. *Internal virtual network and emulated services}*

**The internal virtual network** consists of several virtual machines bridged to an internal physical interface. This network is shown in Fig. 1 (number 4) and it is visible only from the honeypots and the internal virtual network. The motivation for this network is based on the fact that the outgoing connections from the honeypots are restricted. The internal virtual network is visible only for the attackers and its purpose is to study the attacker's behavior, since it mimics a regular school computer study room.

**Emulated services** are different from the internal virtual network in terms of deployment and creation, but they are very similar in usage. In the proposed system the Honeyd are used to implement several services (ICMP, SSH, DNS, HTTP).

### F. *Sandbox*

**Cuckoo sandbox** is an optional part of the proposed data control [14]. It is an open-source software for automating the analysis of suspicious files. In order to do so, it makes use of custom components that monitor the behavior of the malicious processes while running in an isolated environment. Inputs for the sandbox are files, which were modified in honeypots within the internal virtual network. Sandbox makes hash and the hash is subsequently compared with an external database (e.g. virustotal). If the comparison is successful, the administrator is informed. Otherwise, the files are sent to the internal sandbox (a virtual machine – Linux operating system) or external sandbox (a physical machine out of the virtual honeynet – Windows operating system).

## IV. DECISION ALGORITHM

The functioning of the decision module is based on a **decision algorithm**, which handles each packet and decides what happens with it. Its scheme is shown in Fig. 3. There are four final states:

- permit the packet to/out of the honeynet (**state ACCEPT**);
- redirect the packet to the internal virtual network (**state INSIDE**);

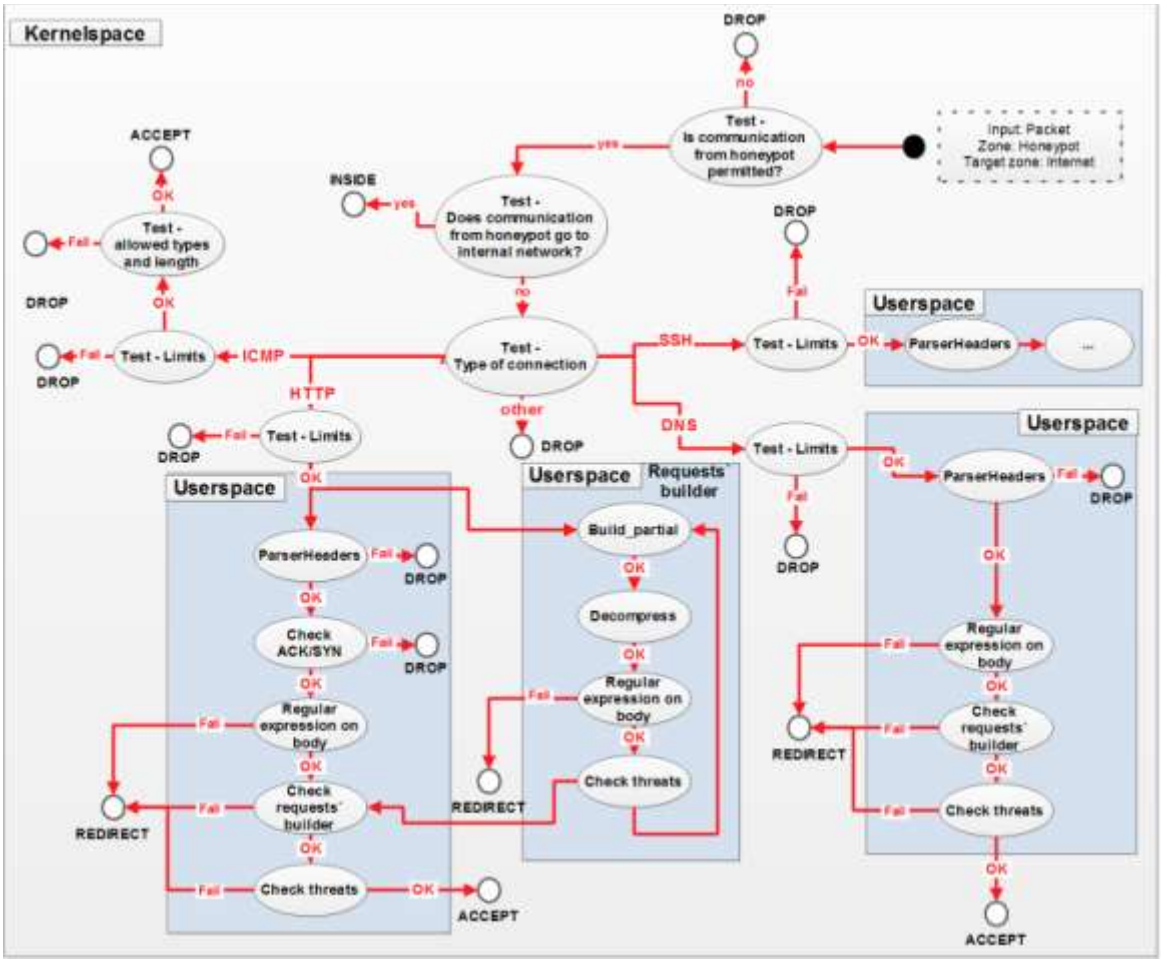


Figure 3. Decision algorithm.

- redirect the packet to the emulated service (**state REDIRECT**);
- drop the packet (**state DROP**).

As we have mentioned before, the proposed framework controls the outgoing communication, not the incoming one. It is due to the nature of honeynets. From this point of view each controlled packet has its source zone: the honeypot and the destination zone: the internet.

At first, the decision algorithm tests whether the packet is allowed to go outside of the honeynet, including the internal network (check destination IP address). The next step is testing the packet whether its destination zone is the internal network. If the control passes, it is redirected there. Otherwise, the decision algorithm tests the packet for its type and handles it according to the type. Our proposed system is able to handle the following network protocols: http (80/TCP), ICMP, SSH (22/TCP) and DNS (53/UDP). In other cases, the packet is dropped.

In the following subsections, we outline the concept of the decision algorithm for each network protocol separately.

In case of SSH protocol it is needed to emphasize some notes. The control of SSH connections is similar to the control of HTTP connection with several differences. One of the differences is the need to obtain the private key for SSH session. As we have mentioned before, the proposed framework is based on operating-system level virtualization. Therefore, the host operating system has the access to the memory where the keys of SSH connections are stored.

### *A. Control of ICMP connections*

At first, the decision algorithm checks the limits of ICMP connections. The default value is 15 packets per second (inspired by the default value, but configurable as needed by setting the amount of packets allowed per time frame). The next step is the control of the type of ICMP packet and its length. The allowed types of packets are 0 (echo request) and 8 (echo reply). This is also configurable, if needed. If the packet passes both tests, communication is accepted and it is allowed to go to the Internet. If that is not the case, communication is dropped after either tests.

*B. Control of HTTP and DNS connections*

The decision algorithm in case of HTTP connection and in case of DNS connection is similar. At first, the decision algorithm checks the limits of HTTP or DNS connection. The default value is 50 connections per second (in case of HTTP) and 15 connections per second (in case of DNS).

Subsequently, it makes a copy of the packet and sends it to the requests´ builder (e.g. Hadoop). This builder is only for HTTP´s connections, not DNS´s connections. The purpose of the requests´ builder is to create a part of HTTP´s connection. In this builder, the second step is to decompress the payload if a known kind of compression was detected, such as gzip. The next step is to check the regular expression on the body of a part of HTTP´s connection. If every test passes, the final step in the Requests´ builder is the check of threat usage by the external frameworks (e.g. virustotal).

The first step of the main control of HTTP and DNS packets is the parsing of the header. This step checks whether the packet is HTTP or DNS. After that the decision algorithm checks ACK and SYN flags (only in control of HTTP packet) and checks next packet (e.g. correct order, check of overlaying of datagram). The next optional step is the usage of a regular expression over the body of packet. In the proposed framework there is a limited set of regular expressions. It might be the regular expression for detection of network paths, such as \verb/^(?:[a-zA-Z]\:|\\\\[\w\.]+\\[\w.\$]+)\\(?: [\w]+\\) *\/ \newline \verb/w([\w.])+\$/,

or similar. The next step is the check of the output of requests´ builder. If the last two steps fail, the packet is redirected. Otherwise, the decision algorithm checks threats. In this step the proposed system uses an external framework to test threats (e.g. virustotal). If this step passes, the packet is accepted. In the opposite case, the packet is redirected.

## V. IMPLEMENTATIONS

For the purpose of implementation, authors have used computer system with following hardware configuration - HP Proliant Gen 4 2U, 2x Intel (R) Xeon (TM) CPU 3.00GHz 6GB RAM. Authors have used Debian 7, x64 based operating system with LXC installation. The proposed data control consists of modules. The implementation of modules was designed using a standard design pattern MVC (Model-View-Controller). At present, the following modules have been implemented:

- decision module (Java);
- database module (MySQL) and
- web module (Java + Rest API) with control center (Python, JavaScript, HTML).

*A. Database module*

As a storage authors decided to use a relational database MySQL. Captured packets are stored in the table called **Packets** in binary form. This form provides us with the opportunity to recover the original data packet to be used by libraries working with packets in Java language. Table **Packets_metadata** contains basic information about the packet, such as source port, source IP address and destination port and destination IP address, protocol type etc.

The table **packet_validator** stores the results of packet validations from decision module. These results are displayed in the web module. The table **regular expressions** contain regular expressions used in the validation process of packet´s body. The table **system variables**contains global variables used to configure the system via web module.

Above mentioned scheme is operated by database module. This module was designed the way so it can be replaced by any other database system, such as an object, graph or other. It is due to our implementation of interfaces (packetDao, expressionDao, honeypotDao). This module was implemented in JAVA 8. Authors have used **Spring Framework data access integration JDBC** version 4.0 to establish database connections and to process all database requests.

*B. Decision module*

**The decision module** is used to validate and redirect packets towards the honeynets. Two protocols were implemented, namely HTTP and ICMP protocol. This module is of the producer-consumer type. Using the library JPcap [15] and jNetPcap [16] the producer captures packets and stores them in a queue(ConcurrentLinkedQueue). Subsequently the packets are stored in the database using a database module.

The decision module stores all communications to the database for further analysis. It is possible to change system to offline mode (**Offline-Packets-Input-Stream).** The consumer processes the data stored in the queue. The first packet from queue is periodically chosen and sent to PacketValidatorFactory where the type of protocol is examined. Subsequently PacketValidatorFactory determines whether it is a HTTP or ICMP protocol (or any else). Packets satisfying the above conditions are subsequently processed pertaining validator (HttpValidator, IcmpValidator). Using PacketValidatorFactory authors have achieved that it is easy to add additional validation protocol or modify existing implementation. Proposed data control allows launching new producers and consumers on the origin server thus we can separately capture all network interfaces and protocols (new producer). If the processing is slow, it is possible to speed it up by addition of one or more consumers.

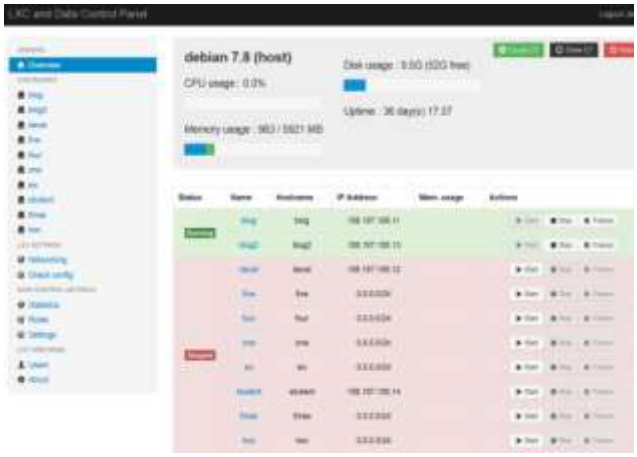


Figure 4. Control center.

### C. *Web module with control center}*

The **web module** is used for communication between the database module and user interface. It processes requests from the user interface. Using the own REST API it can be replaced with another interface (e.g. web-based or mobile interface). For that communication data in JSON (JavaScript object notation) format are used. This module was implemented in Java 8 and Spring Framework.

Honeypots can be added on the fly using the **control center.** It is possible to change their configuration, control their status and modify it. Also there are the statistics and graphs related to data control and its possible settings. Web module with control center is shown in Fig. 4.

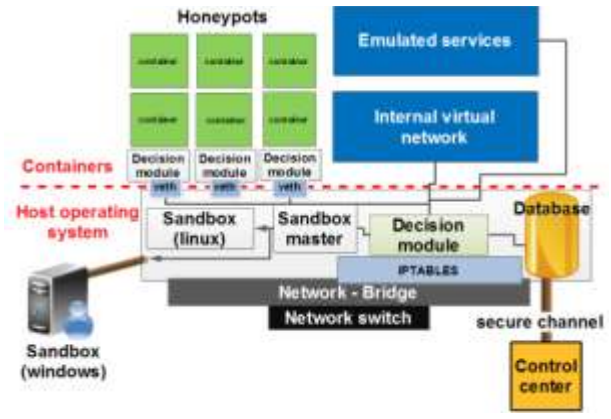


Figure 5. Proposed framework.

## VI. Conclusion

This paper outlines a proposal of a new data control in virtual honeynets. This framework (Fig. 5) consists of six parts from which the decision module is the most important. In this proposal we focus on the low-interaction and the high-interaction honeypots. Within the proposed framework, we have designed a new approach to data control´s decision, which can be used in research, but also in production systems. It can be modified, upgraded an installed to personal and specific use case.

In the future we will focus on implementation of the some parts of proposed system, especially internal virtual network and sandbox. The deployment of the SSH branch in decision module is a challenge for the future research.

## Acknowledgment

We would like to thank our colleagues from the Czech chapter of The Honeynet Project for their comments and valuable input. This paper is funded by the Slovak Grant Agency for Science (VEGA) grant under contract No. 1/0142/15, VVGS project under contract No VVGS-PF-2015-472 and Slovak APVV project under contract No. APVV-14-0598.

## References

[1] Fabien, P., Dacier, M., and Debar, H.: „White paper: honeypot, honeynet, honeytoken: terminological issues, in Rapport technique“, EURECOM 1275, 2003.

[2] W3Techs „Web technology survey, usage of operating systems for web-sites“, http://w3techs.com/technologies/overview/operating_ system/all (online)

[3] Spitzner, L. „Honeypots: Catching the insider threat“, Proceedings of 19th Annual IEEE Computer Security Applications Conference, pp. 170-179, 2003.

[4] The Honeynet project, „Know Your Enemy: Learning about Security Threats (2nd Edition)“, Addison-Wesley Professional, 2004.

[5] Mairh, A. et al., „Honeypot in network security: a survey“, Proceedings of the 2011 International Conference on Communication, Computing and Security, pp. 600-605, 2011.

[6] Joshi, R. C., and Sardana A., „Honeypots: A New Paradigm to Information Security“, CRC Press, 2011.

[7] LXC project, available at https://linuxcontainers.org/ (online)

[8] Sokol P., Andrejko M., „Deploying honeypots and honeynets: issues of liability, in Computer Networks“, Springer International Publishing, 2015.

[9] Shi-wei, Y., Xiu-shuang, M., and Wei-dong, W. A. N. G., „Core Functions Analysis and Example Deployment of Virtual Honeynet“, Computer Science, 3, 2012.

[10] Sharma, N., and Sran, S. S, „Detection of threats in Honeynet using Honey-wall“, International Journal on Computer Science and Engineering, vol. 3, pp.3332-3336, 2011.

[11] Yan, L. K, „Virtual honeynets revisited“, Proceedings from the 6th Annu-al IEEE SMC Information Assurance Workshop, pp. 232-239, 2005.

[12] Zhang, W., He, H., and Kim, T, „Xen-based virtual honeypot system for smart Device“, Multimedia Tools and Applications, pp. 1-18, 2013.

[13] AL-Mukhtar, M. M. A., and Kasim, B. W, „A honeynet framework to pro-mote enterprise network security“, International journal of computer engineering and technology, vol.4, pp. 404-413, 2013.

[14] Cuckoo project, available at http://www.cuckoosandbox.org/ (online)

[15] JPcap library, available at http://jpcap.sourceforge.net/ (online)

[16] jNetPcap library, available at http://jnetpcap.com/ (online)