Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Expert system assessing threat level of attacks on a hybrid SSH honeynet

Matej Zuzčák*, Milan Zenka

Department of Informatics and Computers, Faculty of Science, University of Ostrava, 30. dubna 22, 701 03 Ostrava, Czech Republic

ARTICLE INFO

Article history:
Received 3 September 2019
Revised 25 February 2020
Accepted 2 March 2020
Available online 4 March 2020

Keywords:
Honeypot
Medium interaction
High interaction
Hybrid honeynet
Expert system
SSH
Classification
Transparent redirection

ABSTRACT

Currently, many systems connected to the internet are exposed to hundreds of mostly automated network attacks on a daily basis. These are mostly very simple attacks originating from botnets. However, sophisticated attacks conducted both by automated systems and directly by humans are becoming more common. In order to develop adequate countermeasures, the behaviour of attackers has to be analysed effectively. Honeypots, a sort of lures for the attacks, are used for that purpose. Configuration of honeypots vary depending on the type of attacks they focus on attracting. For simple, analogous attacks that sequentially repeat predefined commands, medium interaction honeypots are sufficient, while more sophisticated attacks require the use of high interactive honeypots. An essential part of the analysis is to differentiate between these types of attacks to make the overall analysis efficient, in terms of efficient use of hardware resources, and effective by providing the attacker with an appropriately emulated environment. This article first analyses the current situation followed by presenting a solution in the form of a system made up of a hybrid honeynet and an expert system. For now, it focuses only on the SSH protocol, as it is widely used for remote system access and is a popular target of attacks. The system has been tested on real data collected over a one-year period. The article also deals with making redirecting SSH connections as transparent as possible.

1. Introduction

Cybersecurity is one of the most dynamic areas of commercial, academic, scientific, and even personal life. Therefore, to be able to react to both existing and new threats effectively, it is necessary to gain awareness of what threats are currently spreading and what is their destination and target. To gather the data, honeypots, and logical networks of honeypots known as honeynets,[1] are used.

The subject of this paper is to propose an expert system made to effectively classify the source of the connection to be either a simple or a sophisticated attacker. A simple attacker is typically a bot or an unskilled human attacker only executing a sequence of predefined, repeating commands, or it is a script-kiddie analysing the system and attempting to draw attention to itself. On the other hand, a sophisticated attacker, whether human or advanced malware, reacts to the situation dynamically. The honeynet is comprised of systems emulating SSH protocol, on network port 22 by default, that is among the most popular means for remote access to Linux shell, and administrators use it to manage remote systems or networks. However, it can also be used by an attacker. The SSH protocol was selected as it is among the most attacked protocols, according to the following reports: F-Secure Attack landscape H2 2018,[2] Akamai - The State of the Internet Q4 2014.[3] Also, the activity and artefacts left behind by an attacker using SSH connection, such as inputted commands or the SSH client used, are analytically useful.

To discern and record practices of attackers mainly medium interaction honeypots were used, namely Cowrie.[4] Cowrie honeypot emulates Linux shell and many of the basic Linux operating system programs, such as wget or SCP. It is a "gateway" capable of providing an overview of the currently spreading simple attacks and dealing with script-kiddies, human attackers us-

* Corresponding author.
E-mail addresses: matej.zuzcak@osu.cz, mzuzcak@secit.sk (M. Zuzčák), milan.zenka@osu.cz (M. Zenka).

[1] Honeynet – in the context of this article, it represents a logical network of multiple independent honeypots connected to various networks, such as academic, ISP, etc., that send the data about the captured connections to a central database for analysis.

[2] F-Secure Attack landscape H2 2018 – https://blog.f-secure.com/attack-landscape-h2-2018/.

[3] Akamai - The State of the Internet Q4 2014 – https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/akamai-state-of-the-internet-report-q4-2014.pdf.

[4] Cowrie – https://github.com/cowrie/cowrie.

https://doi.org/10.1016/j.cose.2020.101784
0167-4048/© 2020 Elsevier Ltd. All rights reserved.

ing simple one-click tools easily available on the internet and requiring no expertise. An experienced human attacker and sophisticated malware can quickly realise the system is emulated, rendering medium interaction honeypots mostly useless in their analysis. A typical bot is just repeating a predefined sequence of commands without deeply probing the environment, so they are not affected by the system being emulated.

To recognise new trends of threats and to identify and understand new attacks, more sophisticated attacks than those effectively revealed by medium interaction honeypots (MIH) need to be analysed. This is achieved by utilising high interaction honeypots (HIH). To ensure the analysis is done effectively, automated attacks should be routed to medium interaction honeypots sufficient to analyse them, while the more sophisticated ones should be routed to high interactive honeypots not easily detectable by the attacker.

Several aspects need to be evaluated to decide where will the particular connection, typically an attack, be redirected. Honeynet capable of discerning and routing attacks to honeypots with the appropriate level of interaction is known as a hybrid honeynet. Details of honeypot and honeynet classification are in the Section 2. The discerning of what level of interaction honeypot redirect the attack is done by the expert system.

The key to the building this honeynet is differentiating between these two types of sessions. This allows the honeynet to transparently redirect only the relevant attacks to the high interaction honeypots that are rather limited in numbers while processing the bot attacks at medium interaction honeypots. The advantages gained by differentiating between the attacks are:

- Saving the limited number of high interaction honeypots for sophisticated attacks.
- Different analytical and statistical approaches can be used for each group.
- Different security measures preventing the spreading of the given threat can be deployed appropriately.
- A bot or an unskilled human attacker can execute their scripts on a medium interaction honeypot without detecting it is an emulated environment.
- A skilled human attacker or sophisticated malware can realise the environment of a medium interaction honeypot is emulated, at which point the attacker usually disconnects. Having human attackers interact with a high interaction honeypot instead, allows longer observation of the attacker's behaviour and more data to be gathered.

2. Honeypot & honeynet background

The primary goal of a honeypot (Joshi and Sardana, 2011; Spotzner, 2002) is the recording and analysing of an activity undertaken within it. The activity in question is most commonly malicious, its goal being using the infected system to spread itself via local network or the internet, conducting DDoS attacks, sending out spam messages, etc. Honeypot can be a single system, a single device, or an entire network. Such a system is usually intentionally vulnerable and lacks anything valuable that could be used or stolen. It is usually operated by people whose goal is to analyse and assess the activity conducted on it. It is essential to isolate a honeypot to the extent that no activity conducted within could negatively affect its surrounding or spread by LAN, WAN or the internet. On the other hand, it must be isolated in a sophisticated way to allow certain, highly controlled, contact of the attacker with the outside, since the attacker should be given an impression of a real system without realising the security boundaries. A suitable compromise between how secure and how realistic the system is has to be established. That depends on what particular threats is the given system supposed to attract and analyse. Therefore, honeypots are divided into the following groups: The classification is based on the theoretical background in Joshi and Sardana (2011),Provos and Holz (2007), Ligh Hale et al. (2011) and Grudziecki et al. (2012):

- **Provided services and the activity of the honeypots:**
  - Passive (server-based) passively providing vulnerable services on certain ports.
  - Active (client-based) actively looking for vulnerabilities.
- **Level of interaction:**
  - Low emulation of vulnerable services. After a session is established, a single pre-programmed action is executed. Low interaction honeypot will be abbreviated to LIH further on.
  - Medium honeypot provides more advanced interaction, e.g. providing complex emulation of a provided protocol. Medium interaction honeypot will be abbreviated to MIH further on.
  - High provides the attacker with an entire system with all services and applications. Most commonly done by virtualising a real, modified system. Virtualisation enables a quick and simple return to a “clean state”. High interaction honeypot will be abbreviated to HIH further on.
  - Hybrid a combination of honeypots with various levels of interaction that, using predefined rules and based on the network connection, redirects individual sessions to a honeypot of an appropriate level.
- **Primary use:**
  - Security often used by organisations to monitor employee activity, malware spread on their networks, etc.
  - Research primarily focused on the detection of new threats and mapping the situation at the time. Usually operated in isolated networks with a secured outside connection.
  - Shadow a subgroup of security honeypots with an overlap with certain aspects of the research ones. They redirect suspicious network traffic to an isolated environment for analysis.

Honeypots can further be grouped by the systems they emulate, e.g. those emulating Windows services, like Dionaea[5]), those emulating Linux shell, like the aforementioned Cowrie, etc.

The term “honeynet” (Abbasi and Harris, 2009) is context-dependent, as it is often used to describe HIHs. In this case, however, it applies to an entire specific network that, besides honeypots, contain other optional components, such as special firewall called honeywall, an IDS[6]/IPS[7] system, various databases and data processing systems, etc. Analytical and control parts of honeynets are commonly grouped into four categories (Balas and Viecco, 2005): Data control, Data capture, Data collection, and Data analysis.

Another meaning of the term “honeynet” is a system of honeypots forming a logical, not a physical system. It is most commonly used for groups of LIHs and MIHs. Use of special parts, such as firewalls, is not necessary. Data from all the honeypots within a honeynet are commonly stored in a single shared database. Honeynet's added value lies in higher relevancy and volume of acquired threat data for further analysis. This is the meaning used in this article.

[5] Dionaea – https://github.com/DinoTools/dionaea.

[6] IDS – Intrusion detection system.

[7] IPS – Intrusion prevention system.

3. Related works

The following section presents the so-called “state of the art” of current hybrid honeynet solutions. It maps their current existing approaches.

The very first implementation of the concept of a hybrid honeynet is described in Bailey et al. (2004), where the authors describe the design of their architecture. It divides parts of the honeynet to so-called frontend and backend. From another point of view, it is composed of three parts: Lightweight LIHs forming the frontend, HIHs forming the backend, and so-called Command & Control part controlling the entire honeynet and processing the data. The goal of the LIHs is to filter out traffic to the HIHs, to send them only the appropriate sessions. The HIHs are run using VMware[8] software, but the authors do not specify what software was used for the LIHs. After an attacker has finished interacting with a HIH, it saves the current state of the system using the snapshot feature of VMware and restores the system to the “clean state”. The snapshot can be used for further forensic analysis. The differentiation between “uninteresting” and “interesting” network traffic is rather straight forward. All packets not belonging to existing sessions, like SYN packets not leading to the establishment of a three-way TCP handshake, or data packets with a previously detected payload, are considered “uninteresting” and processed by the LIHs. On the other hand, packets with an unknown payload that are a part of an established TCP session are routed to the HIHs. When a packet with a new payload is detected, all following packets from its source within the given session are also redirected to HIHs.

Honeybrid (Wicherski and Berthier, 2009) used by some of the papers mentioned further on was made as part of the Google - Google Summer of Code[9]. It was part of The Honeynet Project[10] and it was built by Robin Berthier and his mentor Geroge Wicherski. The goal was to create the part deciding where to redirect sessions and the part doing the rerouting. Therefore, it can effectively differentiate and transparently redirect appropriate network traffic to either HIHs or LIHs. The last version was released in 2013. Besides the above mentioned parts, it also contains a part for controlling outgoing connections from the honeynet, to limit the possibility of using it for malicious activity, and a part for logging. The decision making part is modular, made up of six modules and an interface allowing adding additional modules ad-hoc if needed, such as enabling cooperation with the Snort[11] system. This part can be considered to be a pseudo-expert system. Pseudo, because it does not follow the conventions for creating expert systems, but it does evaluate each session based on multiple criteria in a way an inference engine does, based on rules forming the system’s knowledge base. Typically used modules are:

- Random – introduces a random element when redirecting communication to HIHs or LIHs.
- Yesno – either always accept or always rejects a connection.
- Counter – if the number of packets within a session exceeds a given number, executes a pre-programmed script.
- Source – differentiates between the first session from a source and the consequent ones.
- Hash – calculates a control sum for packet payloads and compares them to a database of known payloads. Continues differently for a match and a no match.
- Control – limits packets arriving from the same source IP address.

[8] VMware – https://vmware.com/.

[9] Google Summer of Code – https://summerofcode.withgoogle.com/.

[10] The Honeynet Project – https://honeynet.org/.

[11] Snort – https://snort.org/.

Honeybrid utilises a configuration file that allows the user to choose which modules are to be active, how should they be used, and define scripted actions specific for HIHs and LIHs. From a practical point of view, Honeybrid forms a gateway deciding to which honeypot should each session be redirected. The honeypots themselves can either be existing honeypot solutions, such as Honeyd[12] for low interaction, or a real web server, such as Apache[13] for high interaction. The user also has to be able to appropriately set IP tables and to process recorded logs Honeybrid generates.

3.1. Payload analysis, TCP connections, redirecting based on ports

Paper (Artail et al., 2006) focuses on creating a hybrid honeynet using the Honeyd solution for security purposes in real life networks. It postulates how to effectively redirect communication from virtual LIHs, using available IP addresses of the given network, to HIHs. This is possible by mapping of available IP addresses. Communication going through certain ports, such as port 80, is automatically always redirected to HIHs, but no other mechanism for selecting sessions to be redirected is used. Therefore it lacks an expert system or its analogue and is meant to be used as an extension of existing solutions by incorporating otherwise unused address space of the given network.

Paper (Kyaw, 2008) is a hybrid honeypot used to distinguish only two types of network traffic - valid TCP connection and everything else. Every correct TCP connection is redirected to HIH while all other connections, such as port scans, is redirected to LIH. The solution uses Honeyd as its LIH, and in case of a correct TCP connection, it acts as a sort of a proxy server. Network traffic classification is rather trivial.

Paper (Qiao et al., 2013) proposes an implementation of network based on a hybrid honeynet, but it does not deal with the decision making regarding reconnecting. It focuses on the deployment of the Honeyd solution, mapping the network, and redirecting connections based on the services used by the connection and those available by honeypots. Snort signatures are used for the attack analysis.

Paper (Bao et al., 2018) presents a hybrid honeynet focused at differentiating between port scan attacks and more complex attacks requiring more interaction. For example, establishing TCP connection of port scan attacks is not complex, while it usually is with different types of attacks. The differentiating mechanism operates at network and transport layer. It uses mainly source and destination port to identify individual connections. For example, port scanning is directed to a low interaction honeypot, while after a successful establishment of the TCP connection, packets containing a payload are redirected at a high interaction honeypot based on the protocol used, eg. HTTP. In the initial intrusion phase, only the low interaction honeypot is active and online, with the high interaction one only turned on after a complex network connection on a transport layer has been established. Redirection is handled by SDN (Software Defined networking (Benzekki et al., 2016). The article does not specify what exact honeypot solutions were used. The paper only mentions the system was only deployed for load testing, not in a live environment.

3.2. Using IDS signatures

Paper (Fan et al., 2016) presents a concept based on the aforementioned Honeybrid, extensively using a TCP proxy. The decision where to redirect communication is done by using the Honeybrid solution, specifically its extension using Snort signatures. The paper does not provide specifics of the decision-making process, but

[12] Honeyd – http://www.honeyd.org/.

[13] Apache web server – https://httpd.apache.org/.

it does not seem to be using an expert system, and it also does not use the SSH protocol. It is mainly concerned with the technical implementation of transparent redirection of sessions. It concludes that the current solution does not support the use of encrypted protocols, such as HTTPS and SSH.

Solution in paper (Chawda and Patel, 2015) uses the unused address range of the real network, Honeyd, and redirecting connections to HIHs providing certain operating systems and services used in the network. Technical details regarding redirecting are not included, but the included schema suggests Honeyd is used as a proxy. Snort signatures and Sebek are used to filter out and analyse the traffic.

Paper (Fan and Fernandez, 2017) focuses on a transparent mean of redirecting connections from the point of deciding to a MIH or a HIH. Establishing and redirecting the TCP connection is transparent, meaning it is not visible for the attacker, using the Software-Defined-Network (SDN)[14] - open-source framework Ryu SDN Framework[15] a OpenFlow[16] switch. The redirection decision is based on Snort signature analysis with the decision-making process not being specified, and admittedly not yet finished. The solution redirects to either MIH or HIH.

Paper (Fan et al., 2015) divides the current standard approaches to deploying HIH into three generations: the first, the second, and the third. All the described solutions sum up their advantages and disadvantages, and they are outdated, as the last, the third, generation was said to emerge in 2005. A new solution is proposed in the paper, built on Virtual Networks over Linux and Honeyd. Monitoring of the HIH is done by the discontinued solution Nitro[17]. Honeybrid is used for connection redirection. Redirection decisions are based on Snort signatures. It is stated the proposed solution can not differentiate between automated and human attacks.

Paper (Baykara and Das, 2018) proposes a system for securing enterprise networks. The system combines three components: Hybrid honeynet, IDS system, visualised monitoring. Honeyd is the low interaction honeypot employed here. Honeyd is used to create system profiles on the link layer of the TCP/IP model.[18]. Those are used to create a simple simulation with a low level of interaction when an open network port is served by a Python script. The concept of a hybrid honeynet is limited to choosing a specific script to serve a port based on basic metrics, such as the protocol used. Details of the high interaction honeypot used, and on how are individual attacks ascertained to be redirected to it, are not mentioned, besides that third-party software was used. In regards to data analysis, mainly data of packets available on network and application layers is analysed and processed for further use in an IDS system, to improve its anomaly detection in packets. The solution was tested on a university campus network.

Paper (Innab et al., 2018) presents a hybrid honeynet system for detection of zero-day attacks. It consists of an anomaly-based detection system and honeypots. All connections to certain predefined ports are assumed to be attacks and are redirected to the honeypot. Honeyd and its extension Honeycomb are used, generating signatures for all new incoming connections, redirecting only new connections to an unspecified high interaction honeypot.

Paper (Wang and Wu, 2019) introduces a hybrid honeynet based on SDN (Software Defined Networking). The mechanism used to decide whether to direct connection to hight or low interaction honeypot is called the Attack traffic migration module, located in the SDN controller. The decision is made based on analysing packets using Snort. Connections evaluated to be potentially dangerous, based on their payload and protocol used, are redirected to an appropriate honeypot. Functional and efficiency tests are included.

3.3. *Expert system*

Paper (Ansiry Zakaria and Kiah, 2012) describes expert systems and the so-called Case-based Reasoning (CBR) systems in both a wider context and specifically used with honeypots. An expert system is briefly described as a system divided into working-memory (WM), a knowledge base (KB), and inference engine (IE). WM is a temporary buffer containing information about the given problem from a real-world user. Based on the information, IE is looking for a solution using KB. KB is defined by IF-THEN rules. As examples of two approaches, MYCIN[19] and DENDRAL[20] systems are mentioned. The CBR systems are based on solving current problems using the knowledge acquired while solving previous problems, represented by the KB. KB is not made of rules, but rather a sort of “experience”. Advantages and disadvantages of both approaches are considered, but neither is considered to be better. Technical implementation of neither approach is included. To compare the approaches, data gathered in an unspecified network was used. Specifically, it was: IP address, used operating system, uptime, and network ports. Based on this data, the expert system, or the CBR, selected the appropriate honeypot to redirect the given connection to. For instance, a connection looking for certain unusual ports was redirected to a Windows XP SP1 system with such vulnerable ports. The level of interactivity is not considered, only the correct matching of vulnerabilities to operating systems is described, meaning for instance that connections heading to certain ports used only by windows will be redirected to Windows honeypot.

3.4. *Other approaches*

Hybrid honeypot in the paper (Kumar et al., 2012) is a combination of client and server-based honeypots, both for low and high interaction. The decision-making process is not described. For the LIH, Nepenthes solution is used, but its support ended some time ago Grudziecki et al. (2012).

Paper (Chovancova et al., 2017) presents a design and deployment example of a “sophisticated hybrid honeypot”, a honeynet scaleable according to the specific network, using HIHs and LIHs. After deployment, it maps the network, its available IP addresses, used operating systems, available network services, etc. Based on the mapped network data, it deploys virtual honeypots mirroring the real network components. It’s structured into four modules. The first contains LIHs based on Honeyd, which are configured according to the mapped data. The specific criteria by which the decision on what communication to redirect are not present in the paper and neither is a technical implementation of the process. It merely states, without going into any details, that LIHs are supposed to differentiate between human and automated attacks. The second module manages the honeynet. It is not specified exactly how, but the pictured schema states it is supposed to be using the Walleye system by Honeynet Project, a part of Roo honeywall. Its development ended in 2009 with version 1.4[21]. The third module manages the isolation of attackers from the outside world. The

[14] Software-Defined-Network (SDN)– https://onlinelibrary.wiley.com/doi/full/10.1002/sec.1737.

[15] Ryu SDN Framework– https://osrg.github.io/ryu/.

[16] OpenFlow– https://dl.acm.org/citation.cfm?id=1355734.1355746.

[17] Nitro– http://nitro.pfoh.net/.

[18] TCP/IP model – RFC 1180 - https://tools.ietf.org/html/rfc1180.

[19] Mycin expert system – http://people.dbmi.columbia.edu/~ehs7001/Buchanan-Shortliffe-1984/MYCIN%20Book.htm.

[20] Dendral – Lederberg, Joshua. How Dendral Was Conceived and Born. ACM Symposium on the History of Medical Informatics, 5 November 1987, Rockefeller University. New York: National Library of Medicine, 1987.

[21] Honeywall WallEye – https://projects.honeynet.org/honeywall/wiki/FAQ.

fourth module is the HIH, again without technical specification. The scheme shows that despite the paper being published in 2017, it uses the outdated solution Sebek (Project, 2003), which is easily detectable by automated attacks (Dornseif et al., 2004; Quynh and Takefuji, 2005), and its development ended in 2010 with the support for Linux kernel 2.6.26 and Windows XP SP2[22]. Therefore, in conclusion, this solution is not appropriate for analysis. Further on the paper describes deployment of the solution in a network and measurements from experimenting in two real networks. The proposed honeynet concept as a whole seems well scalable and adaptable for specific networks, but it lacks the description of the technical side, honeypot monitoring and the decision making regarding redirecting is not sufficiently demonstrated by the experiments. The main issue with this solution is in its use of antiquated software solutions. It also does not mention the possibility of modifications.

The solution proposed in the paper (Mansoori et al., 2012) combines advantages of server and client honeypots to extend the range of possible interactions between the honeypots and the attackers. For instance, incoming connections are passively captured and analysed, but then it passes captured URL addresses to a client honeypot that can interact with the web pages, visit them, analyse their content, etc. Specific software solutions for client honeypots are not provided. The paper states a browser and a so-called "crawler", an Internet bot that systematically browses web sites, are used along with the HoneySpam (Hayati et al., 2009) solution. Regarding honeypot server solutions, low interaction one is not specified, while Sebek is used as the HIH. Sebek is currently easily detectable even by an automated attack, thus as mentioned above in the paper (Chovancova et al., 2017), currently not viable for analysis. In conclusion, the analysis of captured communication such as attacked network ports, countries of origin, etc. are presented.

Paper (Wang et al., 2019) proposes a honeynet based intrusion detection system using machine learning, specifically support vector machine (SVM) (Cortes and Vapnik, 1995), to detect anomalous network traffic by analysing packet abnormalities, such as abnormal size, inconsistent header, specific TCP packet parameters, etc. Conpot[23] is the used honeypot, deployed using a docker[24]. Packets are processed in raw form using tcpdump[25]. Dataset KDDCUP99[26] was used to train the system. The datased contains mostly known harmful packet payloads and abnormalities of parameters in packet headers, such as abnormal size, non-standard flags, wrong fragmentation, etc. The use of this particular dataset is surprising due to its age, as twenty years is a rather long time in the field. The system does not redirect communication, neither does it analyses the interaction of the attacker with the system, it only assesses the abnormalities of the packets. When the SVM classifies a new packet, based on its training, as harmful, it adds its signature to an existing IDS, increasing its effectivity.

Paper (Sadasivam et al., 2017) does not deal with assessing what honeypot to redirect the attack to. However, it proposes an interesting classification of SSH attacks based on TCP header analysis using machine learning algorithms. Two types of attacks are proposed. The first is "severe", when the attacker successfully logs into the system and is able to input commands. The second is "not-so-severe", encompassing unsuccessful login attempts, successful login attempts with no inputted commands, or port scans. As source data, it uses pcap[27] files out of which TCP streams with the assessed parameters, such as number of sent and received packets, number of bytes comprising the payload, number of TCP segments with certain flags, etc. are extracted. A few examples of the algorithms used are Logistic Regression, support vector machine (SVM), k-Nearest Neighbour (Altman, 1992), Decision Tree (J48)[28] (Quinlan, 1993), etc. Accuracy, sensitivity, and precision of algorithms were compared, with the J48 algorithm being the best result. However, the testing was conducted using a small sample size of "severe" attacks, with only 14 samples. The entire set of input pcap files had to be manually verified by the authors of the paper. It concludes that the concept is theoretical, requiring live environment testing.

3.5. Evaluation

The studied solutions are divided into groups for ease of description. They are divided mainly based on the mechanisms deciding what level of interaction honeypot will each connection be redirected to. The groups are:

- One of the solutions (Kyaw, 2008) redirects all packets that are a part of a correctly formed TCP connection to a HIH, excluding only packets that did not follow from a TCP handshake, such as a port scan.
- The solution in Bailey et al. (2004) also decides based on whether packet payload was already detected or not, with the connection containing an unknown payload being redirected to a HIH.
- Snort signatures were often (Chawda and Patel, 2015; Fan et al., 2016; Fan and Fernandez, 2017; Fan et al., 2015; Qiao et al., 2013; Wang and Wu, 2019) used in the decision-making process, but none of the analysed articles provides details on the specifics.
- Some solutions (Artail et al., 2006; Innab et al., 2018; Qiao et al., 2013) redirect all connections going through the ports covered by the HIH to it, and they redirect certain other ports to some available HIH ports.
- The papers (Chovancova et al., 2017; Kumar et al., 2012; Mansoori et al., 2012) do not explain the decision-making mechanism at all.
- The solutions in Fan et al. (2016, 2015) use Honeybrid (Wicherski and Berthier, 2009), a modular system taking multiple aspects into consideration, making it similar to an expert system.
- The solution proposed in Ansiry Zakaria and Kiah (2012) uses an expert system, but not to decide where to redirect a connection, rather to what honeypot to build.
- Even though the solutions Wang et al. (2019) and Sadasivam et al. (2017) are not concerned with hybrid honeynet directly, they do present the possibility of using machine learning techniques, such as SVM or neural networks. Their analysis is limited to packet headers, such as inconsistencies in parameters of TCP segments. The main issue is with data sets for them to learn from. The first uses a freely available set of for IDS systems containing header data, with the second using manual analysis of less than twenty cases.

The solution proposed in Baykara and Das (2018) chooses the level of interaction the attack should be dealt with based solely on which port it uses.

[22] Sebek– https://projects.honeynet.org/sebek/.

[23] Conpot– http://conpot.org/.

[24] Docker – tool for the building and sharing of containerized applications and microservices - https://www.docker.com/.

[25] TCPDUMP – http://www.tcpdump.org/.

[26] KDDCUP99 – https://kdd.ics.uci.edu/databases/kddcup99/task.html.

[27] PCAP – https://github.com/the-tcpdump-group/libpcap.

[28] J48 - open source Java implementation of the C4.5 algorithm in the Weka data mining tool – http://facweb.cs.depaul.edu/mobasher/classes/ect584/weka/classify.html.

Redirecting itself is mostly done using Honeybrid or by other means usually not supporting transparent redirection of SSH protocols, such as Fan et al. (2016).

The authors of this paper think it is prudent to point out that none of the above-mentioned papers dealing with this issue uses "from the ground up" approach, developing every tool and system used. They are all based to some degree on existing tools and solutions, most commonly Honeybrid and Snort. The approaches to redirecting communication are grouped into groups described above in this chapter.

Based on the analysis, the authors of this paper concluded that none of the solutions fit the requirements (presented in chapter 1 and detailed in chapters 4.2, 4.3 and 4.4) of the process deciding what honeypot to redirect individual sessions. Based on the main author's previous measurements, described in previous papers (Sochor et al., 2016; Zuzcak and Sochor, 2017), it was concluded automated attacks are best identifiable by not only ascertaining the data of a fingerprint[29] of the remote operating system, but also dynamically, by analyzing attacker's activity during the attack. Snort and similar solutions can't detect all the new and especially unknown attacks. Redirecting all correct TCP sessions and deciding only based on the packet payload seemed inefficient. Finally, no solution focused solely on the SSH protocol, which is the focus of this paper.

4. Concept of the proposed hybrid honeynet

This paper brings a new approach to the problem by analysing the behaviour of the attacker alongside the metadata. Based on the analysis in chapter 3, the authors of this paper conclude that none of the examined solutions proposes sufficient inference mechanism for SSH hybrid honeynet. Most of the proposed solutions analyse only packet headers, with those that also include payloads only doing it superficially. Most solutions also have not been tested in a live environment, only in a laboratory setting, or with a small data set.

The approach unique to this solution lies in real-time analysis of the attacker's behaviour, alongside analysing the metadata. Examples of what is meant by behaviour are the number of commands sent per a unit of time, pressing of certain keyboard shortcuts such as CTRL+C, using backspace or delete, what commands are inputted, what is downloaded, what files are copied from the honeypot, how long do certain attacker activities take, etc. Combining these and the metadata allows for more complex analysis. Besides human attacker interacting with SSH shell, it is also able to detect advanced malware. Redirection of the SSH session to the high interaction honeypot is transparent. The solution has been tested on real-world data, with a profile of the most common attack approached derived from it. This paper is building on an existing honeynet run by the main author for several years, composed of multiple honeypots placed in various types of networks, such as academic, commercial, ISP, etc.

4.1. Evaluated SSH sessions

This paper focuses only on SSH connections incoming to the honeypots from any IP address, most commonly via the port 22. The honeypots provide a Linux shell console to the attacker.

An attacker is any device identifiable by an IP address, that connects to a honeypot. In the context of this paper, a session is any SSH connection correctly established between an attacker and a honeypot, and accepted by the honeypot. More specifically, an attacker connects to an SSH port, most commonly the port 22, and

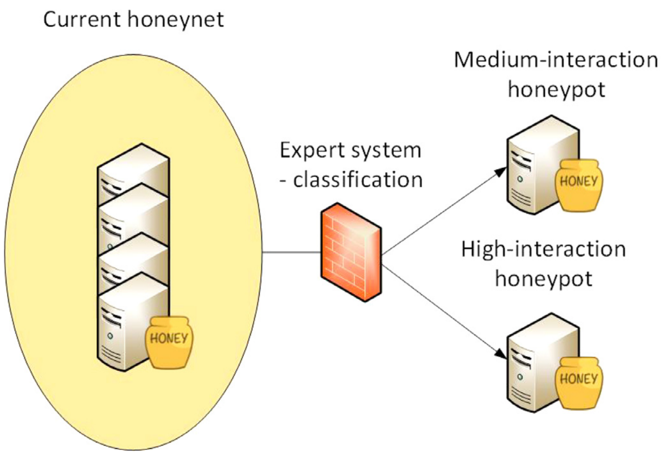

Fig. 1. Schema of the hybrid honeynet using an expert system.

creates an SSH protocol session between the attacker and the given honeypot. The session enables the attacker to interact with the honeypot, to input commands, download and execute files, etc.

4.2. Proposed solution

To realise this concept, an expert system determining where to redirect a given session is used. This is illustrated in Fig. 1. This paper also deals with transparent redirection of SSH sessions from MIHs to HIHs. It does not deal with the software solution used for the HIHs, as this is a part of further research.

When deployed, the expert system will be active on every MIH currently operated by the authors and on other real-life systems, such as servers running web pages of real organisations.

Every attacker first connects to a MIH, for a short amount of time. The attacker's activity there provides the expert system with data based on which it determines whether to redirect the attacker to HIH, whether the attacker is simple or sophisticated. Bot attackers will remain to be observed in the MIH.

4.3. Architecture of the hybrid honeynet

For the implementation of the expert system, EMYCIN[30] expert system type was used, since:

- Only two hypotheses are used, as detailed in chapter 4.4.
- The hypotheses are valid, when suppositions in the form of antecedent-consequent are valid, without the need of fuzzy approach.
- The hypotheses need to express a level of both trust and distrust. For instance, when a file is marked as malware by multiple antiviruses, it increases trust in the presumption that it is a known malware, thus supporting the hypothesis to keep the session at the MIH. However, at the same time, if the file is not marked by the most important, highest rated antiviruses, it decreases trust in the presumption that it is a known malware, as it can also be a new malware only marked by some antiviruses as a false positive, thus indirectly supporting the hypothesis to redirect the session to a HIH.
- The implementation is straightforward, possible in any programming language. It also requires little resources, such as RAM or CPU time, to run its inference engine. All the

[29] Fingerprint – a collection of data on a remote system based on passively acquired data, such as the operating system, running services, etc.

[30] EMYCIN is an expansion of the MYCIN expert system, whose name was derived from antibiotics, as many antibiotics have the suffix "-mycin". While MYCIN was domain dependant, EMYCIN is a domain independent system.

premises are based on exact data represented by rules, such as the number of inputted commands with conjunction with the a priori probability set by an expert, based on previous experience with running the honeynet.

Probability based diagnostic expert systems were based on using modified principles of mathematical probability. They use subjective a priori probability of validity of assertions known as hypotheses. MYCIN and PROSPECTOR were among the prototype probability based expert systems. The expert system proposed in this paper falls under the MYCIN subtype EMYCIN. The MYCIN expert system was a worldwide successful system, used in the medical field. It is used as a basis for other expert systems used for various diagnostic tasks. These solutions are described in articles (Karaci, 2018; Saba et al., 2012; Xie et al., 2015). An effective use of system based on IF-THEN rules can be found in article (Wang et al., 2016).

4.4. Requirements of the expert system

Requirements of the expert system are formulated as follows:

- There are only two hypotheses and each sessions must fit one of them:
  - The sessions should be redirected to a medium interaction honeypot.
  - The sessions should be redirected to a high interaction honeypot.
- The decision making must be based on data provided by the emulated environment, the MIH, such as IP address reputation, country of origin, downloaded malware, etc. These are expressed in numerical values, such as the number of times a given command was inputted into the shell, or how many antiviruses recognise the downloaded malware.
- The decision must be reached quickly, in a matter of a few seconds. The solution must be easily deployable to various Linux based systems with low performance requirements since the MIHs are deployed on mini computers such as the RaspberryPi[31].

4.4.1. Theoretical basis of EMYCIN

The overview of the theoretical basis of EMYCIN is based on the sources (Luo and Zhang, 1999; Luo et al., 1999; Venkata Subba Reddy, 2017; Zhang and Luo, 1999).

The knowledge base is represented by rules in the following form:

$$\langle antecedent \rangle \Rightarrow \langle consequent \rangle \tag{1}$$

expressed by the conditional validity of hypothesis H provided that the evidence E is fulfilled.

$$E \Rightarrow H \tag{2}$$

The uncertainty of a rule is represented by both the level of trust, $MB(H, E)$, and the level of distrust $MD(H, E)$:

$$MB(H,E) = \frac{P(H|E) - P(H)}{1 - P(H)} \tag{3}$$

$$MD(H,E) = \frac{P(H) - P(H|E)}{P(H)} \tag{4}$$

- $P(H)$ a priori probability,
- $P(H|E)$ - conditional probability of validity of H provided that E is fulfilled:

$$P(H|E) = \frac{P(H \cap E)}{P(E)} \tag{5}$$

[31] Mini computer RaspberryPi – https://www.raspberrypi.org/.

If fulfilling E leads to the rise in the level of trust in H:

$$P(HE) > P(H) \tag{6}$$

The level of trust $MB(H, E)$ represents the rise of the probability of H gained using E related to the initial distrust in H:

$$H : 1 - P(H) \tag{7}$$

If fulfilling E leads to decrease of trust in H:

$$P(H) > P(HE) \tag{8}$$

The level of trust $MB(H, E)$ represents the rise of the probability of H gained using E related to the initial trust in H: $P(H)$.

MB and MD have values from the interval $(-1; 1>$. A premise E cannot both support and rebut the same hypothesis, therefore:

$$\text{if } MB > 0, \text{ then } MD \text{ is considered to be } 0, \tag{9}$$

$$\text{if } MD > 0, \text{ then } MB \text{ is considered to be } 0, \tag{10}$$

$$\text{if } MB = MD = 0 \tag{11}$$

then evidence neither supports nor rebuts the hypothesis.

The Certainty Factor CF joins MB and MD into one number, taking on a value from the interval $< -1; 1 >$:

$$CF(H,E) = MB(H,E) - MD(H,E). \tag{12}$$

$$\text{If E increases the level of trust in H, then } CF > 0. \tag{13}$$

$$\text{If E decreases the level of trust in H, then } CF < 0. \tag{14}$$

Every rule of the expert system has a $CF(H, E)$:

$$\text{If } CF > 0, \text{ then } CF = MB(H,E) \tag{15}$$

$$\text{If } CF < 0, \ |\text{CF}| = \text{MD(H,E)} \tag{16}$$

This is only a theoretical example, as in real use, the evidence is rarely fulfilled entirely. The user can subjectively estimate the level of trust of given evidence $CF(H, E')$ based on their observation and experience, which is referred to as *CF_estimate_evidence*. In this paper, mainly in the pseudocode, CF of given evidence, $CF(H, E)$, is referred to as *CF_Evidence*.

Referring to levels of trust and distrust of a hypothesis H as $MB(H, E_1)$ and $MD(H, E_1)$, the overall level of trust or distrust of H can be calculated as:

$$MB(H,E') = (H,E_1) * (max\{0, CF(E_1,E')\}) \tag{17}$$

$$MD(H,E') = (H,E_1) * (max\{0, CF(E_1,E')\}) \tag{18}$$

In this paper, due to ease of implementation, these equations are joined into one while maintaining their meaning as follows: MB and MD are reclassified to M, in the pseudocode it is *M_Evidence*:

- if $M > 0$, then $M = MB$,
- if $M < 0$, then $M = MD$.

therefore, CF of an evidence is defined as:

$$CF(H,E_1) = M(E_1) * max\{0, CF\_estimate(E_1)\} \tag{19}$$

An example is presented in the chapter 4.4.5.

```
Algorithm calculateHypothesis
   for evidence in listOfEvidence:
      CF_Evidence = M_Evidence * max(CF_estimate_Evidence, 0)
      if (CF_Hypothesis * CF_Evidence >= 0):
         CF_Hypothesis = (CF_Hypothesis + CF_Evidence)-(CF_Hypothesis * CF_Evidence)
      else:
         CF_Hypothesis = (CF_Hypothesis + CF_Evidence) / (1 - min(abs(CF_Hypothesis), abs(
             CF_Evidence)))
```

Code 1. Algorithm for calculating the overall level of trust in the validity of a hypothesis.

```
ID Sessions|IP address|SSH client|Termsize|Country|Commands|Times of command inputs|Number of
    inputs|Number of times the malware was detected at VirusTotal.com |Antiviruses with positive
    detection on VirusTotal.com|Passwords used while attempting to log in
```

Code 2. Format of the processed data.

In a situation when multiple pieces of evidence, such as E_1, E_2, E_3 etc., apply to the same H, the overall level of trust in the validity of the hypothesis is calculated as follows:

$$MB(H, E_1 \& E_2) = MB(H, E_1) + MB(H, E_2) - MB(H, E_1) * MB(H, E_2)$$

$$MD(H, E_1 \& E_2) = MD(H, E_1) + MD(H, E_2) - MD(H, E_1) * MD(H, E_2)$$

The overall level of trust in the validity of the hypothesis H, $CF(H, E_1 \& E_2 \& ... \& E_n)$, is calculated as follows:

$$CF(H, E_1 \& E_2) = f(CF(H, E_1), CF(H, E_2)) \quad (20)$$

with f defined as: if $xy \geq 0$:

$$f(x, y) = x + y - x * y \quad (21)$$

else:

$$f(x, y) = \frac{x + y}{1 - min\{abs(x), abs(y)\}} \quad (22)$$

For simplification, $CF(H, E_1 \& E_2 \& ... \& E_n)$ is referred to as $CF\_Hypothesis$ further on.

4.4.2. Implementation of the expert system

The proposed expert system calculates the overall level of trust in the validity of the hypothesis H, $CF\_Hypothesis$, using previously described mathematical and logical approaches, by the following Algorithm 1 written in Python based pseudocode.

An example of a calculation based on the algorithm is in the Section 4.4.5.

4.4.3. Design of the rules

The system is using IF-THEN type rules. The IF part is a supposition and is represented by an evidence E, for instance: “IP address has a bad reputation”. The THEN part represents the hypotheses H, which for the evidence above is: “The connection will be redirected to a medium interaction honeypot”. The rules also have a WITH part, specifying the *MB* or *MD* value.

The expert system itself is implemented in Python 3. Its functionality was verified by feeding it real recorded data via CSV files. Their format is specified in code 2, with three stars, “***” being a separator for attributes with multiple values, such as passwords. The format is used in the Examples 3, 4, 5.

The file contains data for each session representing the connection between the attacker and the honeypot. The expert system evaluates two disjunctive hypotheses:

- H_1: The connection will be redirected to a high interaction honeypot
- H_2: The connection will be redirected to a medium interaction honeypot

The overall level of trust in the validity of a hypothesis is calculated for both H_1 and H_2 by $CF(H, E_1 \& E_2 \& ... \& E_n)$. If the value of CF($H_1$) is higher than the value of CF(H_2) the connection is redirected to a HIH, and if If the value of CF(H_2) is higher than the value of CF(H_1) the connection is redirected to a MIH.

The system uses several suppositions, pieces of evidence E_1, E_2, E_3 etc., along with the level of trust in the validity of the given hypothesis form the rules of the expert system.

4.4.4. Used rules

This chapter presents a representative sample o the rules used in this expert system. Their use is described in the chapters above, mainly in chapter 4.4.1.

If a specific piece of evidence does not influence the trust level of a hypothesis, meaning its certainty factor is zero, $CF\_estimate\_evidence = 0$, due to, for instance, the attacker not downloading any files, then it has no influence on the overall level of trust in the validity of the hypothesis, $CF\_Hypothesis$.

A detailed description of the rules is in the Appendix A. Rules for the hypothesis H_1 are presented in the Tables 6 and 7. Rules for the second hypothesis H_2 are presented in the Tables 8 and 9. A representative sample the rules was selected and their detailed description follows.

- Examples of rules for the hypothesis H_1: The connection will be redirected to a high interaction honeypot.
  - **Evaluation of IP address reputation** (E1_V). An IP address has a positive reputation. The reputation is gained from ReputationAuthority.org that provides a range of $<0;1>$, with 0 meaning no reputation at all and 1 meaning the worst reputation. The range $(0;0.5>$ means, perhaps confusingly, a good reputation, while the range (0.5;1) means bad reputation, the higher the number the worse the reputation is. The expert system modifies this scale, taking everything in range (0;0.5) as 0, meaning good reputation, and projecting the remaining range $(0.5;1>$ to $<0;1>$ range, where for instance 0.5 is projected to zero, 0.6 is projected to 0.2, etc. The projection can also be defined as 2x-1 where x is a value between 0.5 and 1. The only exception is where there is no reputation when its value is 0, as 0 projects to 1, as no reputation at all indicates either new or a sophisticated threat. The resulting projected value is the rule’s CF.
  - **Evaluation of open ports, running services, of the attacker’s system** (E3b_V). Checks if the honeypot detects connection from an IP address where a standard service

is listening, meaning ports such as 80, 22, 443 etc. are opened, and at the same time at least one non-standard service is listening. Every service adds 0.1 to the return value up to a maximum of 1. Database of the Shodan project is used.

  - **Evaluation of commands inputted by the attacker.** (E6_V) If commands that are, based on previous research, typical for human attackers or sophisticated malware are used, every command adds 0.1 to the value up to a maximum of 1. Which commands are typical for a human attacker was determined by analysis described in the article (Sochor et al., 2016). Examples of such commands are history, uname, top, etc.

- Examples of rules for the hypothesis H_2: The connection will be redirected to a medium interaction honeypot.
  - **Evaluation of the SSH client used by an attacker** (E5_S) If the client that established the connection is one of those typically used by bots, such as LibSSH, the CF value is set to 1.
  - **Evaluation of the number of attacks inputted by the attacker** (E6c_S) Evaluates the number of commands inputted in a session. If there were 15 or more identical commands inputted, the value is set to 1. In the range of $(4;14>$, every identical command above 4th adds CF 0.1 to the value up to a maximum of 1.
  - **Evaluation of the time between command inputs** (E9_S) Evaluates times of inputted commands from the start of the session. It calculates the average time per inputted command, by dividing the length of the SSH session by the number of commands. If the average time is less than 1.5 second, the value is set to 1. If the time is in the range of (1.5;2.5) seconds, the initial value at 1.5 is set to 0.9, and each tenth of a second subtracts 0.1 from the value. Over 2.5 seconds the value is 0.

Some of the rules use data from external sources. The reputation of IP addresses is gained from the webpage of ReputatioAauthority.org[32]. The number of positive detections of files as malware is gained from the service of VirusTotal.com. Running services, or open ports, are gained from Shodan[33]. Some of the roles also use an antivirus rating system from AV-Comparatives from December 2018[34], namely ranking antiviruses into groups ADVANCED+ and ADVANCED.

Specific rules for evaluating CF of *E8_S* and *E8_V*. For *E8_S*, if a password attempted for login is shorter than six characters, the attacker is likely a bot trying combinations of short passwords, so it is added to a counter of bot indicating passwords. If a password is longer than six characters, it is compared with a list of the common bot used passwords based on previous research. If the password matches one on the list, one is added to the bot counter. Otherwise, one is added to a counter of human indicating passwords. If the bot counter is more than 20, the CF value is set to 1. Otherwise, the human counter is multiplied by two and divided by the bot counter. If the result is more than 1, the CF value is set to 0. If it is not more than one, it is set to one minus the result. This is done because *E8_V* uses the same value, but subtracted from one. The only exception is when the attacker inputs the correct password with the first attempt, setting the CF value to 0, indicating that human attacker has acquired the password before attacking.

[32] WatchGuard ReputationAuthority – http://www.reputationauthority.org/.
[33] Shodan – https://www.shodan.io/.
[34] AV Comparatives test published in December 2018 – https://www.av-comparatives.org/tests/real-world-protection-test-july-november-2018/.

Table 1
Example of shortened evidence for the hypothesis H_1: The connection will be redirected to a high interaction honeypot.

Evidence	M	CF estimate	CF
E_1	0.3	1	0.3
E_2	0.1	1	0.01
E_3	0.4	0.8	0,32
E_4	−0.5	0.3	−0.15

Table 2
Example of shortened evidence for the hypothesis H_2: The connection will be redirected to a medium interaction honeypot.

Evidence	M	CF estimate	CF
E_1	0.2	1	0.2
E_2	0.5	0.6	0.09
E_3	0.3	1	0.3
E_4	0.62	0.5	0.31

4.4.5. Example of a calculation

This chapter demonstrates a specific calculation based on the equations described in chapters 4.4 and 4.4.1, and on the algorithm described in chapter 4.4.2. Since real calculations contain up to 16 rules, shortened and simplified analogous example is presented here for the sake of simplicity and ease of reading. This example only has 4 rules, but the calculations are still representative of real calculations. A full length example of a real calculation is presented in the Appendix B. An analogous calculation is performed by the expert system for every SSH session connected to a honeypot, based on the data provided to it by Cowrie. There is a certain amount of time between the establishment of the connection and the calculation, to give the attacker some time to act, generating the necessary data for analysis.

Both hypotheses are evaluated and based on the result the attacker will either stay on the MIH (Table 2) or be redirected to a HIH. (Table 1)

The expert system will perform the following calculation to calculate the certainty factor CF of H_1:

$CF(H,E) = M(E)*\max\{0,CF\_estimate(E)\}$

$CF(H_1, E_1) = 0.3*\max\{0;1\} = 0.3$

$CF(H_1, E_2) = 0.1*\max\{0;1\} = 0.01$

$CF(H_1, E_3) = 0.4*\max\{0;0.8\} = 0.32$

$CF(H_1, E_4) = -0.5*\max\{0;0.3\} = -0.15$

$CF_1(H_1, E_1\& E_2) = CF(H_1, E_1) + CF(H_1, E_2) - \{CF(H_1, E_1) * CF(H_1, E_2)\}$

$CF_1(H_1, E_1\& E_2) = 0.3 + 0.01 - \{0.3 * 0.01\} = 0.307$

$CF_2(H_1, CF_1\& E_3) = 0.307+ 0.32 - \{0.307 * 0.32\} = 0.52876$

$CF_3(H_1, CF_2\& E_4) = \frac{0.52876+(-0.15)}{1-\min(|0.52876|,\ |-0.15|)} = \frac{0.37876}{0.85} = 0.4456$

$CF(H_1, E_1\& E_2\& E_3\& E_4) \cong 0.4456$

The overall certainty factor of H_1 has a value of 0.4456.

The expert system will perform the following calculation to calculate the certainty factor CF of H_2:

$CF(H_2, E_1) = 0.2*\max\{0;1\} = 0.2$

$CF(H_2, E_2) = 0.15*\max\{0;0.6\} = 0.09$

$CF(H_2, E_3) = 0.3*\max\{0;1\} = 0.3$

$CF(H_2, E_4) = 0.62*\max\{0;5\} = 0.31$

$CF_1(H_2, E_1\& E_2) = CF(H_2, E_1) + CF(H_2, E_2) - \{CF(H_2, E_1) * CF(H_2, E_2)\}$

$CF_1(H_2, E_1\& E_2) = 0.2 + 0.09 - \{0.2 * 0.09\} = 0.272$

$CF_2(H_2, CF_1\& E_3) = 0.272+ 0.3 - \{0.272 * 0.3\} = 0.4904$

$CF_2(H_1, CF_1\& E_3) = 0.4904+ 0.31 - \{0.4904 * 0.31\} = 0.64838$

$CF(H_2, E_1\& E_2\& E_3\& E_4) \cong 0.64838$

The overall certainty factor of H_2 has a value of 0.64838.

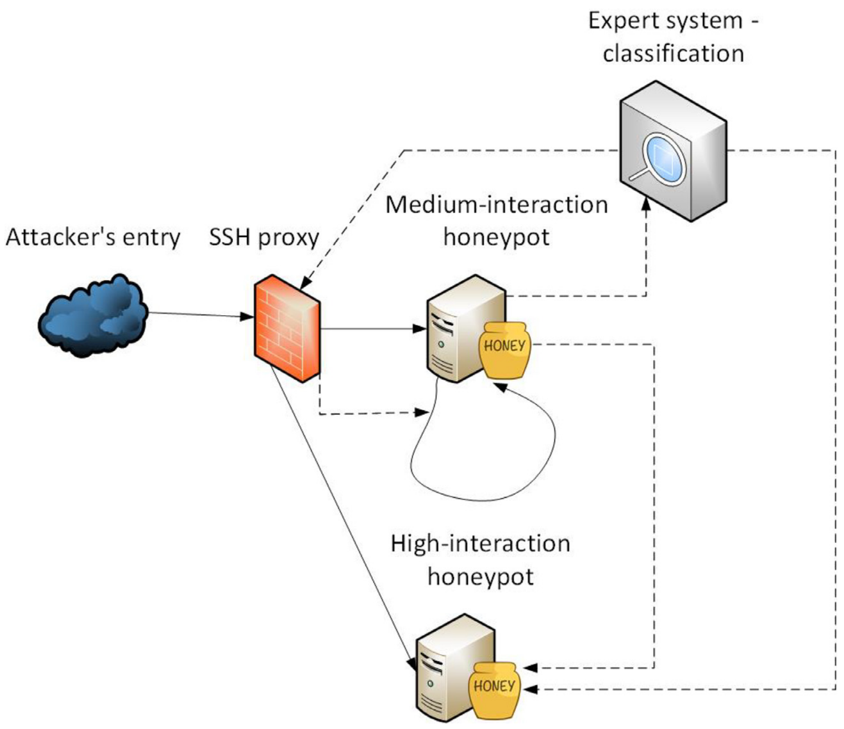
Expert system - classification
Medium-interaction honeypot
Attacker's entry
SSH proxy
HONEY
High-interaction honeypot
HONEY

Fig. 2. Schema of the honeynet with a single MIH. Single MIH is used for the ease of illustration.

Based on these calculations, the expert system decides the given SSH session will be redirected to a MIH, as CF of H_2 is higher than CF of H_1.

5. Implementation of the honeynet

The honeynet is a distributed system. Its nodes are running either on servers or RaspberryPi minicomputers. Every node consists of the following interoperating parts:

- **Cowrie honeypot** - a MIH providing the emulated SSH shell, described in chapters 4.3 and 4.4.5.
- **SSH proxy** - a proxy for transparent redirection of SSH sessions to a HIH in case the expert system evaluates the session as originating from a human attacker.
- **SSH Session Manager** - abbreviated as SSHSM, is a piece of software managing active SSH sessions. It allows the honeynet's administrator to remotely connect to the node and allow them to observe the attacker's activity in real time, increase the time before the decision whether to redirect the attacker is made, redirect them, or disconnect them. It also serves as a bridge between the expert system and the SSH proxy, managing the communication.
- **Expert system** - a system deciding whether to redirect the attacker to a HIH or not, by evaluating data it receives from Cowrie.

A thorough explanation of the interoperation of the parts is presented in the Fig. 2 and expanded further on.

Every SSH session, every attack, is incoming to the SSH proxy, which manages the connection and redirects it to Cowrie. Cowrie provides the attacker with an emulated SSH shell they can interact with, first asking them to enter login and password to the system. After they log in, they can interact with the system, enter commands, download files, etc. Every attacker's action is recorded in two ways. Every inputted command is sent to a central MySQL database, and every keyboard press is stored in a local TTY log[35]. Cowrie also sends every command to the expert system immediately after being inputted, as well as messages informing that the attacker has connected, successfully logged in, disconnected, etc. The main data the expert system needs are summarised in: *ID session, IP address, SSH client, termsize, commands, times of command inputs, number of inputs, attempted passwords*.

When the expert system receives a message that a new attacker has connected to Cowrie, it starts a timer, giving the attacker a certain amount of time to log in. The amount of time is calibrated based on previous experience. If the attacker fails to log in within the allotted time, it is ignored by the expert system not to waste system resources, as there is no further activity to analyse. If the attacker logs in within the allotted time, the first timer is discarded and a second timer starts. This timer gives the attacker a certain amount of time to interact with the system, providing data for the expert system. The amount of time of this timer is also calibrated based on previous experience. After the allotted time passes, the attacker's activity is evaluated by validating the two hypotheses as described in chapter 4.4.3. Based on the result, the attacker is either deemed by the expert system to be redirected to a HIH, or remains on Cowrie. The expert system then sends a message to SSHSM, requesting it to order the SSH proxy to redirect the session to a HIH. At the same time, the expert system also sends a message to a script managing the virtual machines containing HIHs, requesting it to prepare to receive a connection.

After the attacker logs in, the expert system also monitors the session TTY log of the session. If an activity indicative of human behaviour, such as the use of backspace, delete, or the directional arrows, is detected, the timer is terminated and the session is redirected immediately.

Some of the data the expert system needs are gained from external sources, namely ReputationAuthority.org, Shodan.io, and

[35] TTY log – a recording of all activity inside of a terminal in a Unix-like operating systems. http://user-mode-linux.sourceforge.net/old/tty_logging.html.

VirusTotal.com. The expert system also has access to the central database containing where all Cowrie honeypots send their data, and it uses it to get the results from VirusTotal tests on previously downloaded files, to avoid retesting the same file needlessly.

From the moment the session is established, the SSH proxy is sending the SSHSM informative messages. When a session is established, "*REGISTER*" is sent. "*PING*" is sent periodically to signalise a running session. When the attacker disconnects, "*EXIT*" is sent. In the case of a timeout from the attacker, "*TIMEOUT*" is sent.

5.1. Transparent SSH session redirection

The HIH is a full-fledged Linux system running inside a virtual machine. For the honeynet to work effectively, the attacker must be redirected transparently, meaning they must not be able to tell they were redirected.

However, performing a transparent redirection of a running SSH session introduces several problems and inconveniences, compared to a standard honeypot usage. Firstly, the honeynet infrastructure needs to be altered. Second, it is necessary to filter out all effects produced by the redirection, such as extra output messages sent to the attacker by the redirection target.

The architecture of the infrastructure is depicted in Fig. 2. An incoming SSH connection is received by a modified version of an OpenSSH SSH daemon, SSHD, serving as an SSH proxy between the attacker and the Cowrie honeypot. The proxy performs decryption and re-encryption of all traffic exchanged between both sides which implies its ability to log and filter commands sent by the attacker and their output. The proxy connects to the Cowrie honeypot instance through a modified version of the ssh program. Each SSH session is served by an instance of ssh proxy.

One of the challenges hidden behind the transparent redirection was to develop a new or utilise an existing implementation of SSH proxy. Before deciding to modify the SSHD, several projects were examined for promising features.Unfortunately, most of the projects found are based on various Python implementations of the SSH protocol, such as the one integrated into the Twisted framework[36]. Such projects include haas-proxy[37] created by CZ.NIC in their effort to promote honeypots to the broader public and to get a better picture about current threats spreading over SSH or Paramiko[38]. Various SSH implementations, although in accordance with RFC 4251[39], RFC 4252[40], RFC 4253[41] and RFC 4254[42], can be detected by the attacker, revealing them that something is amiss, such as a honeypot is being employed. Since probably most of the servers running Linux use OpenSSH server[43], using the same server as an entry gate to our honeypot infrastructure may make the things more difficult for attackers.

The SSH proxy implementation is based on the sshd_mitm project[44]. The project is barely more than a few scripts and a patch that, when applied to OpenSSH server sources, transforms SSHD into a proxy performing man-in-the-middle attacks on all SSH connection to a given machine. All connections to the target machine are expected to go through attackers via ARP spoofing. The main goal of the project is to collect user credentials.

Although lacking many features needed, sshd_mitm proved a baseline to achieve the required goals. Thus, the necessary features were implemented.

When implementing transparent redirection, two challenges arose:

- how to redirect an instance of the SSH client to another server,
- how to make the attacker unaware of the redirection.

The first one proved to be relatively simple since SSH client relies on file descriptors in order to communicate with the attacker. Data written to stdout or stderr are sent to the attacker. Data read from stdin are received by the SSH instance. Since these descriptors are preserved across the exec operation, the redirection was attempted by restarting the SSH instance with different command-line arguments via the execve() function. This approach proved to work well unless the redirection target is not the same as the current host.

A restart of the SSH client establishes a new connection to the target server. That implies both the client and the server attempt to send some extra data to the attacker, namely:

- information about adding host keys of the target server to known hosts,
- a prompt to enter a user name,
- a prompt to enter a password,
- MOTD,
- last login information,
- initial shell prompt,usually following the user@hostname format.

ssh-mitm suppresses prompts for user name and password and a text informing the attacker that server host keys were added to known hosts. However, it does not prevents outputs from informing about suspicious differences between known hosts and the target server. To resolve this issue with minimal modification to SSHD source code, the code responsible for storing any keys into known hosts was simply removed, thus no suspicious things can be reported, since no keys are ever stored in the file.

The SSHD was modified to suppress all other outputs by using a heuristic approach. When the redirection occurs and the connection to the target is successfully established, including successful authentication, all data sent to the attacker are blocked until a new input is detected from the attacker. Since MOTD, last login information and shell prompt are sent quickly after user login, the attacker sending any input before the sequence finishes is improbable.

Still, one major issue remains: before the redirection from the Cowrie honeypot, the attacker may execute commands changing machine state, and they expect these changes to be visible during the whole session. Such changes include downloading files, creating directories, changing the current directory, etc.

The solution is quite straightforward. After the attacker connects to Cowrie instance, the modified SSHD records all the inputs and replays them to the redirection target after detecting its shell prompt. Such an approach also makes the history of entered commands much more reliable.

6. Testing of the expert system using data gathered by honeynet

Existing data acquired during the tracked time period, August 2017 to July 2018, of the honeynet's run, were used for the development and calibration of the expert system. The data were exported from the central database and provided to the expert system for calibration.

[36] Twisted Framework – https://twistedmatrix.com/trac/.
[37] CZ.NIC Haas-proxy – https://haas.nic.cz/proxy/.
[38] Paramiko – http://www.paramiko.org/.
[39] RFC 4251 – https://tools.ietf.org/html/rfc4251.html.
[40] RFC 4252 – https://tools.ietf.org/html/rfc4252.html.
[41] RFC 4253 – https://tools.ietf.org/html/rfc4253.html.
[42] RFC 4254 – https://tools.ietf.org/html/rfc4254.html.
[43] OpenSSH server – https://www.openssh.com/.
[44] SSH man-in-the-middle tool – https://github.com/jtesta/ssh-mitm.

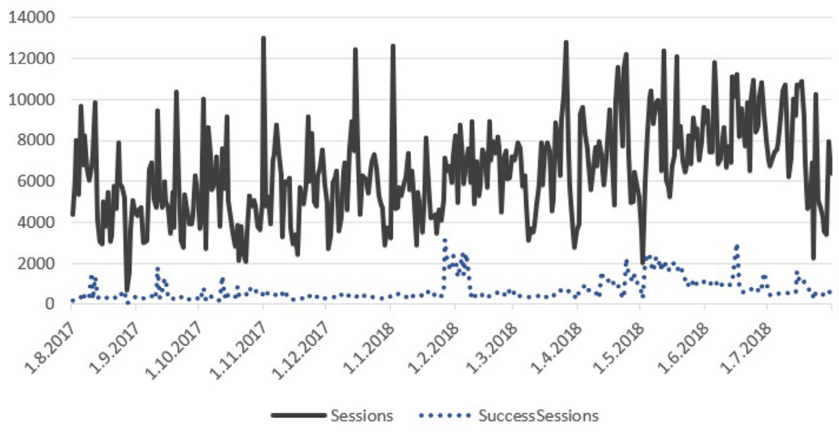

Fig. 3. Number of overall sessions and sessions with successful login over time.

Table 3
Sensors (honeypots making up the honeynet).

Sensor ID	Type of network	Port
HP1	CESNET Czech academic network	22
HP2	Czech VPS hosting grey zone	22
HP3	Common Czech VPS hosting	22
HP4	Czech ISP	22
HP5	Slovak ISP dynamic IP	2222
HP5-B	Slovak ISP dynamic IP	22
HP6	LITNET Lithuanian academic network	22
HP7	VPS hosting - India	22

The honeypots, serving as sensors, send records of all the attacks undertaken against them to a central MySQL[45] database server for further analysis. The composition of the honeynet is presented in the Table 3. Every honeypot runs Cowrie, emulating an SSH server.

Since most of the connections to the honeynet, most of the attacks, so far contained a step downloading and executing a file, such files are analysed by VirusTotal[46].

When deployed, the system uses the same type of data, only the data is generated at the same time as the expert system is running and deciding. It decides after a certain amount of time passes based on all available data that the attack is either sophisticated or simple. An example of such an indicator of a sophisticated attack would be using backspace, while simple attacks may be indicated by running a high number of commands in a very short amount of time. This paper describes the expert system's decision-making mechanism, as well as results of applying it to the data set acquired during the tracked time period.

Only the sessions during which the attacker successfully logged in were used. An overview of the development of the number of sessions directed at the honeynet is presented in the Fig. 3. A more thorough analysis of the data is available in articles (Sochor et al., 2016; Zuzcak and Sochor, 2017). Many other articles have a similar focus of research, such as Vasilomanolakis et al. (2015), Skrzewski (2012) and Fraunholz et al. (2017). Format of the data is described in chapter 4.4.3. The expert system evaluated each recorded fashion just as it would a running one and provided a result saying whether each recorded session would or would not have been redirected to a HIH. The calculations used for the evaluation are described in chapter 4.4.5.

[45] MySQL– https://www.mysql.com/.

[46] VirusTotal.com– http://virustotal.com/.

6.1. Evaluation of the test

Overall, 251 042 sessions were evaluated, with 39 that would have been redirected to a HIH, meaning those attacks were likely performed by humans, or by sophisticated malware. The rest of the attacks would have been left at the MIH, meaning those attacks were likely performed by botnets or script kiddies. Monthly frequency of attacks is presented in Fig. 4.

The vast majority of the sessions, as expected, represented automated attacks. Following are examples of sessions that would have been redirected to a HIH. They generally had some of three objectives:

- **Analysis of files present in the system** - Several valuable looking files were present on the system. In multiple sessions, the attackers copied or manipulated them, the likely goal being stealing possibly valuable files. An example is presented in code 3.
- **Gaining system details** - Multiple attackers used commands giving them system details, likely to verify whether the system is a honeypot, or to evaluate its computational power for other use, such as cryptocurrency mining or botnet spreading. An example is presented in code 4.
- **Using the system for malicious activity** - In some of the sessions, the attackers downloaded software, as seen in code 5, for activities such as cryptocurrency mining. Some have downloaded and ran their own scripts or known malware from GitHub.com and similar webpages. Some were likely starting to prepare the system for future use.

Overall, the test went as expected. The vast majority of attacks were performed by botnets that usually do not scan the system they enter, perhaps besides checking the kernel version. They only execute predefined scripts, most commonly adding the system into the botnet.

Human attackers, or sophisticated malware, usually scan the system much more diligently. There were also cases of probable script kiddies, whos goal was to download a known piece of software and use the system for personal gain.

A part of the attackers tried to "sweep away" their trail, or disable security tools such as IP tables. These operations are, however, sometimes done by bots as well, by loading malicious code into the operating memory that erases downloaded files, erases history, or erases IP tables before executing other scripts.

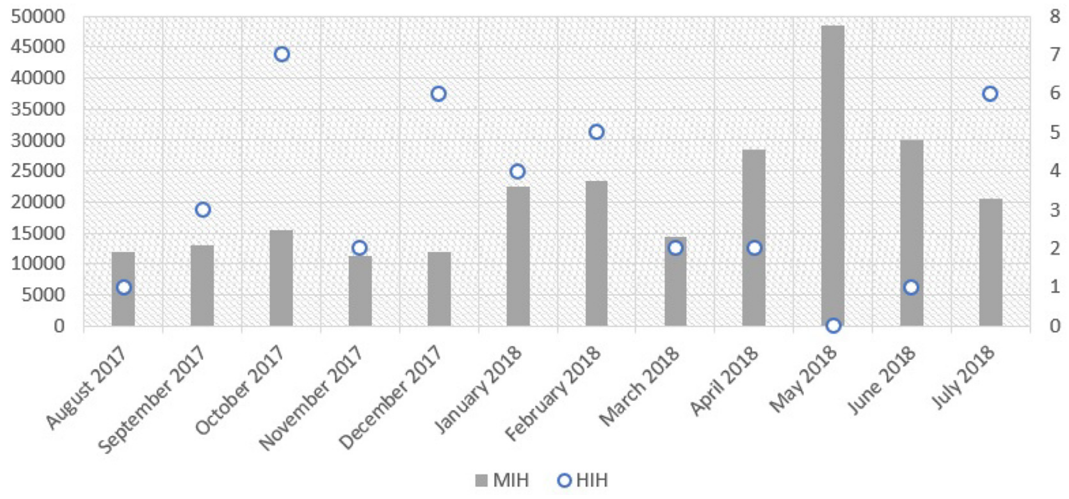

Fig. 4. The number of attacks over the months. The left scale is for the MIHs, the right is for HIHs represented by the circles.

```
d0b87c3524d5|IP|SSH-2.0-PuTTY_Release_0.70|33x97|CZ|ls ***ls ***echo /root/.bash_history /root/.
    bashrc /root/.mysql_history /root/.nano_history /root/.profile /root/.selected_editor /root/
    accountancy_2010.compressed.backup /root/bussinespartners_INTERNAL_DO_NOT_DISTRIBUTE.xls***
    echo ''***gcc***pwd ***ip -a***uname ***uname -a***wget ***man***exit |2017-10-20
    21:27:35***2017-10-20 21:27:38***2017-10-20 21:28:14***2017-10-20 21:28:18***2017-10-20
    21:28:26***2017-10-20 21:28:48***2017-10-20 21:28:51***2017-10-20 21:28:53***2017-10-20
    21:28:56***2017-10-20 21:29:07***2017-10-20 21:29:14***2017-10-20 21:29:17|12|\N|\N|toor
```

Code 3. A session where the likely human attacker wanted to steal the files.

```
fa5f79a46481|IP|SSH-2.0-PuTTY_Release_0.70|24x80|GB|ls ***sudo apt-get update***apt-get update***
    apt-get update***cd ***whoami ***uname -a***uname -rms***lsb_release -rd|2018-01-14
    12:19:13***2018-01-14 12:19:38***2018-01-14 12:19:38***2018-01-14 12:19:59***2018-01-14
    12:20:17***2018-01-14 12:21:39***2018-01-14 12:23:31***2018-01-14 12:23:55***2018-01-14
    12:24:38|9|\N|\N|toor
```

Code 4. A session where the attacker analysed the system they connected to, as suggested by the times of command inputs.

```
8774e846d175|IP|SSH-2.0-PuTTY_Release_0.70|24x80|FR|uname -a***sudo aptitude install git***ls ***
    sudo ***cd ***sudo aptitude install automake***aptitude install automakesudo***ls ***cd ***
    sudo apt-get install libcurl4-openssl-dev libncurses5-dev pkg-config automake yasm git gcc -y
    ***apt-get install libcurl4-openssl-dev libncurses5-dev pkg-config automake yasm git gcc -y***
    git clone https://github.com/pooler/cpuminer.git***cd ***apt-get update|2018-01-14
    15:34:47***2018-01-14 15:36:41***2018-01-14 15:36:46***2018-01-14 15:36:52***2018-01-14
    15:36:54***2018-01-14 15:37:25***2018-01-14 15:37:50***2018-01-14 15:39:04***2018-01-14
    15:44:02***2018-01-14 15:44:05***2018-01-14 15:44:05***2018-01-14 15:45:04***2018-01-14
    15:45:34***2018-01-14 15:48:46|14|\N|\N|toor
```

Code 5. A session where the attacker downloaded software for Litecoin and Bitcoin mining from GitHub.com, likely to mine them on the system.

6.2. A brief analytical and statistical evaluation of the gathered data

The gathered data allows a complex analysis that is going to be a subject of a planned paper. For a basic overview, a selection of basic questions is answered further on:

- Who are the attackers?
- What approach was used to conduct the attack?
- What did the attackers do after breaching the system?

6.2.1. Who are the attackers

From the geographical point of view of the overall number of sessions coming to honeynet, China representing 49.79% (1 173 867 sessions) of the sessions was by far the most common origin of the sessions, followed by the Netherlands with 8.24% (194 176 sessions), France with 6.67% (157 370 sessions), Vietnam with 6.62% (156 027 sessions), and the USA with 6.43% (151 508 sessions). Sessions from all other countries combined represented only 22.26% of the overall number. The authors have previously published a detailed analysis of the relationship between countries and the number of attacks originating in them in article (Zuzčák and Bujok, 2019). Besides basic metrics such as the number of users and computers, other factors, including the enforcement of security principles or the concentration of virtual private server and cloud server services, were shown to be important. These services are easily accessible to users of the entire region, such as Europe, and the companies providing the services often cluster the servers into data centres in a single country, such as the Netherlands, leading to its higher statistical significance as the source of attacks.

When considering only the sessions when the attacker breached the system, successfully logged in, the Netherlands 44.88% (112 800 sessions) was in the lead, followed by China

Table 4
10 most common user names, passwords, and their combinations used by the attackers of the honeynet in the observed timeframe.

username	count	password	count	username+password combinations		count
root	2 828 859	admin	182 223	root	admin	152 950
admin	398 791	root	55 129	admin	root	41 445
test	22 781	123456	46 715	admin	admin	25 720
user	20 984	12345	43 679	root	12345	24 381
support	18 700	password	41 919	admin	password	20 454
pi	15 493	1234	31 764	support	support	16 585
ubnt	12 679	<empty>	26 093	admin	1234	15 383
postgres	10 760	default	20 838	admin	default	14 570
usuario	10 445	ubnt	18 336	admin	12345	14 423
service	9 789	support	17 856	admin	admin123	13 998

14.47% (36 361 sessions), Ukraine 11.50% (28 906 sessions), UK 6.38% (16 034 sessions), and Lithuania 5.57% (14 013 sessions). Sessions from all other countries combined represented only 17.20%. Not all attackers that breached the system actually interacted with it afterwards. When taking only the sessions where the attacker breached the system and then interacted with it, meaning they used console commands, into account, the list of countries with the top five number of sessions was different. It was Poland 59.60% (44 613 sessions) followed by the UK 15.76% (11 794 sessions), France 9.60% (7 187 sessions), the US 3.79% (2 835 sessions), and China 2.22% (1 661 sessions). Sessions from all other countries combined represented only 9.04%. The numbers show the discrepancy between the number of attempted attacks and the attacks that actually breached and did something in the system. Looking at the attacker's IP addresses, they are mostly from dynamic ranges provided by the internet service providers, often placed behind a NAT. Most are likely infected systems, parts of botnets. Another group consists of servers hijacked by attackers and used to spread malware.

Correlating IP addresses the attacks originated from with records from a reputation authority, reputationauthority.org in this case, is analytically interesting as well. Only 37.47% of recorded IP addresses were classified to have a negative reputation, meaning 62.53% of recorded IP addresses were classified to have a good reputation, even though the majority of the attacks originated from them. This indicates that botnets often use victims with dynamically assigned IP addresses, making it difficult for a reputation authority to react to it in a timely manner.

6.2.2. Approach used to conduct the attack

The attacker's approach was rather straight forward, as all the honeypots used user name and password as the authentication method. User name and password combinations were from a set of simple combinations, such as root-toor, to make breaching simple yet not directly effortless. Thus the attacker had to use a dictionary attack at least. The Table 4 contains the overview of ten user name and password combinations most commonly attempted by the attackers in the observed timeframe. As shown, the attackers mostly try out very simple and default login combinations, alongside empty passwords. In the observed timeframe, 83 512 unique passwords and 19 536 unique user names were used.

6.2.3. Attacker activity after breaching the system

The activity of the attackers after breaching the system was varied. Most of the attacks were rather primitive, likely originating from botnets. As mentioned above, such attacks remain on a simple MIH. An overview of the most common inputs is in Table 5. Most commonly detected malware was using the file name "gweerwe323" and was executed by the bot after infection. It is a botnet similar to botnet Mirai or its variant. It erases Mirai if it is on the system (Kambourakis et al., 2017). Following the malware infection, Table 5 presents mainly operations with a file called "nippon". Further on, commands gaining details of the system, such as *ps* and *mount*, and those for masking the attacker's presence, such as modification of "HISTFILE", are present.

Table 5
The most common inputs by successful attackers.

Input
/gweerwe323
uname -a
echo -e \x47\x72\x6f\x70/ > //.nippon
cat //.nippon
rm -f //.nippon
cd /tmp
uname -n -s -r -v
unset HISTORY HISTFILE HISTSAVE HISTZONE HISTORY HISTLOG WATCH
history -n
export HISTFILE=/dev/null

7. Results and further development

Honeypots are a significant part in defence of IT systems. Compared to firewalls and IDS/IPS systems, their goal is not to directly detect or prevent an attack against a live system, but to gather information about attacks, about how to better prevent and how to identify them. This information can then be used by firewalls and IDS/IPS systems to increase their ability to counter such analysed attacks. Specifically, based on honeypot's output, new threat signatures can be identified, specific IP addresses can be blocked, anomalies in the protocols can be mapped, etc. Therefore, honeypots are also an important analytical and scientific tool.

This paper presents a new concept of a hybrid honeynet. Chapter 3 details the existing solutions, leading to the conclusion that none of the analysed solutions provides a viable mechanism for determining the level of sophistication of an attack based on its behaviour and metadata in real-time. An expert system presented in this paper was developed as such a viable mechanism.

Some of the solutions use machine learning methods, such as neural networks to analyse network traffic. However, deployment of such methods requires a learning period, and the data sets used for it were not optimal. The sets were either old or limited in the scope of the data they covered, containing only header and statistical data, or were too small. The expert system was chosen as the parameters for levels of sophistication of attacks can be defined rather precisely. Machine learning, while considered and rejected in favour of an expert system due to the lack of an optimal data set for a machine learning system, will be the next step in the research, using the output of the expert system as a learning data set.

Compared to most of the analysed solutions that were only tested on a small sample of real-world data or in laboratory conditions, this solution has been tested in a real-world deployment. It also presents the most common activities of the attacker in the system after breaching it.

8. Conclusions

This article proposes a new solution for evaluating network traffic using a hybrid SSH honeypot. It analyses approaches of existing solutions and proposes its own, using an expert system based on EMYCIN and rules derived from prior research. The essential points of the article can be summed up as follows: (A) - Currently, there is no solution classifying connections incoming to the honeypot based on their activity into two groups, based on the level of analysis they require. Connections that are potential simple attacks require medium level of interaction, while sophisticated attack require high level of interaction. (B) - An expert system for classifying connections was developed. It utilises two types of data about the attacker. The first type is statistical data, such as the reputation of the attacker's IP address or services running on the attacker's system. The second type is dynamic data, such as their behaviour while interacting with the honeypot, inputted commands, downloaded files, etc. (C) - The proposed system can effectively classify attacks as either simple, requiring only a medium interaction honeypot to analyse, or sophisticated, requiring as real an environment as possible. (D) - The effective classification allows the appropriate analytical and statistical approach to be employed, saving system resources. (E) - The system can process most tasks automatically, lessening the operator's burden. (F) - The developed honeynet can create highly transparent, nearly realistic redirection of SSH sessions, unlike any implemented in the analysed honeynets. The honeynet is being continuously developed and upgraded. The current development focus is on effective monitoring of an attacker within a high interaction honeypot.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Matej Zuzčák: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing - original draft. **Milan Zenka:** Software, Writing - review & editing.

Acknowledgment

Our gracious thanks to Martin Dráb of Excello s.r.o company for assistance with the transparent redirection of SSH sessions, chapter 5.1.

Appendix A. A complete list of rules used by the expert system

Table 6
Evidence for the hypothesis H_1: The connection will be redirected to a high interaction honeypot. PART 1.

IF	Then	With	CF - scale	Description of CF estimate
E1_V	H1	0.3	< 0;1 >	An IP address has a positive reputation. The reputation is gained from ReputationAuthority.org that provides a range of < 0;1 >, with 0 meaning no reputation at all, 1 meaning the worst reputation, the range (0;0.5 > meaning, perhaps confusingly a good reputation, and the range (0.5;1) meaning bad reputation, the higher the number the worse the reputation is. The expert system modifies this scale, taking everything in range (0;0.5) as 0, meaning good reputation, and projecting the remaining range (0.5;1 > to < 0;1 > range, where for instance 0.5 is projected to zero, 0.6 is projected to 0.2, etc. The only exception is where there is no reputation when its value is 0, as 0 projects to 1, as no reputation at all indicates either new or a sophisticated threat.
E2_V	H1	0.1	[0;1]	If a honeypot detects connection from an IP address for the first time, it is set to 1. It uses a local database of all IP addresses that connected to the honeypot. These addresses are not centrally shared by design.
E2b_V	H1	-0.5	< 0;1 >	Checks if a honeypot detects connection from an IP address repeatedly. Based on the number of connections stored in the local database, it adds 0.1 to the value for each connection, up to a maximum of 1.
E3_V	H1	0.3	< 0;1 >	Checks If the honeypot detects connection from an IP address where no service is listening, meaning no ports are open, or where non-standard service, such as opened high number port, is running. It uses the Shodan database that maps running services on IP addresses. No service gives the value of 0.1 with every non-standard service adding additional 0.1 up to a maximum of 1.
E3b_V	H1	0.1	(0;1)	Checks if the honeypot detects connection from an IP address where a standard service is listening, meaning ports such as 80, 22, 443 etc. are opened, and at the same time at least one non-standard service is listening. Every service adds 0.1 to the return value up to a maximum of 1.
E4_V	H1	0.2	[0;1]	If a honeypot detects connection from an IP address from a country that does not belong among the countries with the highest recorded number of automated attacks, the value is set to 1.
E4b_V	H1	0.2	[0;1]	If the client that established the connection has a defined termsize, meaning the user has opened the client in a window, the value is set to 1.
E5_V	H1	0.3	[0;1]	If the client that established the connection is one of those typically used by human attackers, such as Putty or OpenSSH, the value is set to 1.
E6_V	H1	0.7	< 0;1 >	If commands that are, based on previous research, typical for human attackers or sophisticated malware are used, every command adds 0.1 to the value up to a maximum of 1.
E6b_V	H1	0.5	(0;1)	Evaluates the number us unique commands inputted in a session. If the commands are chained by operators such as ||, &&, or a ";" every command in a chain is counted separately. Every unique command adds 0.1 to the value up to a maximum of 1.

Table 7
Evidence for the hypothesis H_1: The connection will be redirected to a high interaction honeypot. PART 2.

IF	Then	With	CF - scale	Description of CF estimate
E6c_V	H1	-0.9	[0;1]	If only one input was sent by the SSH client, and the client is one of those commonly used by bots, the value is set to 1. The rule is based on experience observing bot behaviour.
E7_V	H1	0.4	[0;1]	If the attacker downloads a file that, when tested by VirusTotal.com, is not flagged as malware by any antiviruses, the value is set to 1.
E7b_V	H1	0.3	< 0;1 >	Checks if the attacker downloads a file that, when tested by VirusTotal.com, is flagged as malware by less than three antiviruses. If none flagged it, the value is set to 0. If one flagged it, the value is 0.7, and if two flagged it, the value if 0.5.
E7c_V	H1	0.2	< 0;1 >	Checks if the attacker downloads a file that, when tested by VirusTotal.com, is flagged as malware by less than three antiviruses, but specifically those not in the lists of ADVANCED+ and ADVANCED according to the AV-Comparatives. If none flagged it, the value is set to 0. If one flagged it, the value is 0.5, and if two flagged it, the value if 1.
E7d_V	H1	-0.5	(0;1)	Checks if the attacker downloads a file that, when tested by VirusTotal.com, is flagged as malware by more than two antiviruses. The value is set on the scale < 0.3;1 > , starting at the value of 3 at three detections, adding 0.1 for every other up to a maximum of 1.
E8_V	H1	0.4	(0;1)	Described in 4.4.4.
E9_V	H1	0.4	< 0;1 >	Evaluates times of inputted commands from the start of the session. It calculates the average time per inputted command, by dividing the length of the SSH session by the number of commands. If the average time is higher than 3.5 seconds, it indicates a human and the value is set to 1. If the time is in the range of < 2.5;3.5) seconds, each tenth of a second adds 0.1 to the value. Under 2.5 seconds indicates a bot, thus the value is 0.
E10_V	H1	0.7	[0;1]	Evaluates if the attacker is human by checking if human specific keyboard strokes are detected. The keys are directional arrows, delete, and backspace.

Table 8
Evidence for the hypothesis H_2: The connection will be redirected to a medium interaction honeypot. PART 1.

IF	Then	With	CF - scale	Description of CF estimate
E1_S	H2	0.8	< 0;1 >	The IP address has a negative reputation. How the reputation value is gained and processed is explained in the first line of Table 6. This evidence uses the same value.
E2_S	H2	0.5	< 0;1 >	Checks if a honeypot detects connection from an IP address repeatedly. Based on the number of connections stored in the local database, it adds 0.1 to the value for each connection, up to a maximum of 1.
E2b_S	H2	-0.2	[0;1]	If a honeypot detects connection from an IP address for the first time, it is set to 1. It uses a local database of all IP addresses that connected to the honeypot. These addresses are not centrally shared by design.
E3_S	H2	0.6	< 0;1 >	Checks if the honeypot detects connection from an IP address where a standard service is listening, meaning ports such as 80, 22, 443 etc. are opened. Every service adds 0.2 to the value for each service, up to a maximum of 1.
E3b_S	H2	0.2	[0;1]	If a honeypot detects communication from an IP address from a country belonging to the countries with the highest recorded number of automated attacks, the value is set to 1.
E4_S	H2	0.2	[0;1]	If the client that established the connection does not have a defined termsize, thus it was not run by a user in a window, the value is set to 1.
E5_S	H2	0.4	[0;1]	If the client that established the connection is one of those typically used by bots, such as LibSSH, the value is set to 1
E6_S	H2	0.7	< 0;1 >	If commands that are, based on previous research, typical for bot attackers are used, every command adds 0.1 to the value up to a maximum of 1.
E6b_S	H2	0.5	(0;1)	Evaluates the number of commands inputted in a session. If the commands are chained by operators such as ||or &&, every command in a chain is counted separately. If there are then 60 commands, the value is set to 1. In the range (20;59), each command above the 20th adds 0.025 to the value. If there are less than 20 commands, the value is set to 0.
E6c_S	H2	0.5	< 0;1 >	Evaluates the number of commands inputted in a session. If there were 15 or more identical commands inputted, the value is set to 1. In the range of (4;14 > , every identical command above 4th adds 0.1 to the value up to a maximum of 1.

Table 9
Evidence for the hypothesis H_2: The connection will be redirected to a medium interaction honeypot. PART 2.

IF	Then	With	CF - scale	Description of CF estimate
E7_S	H2	0.6	< 0;1 >	Checks if the attacker downloads a file that, when tested by VirusTotal.com, is flagged as malware by more than five antiviruses. If 5 to 25 antiviruses flagged it, the value is increased by 0.04 for each above 5. If one flagged it, the value is 0.7, and if two flagged it, the value if 0.5, up to the value up to a maximum of 1.
E7b_S	H2	-0.3	< 0;1 >	Checks if the attacker downloads a file that, when tested by VirusTotal.com, is flagged as malware by less than three antiviruses rated as ADVANCED+ and ADVANCED. If none flagged it, the value is set to 1. If one flagged it, it is 0.5, if two it is 0.3 and if three it is 0.
E8_S	H2	0.4	(0;1)	Described in 4.4.4.
E9_S	H2	0.7	(0;1)	Evaluates times of inputted commands from the start of the session. It calculates the average time per inputted command, by dividing the length of the SSH session by the number of commands. If the average time is less than 1.5 second, the value is set to 1. If the time is in the range of (1.5;2.5) seconds, the initial value at 1.5 is set to 0.9, and each tenth of a second subtracts 0.1 from the value. Over 2.5 seconds the value is 0.
E9b_S	H2	0.9	[0;1]	If all the commands were inputted within one second, a bot attacker is strongly indicated, thus the value is 1.

Appendix B. A complex example of the expert system's calculation, based on which specific attacks will be redirected to either a HIH or a MIH

In the following example, the input is represented by the data in 6. Evaluation of individual rules is presented in the Table 10 for H_1 and in the Table 11 for H_2.

The expert system will perform the following calculation to calculate the certainty factor CF of H_1:

Table 10
Evaluation of evidence, based on data gathered by a honeypot, for the hypothesis H_1: The connection will be redirected to a high interaction honeypot.

Evidence	M	CF estimate	CF
E_1	0.3	1	0.3
E_2	0.1	1	0.01
E_{2b}	−0.5	0	0
E_3	0.3	0.1	0.03
E_{3b}	0.1	0	0
E_4	0.2	0	0
E_{4b}	0.2	0	0
E_5	0.3	0	0
E_6	0.7	0	0
E_{6b}	0.5	0	0
E_{6c}	−0.9	0	0
E_7	0.4	0	0
E_{7b}	0.3	0	0
E_{7c}	0.2	0	0
E_{7d}	−0.5	0.3	−0.15
E_8	0.4	1	0.32
E_9	0.4	1	0.4
E_{10}	0.7	0	0

Table 11
Evaluation of evidence, based on data gathered by a honeypot, for the hypothesis H_2: The connection will be redirected to a medium interaction honeypot.

Evidence	M	CF estimate	CF
E_1	0.8	0	0
E_2	0.5	0	0
E_{2b}	−0.2	1	−0.2
E_3	0.6	0	0
E_{3b}	0.2	1	0.2
E_4	0.2	1	0.2
E_5	0.4	1	0.4
E_6	0.7	0.3	0.21
E_{6b}	0.5	0	0
E_{6c}	0.5	0	0
E_7	0.6	0	0
E_{7b}	−0.3	0	0
E_8	0.4	0.02	0.08
E_9	0.7	0	0
E_{9b}	0.9	0	0

CF(H,E) = M(E)*max{0,CF_estimate(E)}
$CF(H_1, E_1) = 0.3*max\{0;1\} = 0.3$
$CF(H_1, E_2) = 0.1*max\{0;1\} = 0.01$
$CF(H_1, E_{2b}) = -0.5*max\{0;0\} = 0$
$CF(H_1, E_3) = 0.3*max\{0;0.1\} = 0.03$
$CF(H_1, E_{3b}) = 0.1*max\{0;0\} = 0$
$CF(H_1, E_4) = 0.2*max\{0;0\} = 0$
$CF(H_1, E_{4b}) = 0.2*max\{0;0\} = 0$
$CF(H_1, E_5) = 0.3*max\{0;0\} = 0$
$CF(H_1, E_6) = 0.7*max\{0;0\} = 0$
$CF(H_1, E_{6b}) = 0.5*max\{0;0\} = 0$
$CF(H_1, E_{6c}) = -0.9*max\{0;0\} = 0$
$CF(H_1, E_7) = 0.4*max\{0;0\} = 0$
$CF(H_1, E_{7b}) = 0.3*max\{0;0\} = 0$
$CF(H_1, E_{7c}) = 0.2*max\{0;0\} = 0$
$CF(H_1, E_{7d}) = -0.5*max\{0;0.3\} = -0.15$
$CF(H_1, E_8) = 0.4*max\{0;0.8\} = 0.32$
$CF(H_1, E_9) = 0.4*max\{0;1\} = 0.4$
$CF(H_1, E_{10}) = 0.7*max\{0;0\} = 0$
$CF_1(H_1, E_1\& E_2) = CF(H_1, E_1) + CF(H_1, E_2) - \{CF(H_1, E_1) * CF(H_1, E_2)\}$
$CF_1\ H_1, E_1\& E_2) = 0.3 + 0.01 - \{0.3 * 0.01\} = 0.307$
$CF_2\ (H_1, CF_1\& E_{2b}) = 0.307+ 0 - \{0.307 * 0\} = 0.307$
$CF_3\ (H_1, CF_2\& E_3) = 0.307+ 0.03 - \{0.307 * 0.03\} = 0.32779$
$CF_{4..13}\ (H_1, CF_{3..12}\& E_{3\ b..7c}) = 0.32779 + 0 - \{0.32779 * 0\} = 0.32779$
$CF_{14}\ (H_1,\ CF_{13}\&\ E_{7d}) = \frac{0.32779+(-0.15)}{1-\min(|0.32779|,\ |-0.15|)} = \frac{0.17779}{0.85} = 0.20916$
$CF_{15}\ (H_1, CF_{14}\& E_8) = 0.20916 + 0.32 - \{0.20916 * 0.32\} = 0.64$
$CF_{16}\ (H_1, CF_1\& E_9) = 0.64 + 0.4 - \{0.64 * 0.4\} = 0.8$
$CF(H_1, E_1\& E_2\& E_{2b}\& E_3\& E_{3b}\& E_4\& E_{4b}\& E_5\& E_6\ \& E_{6b}\& E_{6c}\& E_7\& E_{7b}\& E_{7c}\& E_{7d}\& E_8\& E_9\& E_{10}) \cong 0.8$

The overall certainty factor of H_1 has a value of 0.8.

The expert system will perform the following calculation to calculate the certainty factor CF of H_2:

$CF(H_2, E_1) = 0.8*max\{0;0\} = 0$
$CF(H_2, E_2) = 0.5*max\{0;0\} = 0$
$CF(H_2, E_{2b}) = -0.2*max\{0;1\} = -0.2$
$CF(H_2, E_3) = 0.6*max\{0;0\} = 0$
$CF(H_2, E_{3b}) = 0.2*max\{0;1\} = 0.2$
$CF(H_2, E_4) = 0.2*max\{0;1\} = 0.2$
$CF(H_2, E_5) = 0.4*max\{0;1\} = 0.4$
$CF(H_2, E_6) = 0.7*max\{0;0.3\} = 0.21$
$CF(H_2, E_{6b}) = 0.5*max\{0;0\} = 0$
$CF(H_2, E_{6c}) = 0.5*max\{0;0\} = 0$
$CF(H_2, E_7) = 0.6*max\{0;0\} = 0$
$CF(H_2, E_{7b}) = -0.3*max\{0;0\} = 0$
$CF(H_2, E_8) = 0.4*max\{0;0.2\} = 0.08$
$CF(H_2, E_9) = 0.7*max\{0;0\} = 0$
$CF(H_2, E_{9b}) = 0.9*max\{0;0\} = 0$
$CF_1\ (H_2, E_1\& E_2) = 0 + 0 - \{0* 0\} = 0$

```
52c52970ee3d|XXXX.XXXX.XXXX.XXXX|SSH-2.0-libssh2_1.4.2|\N|CN|ln -sf /usr/sbin/sshd /tmp/su***
/tmp/su -oPort=1987***service iptables stop***cd /tmp***wget http://mdb7.cn:8081/exp|
2018-07-26 15:12:20***2018-07-26 15:12:20***
2018-07-26 15:12:33***2018-07-26 15:12:37***2018-07-26 15:12:39|5|3|Avast***Symantec***Kaspersky|
    alison$1***h%7^2g@1***dunquirk***vocalist***123456
```

Code 6. An example of data gathered during an SSH session, provided to the expert system. The data are used in the complex calculation example.

$CF_2\ (H_2,\ CF_1\&\ E_{2b}) = \frac{0+(-0,2)}{1-\min(|0|,\ |-0,2|)} = \frac{-0,2}{1} = -0.2$

$CF_{3\&4}\ (H_2,\ CF_2\&\ E_{3\&3b}) = \frac{-0.2+(0,2)}{1-\min(|-0,2|,|0,2|)} = \frac{0}{0.8} = 0$

$CF_5\ (H_2,\ CF_4\&\ E_4) = 0 + 0.2 - \{0 * 0.2\} = 0.2$

$CF_6\ (H_2,\ CF_5\&\ E_5) = 0.2 + 0.4 - \{0.2 * 0.4\} = 0.52$

$CF_7\ (H_2,\ CF_6\&\ E_6) = 0.52 + 0.21 - \{0.52 * 0, 21\} = 0.6208$

$CF_{8..11}\ (H_2,\ CF_{7..10.}\&\ E_{6b..7b}) = 0.6208 + 0 - \{0.6208 * 0\} = 0.6208$

$CF_{12}\ (H_2,\ CF_{11}\&\ E_8) = 0.6208 + 0.08 - \{0.6208 * 0.08\} = 0.65114$

$CF_{13..14}\ (H_2,\ CF_{12..13.}\&\ E_{9..9b}) = 0.65114 + 0 - \{0.65114 * 0\} = 0.65114$

$CF(H_2,\ E_1\&\ E_2\&\ E_{2b}\&\ E_3\&\ E_{3b}\&\ E_4\&\ E_5\&\ E_6\&\ E_{6b}\&\ E_{6c}\&\ E_7\&\ E_{7b}\ \&E_8\&E_9\&E_{9b}) \cong 0.65114$

The overall certainty factor of H_2 has a value of 0.65114.

Based on these calculations, the expert system decides the given SSH session will be redirected to a HIH, as CF of H_1 is higher than CF of H_2.

References

Abbasi, F.H., Harris, R.J., 2009. Experiences with a generation III virtual honeynet, australasian telecommunication networks and applications conference. In: ATNAC 2009 - Proceedings, art. no. 5464785.

Altman, N.S., 1992. An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. 46 (3), 175–185.

Ansiry Zakaria, W.Z., Kiah, M.L.M., 2012. A review on artificial intelligence techniques for developing intelligent honeypot. In: Proceedings - 2012 8th International Conference on Computing Technology and Information Management. ICCM, pp. 696–701. 2012, 2, art. no. 6268588

Artail, H., Safa, H., Sraj, M., Kuwatly, I., Al-Masri, Z., 2006. A hybrid honeypot framework for improving intrusion detection systems in protecting organizational networks. Comput. Secur. 25 (4), 274–288.

Bailey, M., Cooke, E., Watson, D., Jahanian, F., Provos, N., 2004. A hybrid honeypot architecture for scalable network monitoring. In: Technical report. University of Michigan and Google Inc, pp. 1–18.

Balas, E., Viecco, C., 2005. Towards a third generation data capture architecture for honeynets. In: Proceedings from the 6th Annual IEEE System, Man and Cybernetics Information Assurance Workshop. SMC, pp. 21–28. 2005, art. no. 1495929

Bao, N.K., Ahn, S.W., Park, M., 2018. An elastic-hybrid honeynet for cloud environment. In: Nagamalai, D., et al. (Eds.), CSEIT, NCS, SPM, NeTCoM, p. 117127.

Baykara, M., Das, R., 2018. A novel honeypot based security approach for real-time intrusion detection and prevention systems. Journal of Information Security and Applications 41, 103–116. ISSN 2214-2126

Benzekki, K., El Fergougui, A., Elbelrhiti Elalaoui, A., 2016. Software defined networking (SDN): a survey. Security Comm Networks 9, 5803–5833.

Chawda, K., Patel, A.D., 2015. Dynamic & hybrid honeypot model for scalable network monitoring. In: 2014 International Conference on Information Communication and Embedded Systems. ICICES. 2014, art. no. 7033844

Chovancova, E., Adam, N., Balaz, A., Pietrikova, E., Fecilak, P., Simonak, S., Chovanec, M., 2017. Securing distributed computer systems using an advanced sophisticated hybrid honeypot technology. Comput. Inf. 36 (1), 113–139.

Cortes C., Vapnik V.. Mach learn 20: 273. 1995.

Dornseif, M., Holz, T., Klein, C.N., 2004. NoSEBreak - attacking honeynets, proceedings fron the fifth annual IEEE system. In: Man and Cybernetics Information Assurance Workshop. SMC, pp. 123–129. Art. no. 450

Fan, W., Du, Z., Fernández, D., Hui, X., 2016. Dynamic hybrid honeypot system based transparent traffic redirection mechanism. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9543, pp. 311–319.

Fan, W., Fernandez, D., 2017. A novel SDN based stealthy TCP connection handover mechanism for hybrid honeypot systems. In: 2017 IEEE Conference on Network Softwarization: Softwarization Sustaining a Hyper-Connected World: en Route to 5G. NetSoft. 2017, art. no. 8004194

Fan, W., Fernández, D., Du, Z., 2015. Adaptive and flexible virtual honeynet. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9395, pp. 1–17.

Fraunholz, D., Krohmer, D., Anton, S.D., Dieter Schotten, H., 2017. Investigation of cyber crime conducted by abusing weak or default passwords with a medium interaction honeypot. In: 2017 International Conference on Cyber Security And Protection Of Digital Services. Cyber Security 2017, art. no. 8074855.

Grudziecki, T., Jacewicz, P., Juszczyk Å, ., Kijewski, P., Pawliński, P., ENISA, 2012. Proactive Detection of Security Incidents Honeypots. ENISA publication, Greece.

Hayati, P., Chai, K., Potdar, V., Talevski, A., 2009. Honeyspam 2.0: Profiling web spambot behaviour. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5925. LNAI, pp. 335–344.

Innab, N., Alomairy, E., Alsheddi, L., 2018. Hybrid system between anomaly based detection system and honeypot to detect zero day attack. In: 21st Saudi Computer Society National Computer Conference. NCC, Riyadh, pp. 1–5.

Joshi, C.R., Sardana, A., 2011. Honeypots A New Paradigm to Information Security. Science Publishers, USA.

Kambourakis, G., Kolias, C., Stavrou, A., 2017. The mirai botnet and the iot zombie armies. In: MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM). Baltimore, MD, pp. 267–272.

Karaci, A., 2018. Intelligent tutoring system model based on fuzzy logic and constraint-based student model. Neural Comput. Appl. 1–10.

Kumar, S., Sehgal, R., Bhatia, J.S., 2012. Hybrid honeypot framework for malware collection and analysis. In: IEEE 7th International Conference on Industrial and Information Systems. ICIIS. 2012, art. no. 6304786

Kyaw, K.L., 2008. Hybrid honeypot system for network security. world academy of science. Eng. Technol. 2 (12), 232236.

Ligh Hale, M., Adair, S., Hartstein, B., Matthew, R., 2011. Malware Analyst's Cookbook and DVD - Tools and Techniques for Fighting Malicious Code. Wiley Publishing, Inc, USA.

Luo, X., Zhang, C., 1999. Proof of the correctness of EMYCIN sequential propagation under conditional independence assumptions. IEEE Trans. Knowl. Data Eng. 11 (2), 355–359.

Luo, X., Zhang, C., Leung, H.F., 1999. A class of isomorphic transformations for integrating EMYCIN-style and PROSPECTOR-style systems into a rule-based multi--agent system. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1733, pp. 211–225.

Mansoori, M., Zakaria, O., Gani, A., 2012. Improving exposure of intrusion deception system through implementation of hybrid honeypot. Int. Arab J. Inf.Technol. 9 (5).

Project, T.H., 2003. Know Your Enemy: Sebek A Kernel Based Data Capture Ttool. Honeynet.org.

Provos, N., Holz, T., 2007. Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison Wesley Professional, USA.

Qiao, P.L., Hu, S.S., Zhai, J.Q., 2013. Design and implementation of dynamic hybrid honeypot network. In: Proceedings of SPIE - The International Society for Optical Engineering, 8752. Art. no. 87520L

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.

Quynh, A.N., Takefuji, Y., 2005. A novel stealthy data capture tool for honeynet system. In: Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, pp. 207–212.

Saba, T., Al-Zahrani, S., Rehman, A., 2012. Expert system for offline clinical guidelines and treatment. Life Sci. J. 9 (4), 2639–2658.

Sadasivam G., K., Hota, C., Anand, B., 2017. Detection of severe SSH attacks using honeypot servers and machine learning techniques. In: Software Networking. River Publishers, pp. 79–100.

Skrzewski, M., 2012. Network malware activity a view from honeypot systems. In: KwieceÅ, A., Gaj, P., Stera, P. (Eds.), Computer Networks, Communications in Computer and Information Science. Cham: Springer International Publishing, pp. 198–206.

Sochor, T., Zuzcak, M., Bujok, P., 2016. Statistical analysis of attacking autonomous systems: Detailed statistics of attackers against linux honeynet. In: International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2016.

Spotzner, L., 2002. Honeypots: Tracking Hackers. Addison Wesley Longman Publishing Co., Inc., USA.

Vasilomanolakis, E., Karuppayah, S., Kikiras, P., Mühlhäuser, M., 2015. A honeypot–driven cyber incident monitor: lessons learned and steps ahead. In: Proceedings of the 8th International Conference on Security of Information and Networks, pp. 158–164.

Venkata Subba Reddy, P., 2017. Fuzzy logic based on belief and disbelief membership functions. Fuzzy Inf. Eng. 9 (4), 405–422.

Wang, H., Wu, B., 2019. SDN-based hybrid honeypot for attack capture. In: IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference. ITNEC, Chengdu, China, pp. 1602–1606.

Wang, O., Liberti, L., D'Ambrosio, C., Marie, C.S., Ke, C., 2016. Controlling the average behavior of business rules programs. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9718, pp. 83–96.

Wang, Z., Li, G., Chi, Y., Zhang, J., Liu, Q., Yang, T., Zhou, W., 2019. Honeynet construction based on intrusion detection. In: Proceedings of the 3rd International Conference on Computer Science and Application Engineering, CSAE 2019. ACM. 80:1–80:5

Wicherski, G., Berthier, R., 2009. Honeybrid: hybrid honeypot framework. Software Project - Google Summer of Code, The Honeynet Project.

Xie, N., Han, Y., Z., L., 2015. A novel approach to fuzzy soft sets in decision making based on grey relational analysis and MYCIN certainty factor. Int. J. Comput. Intell.Syst. 8 (5), 959–976.

Zhang, C., Luo, X., 1999. Isomorphic transformations of uncertainties for incorporating EMYCIN-style and PROSPECTOR-style systems into a distributed expert system. J. Comput. Sci. Technol. 14 (4), 386–392.

Zuzcak, M., Sochor, T., 2017. Behavioral analysis of bot activity in infected systems using honeypots. Commun. Comput. Inf. Sci. 718, 118–133.

Zuzčák, M., Bujok, P., 2019. Causal analysis of attacks against honeypots based on properties of countries. IET Information Security.

Dr. Matej Zuzčák attained his Mgr. and RNDr. degrees in Information Systems from Department of Informatics and Computers of the Faculty of Science at University of Ostrava in 2016 and 2017. Since 2016 he has been working on his dissertation thesis within his Ph.D study at the Department of Informatics and Computers. His scientific research is focused mainly on honeypots, honeynets, network security, expert systems and data analysis. Since 2017 he is the team leader of CSIRT OU. He is also a member of The Honeynet Project in Czech chapter.

Milan Zenka attained his Mgr. degree in Information Systems from Department of Informatics and Computers of the Faculty of Science at University of Ostrava in 2017. He has been working towards attaining Ph.D at the Department of Informatics and Computers since 2017. His research is based on development of Windows based IDS. He has been a member of CSIRT OU since 2017.