Krzysztof CABAJ, Piotr GAWKOWSKI

Warsaw University of Technology

doi:10.15199/48.2015.02.16

HoneyPot systems in practice

***Abstract**. The paper presents the HoneyPot technology as well as the experience gained from their usage in the network of the Institute of Computer Science Warsaw University of Technology. On this background the concept of HoneyPot systems is presented and discussed. The paper is illustrated with the real-life cases of some recent vulnerabilities observed on our HoneyPots.*

***Streszczenie.** Praca przedstawia technologię systemów HoneyPot oraz doświadczenia zebrane z ich użycia w sieci Instytutu Informatyki Politechniki Warszawskiej. Na tym tle zaprezentowano i omówiono koncepcję systemów HoneyPot oraz prawdziwe przypadki najnowszych zagrożeń zaobserwowane na naszych systemach HoneyPot. (**Systemy HoneyPot w praktyce**).*

Keywords: Network Security, HoneyPot systems, Network Attacks, Exploits.
Słowa kluczowe: bezpieczeństwo sieci, systemy HoneyPot, ataki sieciowe, exploity.

Introduction

Security of information is one of the most important issues in the IT systems nowadays. The growing number of devices connected to the Internet pushed forward the necessity of secure communication and data storage. Internet of Things, Bring Your Own Device (BYOD), Intelligent Cars and Houses, Systems of Systems, Cloud-based solutions are no longer scientific nor marketing concepts but a reality – reality introducing new challenges for security and personal privacy.

The undisputable advantage of using cloud solutions is high availability of the users' data from any location on any device. It is however depended on the Internet connectivity. Fortunately, the high speed broadband is more and more popular [1]. Despite of the user privacy concerns related to the storage of data on third party servers, collection of such information like mobile device locations (e.g. Android devices) or web pages viewed by the user for advertisement, the big problem is the security of network-based applications and the network itself. That brings us to the problems of dependable networking.

Several types of threats can be pointed out in the mentioned above context. One of this is related with the confidentiality of the communication channels (e.g. robustness of routers in public Wi-Fi installations). In [2] it is showed that single-bit faults can lead to major security issues. However, the most of the problems are related to the quality of software. It may contain (or rather contains) some yet-undiscovered bugs that can be exploited, in the security field called *vulnerabilities*. This can be done through malicious documents sent by e-mail, accessing the infected web page or direct attack on the services served over the network (e.g. remote code injection through malicious network request). In this last scenario the infection does not require user actions on the affected machine to activate malicious code. Thus, it is more dangerous as without the proper system monitoring the infection could be undiscovered for long time. System user might be unaware of the problems. Here arises the problem of abnormal system behaviour – how to recognise any anomalies from normal operation [3].

Invaluable data upon security threats in the network can be extracted from the HoneyPot systems [4, 5]. In this paper we present some of our experience from over two years of several HoneyPot systems usage within our university network. In the next section we describe the main concepts of HoneyPots as well as some important aspects of their usage. The following sections introduce our HoneyPot systems in the context of the gained experience and observed intrusion attempts.

Network security threats

Typical remote attacks exploiting vulnerability start with the recon phase – vulnerable machines directly connected to the Internet are sought. This process could be performed in advance due to massive scanning of large ranges of IP addresses or just before the attack. Currently, access to the services like Shodan [6], which sequentially scans all machines connected to the Internet and records all enabled services and used software versions significantly ease this process. When potential victim is found the attacker sends so called *exploit*. The exploit is specially crafted packet, communication session or user data that using a bug (or imperfection) in software performs some actions, not intended by a programmer. Two most popular ways of attacks are remote code execution and command injection. In the first case a bug allows the attacker to execute provided machine code. This can be done for example by buffer overflow vulnerability (leading to overwritten return address). The other kind of software errors allows direct execution of OS commands. In both cases the attacker's code or commands are very limited. In effect, this code is called "a first stage", and used only for downloading and execution of larger, main malicious software used by the attacker.

One of the main problems of Internet security is the number of new, so far unknown threats due to existing security holes in the software – called "zero day threats". At the beginning there is no knowledge upon actions taken by the attacker. Following and understanding these actions and mechanisms behind them allow identifying software bugs that lead to the successful attack. It probably will also show the potential scope of damage made by the exploit and malicious software on the compromised systems. Moreover, it can help to identify the other web resources engaged in spreading the malicious software over the Internet. Nowadays most of the security systems, for example, antivirus systems or intrusion detection systems, detect malicious activity using databases of know patterns that describes malicious activity. Due to this fact, appearance of "zero day threat" cause that all users of vulnerable software are effectively not protected unless new patterns are produced by security companies and downloaded by users.

In order to prevent the malicious code from spreading across the network several steps have to be taken:

- discovery of the new threat existence;
- identification of mechanisms of compromising the target;
- assessing disaster made in the overtaken systems and further actions taken by the malicious code;
- recovery of compromised systems;

- development of security paths;
- deployment of security paths.

In the timeline, two of these steps takes most of the time: discovery of the new threats and securing the vulnerable systems. In case of (one of the latest) Heartbleed vulnerability of the OpenSSL implementation it is reported that the security hole was present for almost two years before officially announced. Some companies were informed about the vulnerability earlier [7]. Another issue is how fast the vulnerable software is updated by system administrators. Even if the OpenSSL implantation was fixed before the official announcement (which was at the 7th of April 2014), it is apprised that more than 300 thousands systems is still (as for June 23) vulnerable to Heartbleed attacks [8]. So, the life-cycle of malicious software and vulnerable systems is quite similar to epidemic behaviour – threats can spread unnoticed and last quite long after identification and availability of fixed version of vulnerable software. In this sense there are works towards modelling of this life-cycle [9]. There is also a threat that after some time after the bug fix, when the vulnerability is slowly forgotten in the community, some new installations of old vulnerable software versions can take place.

HoneyPots concept

Hackers are almost always first to be aware of some vulnerabilities, so, it is crucial to identify somehow their activities. Here the HoneyPot systems are handy. The key role of the HoneyPot can be performed by any resource that can be used for observing hostile or unexpected activity. The only common feature of this resource is that it is not used for production purposes. The HoneyPot is mostly specialized machine or software; however, in this role a fake record in the data base can be used or a fake account in the important computer system. Any access to the resource, for example, an attempt to read or login, is a sign of unexpected activity. A good survey on HoneyPots can be found in [5]. In the [10] details concerning one of the first well documented development of the HoneyPot and description of further monitoring and tracing real attacker can be found.

The suspicious activity starts with connection performed to the HoneyPot system. In effect, the HoneyPot responds with appropriate response that in most cases contains details concerning the used software (name and version). This information is specially crafted to entice as many attackers as possible presenting older (vulnerable) versions. In some cases, in this step the exchange of messages stops. This can be caused by a few facts: this is harmless connection due to erroneous usage of HoneyPot address instead of the real destination address, presented version of the software is not vulnerable to the given attacker or the attacker performs only recon activity and maybe he will be back later. If following messages are observed then it is a sign that with high probability the attacker tries to use the detected vulnerability and sends an exploit (see the previous section). In case of successful vulnerability exploitation the victim downloads larger primary malware.

We can distinguish two kinds of HoneyPot systems: *high* and *low* interaction ones. The first one provides a fully operable version of the target system enriched with monitoring functions. This solution was ideal for caching and tracking a human attacker. However, in the era of automatic threats, like worms, e-mail viruses or auto-rooters, dedicated high interaction HoneyPots systems used for gathering of copies of malicious code are not efficient and very risky. If the attacker detects and disables all monitoring mechanisms, the HoneyPot can be used for other hostile activity. Additionally, after each infection, the HoneyPot system must be cleaned. The initial deployment or cleaning the HoneyPot after a successful attack is very labour intensive. This process, even with the support of virtualization, is relatively slow. In this context, a better solution for gathering information related with malware is usage of low interaction HoneyPots. They are dedicated software that imitates vulnerable services. Depending on purpose, it can be very simple (e.g. only listing for incoming connections and returning standard banners of simulated service) or very complicated system dedicated to downloading new samples of malware. This kind of low interaction HoneyPot simulates high level protocols in which vulnerabilities appear, emulates incoming shellcode used by worms during vulnerability exploitation and downloads next stages of the malware. The most important low interaction HoneyPots are HoneyD [4], Nepenthes [11] and its successor Dionaea [12].

HoneyPot systems can gather detailed information concerning attacker's IP address, used attack technique and even executable code that attacker want to execute on victims machine. The HoneyPot configuration enables various monitoring mechanisms that during attacks gather as many as possible data concerning the attacker's activity. For this purpose the logs from operating systems, network devices placed between the HoneyPot and Internet or even traces of all the traffic directed to it can be used.

In typical attack two stages can be distinguished: code injection and then malware installation on the compromised system. Analysing an attack the one can have two different goals related to these two stages of attack. The problem is with the amount of data for the analysis and differences in the types of information related with these two stages (i.e. a set of network packets and connections data against a dynamic model of malware execution). So, different methods and tools are needed for both stages of attack. As a result it is not sufficient (nor practical) to set only one HoneyPot system in the organization network. We faced this problem in practice. To identify potential threats we use low-interaction HoneyPots collecting all connections data. Based on the analysis (e.g. data mining, clustering and classification) and trend analysis some specialized HoneyPot systems are developed and setup to catch the exploits' code and the malware (presented in further sections). Then, a separated system emulates the behaviour of the collected malware to analyse its execution.

Another problem is related to the attractiveness of the HoneyPot to the attacker (no matter if it is a human or a bot). The more interesting target system is, the faster the attack will occur. That principle has been proven in our practice – examples are given in the further sections. The HoneyPot is not production system and naturally delivers only limited service for the attacker (e.g. emulates only the vulnerable parts of a service). This leads to an efficient and effective solution. However, if the provided service is too simple the attacker will not be attracted to exploit the system. At the other side, if the HoneyPot provides unnaturally rich set of vulnerable services (i.e. the old ones), the attacker can assume that it is not a true production system but a HoneyPot. So, it is important to provide the dedicated services only if there is observable interest from the attackers to assure the creditability of the HoneyPot. Here the implementation details are also important. For instance, in [13] the authors pointed that high-interaction honeypots implemented on virtual machines have less success in identifying websites that host malware in drive-by-download attacks if malware is able to detect VM implementations of honeypots.

In [14] authors proposed a framework to dynamically enable HoneyPots for interesting services. In the presented system only network activity was monitored and using this information new HoneyPots were deployed. In our research we enhanced this idea using simpler HoneyPots for gathering initial data. The analysis shows then what new HoneyPots should be later deployed. Nowadays, with many attacks directed to various Web applications, simply enabling a web server is not efficient. For attracting the attackers, some more complicated systems (e.g. with advanced web applications) must be deployed. In the network of our Institute we use a few types of HoneyPots. The simplest one is based on well-known open source Dionaea low interaction HoneyPot. The second type of HoneyPot is custom made WebHP system, devoted to the http protocol. Additionally, after analysis of the requested URLs some specialized applications, monitored by the own developed WebHP systems are manually enabled. Unfortunately, due to the complicated analysis, which finds vulnerable applications using the observed requests, fully automatic HoneyPot deployment is almost impossible, so, we use so far only manual deployment. To summarize, currently, in the network of the Institute of Computer Science we have deployed the following HoneyPots and analytical systems: Dionaea, WebHP, HPMS and HeartBleedHP. Except the Dionaea, they all are designed and developed in our Institute. In the following sections we present these systems with some practical cases.

Dionaea

The Dionaea low interaction HoneyPot is successor of the Nepenthes system. Its main purpose was easy gathering of worm samples, especially those directed to the Microsoft Windows. So, it implements full Windows remote procedure call (RPC) stack used for example by DCOM and various Windows network services. Additionally, system can simulate other services, for example, telnet, ftp, http and https. For analysis purpose and exploit detection in suspicious messages for x86 processor shellcodes a libemu library can be used. Dionaea system can download malicious code using various protocols, for example, ftp, tftp and http. All data concerning activity of the systems are stored in Postgress database. Easy access to data is provided by additional Web panel called *carniwwwhore* [15]. The main drawback of the Dionaea is poor emulation of http service. So, we developed some dedicated HoneyPots described later on.

Using the Dionaea the one can identify new attempts of attacks. For instance, we identified the first connections related to the most recent Supermicro IPMI implementation bug [16] in advance for one and a half months before the official bug announcement (19th of June 2014). We noticed the first connection to the 49152 port at 29th of April 2014. Before that date there were neither connections to this port nor any ports from the neighbourhood. Then, no connections were observed during the whole May. In June, before the bug announcement, there was only 5 connections but after the announcement over ten days we noticed 16 attempts.

WebHP and HPMS

WebHP was developed due to limited Dionaea capabilities associated with gathering details on data exchange in application layer between the attacker and the HoneyPot system, especially using HTTP protocol. To address the problem of the analysis of the gathered data we also developed HPMS (HoneyPot Management System). Analysing the data is a serious problem when we have in mind the amounts of data that can be gathered by HoneyNets – installations in which hundreds or even thousands of HoneyPot systems can be deployed. Known web consoles for HoneyPots, for example Carniwwwhore [15] and DionaeaFR [17], show only sample stats, graphs and plots. These systems do not allow performing any analysis on gathered data. Contrary, the main aim of HPMS system is associated with performing analysis and presenting the results to the user. For this purpose HPMS system uses data mining algorithms, frequent sets analysis in particular [18]. In effect, system shows not only all connections but some patterns that group many connections sharing similar features. For example, when an attacker requests a given URL multiple times, the system detects and shows only one pattern. More details concerning analytic features of HPMS system are presented in [19].

WebHP was developed as specialized data capture script implemented in PHP language. It must be placed in each monitored page of prepared Web HoneyPot static pages or an application. It is responsible for logging all requests sent from the attacker to the data base used by HPMS management system. Additionally, in the implemented Web HoneyPot a custom error page was prepared, which included data logging script, too. This allows capturing any requests, even if the requested page is not present in the Web HoneyPot.

The HPMS system was implemented in Python language using Django framework. It allows easy access to all the data captured by Web HoneyPot, for example, searching for interesting requests and plotting activity in given time frame. Moreover, the user can define rules, which automatically tag all the requests matching certain conditions [19].

Below a detailed description of a sample attack directed to the Web server that hosts PHP My Admin (PMA) application simulated by our WebHP HoneyPot is presented. The first attacker's request looks like innocent attempt to gather the file /phpmyadmin/scripts/setup.php (see Fig. 1). The file name is little strange, but the Web server responds with appropriate data. What is important to the attacker, the returned file contains unique session token that is later used during the injection attack.

```
GET /phpmyadmin/scripts/setup.php HTTP/1.1
Host: 194.29.XX.YY
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MSIE 5.5;
Windows NT 5.1) Opera 7.01 [en]

HTTP/1.1 200 OK
Date: Thu, 19 Jun 2014 08:51:35 GMT
Transfer-Encoding: chunked
Content-Type: text/html

24a7
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"><head>
. . .
href="http:///app/phpMyAdmin/setup/index.php?version_chec
k=1&amp;token=a60b82c06fa123c868288b29584d345">Check for
latest version</a>
. . .
```

Fig. 1. Initial message and response during the attack to the PMA application gathered by WebHP system.

The second message that is sent from the attacker to the victim contains an exploit. In this case the attacker utilizes code injection error in PMA application that allows remote change of configuration object. In effect the attacker can download and execute on the target machine any malicious code. Message containing the exploit, which was sent to the HoneyPot is presented in the figure 2.

```
POST /phpmyadmin/scripts/setup.php HTTP/1.1
Host: 194.29.XX.YY
Referer: http://194.29.XX.YY/phpmyadmin/scripts/setup.php

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MSIE 5.5;
Windows NT 5.1) Opera 7.01 [en]
Content-Type: application/x-www-form-urlencoded
Content-Length: 238

action=lay_navigation&eoltype=unix&token=a60b82c06fa123c8
68288b29584d345&configuration=a%3A1%3A%7Bi%3A0%3BO%3A10%3
A%22PMA%5FConfig%22%3A1%3A%7Bs%3A6%3A%22source%22%3Bs%3A2
9%3A%22ftp%3A%2F%2F37%2E59%2EAA%2EBB%2Fpub%2F124%2Ephp%22
%3B%7D%7D
```

Fig. 2. Message containing an exploit used during the attack to the PMA application gathered by WebHP system.

The first marked in grey part of the message is the session token gathered already in the initial phase of the attack. The second marked part (after decoding) contains the following text:
"*a:1:{i:0;O:10:"PMA_Config":1:{s:6:"source";s:29: "ftp://37.59.AA.BB/pub/124.php";}}*"

This text contains specially crafted PMA software configuration object which due to an error in the application can be remotely changed. In effect, vulnerable software injects in the currently executed instructions the code downloaded from provided malicious URL – file 124.php from IP 37.59.AA.BB. Because this exploit is sent to the low interaction HoneyPot, this attempt was only stored in the data base and malicious code did not execute. However, later manual analysis proved that the file 124.php contains malicious code that is used to remotely control infected machine and create a BotNet.

Presented in this real-life example data are gathered from communication exchange observed by the developed in the Institute of Computer Science WebHP system. The information stored in the data base concerns attackers' IP addresses and used URLs almost uniquely identifies vulnerable application and exploit code. In most cases this exploit code contains IP address and file names associated with the malware to be downloaded.

In currently deployed installation of WebHP HoneyPot it runs on three distinct IP addresses. Web pages with logging script are executed in various configurations, using four distinct ports numbers: 80, 8080, 443 and 5000. Additionally, various applications are provided, for example: simple "in construction" page, a guest book and PHP My Admin application. New ports and applications are constantly added to study new malicious activities. For example, port 5000 was added after observing a huge rise of activity in this port by Dionaea HoneyPot deployed in the network earlier. It is related to the bug in Synology NAS vulnerability [20] (presented in further section).

HeartBleedHP

The HeartBleedHP is the most recently deployed HoneyPot system. It is aimed on answering the question if anyone interacts with OpenSSL HeartBleed bug (CVE-2014-0160) [7]. This HoneyPot provides a simple page by HTTPS protocol. Moreover, this page is integrated with WebHP script, thus all requests to this page are intercepted. HeartBleedHP is developed using Apache web server and specially modified OpenSSL library which enables HTTPS protocol. Implemented additional functionality adds full logging to the vulnerable implementation of HeartBeat message [21]. In effect this installation intercepts all requests which utilize this rarely used message. Moreover, all data transmitted by the attacker and provided to him by the server are stored too. Figure 3 presents sample log produced by HeartBleedHP.

```
Data: Wed Apr 30 22:56:07 2014
RREC_length:60
Payload_length:79
..Oheartbleed.filippo.io YELLOW SUBMARINE
194.29.XX.YYY:443.¸ŢÂ.       "-.Ţ
ýŠčŢD!.7ÄŐ8+4,.¸Ňő @Ę.őg.U1
```

Fig. 3. Sample log concerning HeartBeat massages.

This HeartBeat message logged on HeartBleedHP system originates from the vulnerability scanner. First three lines are added by implemented modification and contain respectively date and time of the event, received, requested by the attacker and transmitted data lengths. Later, the exact data sent to the requester is presented. As can be seen, the attacker received 19 additional bytes of HoneyPot memory (the difference between declared and real payload length: 79-60). For more than a month of HeartBleedHP deployment we observed more hostile activity. The HeartBleedHP was deployed at 26th of April. The first connection was noticed next day after the deployment. Till 26th of June 8 connections from 4 different addresses were logged. The most aggressive attacker requested the maximal allowable size of 16 KiB and did such request eleven times. That means that the attacker downloaded 160KiB of internal OpenSSL data.

Synology NAS case

HoneyPot systems can point out protocols and applications that are currently very interesting for attackers. This fact is with high probability a sign that some exploitable vulnerability was discovered. Monitoring deployed HoneyPot systems allows detection of such situation. Figure 4 presents the activity observed by our Dionaea instance concerning port 5000. As can be easily seen at the beginning of March 2014 we recorded huge number of connection attempts to this port. What is interesting, in the whole 2013 year HoneyPot observed only 21 such events.

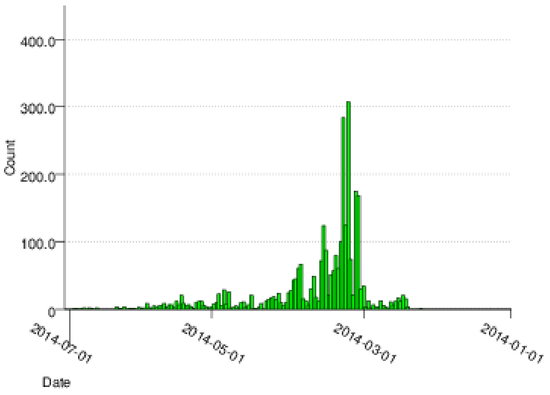

Fig. 4. Activity observed by Dionaea HoneyPot concerning port 5000 daily activity, at the beginning of 2014. Plot from Carniwwwhore web panel.

The rise in the number of observed events and some rumours that activity is associated with HTTP protocol encourage us to deploy WebHP using this port, which initially was deployed at the beginning of April. Just after the deployment first appropriate HTTP requests were recorded. From April to middle of June more than 500 connections were observed. All the requests belong to two classes associated with URL /webman/info.cgi and /webman/imageSelector.cgi. The first URL uses HTTP protocol's GET method, which probably is a sign of searching for vulnerable machines. The second URL uses POST method and exploits vulnerability in imageSelector.cgi script. As presented in the Figure 5, the vulnerable script has a bug that allows execution of commands provided by the attacker. This kind of activity is commonly known as command injection attack. In the presented figure commands sent by the attacker in the exploit are marked in grey.

```
   --shit_its_the_feds Content-Disposition: form-data;
name="source" login --shit_its_the_feds Content-
Disposition: form-data; name="type" logo --
shit_its_the_feds Content-Disposition: form-data;
name="foo"; filename="bar" Content-Type:
application/octet-stream sed -i -e '/sed -i -e/,$d'
/usr/syno/synoman/manager.cgi export
TARGET="stratum+tcp://5.104.XX.YY:3344" && curl -L
http://109.163.AA.BB/.h/run.sh | sh 1>/dev/null && unset
TARGET && echo QQQ --shit_its_the_feds--
```

Fig. 5. Recorded by WebHP exploit to imageSelector.cgi using command injection attack. Commands sent by the attacker are marked in grey.

Analysis of executed commands reveals that the attacker set some shell environment variable and downloads and executes additional software. Analysis of downloaded software and word "stratum" in the exploit proved that with high probability this activity is associated with bitcoins mining on victims machine. What is interesting, the URL suggests that this attack is directed to the NAS devices manufactured by Synology company. More details concerning this particular attack that proves our observations are described in the dshield handler blog [20].

Summary

The HoneyPot systems deployed in the Institute of Computer Science proved their practical suitability. This year (as for the end of June) already 56500 connections to our Dionaea system on different ports were observed. Through the whole previous year we got 174380 connections. At the same time the WebHP HoneyPot got 182000 connections just to the http service. We observed that the number of attacks is correlated with the complexity of the web application on the HoneyPot. More sophisticated application (e.g. PHP MyAdmin, guest book) attracts more attacks.

Analysing trends in the connections to the HoneyPot systems it is easy to identify some new trends in exploits. Described (in the previous section) rise in the number of attacks to port 5000 revealed a serious flow in Synology NAS software. Usage of HoneyPots allows obtaining such results even without access to vulnerable software. Rapid detection of such event can decrease so called "zero-time exploit" activity. In this context HoneyPot systems can be very helpful in identification of bugs for improving software quality and security.

The further work will be focused on developing analytical systems to support identification and classification of security threats as well as on integration with the infrastructure monitoring systems in the Institute [3].

REFERENCES

[1] January 28, 2014 – Akamai Releases Third Quarter, 2013 'State of the Internet' Report http://www.akamai.com/html/about/press/releases/2014/press_012814.html

[2] Nazimek P. P., Sosnowski J., Gawkowski P.: Checking fault susceptibility of cryptographic algorithms, *Pomiary-Automatyka-Kontrola*, (2009), nr 10, 827-830

[3] Sosnowski J., Gawkowski P., Cabaj K., Exploring the Space of System Monitoring, in: Intelligent Tools for Building a Scientific Information Platform: Advanced Architectures and Solutions / Bembenik R. [et.al.] (eds), *Studies in Computational Intelligence*, 467 (2013), 501-517

[4] Niels P., Thorsten H., Virtual Honeypots: From Botnet Tracking to Intrusion Detection, Addison-Wesley Professional (2007)

[5] Bringer M. L., Chelmecki Ch. A., and Fujinoki H., A Survey: Recent Advances and Future Trends in Honeypot Research, *I. J. Computer Network and Information Security*, (2012), 10, 63-75

[6] Shodan search engine web page: http://www.shodanhq.com/

[7] Heartbleed Bug Health Report, https://zmap.io/heartbleed/

[8] Brewster T., theguardian.com, More than 300k systems 'still vulnerable' to Heartbleed attacks: http://www.theguardian.com/technology/2014/jun/23/heartbleed-attacks-vulnerable-openssl

[9] Staniford S., Paxson V., and Weaver N., How to Own the Internet in Your Spare Time. *Proceedings of the 11th USENIX Security Symposium*, Dan Boneh (Ed.). USENIX Association, Berkeley, CA, USA, (2002) 149-167

[10] Cheswick B. An Evening with Berferd in which a cracker is Lured, Endured, and Studied, *In Proc. Winter USENIX Conference,* (1992)

[11] Baecher P., Koetter M., Dornseif M., Freiling F., The nepenthes platform: An efficient approach to collect malware, In Proceedings of the 9 th International Symposium on Recent Advances in Intrusion Detection (RAID06), (2006)

[12] Dionaea home page, http://dionaea.carnivore.it/

[13] Narvaez J., Aval Ch., Endicott-Popovsky B., Seifert C., Malviya A., and Nordwall D., Assessment of Virtualization as a Sensor Technique,‖ *Proceedings of the IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, May (2010), 61-65

[14] Jiang X., Xu D., BaitTrap: A Catering HoneyPot Framework, Department of Computer Science Technical Report CSD TR 04-0xx, Purdue University, August (2004), http://friends.cs.purdue.edu/pubs/BaitTrap.pdf

[15] Carniwwwhore project page: http://carnivore.it/2010/11/27/carniwwwhore

[16] Supermicro BMC vulnerability, http://blog.cari.net/carisirt-yet-another-bmc-vulnerability-and-some-added-extras/

[17] DionaeaFR project page: http://rubenespadas.github.io/DionaeaFR/

[18] Agrawal R., Imielinski T., Swami A., Mining Association Rules Between Sets of Items in Large Databases, *Proceedings of ACM SIGMOD Int. Conf. Management of Data*, (1993)

[19] Cabaj K., Denis M., Buda M.: Management and Analytical Software for Data Gathered from HoneyPot System, in: Information Systems in Management, WULS Press Warsaw, 2 (2013), nr 3, 182-193

[20] Ullrich J. B., More Device Malware: This is why your DVR attacked my Synology Disk Station (and now with Bitcoin Miner!), InfoSec Handlers Diary Blog, 2014.03.31,http://dshield.org/diary/More+Device+Malware%3A+This+is+why+your+DVR+attacked+my+Synology+Disk+Station+%28and+now+with+Bitcoin+Miner!%29/17879

[21] Seggelmann R. , Tuexen M., Williams M. ,Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension, RFC 6520, (2012), ISSN: 2070-1721

***Authors**: dr inż. Krzysztof Cabaj, Politechnika Warszawska, Instytut Informatyki, ul. Nowowiejska 15/19, 00-665 Warszawa, E-mail: K.Cabaj@ii.pw.edu.pl; dr inż. Piotr Gawkowski, Instytut Informatyki, ul. Nowowiejska 15/19, 00-665 Warszawa, E-mail: P.Gawkowski@ii.pw.edu.pl.*

The correspondence address is:
e-mail: K.Cabaj@ii.pw.edu.pl