Data Collection and Data Analysis in Honeypots and Honeynets

Pavol Sokol, Patrik Pekarčík, Tomáš Bajtoš

pavol.sokol@upjs.sk, patrik.pekarcik@upjs.sk, tomas.bajtos@student.upjs.sk

Institute of Computer Science, Faculty of Science
Pavol Jozef Šafárik University in Košice
Košice, Slovakia

Abstract

Honeypots and honeynets are unconventional security tools to study techniques, methods, tools, and goals of attackers. Therefore, data analysis is an important part of honeypots and honeynets. In paper we focus on analysis of data collected from different honeypots and honeynets. We discuss framework to analyse honeypots' and honeynets' data. Also, we outline a secure way to transfer collected data from honeypots to the analysis itself. At last, we propose a framework for analysis of attack based on data collected by honeypots and honeynets.

Keywords: honeypot, honeynet, data analysis, data collection, event

1 Introduction

The traditional ways of defence (e.g. firewalls, IDS, IPS) are becoming less and less effective. It is due to the changing nature of the attackers' behaviour, methods, and tools. Therefore the attackers are several steps ahead of defensive mechanisms. From this perspective, we need to find new approaches to protect information and infrastructure of the organizations. One of the effective approaches to protect them is concept of honeypots and honeynets.

A **honeypot** is "a computing resource, whose value is in being attacked" [1]. Lance Spitzner defines **honeypots** as "an information system resource whose value lies in unauthorized or illicit use of that resource" [2]. Honeypots are a very useful tool for learning about tools, procedures, targets, and methods of attackers.

For the purpose of this paper, we classify the honeypots according to their level of interaction and role. The first classification is based on the **role** of honeypot. According to this classification, honeypots are divided in server-side honeypots and client-side honeypots. **Server-side honeypots** are useful in detecting new exploits, collecting malware, and enriching research of the threat analysis (e.g. Conpot [3]).

On the other hand, honeypots for client-side attacks are called **client-side** (e.g. Thug [4]). The prime motive of client-side honeypots is to identify and detect malicious activities across the Internet [5].

The second classification is **based on the level of interaction**. The level of interaction can be defined as the range of possibilities that a honeypot allows an attacker to have. **The low-interaction honeypots** detect attackers using software emulation of characteristics of a particular operating system and network services on the host operating system. Advantage of this approach is in a better control over attacker's activities, since attacker is limited to software running on a host operating system. On the other hand, disadvantageous about this approach is the fact that the low-interaction honeypot emulates service, or couple of services, but it does not emulate complete operating system. Examples of this type of honeypots are Dionaea [6], HoneyD [7].

Honeypots that offer attackers more ability to interact than do the low-interaction honeypots, but less functionality than high-interaction solutions, are called **medium-interaction** honeypots. They can „expect certain activity and are designed to give certain responses beyond what a low-interaction honeypot would give" [1]. Examples of this type of honeypot is Kippo [8].

In order to get more information about attackers, their methods and attacks, we use a complete operating system with all services. This type of honeypot is called **high-interaction honeypot.** Main aim of this type of honeypot is to provide the attacker access to a real operating system [9]. Examples of this type of honeypots are HonSSH [10], Sebek [11].

The concept of honeypots is extended by **honeynets**. Honeynet can be defined as "a highly controlled network of honeypots" [12]. At present, complete honeynet, running on a single computer in virtual environment is used [12]. This type of honeynet is defined as a **virtual honeynet**.

To successfully deploy a honeynet, we must correctly deploy the honeynet architecture. There is "no single rule on how one should deploy this architecture" [13]. There are three core elements of the honeynet architecture that define honeynet architecture [2,12]:

- **Data capture** - monitors and logs all activities of attacker within the honeynet.
- **Data control** - purpose of which is to control and contain the activity of attacker.
- **Data collection** - all data are captured and stored in one central location.

The first two core functions are the most important, and they are applicable to every honeynet deployment. The last core function - data collection - is applied by organization in case that organization has the multiple honeynets in distributed environments.

Some authors [14,15] add **data analysis** to the above-mentioned core elements. Data analysis is an ability of honeynet to analyse the data, which is being collected from it. Data analysis is used for "understanding, analysing, and tracking the captured probes, attacks or some other malicious activities" [1]. Example of this core element is combination of security devices, such as firewall (IPtables), intrusion prevention system (Cisco IPS) and intrusion detection system (Snort, Suricata), where this security devices can analyse the network traffic in detail, and return the result of analysis in a visible way. In this paper we focus on data analysis.

Deployment and usage of honeypots and honeynets brings many benefits, e.g. the possibility of discovering new forms of attacks. On the other hand, usage of honeypots and honeynets brings about some problems. The primary motivation for elaborating this paper is the fact that there are several problems in field of analysis of data. There are a lot of implementations of honeypots that collect data. In most cases they use different format for their storage, or collected data differ. Therefore, it is difficult to analyse the attack from various types of honeypots. Another problem represents a secure way of transferring the collected data from honeypots to the analysis itself.

To formalize the scope of our work, we state **three research questions**:

- **How to collect data for their further analysis securely?**
- **How to analyse data from different types of honeypots?**
- **How to analyse incident according to data collected from honeypots?**

This paper is organized into five sections. In Section II, it is focused on the papers related to data analysis of honeypots and honeynets. Section III is the main part of paper. It provides framework for data analysis. In this section, answers to the first and the second research question are provided. Section IV outlines incident taxonomy, based on honeypots' and honeynets' data. In this section, the third research question is answered. Section V concludes the paper and contains suggestions for future work.

2 Related works

This paper is combination of two research areas, namely the data analysis in honeypots and honeynets and the incident taxonomy. Research in area of the data analysis has resulted in a number of papers, discussing specific topics, concerning design of data analysis and analysis of specific aspect of attack.

Kaur [16] proposes data analysis module in the third generation of honeypots. This module is based on data access method: a relational model based fast path. Sharma [17] focuses on honeywall and how it is used in honeynet environment to trap attackers. One of the modules of honeywall is data analysis module. Another interesting paper [15] focuses on design and implementation of virtual honeynet. Its data analysis can be devided into three levels, such as swatch software, Hflow2, and walleye.

Project **Modern Honey Network** [18] focuses on deploying and managing honeypots and analysis data from honeypots. It allows easy analysis of data from low-interaction and medium-interaction honeypots. Unfortunately, this framework doesn't focus on high-interaction honeypots and issues of secure connection between honeypots and itself.

Thonnard [19] proposes the application for analysing one specific aspect of the honeynet data, i.e. the time series of the attacks. The result of paper is identification of the activities of several worms and botnets in the collected traffic according to time series analysis.

In research **field of incident taxonomy**, there is a very interesting research in [20]. This paper presents the results of a project of Sandia National Laboratories to develop a common language for computer security incidents. Proposed honeypot-incident taxonomy is based on this paper. This paper represents the basis for a large number of papers. Example is Brooks [21], who maps taxonomy of mobile code paradigms to taxonomy of network security vulnerabilities. Another example is report [22]. Authors of this report create mapping of various incident taxonomies to each other and identify some practical deficiencies and omissions in most recent of them – MKII.

3 Proposed framework for data analysis

Purpose of **proposed framework** is analysis of data collected by different types of honeypots. This framework is shown in Fig. 1 and consists of two parts, such as client part and server part. **Client part** is located in honeypots or in honeynets (especially virtual honeynets). Its purpose is to convert data from honeypot or honeynet and transfer them to central database. Client part consists of:

- **client's collectors**
- **secure connection** - the communication between each collector and central database is secured by Stunnel4 network service with mutual PKI authentication. We can find more information about Stunnel in [23].

Client part of proposed system is done in 2 ways:

- Collect data directly from standalone honeypots (Fig.2 left)
- Collect data from virtual honeynet based on OS-virtualization (Fig.2 right)

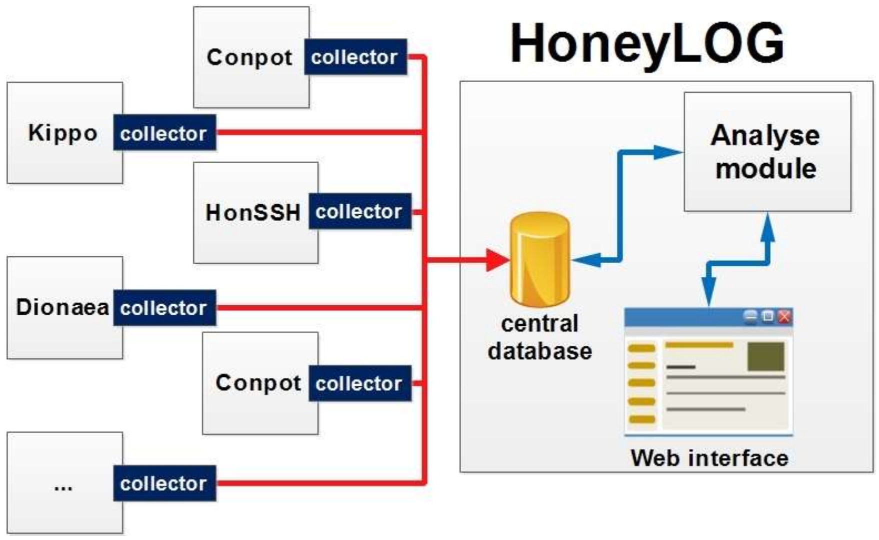
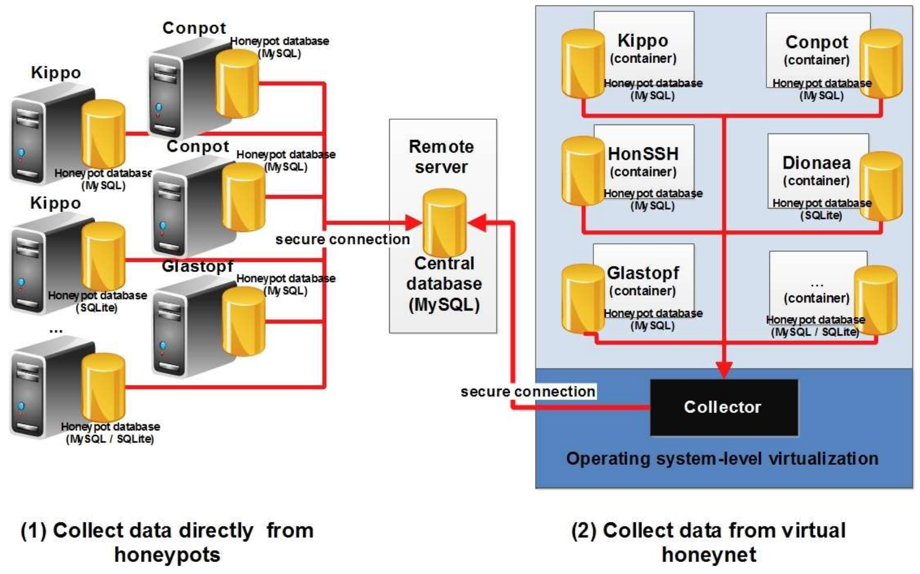


Figure 1: Scheme of proposed framework.

On the other hand, data collected from honeypots or honeynets is stored and analysis in one place – **server part**. This part consists of:

- central database,
- analytical module with web interface.



Conpot
Honeypot database (MySQL)
Kippo
Honeypot database (MySQL)
Conpot
Honeypot database (MySQL)
Kippo
Honeypot database (SQLite)
Glastopf
Honeypot database (MySQL)
...
Honeypot database (MySQL / SQLite)
secure connection
Remote server
Central database (MySQL)
Kippo (container)
Honeypot database (MySQL)
Conpot (container)
Honeypot database (MySQL)
HonSSH (container)
Honeypot database (MySQL)
Dionaea (container)
Honeypot database (SQLite)
Glastopf (container)
Honeypot database (MySQL)
... (container)
Honeypot database (MySQL / SQLite)
secure connection
Collector
Operating system-level virtualization
(1) Collect data directly from honeypots
(2) Collect data from virtual honeynet

Figure 2: Scheme of data collection.

3.1 Collectors in honeypots and virtual honeynets

Data for analysis are collected from honeypots and honeynets by collectors. **Collector** is a simple python script (with external tools – e.g. mysqlbinlog), which is able to process data from the honeypot or honeynet and send it to central database.

There are two options how to collect data by collectors. In first case the collector is placed **directly into honeypot**. This collector accesses the honeypot's database. In this case this collector has information about central database. Therefore, this collector is applicable only in low-interaction or medium-interaction honeypot. High-interaction honeypots are real systems and attacker would gain access to collector.

For elimination of weaknesses of previous collector we use the second option, which is **data collection from virtual honeynet based on operating system-level virtualization**. Advantages of this approach of virtualization are described in [24]. One of them is direct access to containers (virtual machines) without knowing the virtual machines about this access. In this paper we use implementation of operating system-level virtualization – Linux containers (LXC) [25]. Collector is placed in **host operating system**, which is the actual operating system, on which the virtualization takes place. Host operating system has access directly to containers.

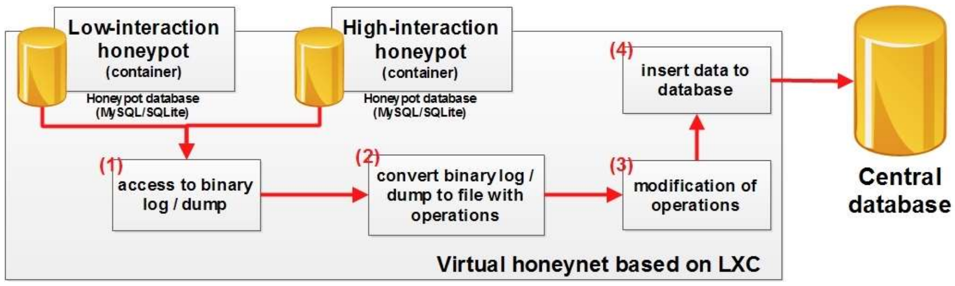

Low-interaction honeypot (container)
Honeypot database (MySQL/SQLite)
High-interaction honeypot (container)
Honeypot database (MySQL/SQLite)
(4) insert data to database
(1) access to binary log / dump
(2) convert binary log / dump to file with operations
(3) modification of operations
Central database
Virtual honeynet based on LXC

Figure 3: Process of data collection in virtual honeynet based on LXC.

Honeypots' implementations use in most cases **MySQL database** [26] and **SQLite database** [27]. Process of data collection in the virtual honeynet based on LXC is shown in Fig. 3 and consists of four steps. In case of MySQL database, host operating system has access to binary logs. These logs are created at every change in the database. In case of SQLlite database the host operating system is able to copy database´s file and read necessary data from it (create dump of database). In the **second step** collector convert binary logs (MySQL) and dump (SQLite) to file with database's operations (inserts or updates). The all insert operations are allowed, but several update operations are allowed. For example, terminate of connection in Kippo, Other database's operations are not allowed (e.g. delete). In the **third step** collector modifies the operations according to regular expressions. In case of MySQL database, column "sensor_id" is added. In case of SQLite database, minor syntax changes are made. These modifications are necessary to prepare data to insert into central database. In the **last step** collector insert data into central database, using secure connection.

3.2 Central database

Central database is the main storage part of proposed system and it is shown in Fig. 4. We stored data from different honeypots into this database in in original format except for one exception. The proposed framework adds column, which identifies the sensor (sensor_id). This change allows collecting data from different honeypots (sensors) without creation of table for each sensor.

Record collected by honeypots or honeynets consists of:

- **timestamp** of each events – 3 types – created, collected and analytical timestamp
- **protocol** – network protocol used to communicate with honeypots
- **IP address** of attacker – location
- **identification of honeypots** – vulnerability, IP address, location
- **information about event**

The main table called Event focuses on events. Columns from this table (id, sensor_id, sensor_table) exactly identify each event (with timestamps) from honeypots.

In database **sensors** are considered as low-interaction honeypots, medium-interaction honeypots and sensors of high-interaction honeypots. High-interaction honeypots are divided into several sensors (e.g. network, CPU, HDD, etc.). This is one of advantages of proposed system's design. This design allows correlating data from each type of honeypots.

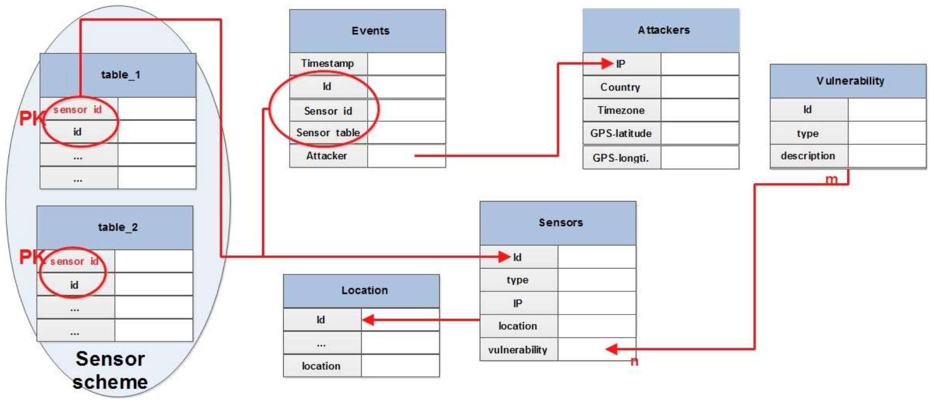


Figure 4: Scheme of central database.

3.3 Analytical module with web interface

The Analysis module provides a standardized interface for analysis in proposed framework. It allows administrators integrate low-interaction and high-interaction honeypots. This module consists of two main parts:

- detail information about event (directly connection with sensors),
- reports from data collected by honeypots and honeynets (reports of all honeypots, report of stand-alone honeypot).

In the central database there is an object, which connects the shared information from each sensor. It provides basic information about events (e.g. timestamp, IP address of attacker). For the purpose of the data analysis we also need extended information about event collected by particular sensors (e.g. input´s commands). The first part of analytical module is interface to get information from particular sensors for each event.

The second and integral part of the analytical module is represented by data reports. From the perspective of the diversity of honeypots there are **local reports** (reports only for one honeypot) and **global reports** (reports for group of honeypots or all honeypots and honeynets). Example of global report is analysis of frequent occurrence passwords from multiple honeypots. In other words, analytical module is a basic interface to generate reports. Interface additionally contains a menu of reports and the form, where we specify the selected report. For example, filter data according to a certain period, service, IP address.

Analytical module consists of **submodules**. It can be defined as a specific implementation of the interfaces analytical module for a specific type of sensor. Each submodule additionally contains a database schema of sensor that is recorded into the central database by migration scripts. Web interface is an indispensable part of proposed framework and its dashboard is shown in Fig. 5. Its purpose is communication with analytical module and creation of reports from collected events.

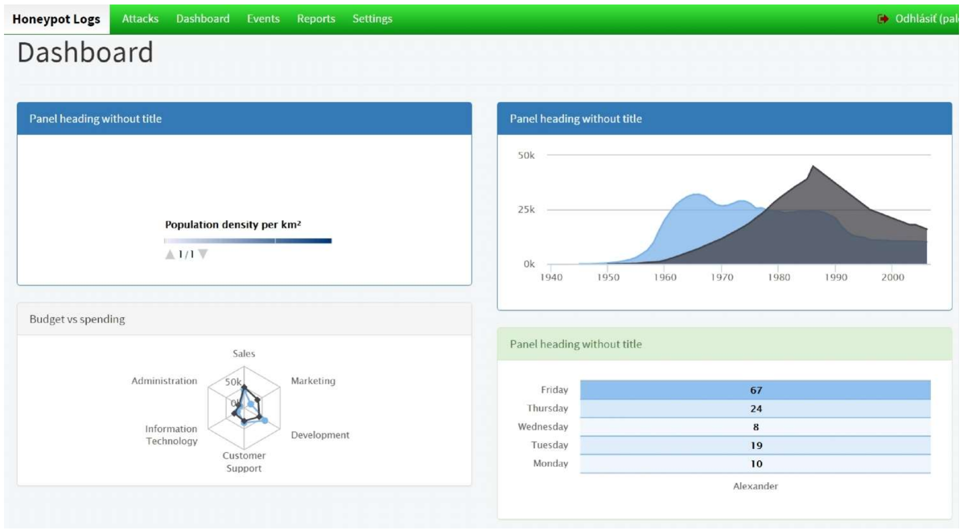


Figure 5: Scheme of web interface.

Communication between web application, analytical module and database (hn subpackages) is shown in Fig. 6. Web application communicates with analytical module and this module communicates with part of the database, where are data from particular honeypot.

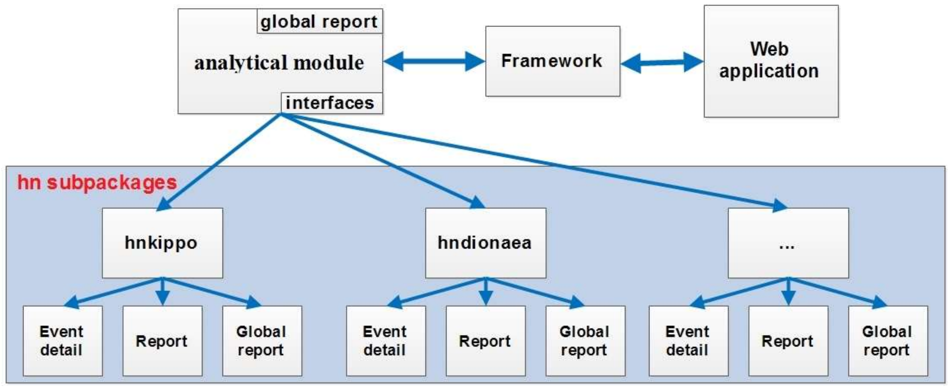


Figure 6: Communication between server-side parts of proposed system.

4 Honeypot-incident taxonomy

Above we propose framework for data analysis. This section outlines incident taxonomy based on data collected from honeypots and honeynets. **Taxonomy** is a classification scheme that partitions a body of knowledge and defines the relationship of the pieces [28].

According to taxonomy developed at Sandia National Laboratories (Sandia's taxonomy) [20], each security incident is a combination of one or more attacks, which use tools to exploit system vulnerabilities and create an unauthorized results. We propose **honeypot-incident taxonomy**, which is based on Sandia's taxonomy and takes into account the specificities of honeypots and honeynets.

On the basis of proposed framework discussed in Section 3 we define **incident in honeypot** as a combination of one or more events at a certain time, perpetrated by attacker, who use tools to exploit vulnerability of honeypots to achieve his objective.

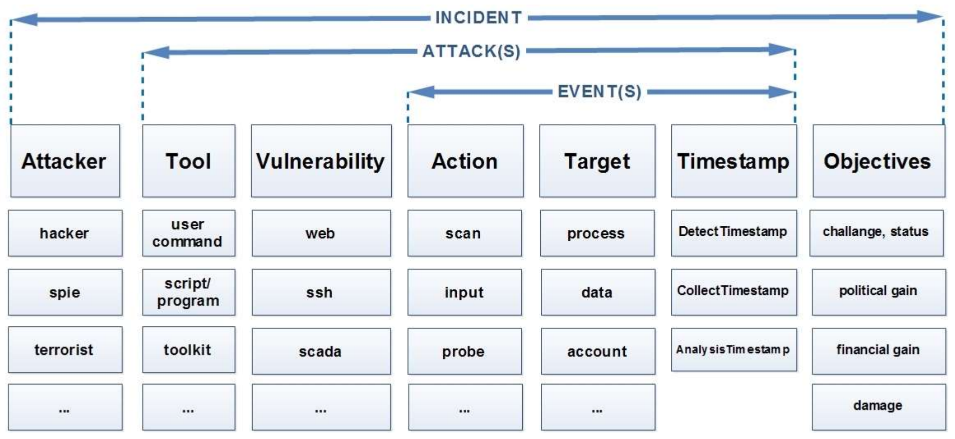


Figure 7: Incident taxonomy based on honeypots' and honeynets' data.

4.1 Incident

Incident – unplanned interruption to a service or a reduction in the quality of a service at a specific time [29]. Howard [20] defines incident as "a group of attacks that can be distinguished from other attacks because of the distinctiveness of the attackers, attacks, objectives, sites, and timing." Taxonomy of incident based on honeypot's and honeynet's data is shown in Fig. 7.

Incident = {incident_number, attacker, attacks, objectives}

Incident number is a number used to identify incident and its information.

Attacker is defined as "an individual who attempts one or more attacks in order to achieve an objective" [20]. Attackers can be classified according to their intentions into six categories:

- **hackers** - attackers who attack honeypots for challenge, status
- **spies** - attackers who attack honeypots for information to be used for political gain.
- **terrorists** - attackers who attack honeypots for information to be used for political gain.
- **corporate raiders** –employees who attack honeypots for financial gain.
- **professional criminals** - attackers who attack honeypots for personal financial gain.
- **vandals** - attackers who attack honeypots to cause damage.

Hence, the attackers may use similar methods, they may achieve a distinctive or similar **objective**. Objective is what attackers want to accomplish. Examples of an objective are, e.g. challenge, status, political gain, financial gain, and damage.

Howard in Sandia's taxonomy states that the logical end of a successful attack is an unauthorized result (e.g. increased access, disclosure or corruption of information etc.). Therefore, honeypot is a trap for attackers and one of the honeypot's roles is prevention of damage, successful attack is an expected result of analysis.

4.2 Attack

Attack is defined as "a series of steps taken by an attacker to achieve an unauthorized result" [20].

Attack = {tool, vulnerability, events}

Tool is defined as "an exploiting a computer or network vulnerability" [20]. In proposed taxonomy we use all tools from Sandia's taxonomy except **physical attack,** which means "a physically stealing or damaging a computer, network, its components, or its supporting systems (such as air conditioning, electric power, etc.)." From the nature of honeypots we don't include this type of attack. In case that attacker has access to honeypot, he/she easily identifies honeypots. We distinguish the following types of tools [20]:

- **information exchange** is defined as obtaining information either from other attackers or from the people being attacked (e.g. data from spam honeypot).
- **user command** is defined as exploiting a vulnerability by entering commands to a process through direct user input at the process interface (e.g. entering command in Kippo honeypot).
- **script or program** is defined as exploiting a vulnerability by entering commands to a process through the execution of a file of commands (script) or a program at the process interface (e.g. shell script to exploit a software bug in Kippo or Dionaea honeypot).
- **autonomous agent** is defined as exploiting a vulnerability by using a program, or program fragment, which operates independently from the user (e.g. viruses, worms).
- **toolkit** is defined as a software package which contains scripts, programs, or autonomous agents that exploit vulnerabilities (e.g. rootkit stored in high-interaction **honeypot).**

Shirey defines **vulnerability** as a flaw or weakness in a system design, implementation, or management that could be exploited to violate the system security policy [30] It can be defined as "a weakness in a system allowing unauthorized action" [31]. Sandia's taxonomy uses three categories of vulnerabilities, such as design vulnerability, implementation vulnerability and configuration vulnerability. Purpose of honeypot is to provide vulnerabilities to attackers or to seek vulnerabilities using the services and implementations. For this reason we use the vulnerabilities in network services (e.g. web service, ssh, service, dns service, scada system etc.)

4.3 Event

The operation of computers and networks involves innumerable events. An **event** is a discrete change of state or status of a system or device [22]. Arlat et al. define event as the instantaneous image of a system [32]. Event can be defined as "a low level entity (TCP packet, system call, syslog entry, for instance) from which an analysis is performed by a security tool" [33].

Event = {action, target, timestamp}

Action means "a step taken by a user or process in order to achieve a result" [28]. In our taxonomy we use the actions according to Sandia's taxonomy, such as to probe, scan, flood, authenticate, bypass, spoof, read, copy, steal, modify, or delete. We add new actions:

- **download**
- **input** - execution of action (e.g. read, copy) without any change

Examples of actions in honeypots are as follows: to scan, probe (low-interaction honeypot - HoneyD), to authenticate, probe, scan (medium-interaction honeypot Kippo), and to authenticate, scan, probe, modify, read, copy (high-interaction honeypot HonSSH). With the increase of interaction the number of actions is increasing, as well.

Target can be defined as "a computer or network logical entity (account, process, or data) or physical entity (component, computer, and network)" [20]. For example, ssh account in Kippo, each process in high-interaction honeypot.

The common attribute of all events is the **timestamp**. The Request for comments 4765 - IDMEF [34] uses three distinct timestamps, such as Detectime, CreateTime, AnalyzerTime. The timestamps of proposed taxonomy are based on this request for comments. In proposed taxonomy we use:

- **DetectTimestamp** – the event was created (exactly one),

- **CollectTimestamp** – the event was inserted to central database (exactly one),
- **AnalysisTimestamp** - the event was used in the analysis (zero or one).

In terms of the following analysis, we know the events in honeypots and honeynets and vulnerabilities of honeypots. Purpose of data analyse is to find information about attackers and their objectives and tools.

5 Conclusion

The honeypots and honeynets are unconventional tools. Their main purpose is to collect data and subsequently analyse them to learn information about attackers and their methods, tools and objectives. Therefore, we proposed framework for analysis of data collected from honeypots and honeynets. In paper we outline the main parts of proposed framework – client side and server side.

Location of collector is an important issue in data collection. Usage of secure communication channel is sufficient only for low-interaction and medium-interaction honeypots. For high-interaction honeypots it is needed to use another method of data collection. Collector located on virtual honeynet based on operating-system level virtualization is solution for this type of honeypots. **Analysis of data from different honeypots** is the second important issue. In paper we propose framework, which adds a new abstract layer – layer of event over all data from honeypots. Events are essential parts of data from each honeypots. Sandia's taxonomy discusses the taxonomy of incident in general. In paper we focus on **taxonomy of incident** from the perspective of honeypots and honeynets. We propose new honeypot-incident taxonomy, which is modification of already mentioned Sandia's taxonomy.

In future, we will focus on analysis of incident, based on data collected by honeypots and honeynet in more detail. We want to develop proposed taxonomy. Also, we would like to propose and implement a module for creation of intrusion detection system's schemes.

Acknowledgments

We thank colleagues from Czech chapter of The Honeynet Project for their comments and valuable input. This paper is partially funded by the Slovak Grant Agency for Science (VEGA) grant under contract No. 1/0479/12 and VVGS grant under contract No. VVGS-PF-2013-109.

References

[1] Spitzner, L.: The honeynet project: Trapping the hackers, in *IEEE Security & Privacy, 1(2)*, pp. 15-23, 2003.

[2] Spitzner, L.: Honeypots: Tracking Hackers, in *Addison Wesley*, pp. 1-430, 2002.

[3] The Conpot Project. 2015: http://conpot.org/

[4] Thug. 2015. http://buffer.github.io/thug/

[5] Biswas, J.: Analysis of Client Honeypots, in *International Journal of Computer Science & Information Technologies, 5(4)*, 2014.

[6] Dionaea. 2015: http://dionaea.carnivore.it/

[7] HoneyD, 2015: http://www.honeyd.org/

[8] Kippo, 2015: https://github.com/desaster/kippo

[9] Joshi, R. C., Sardana, A. (Eds.): Honeypots: A New Paradigm to Information Security, in *CRC Press*, 2011.

[10] HonSSH, 2015: https://github.com/tnich/honssh

[11] Sebek, 2015: https://projects.honeynet.org/sebek/

[12] Abbasi, F., Harris, R.: Experiences with a Generation III virtual Honeynet, in *Telecommunication Networks and Applications Conference (ATNAC), 2009 Australasian*, 2009.

[13] The Honeynet project: Know Your Enemy: Learning about Security Threats (2nd Edition), Honeynet Alliance, p. 768, 2004.

[14] Balas, E., Viecco, C.: Towards a third generation data capture architecture for honeynets, in *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC. IEEE,* pp. 21-28, 2005, June.

[15] Shi-wei, Y., Xiu-shuang, M., & Wei-dong, W. A. N. G.: Core Functions Analysis and Example Deployment of Virtual Honeynet, in *Computer Science*, 3, 021, 2012.

[16] Kaur, M. M., Singh, D. Attack Methodology Analysis By Using Honeypot Technology.

[17] Sharma, N., Sran, S. S.: Detection of threats in Honeynet using Honeywall, in *International Journal on Computer Science and Engineering, 3(10)*, 3332-3336, 2011.

[18] Modern Honey Network, 2015: http://threatstream.github.io/mhn/

[19] Thonnard, O., Dacier, M.: A framework for attack patterns' discovery in honeynet data, in *Digital investigation, 5*, pp. 128-S139, 2008.

[20] Howard, J. D., Longstaff, T. A.: A common language for computer security incidents, in *Sandia National Laboratories*, 1998.

[21] Brooks, R. R.: Mobile code paradigms and security issues, in *Internet Computing, IEEE, 8(3)*, pp. 54-59, 2004.

[22] Kácha, P.: IDEA: Classification of security events, their participants and detection probes, in *WSEAS TRANSACTIONS on COMPUTERS, pp.* 213-223*,* 2015.

[23] Stunnel, 2015: https://www.stunnel.org/index.html

[24] Sokol, P., Pisarcik, P.: Digital evidence in virtual honeynets based on operating system level virtualization, in *Proceedings of the Security and Protection of Information*, pp. 22-24, 2013.

[25] LXC project, 2015: https://linuxcontainers.org/

[26] MySQL, 2015: http://www.mysql.com/

[27] SQLite, 2015: http://www.sqlite.org/

[28] Radatz, J.: The IEEE standard dictionary of electrical and electronics terms, in *IEEE Standards Office*, 1997.

[29] International Organization for Standardization, "ISO/IEC/IEEE 24765c:2014 - Systems and software engineering -- Vocabulary

[30] Shirey, R.: RFC 2828: Internet security glossary, in *The Internet Society*, 2000.

[31] Álvarez, G., & Petrović, S.: A new taxonomy of web attacks suitable for efficient encoding, in *Computers & Security, 22(5)*, 435-449, 2003.

[32] Arlat, J. e. al.: Guide de la suret´e de fonctionnement, in *Cepadues editions*, 1996.

[33] Morin, B., Mé, L., Debar, H., & Ducassé, M.: M2D2: A formal data model for IDS alert correlation, in *Recent Advances in Intrusion Detection. Springer Berlin Heidelberg*, pp. 115-137, 2002, January.

[34] Debar, H., Curry, D., & Feinstein, B.: RFC4765: The intrusion detection message exchange format (IDMEF), in *Network Working Group*, IETF, 1-157, 2007.