How to Mock a Bear: Honeypot, Honeynet, Honeywall & Honeytoken: A Survey

Paul Lackner
Institute of IT Security Research, St. Pölten University of Applied Sciences, Austria

Keywords: Honeypot, Honeynet, Honeywall, Honeytoken, Survey.

Abstract: In a digitized world even critical infrastructure relies on computers controlled via networks. Attacking these sensitive infrastructures is highly attractive for intruders, who are frequently a step ahead of defenders. Honey systems (honeypots, honeynets, honeywalls, and honeytoken) seek to counterbalance this situation. Honey systems trap attackers by generating phoney services, nets, or data, thereby preventing them from doing damage to production systems and enable defenders to study attackers without letting intruders initially notice. This paper provides an overview of existing technologies, their use cases, and pitfalls to bear in mind by illustrating various examples. Furthermore, it shows the recent efforts made in the field and examines the challenges that still need to be solved.

1 INTRODUCTION

Due to continuously improving endpoint protection and network protection, attacks against computer systems are becoming more complex. Various new attack vectors are found continuously and multiple system components, hosts, and services in combination are necessary to successfully attack a system (Simmons et al., 2014) (Papp et al., 2015). Automated attacks based in machine learning are increasing, which reveals the need for adjusted defense methods (Bland et al., 2020) (Cui et al., 2020). To learn new techniques from attackers, honeypots are one of many tools to implement (Spitzner, 2002), especially in Supervisory Control and Data Acquisition (SCADA) networks, as there is little to no human interaction required to manage the networks (Disso et al., 2013).

This survey paper offers an introduction to the respective purposes of honeypots, honeynets, honeywalls, and honeytokens and shows the benefits of implementing them. These four systems study attackers by generating fake networks, hosts, services, and data.

The motivation to write this survey was to create a new collection of relevant literature and a structured overview to get more insight in this already established, but still promising topic. This survey is intended to give a brief introduction into the topic and to show what has already been done with honey systems to see the variability of them and attract more attention to this topic.

This paper is structured as follows: First, section 2 explains the differences between the honey tools. The following sections describe the properties of the different honey tools and illustrate some use cases as well as implementation considerations. Furthermore, section 5 describes monitoring, detection and hiding methods followed by a standardised threat information exchange approach. Finally, legal aspects are described and a list of some implementations and honeypot tools finalises the paper. Further research that needs to be done concludes the paper.

2 TERMINOLOGY

Honeypots, honeynets, honeywalls, and honeytoken all serve the same purpose, namely to detect intruders and analyse their intrusive behaviour. No legitimate user would ever access a honey system (Petrunić, 2015), even a connection attempt is considered an attack (Provos, 2003).

Invented in the 1990s, a *honeypot* is the best known application of the honey systems (Cheswick, 1992). It is a single host, network device or daemon (Provos, 2003) luring potential attackers to distract them from valuable network ressources (Pouget et al., 2013). “A honeypot is a security resource whose value lies in being probed, attacked, or compromised.” (Spitzner, 2002). “A honeypot is a re-

Lackner, P.
How to Mock a Bear: Honeypot, Honeynet, Honeywall Honeytoken: A Survey.
DOI: 10.5220/0010400001810188
In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) - Volume 2*, pages 181-188
ISBN: 978-989-758-509-8

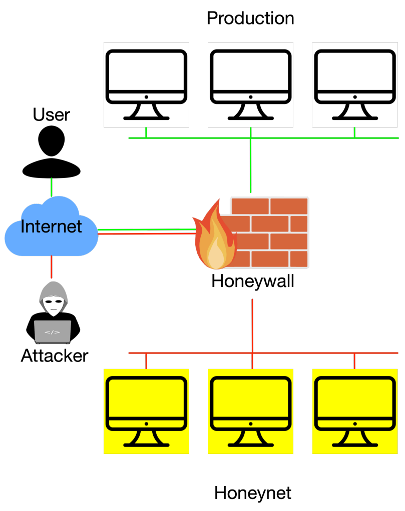

Figure 1: A honeynet, which consists of honeypots, duplicating the production system and being separated by a honeywall. The honeywall routes legimite users to the production network (white hosts) and attackers to the honeynet (yellow hosts).

source which pretends to be a real target." (Provos, 2003). A honeypot itself is not a security fix but helps fixing issues by gaining information about attacks. "The main goals are the distraction of an attacker and the gain of information about an attack and the attacker" (Baumann and Plattner, 2002).

A *honeynet* is a network of multiple honeypots (Project, 2002; Project, 2001). Analogous to honeypots, the entire network is to be attacked. The honeynet is not emulated and therefore can be a copy of the production system (Spitzner, 2003a). "Honeynets represent the extreme of research honeypots. They are high interaction honeypots, which allow learning a great deal; however they also have the highest level of risk. Their primary value lies in research and gaining information on threats that exist in the Internet community today. Little or no modifications are made to the honeypots of the honeynet to provide a plausible copy of the production net. This gives the attackers a full range of systems, applications, and functionality to attack. From this it can be learnt a great deal, not only their tools and tactics, but also their methods of communication, group organization, and motives" (Project, 2002). Low and high interaction honeypots are further explained in section 3. Figure 1 shows an example of a honeynet.

As also visualised in Figure 1, a *honeywall* is the perimeter border between honeynets and productive systems, although the term honeywall is not clearly defined. Some literature describe it as a gateway (Spitzner, 2003a), some describe it as a layer-2 or layer-3 filtering bridge firewall/Network Intrusion Detection System (NIDS) (Dittrich, 2004). It is also not clear whether the term "honeywall" refers to the product name of *The Honeynet Project* or the basic functioning in honey environments itself. The product *honeywall* is based on *iptables* rules, *Snort*, and *Snort-inline* (Dittrich, 2004). Other implementation use similar tools. In this paper, the term honeywall refers to a general implementation. Honeywalls support interception of SSL connections and decide if incoming traffic is malicious and therefore needs to be redirected to a honeynet, or if it is valid and therefore redirected to the productive system.

Unlike honeypots or honeynets, *honeytokens* are just data in the form of files, entries within files, or special strings (Spitzner, 2003a) (Malin, 2017). The data looks valid even though it is fictional and does not have any production use. The files are monitored in case of their modification. Data and strings are monitored as well, e.g. via *Google Alerts*. Google Alerts, a web service, notifies a user if a string appears in Googles search. Therefore, when a user is notified about a published honeytoken, it is likely that this honeytoken has been stolen and a successful intrusion occurred. A paper defines the following honeytoken properties (Bowen et al., 2009):

- Believable: A honeytoken looks like valid data.
- Enticing: A honeytoken lures an attacker.
- Conspicuous: A honeytoken is easily found.
- Detectable: Interacting with a honeytoken generates an alert.
- Variability: Various honeytoken do not contain the same information that create a connection between them.
- Non-interference: A Honeytoken does not interfere with desired data or system interactions.
- Differentiable: A legitimate user can differentiate between a honeytoken and actual data while an intruder cannot.

3 PROPERTIES OF HONEYPOTS

Another survey (Mokube and Adams, 2007) defines honeypots as devices which distract attackers from valuable machines, provide early warnings about attacks and allow in-depth examination of adversaries during and after the exploitation. To this end, it is in the interest of a defender that an attacker interacts with the honeypot over an extended period of time, while being closely monitored. There are several types (Spitzner, 2002) and several use cases (Mokube and Adams, 2007) of honeypots. While each type can be used in each use case, there are different advantages and disadvantages for each honeypot configuration.

3.1 Types

A *low interaction honeypot* has a very limited set of commands available for an attacker. While there is not much information one can obtain about attackers, the risk of damaged production systems due to a successful attack through that honeypot is also low (Mairh et al., 2011).

A *high interaction honeypot* has the goal to obtain a maximum amount of information about the attacker. The honeypot allows itself to be used, tampered with, or even be damaged. High interaction honeypots often are a clone of a production server. The goal is mainly to learn about novel attack techniques (Mairh et al., 2011). When deploying a high interaction honeypot, considerations about an extremely resilient monitoring system have to be made, to obtain authentic information of a partially compromised honeypot.

A *medium interaction honeypot* is in between of a low and a high interaction honeypot. While a low interaction honeypot only provides a limited set of commands and a high interaction honeypot grants access to the operating system (OS) of the honeypot, a medium interaction honeypot emulates a service (Mokube and Adams, 2007).

3.2 Use Cases

Widely known literature regards two use cases; research and production. This paper introduces a third use case: vulnerability scan.

The idea of a *research* honeypot is to provoke an attack and learn about the attacker. The purpose of a research honeypot is to learn about new techniques and tools of attackers. It also serves to learn about new combinations of tools attackers use to tamper with systems. There is no intention to defend the system against attackers (Mairh et al., 2011).

A *production* honeypot is only used in productive networks to obtain information about attackers invading the productive network. The purpose is not to gain maximum intel about the attacker but to detect the interest of the attacker. Spitzner and Schneider defined three interest groups within security issues: Detection, Prevention and Response (Mairh et al., 2011). A honeypot can provide value to Detection and Response.

HosTaGe constitutes a further use case, the non intrusive *vulnerability scan* (Vasilomanolakis et al., 2014). *HosTaGe* is a honeypot for mobile phones which scans public wireless networks. It detects malware spreading from devices within the same network and checks the basic security of public networks. It is considered to be a honeypot-to-go. *HosTaGe* also supports Industrial control system (ICS) protocols, making it relevant for industrial companies (Vasilomanolakis et al., 2015).

When implementing a honeypot, several problems should be considered (Mokube and Adams, 2007):

- Data Types: Data provided should look authentic to gain the attention of an attacker but it should not be possible to harm the company with this data.
- Uplink Liability: Honeypots can and will get compromised, which enables an attacker to attack other systems with it. Considerations about placing a honeypot in separate networks, maybe even a different public IP range than the production network of the enterprise have to be made (see more in section 6).
- Build Your Own Honeypot?: Most of the honeypot frameworks are open source and therefore customisable. The established honeypots professionally emulate the services they offer, when configured correctly. Additional services have to be programmed by the respective end-users themselves.
- Hiding: A honeypot should not obviously appear as a honeypot (see subsection 5.3).

Some researcher created a method to evaluate the potency of a honeynet which can be deployed to any honeynet (Ren et al., 2020).

3.3 Advantages and Disadvantages of Honeypots

Honeypots have particular advantages over traditional NIDS and network security approaches (Mokube and Adams, 2007):

- Small data sets: Only traffic addressing the honeypot is being observed. This can also be a disadvantage however.
- Minimal resources: Only attackers access a honeypot; normal users have no intention of using it. There is not always the need to use state-of-the-art hardware as the system does not have to carry normal production traffic and interactions.
- Simplicity: Honeypots, especially low interaction honeypots, are “simple and flexible”. They feature easy deployment and update routines as they do not offer the whole functionality of the original service.

- Discovery: Honeypots can help discover new tactics and tools directed at them.

Naturally, honeypots also have some disadvantages. To decide if a honeypot is the right device to monitor one's network, one has to consider the following points (Mokube and Adams, 2007):

- Limited Vision: Only the traffic that hits the target is visible and analysed. Traffic that does not target the honeypot in any way will not be detected. This can also be an advantage.
- Discovery and Fingerprinting: Honeypots can either emulate a service or provide the service itself. All services have different fingerprints in terms of reaction time and properties when using different network stacks, operating systems, and hardware. Nevertheless some fingerprints can reveal the implementation of a service. If a service is emulated, the fingerprint can reveal the emulating service (e.g.: honeypot tool). It might be possible to distinguish a real service from a honeypot from network meta data.
- Risk of Takeover: A honeypot may be used as an attacking device if it is taken over. So a honeypot should be monitored to determine, if it is still working as intended or already got compromised.

4 IMPLEMENTATION EXAMPLES OF VARIOUS HONEY SYSTEMS

Honey systems are used in various ways with various goals to achieve. The following subsections enumerate some examples of the possibilities.

4.1 Honeynets as a NIDS

A Network Intrusion Detection System is a system which detects known attacks (Karen A. Scarfone, Peter M. Mell, 2007) and warns administrators mostly through a Security Information and Event Management (SIEM) tool about incidents. During a project a honey net was created through container orchestration and *Conpot* as a honeypot backend (Wang et al., 2019). They define three modules to achieve an easy-to-work-with honeynet: the deployment module deploys a honeypot as a container with the desired properties. The management module manages the deployed container in terms of power management and honeypot properties. The intrusion detection module is the brain of this system. It processes traffic provided by the honeypots, trying to find attack patterns. Using a Support-Vector-Machine (SVM) algorithm the traffic data, especially the meta data (connection duration, protocol type, number of source bytes, number of destination bytes, whether it comes from the same host, the number of wrong segments, the number of urgent packets, etc.) is used to train a model, which was initally trained with the KDDCUP99[1] data set (a data set from 1999). The intrusion detection module uses this model to create alerts. They argue, since every connection to a honeypot is an attack per definition (Provos, 2003), this is a way to determine whether an attack is just random noise or a targeted attack against the system and to achieve a better detection rate with a lower amount of false positives (Wang et al., 2019). The detection rate of 89% still remains low compared to common detection rates using the SVM algorithm. A newer approach describes a similar concept (Fan et al., 2019).
A different approach describes a honeynet out of high interaction honeypots to detect a botnet in its creation phase using machine learning to detect hidden patterns (Martínez Garre et al., 2020). Furthermore, honeypots can organize themselves with a (private) blockchain to act and save data distributed and be more resistant to takedowns (Shi et al., 2019).

4.2 Honeywall

As seen in Figure 1, a honeywall is the perimeter border between a honeynet, a possible production net, and the internet. Its three main goals are: data capture, data control, and automated alerting. This is achieved by combining a layered firewall with NIDS/IPS and a monitoring tool. The honeywall decides if traffic is malicious or valid, and helps to correlate traffic from or to various honeypots in the honeynet (Chamales, 2004). A problem is the generation of good training material. To generate malware traffic signatures which let the honeywall decide if traffic is valid or malicious, malware traffic needs to be routed to the honeynet by the honeywall. Obviously, this is a chicken-and-egg problem, which can be mitigated by reversing the function of a NIDS. While a NIDS usually works by block-listing malicious traffic (Karen A. Scarfone, Peter M. Mell, 2007) it can also pass-lists legitimate traffic: All known legitimate traffic is sent to the productive network, everything else to the honeynet.

4.3 Honeytokens as an Active Defense

Honeytokens could be used as an active defense (Petrunić, 2015).

[1]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99

Zone 0: acts as the perimeter zone and may contain a firewall (honeywall), NIDS, and Web Application Firewall (WAF). Honeytokens are not placed there, because they are meant to address attackers who are able to circumvent these defense technologies. Although, on the perimeter border, an IDS rule can detect the usage of a honeytoken (Qassrawi and Hongli, 2010).
Zone 2: represents the web application. Honeytokens are placed here as they enable the application to react just-in-time to an attack. A URL parameter (*https://example.com/site.php?admin=false*) is an example of how applications react to the usage of honeytokens. When accessing the site with the parameter *admin=true*, the application knows to call a specific function defending against an attack. A further implementation is a web application honeypot to track bot crawls on a website (Lewandowski et al., 2020).
Zone 1 and 3: represent the web server and possible databases. These zones contain classic file or database-entry tokens to collect as much data as possible about the attacker if zone 2 is compromised. A webserver for example, storing honeytokens as files, can monitor them through the meta data. An exfiltration of these honeytokens is noticed by monitoring the file metadata, or by creating a trigger for the usage of a query for these honeytoken database entries.
The higher the zone number, the further an attacker has advanced into a system.

An example of the usage of honey token and the reaction of webservers follows: (Petrunić, 2015):

- Session Manipulation: If an attacker accesses a honeytoken, the session in use may be terminated or isolated into a sandbox mode. Also, every or randomly chosen requests may be answered with a HTTP 200 OK status code to feed the attacker with false information. A request rate limiting or intentional connection timeouts may further slow down the attacker.
- Generated Vulnerabilities: Once the system detects attackers it dynamically generates vulnerabilities to “keep the attackers happy” and to gain more time to study the attackers.
- Attack the Attacker: The generated vulnerabilities can deface attackers and reveal their identity by forcing them to establish a direct communication channel (Djanali et al., 2014) (Petrunić, 2015).
- *robots.txt*: Fake URLs and entries in the *robots.txt* help to identify attackers and malicious web spiders. *robots.txt* defines whether a web spider is allowed to index an URL or is not. When one of the fake URLs in *robots.txt* is accessed by either an attacker or malicious web spider, a specific reaction may get triggered.

Honeytokens are either handcrafted or generated through a tool like *HoneyGen* (Bercovitch et al., 2011). Like honeypots, honeytokens slow down attackers (Sandescu et al., 2017), and feed them with false information[2].

4.4 Real World Implementations

Honey systems are already used by different companies with different implementations. While companies usually do not publish their use of honey systems, some are well known to use them and even openly admit to use honey systems. Nevertheless, enterprises do not publish the exact implementation and version of their honey system. Many telecom providers, as well as security enterprises (firewall/NIDS manufacturers, antivirus/HIDS manufacturers, etc.) sustain many honeypots in a honeynet. They are distributed worldwide and create threat information and an attack map, among other things. Associations hide their critical infrastructure with honey systems and improve security efforts with the obtained information. Also, hospitals and medical facilities test the integrity of their personnel with a bogus patient record, for example a honeytoken called “John F. Kennedy” (Spitzner, 2003b).

5 MOCK THE BEAR, MANIPULATE THE HONEY

When operating any honey system there is always the risk of a compromised system as well as a defaced honey system. Either way, a honey system should work as intended. Monitoring and logging tools must be resistant to attacks to still record trustworthy information about an attack and an attacker. Also, attackers are capable to detect honey systems with various methods. This demands various hiding methods. The following paragraphs explain monitoring and logging methods, ways to distinguish honey systems from productive systems, and methods to mitigate a detection and to hide honey systems.

5.1 Find the Bear, Monitor the Honey

Various ways of monitoring honeypots exists. When honeypots became popular, *SEBEK* was a popular

[2]https://arstechnica.com/information-technology/2017/05/macron-campaign-team-used-honeypot-accounts-to-fake-out-fancy-bear/

monitoring tool, superseded by *XEBEK*, a *SEBEK*-like tool optimised for virtual machines. These are data/traffic capture kernel modules which send data to a central logging server to monitor and visualise events (Quynh and Takefuji, 2006). Nowadays, standard tools like *syslog* and Trusted Automated eXchange of Indicator Information (TAXII) in combination with a central monitoring server are more common (Wagner, 2019). Since a honeypot is accessed by an attacker, there has to be a focus on log authenticity. Sending logs immediately to a remote logging station and not storing them locally on a compromised honeypot is part of a good log handling.

5.2 Detect the Honey

A recent paper describes a method to monitor and analyse botnets with honeynets (Bajtoš et al., 2018). To get authentic results, considerations must be made on the fact, that many different ways of detecting honeypots have been developed by attackers to avoid honeypots. A quite old paper describes a way to detect honeypots in a botnet (Zou and Cunningham, 2006). The underlying problem is still present: “Should a honeypot attack other systems to look authentic?” A botnet is a network of infected machines controlled by a bot controller (Puri, 2003). A bot controller commands a new bot to attack a list of hosts. This list of hosts or addresses include sensors, controlled by the attacker, which report every attempt to contact them to the bot controller. When the bot controller receives an IP address, to which the controller commanded an attack before, it is assumed a valid bot, otherwise it might be a honeypot trying to sanitize its outgoing traffic to not attack others (Wang et al., 2010). This introduces the dilemma of sanitising outgoing traffic due to legal affairs and getting authentic results.

5.3 Hide the Honey

Honeypots, honeynets, honeywalls, and honeytokens need to be concealed from the attackers. The first step to hide them is to determine the type of attacker they are hidden from. Usually, “attacker” means everything outside of an organisation or company. To hide a honeypot from the outside, it should behave like the emulated service in terms of user interaction. This means that the front-end and user responses, especially the reaction time of the service, remain the same as the original service (Litchfield et al., 2016). The back-end can be adjusted to the operators’ needs. In the best case, an attacker does not know about the structural design of an organisation, so little to no management effort is required to hide honey systems. The emulation of several things, including the network stack and processing must be authentic to the real service to disguise the honeypot (Litchfield et al., 2016). Network scanning tools like *nmap*[3] look for protocol quirks to identify the exact implementation. Depending on the purpose, honeypots should or should not emulate the exact quirks. Various methods to properly emulate services are suggested (Qassrawi and Hongli, 2010):

- Vulnerability Emulation: Only the vulnerable part of a service or OS is emulated. *PHP.HoP* (Qassrawi and Hongli, 2010) uses this technique. This obviously works only for already known vulnerabilities!
- Connection Tarpitting: Tarpitting in terms of honeypots means delaying network traffic and service responses. Network traffic may be delayed by manipulation of the window flag in Transmission Control Protocol (TCP/IP) packets. Service responses may be delayed deliberately by waiting milliseconds before sending them. Appropriate waiting times might help to make a honeypot more authentic; e.g. sending a command to a machine via a controller implies waiting for a response from a controller which waits for the response of the machine, which takes time. Immediately sending a response likely unmasks a honeypot.
- Traffic Redirection: Suspicious traffic gets redirected to a honeypot. For example, a connection establishment to an unused IP or suspicious traffic, detected with help of a NIDS, can be redirected to a honeypot. However, a NIDS only detects already known attack vectors and needs to know the network topology or at least the used IP addresses to detect unused IP addresses.

The techniques described here hide a honey system from an external attacker. Hiding honey systems from personnel or specific groups of personnel of an organisation or company was not found in literature.

6 LEGAL AFFAIRS AND PRIVACY

When operating a honeypot one has to consider several legal aspects.

Liability is an important factor to consider. Activity and traffic from the honeypot is linked to the organisation hosting the honeypot. If a honeypot is compromised and attacks other systems or acts as an

[3]https://nmap.org/

illegal market place for example, there are legal consequences for the organisation. Especially neglected honeypots, which are not monitored regularly, are a popular target. To prevent consequences, aside from the technical aspects, the goal and methods of a honeypot should be documented in detail (Mokube and Adams, 2007).

The right to monitor is competitive to the right of *privacy*. As this tension is part of the daily life of an IT-security engineer, this applies to honey systems as well. Most countries already passed laws regarding privacy matters, such as the Fourth Amendment, the Wiretap Act, the Patriot Act, and the Freedom Act in the USA and the GDPR in the EU. A decision, whether and how information can be logged, is made regarding these laws (Mokube and Adams, 2007). A profound analysis of privacy matters, concerning honey systems, has already been done based on EU laws, already including GDPR, in 2017 (Sokol et al., 2017).

7 CONCLUSION

This paper describes the different types of honeypots, honeynets, honeywalls, and honeytoken. It explaines the terminology and properties of the honey systems and describes the fields where the systems are deployed. Additionally, advantages and disadvantages are discussed. Implementation thoughts, including legal aspects and different techniques to detect and hide honey systems, are given as well as techniques to transmit threat information. As the purpose of honeypots is to communicate with an attacker they are at risk of being compromised. Therefore, a honey system should always be operated with a NIDS and should never be left unmonitored. Also, any logging data should be transferred to a safe place immediately. Honey systems are not a new field in computer science anymore. New insights into the topic based on public research are limited, even though a lot of implementation methods and tools have been developed. Further research needs to be done to evaluate the honeypot implementation of new services and protocols. Recent papers combine honeysystems with modern technologies like blockchains and machine learning to mitigate honeysystem detection methods and to increase malware detection rates. Although the methods are new, the data used to train systems is either old (Wang et al., 2019) and is not comparable to modern internet traffic or not described (Martínez Garre et al., 2020). The main challenge in upcoming honey system development will be trusted, automated, dynamic attack pattern matching. Unfortunately, no solution to hide a honey system from employees and administrators in the same company has been developed so far which is why further research into the matter is necessary.

ACKNOWLEDGEMENT

This work has been supported by the Austrian State Printing House. Furthermore, I would like to thank Martin Pirker and Patrick Kochberger for giving helpful feedback about this paper.

REFERENCES

Bajtoš, T., Sokol, P., and Mézešová, T. (2018). Virtual honeypots and detection of telnet botnets. In *Proceedings of the Central European Cybersecurity Conference 2018*, CECC.

Baumann, R. and Plattner, C. (2002). Honeypots. Master's thesis, ETH Zürich.

Bercovitch, M., Renford, M., Hasson, L., Shabtai, A., Rokach, L., and Elovici, Y. (2011). Honeygen: An automated honeytokens generator. In *Proceedings of 2011 IEEE International Conference on Intelligence and Security Informatics*. IEEE ITSS.

Bland, J. A., Petty, M. D., Whitaker, T. S., Maxwell, K. P., and Cantrell, W. A. (2020). Machine learning cyberattack and defense strategies. *Computers & Security*.

Bowen, B., Hershkoop, S., Keromytis, A., and Stolfo, S. J. (2009). Baiting Inside Attackers Using Decoy Documents. In *International Conference an Security and Privacy in Communication Systems*. SecureComm, Springer.

Chamales, G. (2004). The honeywall cd-rom. *IEEE Security Privacy*.

Cheswick, B. (1992). An evening with Berferd in which a cracker is lured, endured and studied. In *In Proc. Winter USENIX Conference*. USENIX.

Cui, M., Wang, J., and Chen, B. (2020). Flexible machine learning-based cyberattack detection using spatiotemporal patterns for distribution systems. *IEEE Transactions on Smart Grid*.

Disso, J. P., Jones, K., and Bailey, S. (2013). A plausible solution to SCADA security honeypot systems. In *2013 Eighth International Conference on Broadband and Wireless Computing, Communication and Applications*.

Dittrich, D. (2004). Customizing ISOs and the Honeynet Project's Honeywall.

Djanali, S., Arunanto, F. X., Pratomo, B. A., Baihaqi, A., Studiawan, H., and Shiddiqi, A. M. (2014). Aggressive web application honeypot for exposing attacker's identity. In *2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering*. ICITACEE.

Fan, W., Du, Z., Smith-Creasey, M., and Fernández, D. (2019). Honeydoc: An efficient honeypot architecture enabling all-round design. *IEEE Journal on Selected Areas in Communications*.

Karen A. Scarfone, Peter M. Mell (2007). Guide to Intrusion Detection and Prevention Systems (IDPS). Technical report, NIST.

Lewandowski, P., Janiszewski, M., and Felkner, A. (2020). Spidertrap - an innovative approach to analyze activity of internet bots on a website. *IEEE Access*.

Litchfield, S., Formby, D., Rogers, J., Meliopoulos, S., and Beyah, R. (2016). Rethinking the honeypot for cyber-physical systems. *IEEE Internet Computing*.

Mairh, A., Barik, D., Verma, K., and Jena, D. (2011). Honeypot in network security: A survey. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*. ICCCS, ACM.

Malin, C. (2017). LureBox - Using Honeytokens for Detecting Cyberattacks. Master's thesis, UAS St. Pölten.

Martínez Garre, J. T., Gil Pérez, M., and Ruiz-Martínez, A. (2020). A novel machine learning-based approach for the detection of ssh botnet infection. *Future Generation Computer Systems*.

Mokube, I. and Adams, M. (2007). Honeypots: Concepts, approaches, and challenges. In *Proceedings of the 45th Annual Southeast Regional Conference*. ACM-SE.

Papp, D., Ma, Z., and Buttyan, L. (2015). Embedded systems security: Threats, vulnerabilities, and attack taxonomy. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*.

Petrunić, R. (2015). Honeytokens as active defense. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. MIPRO.

Pouget, F., Dacier, M., and Debar, H. (2013). Honeypot, Honeynet, Honeytoken: Terminological issues. *Institut Eurécom (EURECOM), Sophia Antipolis, France, Research Report RR-03-081*.

Project, T. H. (2001). Know Your Enemy: Honeynets.

Project, T. H. (2002). Know Your Enemy: Defining Virtual Honeynets.

Provos, N. (2003). HoneyD: A Virtual Honeypot Daemon.

Puri, R. (2003). Bots & botnet: An overview.

Qassrawi, M. T. and Hongli, Z. (2010). Deception methodology in virtual honeypots. In *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*. NSWCTC.

Quynh, N. A. and Takefuji, Y. (2006). Towards an invisible honeypot monitoring system. In *Australasian Conference on Information Security and Privacy*. ACISP.

Ren, J., Zhang, C., and Hao, Q. (2020). A theoretical method to evaluate honeynet potency. *Future Generation Computer Systems*.

Sandescu, C., Rughinis, R., and Octavian, G. (2017). Hunt: Using honeytokens to understand and influence the execution of an attack. In *Proceedings of the 13th International Scientific Conference "eLearning and Software for Education"*. eLSE.

Shi, L., Li, Y., Liu, T., Liu, J., Shan, B., and Chen, H. (2019). Dynamic distributed honeypot based on blockchain. *IEEE Access*.

Simmons, C., Ellis, C., Shiva, S., Dasgupta, D., and Wu, Q. (2014). Avoidit: A cyber attack taxonomy. In *9th Annual Symposium on Information Assurance (ASIA'14)*. ASIA.

Sokol, P., Míšek, J., and Husák, M. (2017). Honeypots and honeynets: issues of privacy. *EURASIP Journal on Information Security*.

Spitzner, L. (2002). *Honeypots: Tracking Hackers*. Addison Wesley.

Spitzner, L. (2003a). Honeypots: Catching the Insider Threat. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.* IEEE.

Spitzner, L. (2003b). Honeytokens: The other honeypot. In *Endpoint Protection*. Broadcom.

Vasilomanolakis, E., Karuppayah, S., Mühlhäuser, M., and Fischer, M. (2014). Hostage - a mobile honeypot for collaborative defense. In *Proceedings of the 7th International Conference on Security of Information and Networks*. SIN.

Vasilomanolakis, E., Srinivasa, S., and Mühlhäuser, M. (2015). Did you really hack a nuclear power plant? an industrial control mobile honeypot. In *2015 IEEE Conference on Communications and Network Security (CNS)*.

Wagner, T. D. (2019). Cyber threat intelligence for "things". In *2019 International Conference on Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*.

Wang, P., Wu, L., Cunningham, R., and Zou, C. C. (2010). Honeypot detection in advanced botnet attacks. *Int. J. Inf. Comput. Secur.*

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

Zou, C. C. and Cunningham, R. (2006). Honeypot-aware advanced botnet construction and maintenance. In *International Conference on Dependable Systems and Networks (DSN'06)*. DSN.