Evaluating Digital Forensic Readiness: A Honeypot Approach

Philip Zimmermann[a] and Sebastian Obermeier[b]
Hochschule Luzern Informatik, Campus Zug-Rotkreuz, Suurstoffi 1, 6343 Risch-Rotkreuz, Switzerland

Keywords: Digital Forensics, Forensic Readiness, Honeypot.

Abstract: Digital forensic readiness has proven to be a challenging undertaking for small to medium-sized companies. To improve, it is important to evaluate the effectiveness of forensic processes. In this paper, an approach for a forensic honeypot is proposed that simulates an environment based on real company devices and is hosted in the cloud. The data collected is used for the evaluation of the forensic process, enabling the identification of discrepancies within the forensic readiness approach. The experimental results show the feasibility of the approach in collecting forensic evidence in a short time. The paper also discusses limitations with regard to the introduction of new security threats and the use and placement of endpoint intrusion detection systems.

1 INTRODUCTION

Malware attacks pose a serious threat to the security and integrity of digital systems, especially for small-to-medium enterprises (SMEs), which may lack adequate resources and expertise to deal with them. However, not all SMEs are equally exposed to malware risks, and some may never encounter a malware incident in their operations. This does not mean that they can afford to be complacent or unprepared, as malware attacks can occur at any time and cause significant damage. Therefore, it is essential for SMEs to adopt a proactive approach to digital forensics, which is the process of collecting, preserving, and analyzing digital evidence in the event of a cyberattack. This approach is known as *digital forensic readiness* (DFR), and aims to minimize the impact of malware incidents and facilitate recovery and investigation processes. Measuring the effectiveness and functionality of the DFR process is challenging without a genuine or simulated incident. This paper aims to transform a company device into a virtual honeypot, allowing SMEs to evaluate their DFR capabilities.

1.1 Problem Description

Achieving forensic readiness is not a trivial task, as it requires careful planning, testing, and evaluation of the forensic capabilities and procedures of an organization. Furthermore, testing the forensic readiness of an organization is challenging, as it cannot be done realistically without simulating a malware attack, which may be unethical or not practicable, e.g., due to the criticality of the system. Therefore, there is a need for novel methods and tools that can help SMEs to assess and improve their forensic readiness in a safe and effective manner.

One method of simulating cyber attacks within a company is to use red teaming (Mansfield-Devine, 2018) (Oakley, 2019), which is a simulated cyberattack that tests the security posture and incident response capabilities of an organization. Although it can provide valuable information on the strengths and weaknesses of defensive measures as well as the effectiveness of the forensic readiness, red teaming also has some potential downsides, especially for small and medium enterprises (SMEs) that may have limited resources and expertise: First, red teaming is a costly and time-consuming exercise that can divert resources and attention from other business priorities. Second, red teaming may not cover all the possible scenarios and attack vectors that affect the forensic capabilities and forensic readiness of an organization. Third, red teaming can introduce new risks and vulnerabilities and can generate a large amount of data that could overwhelm the forensic analysis capabilities of an SME.

Another approach would be the use of dedicated honeypots such as T-Pot[1](Washofsky, 2021). However, as the company's internal systems are typically different from the honeypot, the resulting data may

[a] https://orcid.org/0009-0007-8765-9703
[b] https://orcid.org/0000-0003-2802-7427

[1] https://github.com/telekom-security/tpotce

Zimmermann, P. and Obermeier, S.
Evaluating Digital Forensic Readiness: A Honeypot Approach.
DOI: 10.5220/0012713600003767
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024)*, pages 615-621
ISBN: 978-989-758-709-2; ISSN: 2184-7711
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

not reflect the actual company infrastructure. In addition, it may not yield enough data to test the forensic approaches. Nevertheless, we keep the idea of using a honeypot, but more tailored to the actual SME.

1.2 Contributions

This paper

1. creates a concept for transforming a company device into a virtual honeypot
2. defines the steps that are needed to deploy and run the honeypot securely in a cloud environment in order to gather data for forensic analysis
3. evaluates the concept in a real-world experiment and discusses the feasibility of the approach

Note that neither the process of sanitizing a system such that no identifiable personal information is present nor the actual analysis of the gap in the forensic process is part of this document, as this is highly depending on the type of company.

2 RELATED WORK

Numerous publications exist that explore the concept of honeypots or forensic readiness, but linking both does not seem common. (Kebande et al., 2016) focuses on utilizing honeypots for the collection of potential digital evidence (PDE) within a BYOD (Bring Your Own Device) setting, but does not transform actual devices into honeypots.

A comparison of the major cloud platforms for the deployment of honeypots is made in (Kelly et al., 2021). The paper confirms that threat actors try to abuse cloud-based services.

During our investigations, we concluded that the main use of honeypots is to detect attacks in an early state – both in IT (Dodson et al., 2020) and OT networks (Priya and Chakkaravarthy, 2023) – since interaction with the honeypot is typically not a normal operational activity and can reveal security issues in the cloud environment (Singh, 2021).

Regarding an optimal honeypot deployment, (Verma and Dubey, 2020) discusses the advantages and disadvantages of different deployment approaches.

The authors of (Biedermann et al., 2012) utilize a cloning mechanism to duplicate a live VM facing an attack, redirecting the assault for analysis and learning from the attacker. To ensure rapid cloning during live attacks, an approach is used to adapt the Xen hypervisor (Barham et al., 2003), enabling rapid duplication of existing VMs. However, this approach is not compatible with modern cloud environments, as users lack the ability to make modifications to the hypervisor in that setting.

To analyze data of a honeypot system, machine-learning based approaches are becoming a viable option (Setianto et al., 2022).

To summarize, honeypots are frequently used to detect attacks in an early stage, but the use for evaluating a company's forensic readiness is currently not a common practice.

Once data is collected through the approach proposed in this paper, forensic readiness capabilities should be evaluated. Different models exist for this approach, for example, (Taiwo and Claims, 2022). Forensic analysis is also standardized in ISO/IEC 27043. However, such models are typically only applicable to companies that have a forensic footprint and knowledge about forensic readiness. Our approach helps speed up this process, as our experiments confirmed that incident data will most likely be generated in less than a week.

3 SOLUTION APPROACH

In order to fulfill the two goals of "forensic readiness" as described by Tan (Tan, 2001), it is necessary to assess the ability to gather pertinent digital evidence. To evaluate this capability, a forensic honeypot is proposed that simulates an environment based on real company devices. This environment is exposed to the public internet, and all interactions and potential incidents are recorded by the forensic honeypot. The data from the system used as the attack surface is then processed using the existing forensic analysis approach. The investigator can determine whether the collected data are sufficient or whether the forensic readiness approach should be extended or modified.

For the purpose of this paper, the following requirements have been formulated.

Attack Detection. The system must be capable of detecting attacks and generating corresponding alerts over the duration of the data collection

Traceability. The attacker's actions must be accurately tracked, i.e., what, when, how, and by whom actions were taken.

Security. An attack on the honeypot must not jeopardize the security of the surrounding systems.

Transparency. The honeypot should not be recognizable as such to external parties. It shall be transparent.

High Interaction. The honeypot should have a high-level interaction capability.

The basic idea behind the construction of the honeypot is to transform an existing system into that honeypot and use it to attract attackers and collect forensically relevant data. To achieve this, an existing system is selected and rebuilt within the honeypot environment. To allow investigation of security incidents, the vulnerable system is periodically cloned. Backups are created periodically from these clones to simplify analysis, such as recovering traces obscured by the attacker. Services like a Host Intrusion Detection System (HIDS) are then installed on the clone, allowing monitoring of the environment without revealing this to the attacker.

For proof of concept, two virtual machines with restricted scope serve as the vulnerable system. However, the described method can also be applied to complex systems as potential attack surfaces.

The operation of the honeypot is divided into three phases.

- Normal Operation: The system is deployed. To stimulate an investigatable security incident, vulnerabilities can be introduced into the services being used. This can be done, for example, by employing services with known security flaws. Once an attack occurs, an alert is generated, and the process transitions to the next phase.
- Data Collection: Upon detecting an attack on the honeypot, the duration for data collection can be determined based on the severity of the attack. After this period elapses, the vulnerable system is shut down and forensic analysis starts.
- Forensic Analysis: During the forensic analysis of the honeypot, the resources that participated in the security incident are analyzed: the image of the vulnerable system and any data from the respective surrounding systems like HIDS.

Upon completion of these three phases, the implementation of DFR will be reviewed. The data collected by the honeypot are used for the review, allowing the identification of discrepancies between the data collected from the forensic process and the actual data generated during the incident.

3.1 Special Case: HIDS and Remote Management

A special case concerns the management system of intrusion detection systems used, especially endpoint detection and response (EDR) systems. In this context, there are two viable options:

A) Setting up a new HIDS-Management Server instance within the Honeypot environment may result in higher costs due to licenses. Nevertheless, the honeypot remains self-contained, reducing the risk of an attacker using the HIDS-Management Server for lateral movement into the company network.

B) Alternatively, connecting an existing HIDS-Management Server in the SME environment to the honeypot eliminates the need for extra license fees. However, this approach expands the attack surface, potentially allowing attackers to exploit the connection for lateral movement into the company network.

To come to an informed decision, the SME should first evaluate its current licensing situation for HIDS/ EDR. Following this, it is crucial to take into account any regulatory requirements that may apply. A comprehensive risk assessment should then be conducted, recognizing the financial limitations and the overall risk tolerance of the organization.

4 IMPLEMENTATION

For the implementation, we have used a system based on the university notebooks. As there is no standard HIDS throughout the university, we have employed OSSEC (Open Source HIDS SECurity)[2]. It is crucial to recognize that any HIDS could be appropriate for the task. OSSEC+ was selected due to its open-source nature. The honeypot experiment was set up in Google Cloud using Ansible for automation.

In order to import an existing system as an attack surface into the honeypot, it needs to be ensured that this is done securely. For this purpose, the suggested process is outlined, comprising both a setup phase (1-4) and an operational phase (5-8).

1) Sanitization: The system must undergo sanitization to prevent any potential leakage of sensitive information or credentials to an attacker

2) Import: To enable the import of the system, it should be exported in a compatible format using suitable tools like Disk2VHD[3]. For Google Cloud, the Open Virtualization Format (OVF)[4] is a suitable choice, the virtualized system should be exported in such a format.

3) Prevention: Implement firewall rules that block outgoing traffic (egress) from the system to ensure that attackers cannot use it as a staging area.

4) Introduce Vulnerability: To gain insights into the DFR capabilities, an attack should be analyzed. To facilitate this, intentionally introducing a vulnerability can increase the likelihood of a successful at-

[2]https://www.ossec.net/

[3]https://learn.microsoft.com/en-us/sysinternals/downloads/disk2vhd

[4]https://www.dmtf.org/standards/ovf

tack. This step is optional and can also be completed at a later time.

5) Deploy/ Simulate Attack: The system can then be deployed, and specific scenarios can be tested by simulating attacks. This proactive approach eliminates the need to wait for malicious parties to initiate an attack. Alternatively, the system can be deployed without simulating an attack. This allows the collection of data that closely resembles real-life scenarios.

6) Preparing HIDS-Server: The Connection from the HIDS-Agent to the HIDS-Server needs to be prepared so it can be established after the cloning.

7) Clone: The system undergoes cyclical cloning, followed by the installation of the HIDS-Agent on the cloned system to enable monitoring.

8) Monitor: Active monitoring of the HIDS and firewall logs is crucial to prevent any abuse of the system.

Figure 1 illustrates a run of the cloning task. As this cloning task introduces a novel concept that has not been previously observed in other honeypots, it will be thoroughly illustrated.

- Step 1: At first, a snapshot of the attack surface or vulnerable host is created. Any existing snapshots are deleted.
- Step 2: A disk is generated based on the snapshot, with the current date as its label. Disks with the same label are removed, while those with a different label are retained, this enables us to retain a daily backup of the disk.
- Step 3: The newly created disk is used to create a new instance.
- Step 4: Involves configuring the OSSEC+ agent on the cloned attack surface.

The entire sequence repeats at a predefined interval of 45 minutes. At the heart of the cloning operation is the cloning of the disks outlined in step 2. The corresponding ansible code is shown in Figure 2. The ansible module google.cloud.gcp_compute_disk is utilized. This module enables the manipulation of disks, involving actions such as deletion and creation. Parameters for this module include the zone where the disk will be generated, the disk's name (which serves as a unique identifier for the associated instance, including the creation date), and the size. Notably, disks will undergo continuous creation and deletion, with one disk retained at the conclusion of each day.

Figure 3 presents the network topology of the approach, comprising of four key components:

- OSSEC+-Server: This server functions as the HIDS server, responsible for aggregating and processing alerts generated by agents on the attack surface clone and syslog messages from the attack surface.

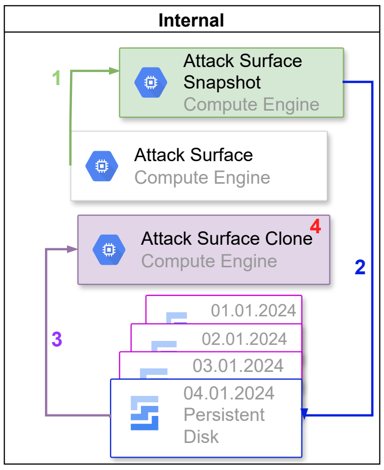

Figure 1: Process of the cloning task.

```
- name: 'Delete old disk'
 google.cloud.gcp_compute_disk:
   project: "{{ project }}"
   name: "{{item.sign +  ansible_date_time.date}}"
   auth_kind: "{{ auth_kind }}"
   service_account_file: "{{ service_account_file }}"
   state: 'absent'
   zone: "{{ zone }}"
 ignore_errors: true
 loop:
   - {sign: 'v'}
   - {sign: 'w'}
   - {sign: 'm'}

 name: 'Create disk from snapshot'
 google.cloud.gcp_compute_disk:
   name: "{{item.sign + ansible_date_time.date }}"
   project: "{{ project }}"
   source_snapshot:
     selfLink: "{{item.link}}"
   auth_kind: "{{ auth_kind }}"
   service_account_file: "{{ service_account_file }}"
   state: 'present'
   zone: "{{ zone }}"
   size_gb: "{{item.size}}"
 loop:
   - {sign: 'v', link: "{{self_link}}", size: 15}
   - {sign: 'w', link: "{{web_self_link}}", size: 15}
   - {sign: 'm', link: "{{win_self_link}}", size: 125}
```

Figure 2: Ansible code for disk rotation.

- Ansible Coordinator: This component orchestrates the cloning process of the attack surface.
- Attack Surface: These are virtual machines (VMs) that serve as the attack surface. They collect syslog messages and transmit them to the OSSEC-Server.
- Attack Surface Clone: The attack surface clone replicates the original attack surface. An OSSEC Agent operates on this machine, generating alerts for further processing.

Firewall rules serve as a robust protective measure, efficiently intercepting and mitigating unauthorized communication attempts and the activation of malicious software originating within the honeypot environment. These regulations endorse exclusively inbound network traffic, while strictly limiting activi-

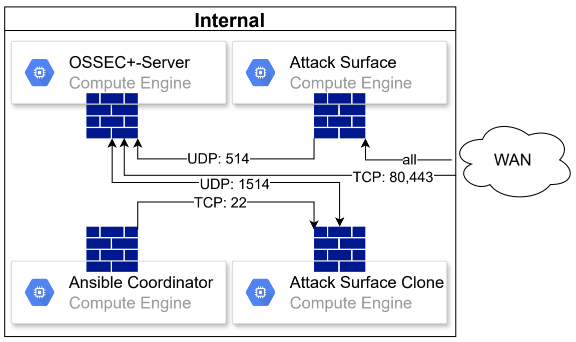

Figure 3: Network topology and firewall rules.

ties such as initiating reverse shells and executing attacks on external systems located outside the predefined attack surfaces. The establishment of connections between the defined attack surfaces remains possible. Due to the utilization of software-defined networking in the Google Cloud network, each instance functions as if it possesses its own firewall. Consequently, firewall rules must be configured on both endpoints.

5 EXPERIMENTAL EVALUATION

To validate the forensic honeypot, two attack surfaces based on university devices were provisioned and configured according to the eight step plan defined earlier.

- Linux host with SSH: A Debian system with a ssh server that allows user login with standard passwords.
- Windows 10 host with RDP: A Windows 10 host with RDP enabled that can be accessed with a blank password.

While the Linux host was sourced from a server in the university environment, the Windows host was cloned from a live system on a laptop, converted to the OVF format, and subsequently integrated into the GCP environment. Following this integration, firewall rules were established to enable communication. A crucial rule implemented is an implicit deny rule, restricting all traffic that is not explicitly permitted. The vulnerability introduced pertained to preconfigured blank or standard passwords. Additional vulnerabilities were later incorporated for simulated tests. The system was then deployed to attract real attackers. The Agent connections were established on the OSSEC-Management Server before executing the initial cloning task, allowing the Agents to connect as soon as they were installed on the clones. The Ansible playbook for continuous cloning was initiated, and the logs of the HIDS and firewall were intermittently monitored manually.

The assessment period was 24. Nov. 2023 to 29. Nov. 2023. The first successful login for the user ”pi” occurred following a series of unsuccessful login attempts by the same attacker. This event triggered an alert by OSSEC at 9:31. The initiation of the second phase, namely the data collection, is indicated by this alert. The data collection timeframe has been established with a deadline set for the 29th. Shortly thereafter during the next clone cycle, OSSEC detected the generation of a new file that according to VirusTotal[5] is classified as a ”trojan” malware. Subsequently, OSSEC signaled a modification in the /etc/passwd file, indicating that the user's password had been changed to prevent unauthorized access. In the Google Cloud Platform (GCP) logs, this marked the beginning of an increase in the firewall logs. Approximately every 30 minutes, 50,000 connection requests from the now infected host were blocked. It is important to note that the vulnerable machine is cloned, resulting in the generation of only 25,000 requests per machine in reality. Following the data collection phase, an image file of the virtual machine was acquired. The honeypot's findings could be corroborated with the actual data on the system. In a real-world scenario, the disparities between the intelligence gathered from the virtual machine image and logs from other systems would be assessed in comparison to the intelligence collected by the honeypot. These differences provide insights into how the DFR process can be enhanced and whether adjustments are needed.

Table 1 provides information on the usage of the honeypot during the evaluation phase.

Table 1: Statistics honeypot evaluation.

	Windows	**Debian**
Login Attempts	49807	72766
Successful Logins	1257	249
Actions on Host	None	Malware Installed

In order to illustrate how a real-world attack scenario would occur, an attack was conducted on both machines as part of a student exercise. The Debian machine was configured with sudo version 1.8.19p1-2.1, which contains a vulnerability identified as CVE-2021-3156. This vulnerability enables an attacker to exploit a heap-based buffer overflow, thereby acquiring root privileges.

During an attack, OSSEC detected a brute force attack on SSH credentials, marking a successful login

[5]https://www.virustotal.com

after numerous failed attempts as suspicious. Subsequently, the alert system flagged the creation of the file linpeas.sh, with relevant details such as the alert timestamp, originating host, alert type, and MD5 and SHA1 hashes of the new file, as illustrated in Figure 4. Similarly, OSSEC signaled the creation of the exploit file exploit_userspec.py[6]. The exploit, generating approximately 2000 alerts, indicated a segmentation fault caused by the exploit. Notably, OSSEC did not recognize the multitude of segmentation faults as indicative of a buffer overflow. The creation of a new user ”gg”, resulting from the exploit script, triggered another alert. OSSEC also detected the password change associated with this newly generated user.

The attack executed on the Windows machine differed, as the machine did not possess any well-known security flaws. The attack focused on installing malware and involved the following steps: Login to the machine with the blank password, install executable on the machine, simulate a dropped file in the autostart directory, and simulate changed and added registry keys. OSSEC detected each of these actions and created alarms.

```
"_index": "ossec-alerts-3.x-2023.12.07",
"_type": "_doc",
"_id": "2RRiRIwBmKL8NDW4whs9",
"_version": 1,
"_score": 1,
"_source": {
 "@timestamp": "2023-12-07T13:06:11.155Z",
 "hostname": "(debvuln) 10.182.0.31->syscheck",
 "decoder": "syscheck_integrity_changed",
 "SyscheckFile": {
   "path": "/tmp/linpeas.sh"
 },
"rule": {
  "firedtimes": 3369,
  "groups": [
    "local",
    "syslog",
    "syscheck"
  ],
  "level": 10,
  "comment": "File added to the systen.",
  "sidid": 554
},
"agent_name": "debvuln",
"agentip": "10.182.0.31",
"timestamp": "2023 Dec 07 13:06:10",
"id": "1701954370.1141060"
```

Figure 4: Json alert.

6 DISCUSSION

The experimental evaluation revealed that real-world attacks manifested within a few hours of online presence. The honeypot demonstrated effective detection of intrusions and malicious actions on hosts such as the installation of malware or the creation of a new user as the result of a privilege escalation. For showcasing advanced attacks, we included two vulnerabilities into the VMs. The requirements of Section 3 can be evaluated as follows:

Attack Detection. In both, simulated and actual scenarios, the honeypot identified the attacks.

Traceability. In both simulated and real-world situations, it was feasible to precisely trace the actions of the attacker, including what, when, who, and how of their activities.

Security. An attack on the honeypot must not compromise the security of the surrounding systems. The vulnerable host and its clone establish two outgoing connections to the rest of the system: one for sending Syslog messages to the OSSEC server and another for the clone’s communication with the OSSEC server via the OSSEC agent. With firewall rules blocking additional communication, an attack on the honeypot does not pose a threat to other honeypot systems. The productive subsystems are in a separate environment, and the use of GCP ensures galvanic isolation of the honeypot from the productive system.

Transparency. The sole configuration on the attack surface host involves the ”rsyslog.conf,” enabling the transmission of Syslog messages to the OSSEC server. This host, which functions as a complete virtual machine with either Linux or Windows OS, hides its honeypot nature from external attackers.

High Interaction As the vulnerable host is a fully functional instance of a virtual machine, the interaction level of the honeypot can be classified as ”high.”

We have observed no automatic containment by the cloud provider when the firewall blocks outgoing requests, making it technically possible to replicate a realistic environment. Real-world attacks manifested shortly after deployment, confirming the viability of using it as a honeypot and creating valuable data to evaluate the effectiveness of DFR processes. Our firewall configuration has ensured that the honeypot is not exploited as an entry point for attacks on additional systems. Moreover, costs can be adjusted as required since the infrastructure is in the cloud.

Known limitations of the system include the inability to create an exact 1:1 replica of an existing environment. Sanitization of data before deployment includes, for example, the deletion of sensitive personal or sensitive company data, which inherently modifies the system. However, as relevant aspects of the forensic readiness system are preserved, we consider the impact of this modification to be negligible.

While it is crucial that firewall rules limit outbound traffic for security reasons, yet this also provides attackers a way to detect the honeypot.

Furthermore, the mere presence of critical components in the honeypot could potentially leak information, leading to a decision to exclude them. Striking

[6]https://github.com/worawit/CVE-2021-3156

a balance between accurately portraying the system and limiting the disclosure of information to potential attackers becomes an important consideration that is based on the individual company's business.

Another limitation concerns the host-based intrusion detection systems (HIDS), in our experiments OSSEC. It requires proper configuration as the default configuration fails to provide alerts for newly created files and the inability to recognize buffer overflows generated in simulations as attacks. Custom rules can be set up to configure these aspects, but it remains unclear which other attack scenarios may not be detected by OSSEC. However, this is based on the HIDS used in the individual company, and other HIDS may perform differently.

However, regardless of the HIDS, there is no assurance that the system is completely secure and that attackers cannot exploit the honeypot; however, since the honeypot is an isolated system, the impact of such an incident is limited. By implementing robust security mechanisms and incorporating additional manual controls (e.g., limited storage space and computational power), a high level of confidence in the system's security can be achieved. In addition, our experiments have shown that the time span until an attacker exploits the system is in the range of days. This means that the risk of the honeypot being used as a dissemination facility for illegal content is limited when its monitored frequently.

7 SUMMARY AND CONCLUSION

Forensic readiness and its evaluation are important elements for a holistic security architecture. The presented approach can be used to analyze forensic readiness capabilities before an actual incident occurs or even before an attack simulation is conducted. It is based on cloning an existing system, sanitizing it to transform it into a honeypot, and deploying it in a cloud environment. The experiments carried out have shown that this approach is feasible, reasonably secure, and generates actual data for forensic investigations as attacks on the honeypots are conducted. Therefore, we consider our approach an important step in enabling a forensic readiness assessment using existing company systems and processes, especially for small and medium enterprises.

REFERENCES

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177. ACM.

Biedermann, S., Mink, M., and Katzenbeisser, S. (2012). Fast dynamic extracted honeypots in cloud computing. In *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, pages 13–18. ACM.

Dodson, M., Beresford, A. R., and Vingaard, M. (2020). Using global honeypot networks to detect targeted ics attacks. In *2020 12th International Conference on Cyber Conflict (CyCon)*, volume 1300, pages 275–291.

Kebande, V. R., Karie, N. M., and Venter, H. S. (2016). A generic digital forensic readiness model for BYOD using honeypot technology. In *2016 IST-Africa Week Conference*, pages 1–12.

Kelly, C., Pitropakis, N., Mylonas, A., McKeown, S., and Buchanan, W. J. (2021). A comparative analysis of honeypots on different cloud platforms. *Sensors*, 21(7).

Mansfield-Devine, S. (2018). The best form of defence–the benefits of red teaming. *Computer Fraud & Security*, 2018(10):8–12.

Oakley, J. G. (2019). *Professional Red Teaming: Conducting Successful Cybersecurity Engagements*. Apress.

Priya, V. D. and Chakkaravarthy, S. S. (2023). Containerized cloud-based honeypot deception for tracking attackers. *Scientific Reports*, 13(1):1437.

Setianto, F., Tsani, E., Sadiq, F., Domalis, G., Tsakalidis, D., and Kostakos, P. (2022). Gpt-2c: A parser for honeypot logs using large pre-trained language models. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, page 649–653, New York, NY, USA. Association for Computing Machinery.

Singh, K. D. (2021). Securing of cloud infrastructure using enterprise honeypot. In *2021 3rd Intl. Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 1388–1393.

Taiwo, A. and Claims, I. (2022). An extended digital forensic readiness and maturity model. *Forensic Science International: Digital Investigation*, 40:301348.

Tan, J. (2001). Forensic readiness. *Cambridge, MA:@ Stake*, 1. Publisher: Citeseer.

Verma, A. S. and Dubey, A. (2020). A review on honeypot deployment. *LJP London Journal of Research in Computer Science and Technology*, 20(1).

Washofsky, A. D. (2021). *Deploying and analyzing containerized honeypots in the cloud with T-Pot*. PhD thesis, Monterey, CA; Naval Postgraduate School.