Research Report

AI Enabled Honeypot

Conor Hendley
C00257507

Bachelor of Science (Honours)
in Cybercrime and IT Security

Supervisor: Christopher Staff

Table of Contents

1.0 Abstract

Honeypots are one of the most effective and proactive cybersecurity tools that are deployed in the fight against cybercriminals. However, honeypots can only be effective as long as they remain undiscovered by the attackers they are trying to deceive. This research investigates the common issues that traditional honeypots suffer, specifically in regard to their discoverability and attackers' abilities to fingerprint these systems. It also delves into current work being done with the utilisation of generative AI to solve these problems.

This research report begins by exploring traditional honeypot technologies, including the different interaction level models: low, medium, and high interaction levels. Then we look at the different honeypot deployment models and methods. Next is a deep dive into the issues associated with traditional honeypots, with emphasis on the challenge of detectability, along with the consequences that may arise if a honeypot system were to be detected.

The report then highlights the emerging applications of artificial intelligence in honeypot systems, examining previously accomplished systems that integrate LLM technology in their honeypots in order to produce realistic and contextually accurate responses to attacker inputs in an attempt to reduce the chance of an attacker fingerprinting the system bases on incorrect system behaviour and responses.

Finally, the report highlights some of the potential technologies that could be utilised for the development of an AI driven honeypot system. Including technology for SSH emulation, LLM response generation, logging, configuration storage, and deployment methods. Currently available honeypot frameworks are also explored with the goal of identifying popular options that would be suitable for comparison testing between it and an AI driven honeypot.

The findings of this research showcase a clear gap in current honeypot system capabilities. The need for a safe, highly realistic honeypot is extremely prevalent in the modern security world, and LLM technology may be the solution to these issues.

2.0 Introduction

Honeypots are powerful pieces of technology that assist cybercrime researchers, organisations, and security conscious individuals keep their systems secure from malicious actors. Honeypots are designed to deceive malicious actors (attackers) into interacting with them, this has benefits including but not limited to: **Protection of real systems**, honeypots help keep attackers engaged, while they waste time attacking a fake system the real system is safely kept unknown from the attacker. **Early warning system**, honeypots can act as early warning systems for the user, if an attacker is caught engaging with your organizations honeypot it could potentially be attributed to a larger more targeted attack on your organization or business. **Research and intelligence gathering**, honeypots are also incredible at assisting cybercrime researchers and organizations with gathering data on the attackers themselves. When an attacker in engaged with a honeypot system they become prime research subjects, researchers and organisations can monitor the attacker, observing the actions they take and the behaviours they present. This research and insight are invaluable as it allows for the creation of countermeasures to the attacker's tactics and highlights potential vulnerabilities in the real systems that the honeypot is mimicking. However, a problematic issue that has faced traditional honeypots since their adoption is that of fingerprinting. These days knowledgeable attackers can fingerprint and identify honeypot systems with relative ease and leave just as quick as they arrived. This severely limits the amount of actionable data that could be collected on the attackers' behaviours, as well as fails to keep them engaged and away from the live systems they were mimicking. This is where large language model technologies can help. Many low-level honeypots are static in nature; some may attempt to replicate the original system through pre-set outputs that are designed to produce an output based on the attacker's input. These fall short if an attacker issues a command that was not originally accounted for, regardless of its level of complexity. Other systems may set up to be a live system but without the defining information that the real system contains, these will often fall short as they do not contain believable information for the attacker and as such are fingerprinted. Both issues could be resolved by a correctly trained LLM.

This research report will aim to explore the potential of developing an “AI Enabled Honeypot” that is designed to take in attacker input and generate a realistic, contextually correct, and consistent response for it. The aim of the AI Enabled Honeypot will be to prevent fingerprinting from incorrect, confusing, or missing responses. This will ideally allow for longer attacker engagement with the honeypot leading to an increase into actionable data gathered on attacker behaviour and tactics.

This report will offer an in-depth look at honeypot systems and currently popular honeypot solutions available. This includes but is not limited to information on different types of honeypots, how they currently work, there advantages and disadvantages, preferable use cases of each type. It will also provide information on the impact working honeypots provided, along with a closer look at the short comings that honeypots present, specifically regarding attacker’s abilities to fingerprint these systems.

Following this, the report will focus on how a large language model (LLM) could help improve honeypot systems through the generation of realistic, contextually correct, and consistent responses to attackers' inputs in attempts to mitigate fingerprinting of the system. The report will cover items such as; research into past attempts to integrate AI with honeypots, suitable models to use for response generation, training of the model, integration of the model, and safeguards needed to ensure the model is not misused
Next, the report will examine the merits of building a honeypot system independently compared with modifying an open-source honeypot to work with an LLM response engine.

2.1 System Objectives

The success of the AI Enabled Honeypot will rely on its ability to produce realistic, contextually correct, and consistent responses to attackers' inputs. To achieve this result there are several main objectives that the honeypot system must achieved.

- **Generate Realistic Responses:**
  The AI Enabled Honeypot should produce terminal like outputs that resemble those of the system it has been designed to mimic.
- **Maintain Context Awareness:**
  Generated responses should reflect the current session state, previous issued commands, and outputs along with environment details should be considered when responses are generated to ensure the interaction remains coherent and logical.
- **Ensure Consistency:**
  Generated responses should follow a consistent narrative with previous generated responses; this will prevent contradictions and signs that the system is artificial.
- **Integration:**
  The developed AI response engine should be able to integrate with the chosen open-source (or independently developed) honeypot system, the response engine should be able to take over response outputs when required.
- **Believable Shell Prompts and TTY Behaviour**
  The emulated shell should provide key basic TTY behaviour to ensure that the interaction feels authentic, this includes items such as echo, accurate prompts, carriage returns. The exact emulation will depend on the service being mimicked.

3.0 Research Methodology

Several research methods will be employed during the creation of this research report. This will ensure that each section of this report is explored in thoroughly.

- **Academic Literature Review:**
  Academic papers such as *"Hypnotic Honey: Adaptive Honeypots Managed by Local Large Language Models"* by Lewis O. P. Childs and Mark A. Wood, will be reviewed to learn from previous applications of these technologies.
- **Technology Testing:**
  A hands-on approach to potential technologies being researched will be implemented for this research report. This will entail the set-up, configuration and testing of these technologies to determine if their use is appropriate for the project.
- **Real World Deployment Review:**
  Real world deployments of honeypots will be examined to better understand how the function in practice. Reviewing these deployments will help identify common shortcoming of existing systems and assist with identifying where the AI Enabled Honeypot can assist.

4.0 Overview on Honeypots

Honeypots are systems that are deliberately designed and created to look purposely vulnerable to outside users and potential malicious actors. These systems often function as traps and diversions for cybercriminals, drawing them in so that their activities and behaviours can be closely observed by security professionals. These observations allow security professionals to learn from the attackers, such as by exposing potential flaws in the real systems current security systems, or by allowing the attacker to deploy malware in the honeypot that may have been previously unseen. These observations then move onto shape future security policies and systems.

Honeypot systems are typically modelled after the legitimate asset they are designed to protect. For example: A security team at a large insurance company could deploy a honeypot system that resembles an insecure FTP server. This decoy server would be populated with fake but realistic looking data, such as fake customers PII, financial documents, internal reports and memos, etc. The data must appear real enough to entice an attacker to stay in the system for extended periods of time. For the honeypot to be successful it cannot appear completely vulnerable. It must mimic the security posture of the legitimate asset as close as possible. This is important for two reasons. First if the system appears fully open and unprotected, they attackers are likely not to trust it is real. Secondly organisations are given the opportunity to have their defences tested in a relatively safe environment. They can observe how the defences are bypassed so they can improve their legitimate systems security.

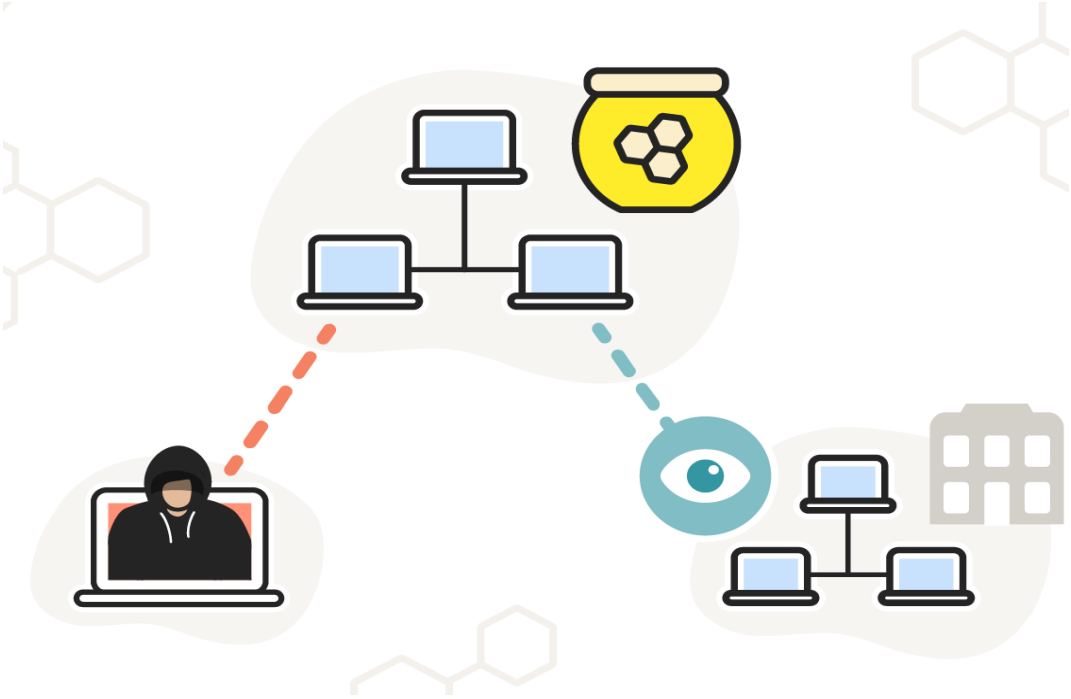


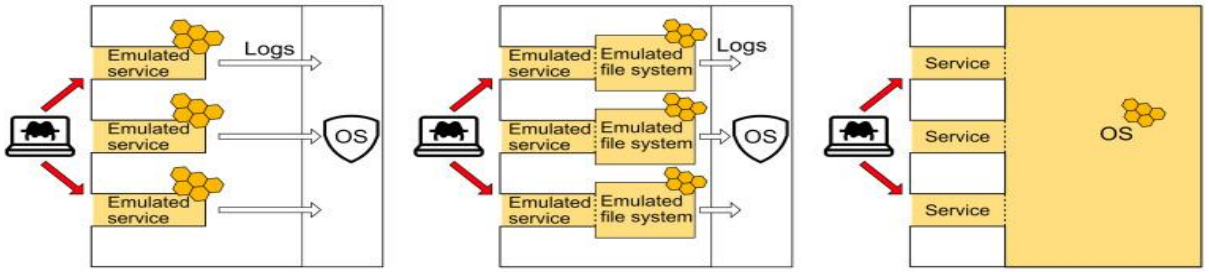
Figure 1: Simple Honeypot Diagram (Bodnar, 2023)

Honeypots are indispensable in the modern security landscape.
Dodson, er al. (2020) demonstrated the effectiveness of a large-scale honeypot deployment in their case study *Using Global Honeypot Networks to Detect Targeted ICS Attacks.* Their network of high scale honeypots allowed then to expose four separate zero-day attacks targeting industrial control systems, noting "We have demonstrated that a network of high-interaction honeypots can identify and profile previously unknown, targeted ICS attacks. Specifically, we exposed four zero-day attacks against devices running common ICS protocols such as S7comm and Modbus, which were disclosed to the applicable manufacturers." (Dodson, et al. 2020).
This case study highlighted how honeypots can massively assist safeguarding against targeted attacks on systems by skilled individuals

Honeypots can also help with research into botnet activity.
Mahmoud and Pedersen (2019) utilised a honeypot to "achieve current information about actual attempts of attacking universities". They deployed a low-level honeypot on Aalborg University's network in an attempt to discover more information about the attack behaviour against universities. There attempts showcase large numbers of attempted connections to the honeypot that they attributed to botnet activity based of the connections source Ip addresses. Studies such as this highlight the increased pressure organisations and institutions are under in terms of potential cyberattacks.

4.1 Honeypot Classifications

Honeypots can generally be put into three distinct classification types based off there interaction level.
These classifications are: Low Interaction Honeypots, Medium Interaction Honeypots, and High Interaction Honeypots.



(a) Emulated network services that are used as LH. (b) MH containing an emulated system to track attacker movement. (c) HHs deploy a real operating system for a realistic attack surface.

Figure 2: Honeypot classifications (Ilg, et al. 2023)

4.1.1 Low Interaction Honeypots

Low interaction honeypots typically won't simulate an operating system but will instead simulate the few services that are typically be used by attackers, as such they provided the least amount of interaction possible to attackers. This type of honeypot if ideal for catching credential brute-force attacks or the monitoring of connection attempts (Ilg, et al. 2023). Low interaction honeypots also have the benefit of requiring fewer resources needed to run and can be easier to deploy then higher interaction honeypots. Another benefit is the honeypots low risk of compromise since there is not real operating system to exploit. The main disadvantage of a low interaction honeypot is that experienced attackers will likely have the ability to identify these honeypots and avoid them. (GeeksforGeekes, 2025)

Examples of low interaction honeypots include: KFSensor (KeyFocus Security) or HoneyBot (Atomic Software).

4.1.2 Medium Interaction Honeypots

Medium interaction honeypots operate in a similar manner to low interaction honeypots in that they also do not emulate an operating system. However, unlike low interaction honeypots, which typically simulate only a single service or a set of services, medium interaction honeypots provide a simulated shell that allows attackers to run commands (Ilgm, et al 2023). These honeypots are designed to maintain attacker engagement for longer periods while keeping the risk of compromise low, as no real operating system is behind the scenes. While medium interaction honeypots are more complex to develop and deploy than low interaction honeypots, they are sizably less so than high interaction honeypots.

Examples of low interaction honeypots include Cowrie (Cowrie) or Thinkst Canary (Canary by Thinkist Applied Research)

4.1.3 High Interaction Honeypots

High interaction honeypots differ significantly from low and medium interaction honeypot systems because they provide the attacker with access to a real operating system that has been intentionally configured to appear vulnerable. These honeypots offer unrestricted access to the operating system, enabling attackers to execute commands, deploy malware, perform real privilege escalation and lateral movement. As a result, they provide greater insight into the attacker's movements, behaviours, and techniques (Ilgm, et al 2023) (Mahmoud, et al (2019).

High interaction honeypots are the most convincing type of honeypot and, as such, have the highest likelihood of successfully deceiving attackers. However, they are not without their significant drawbacks. They are the most complex and resource intensive honeypot to deploy and maintain due their complex system environments, along with the needed monitoring infrastructure and containment controls. More critically, high interaction honeypots also come with a sever security risk: if an attacker were to discover the honeypot, they could compromise the system and utilise the real operating system to launch attacks on the organisation’s network.

Examples of high interaction honeypots include: Cymmetria MazeRunner or TrapX DeceptionGrid

4.2 Honeypot Deployment Models and Architecture

Honeypots can be deployed in a variety of ways depending on the objectives and desired outcomes of the organisation or individual implementing them. Deployment methods may be production based or research based, client-side or server-side, distributed across networks hosted in the cloud or integrated into larger honeynet architectures.
Each deployment model offers clear advantages and limitations, and the suitability of a particular approach depends heavily on the stated purpose of the honeypot and the environment in which it will operate.

4.2.1 Production vs Research Honeypots

Production Honeypot Deployments
Production honeypot deployments are the more common deployment type between these two (Production and Research). They are effectively utilised by businesses and organisations to help increase the security of their live systems. These honeypot deployments are usually comprised of either low interaction honeypots or medium interaction honeypots, or a mix of the two. These deployments are configured to fulfil two main goals. First, they are effective early warning systems, alerting organisations’ security teams as early as possible to potential intrusions from attackers. Secondly, medium interaction level production honeypots are effective decoys for attackers, diverting attention away from the organisation’s live systems. Organisations will typically use low and medium interaction honeypots for their production deployments; this is because low and medium interaction level honeypots largely fill the goals set out in production deployments, while high interaction honeypots introduce

increased risk due to potential compromise and are more resource intensive to run. Production deployments will typically be continually monitored by a Security Operations Centre (SOC) team.
(Beres, 2022; Ilg, et al. 2023)

Research Honeypot Deployments
Research honeypot deployments are the less common deployment type between these two (Production and Research). They are utilised by research organisations and governments with the goal of collecting data on cybercriminals' and cyber attackers' behaviours and strategies. Unlike with a production honeypot deployment, where the focus is to have the honeypot be an early warning system for intrusions and as a diversion for attackers away from a live system. Research honeypots are primarily focused on a single goal: data collection. This data can take the form of testing new security systems, observing attacker behaviours, obtaining samples of attacker malware, etc. Research deployed honeypots typically benefit much more if they are high interaction honeypots. As previously mentioned, high interaction honeypots are designed to truly convince an attacker they have compromised a real system when in fact they are in a honeypot. Research deployments benefit from this more than production deployments, as research organisations need attackers to engage much longer with the honeypot to gather as much data as possible as opposed to just detection and distraction. High interaction honeypots also allow for the researcher to gather malware samples, as there is a real operating system being the honeypot where the malware can be deployed. The risks associated with a high interaction honeypot are still present, so research groups must keep this in mind during deployment and mitigate them as much as possible.
(Beres, 2022; Ilg, et al. 2023)

4.2.2 Honeynets

A honeynet is a collection of multiple interconnected honeypots deployed to observe and gather a larger and more diverse range of attack data. Honeynets are almost exclusively utilised for research purposes, as they require significantly more resources to deploy, manage, operate, and monitor than standalone honeypots. Honeynets extend beyond a single honeypot system. Instead of mimicking a single decoy system, a honeynet simulates an entire network environment consisting of multiple honeypot hosts, services, and infrastructure that attackers can interact with. This enables researchers to capture similar data to a high interaction honeypot deployment but at a much larger scale. Honeynets are typically comprised of high interaction honeypots due to their research nature.

The paper "An Overview of Honeypot Systems" outlines four core elements that make up a honeynet. (Titarmare, et al. 2019)

- Data Control - The control of the intruder's activity.
- Data Capture - The capture and recording of the malicious activity by the intruder.
- Data Collection - The storing and preserving of data in a centralised location.
- Data Analysis - The investigation of all collected data.

A great real world example of honeynets being deployed is with “The Honeynet Project”. From The Honeynet Project’s ‘About Us’ section:
“The Honeynet Project is a leading international 501c3 non-profit security research organization, dedicated to investigating the latest attacks and developing open source security tools to improve Internet security.”
(The Honeynet Project, 2025)
The Honeynet Project aims to promote awareness to the security community and general public on the threats and vulnerabilities that exit on the internet today; the deployment of honeynet systems is a large part of this research.

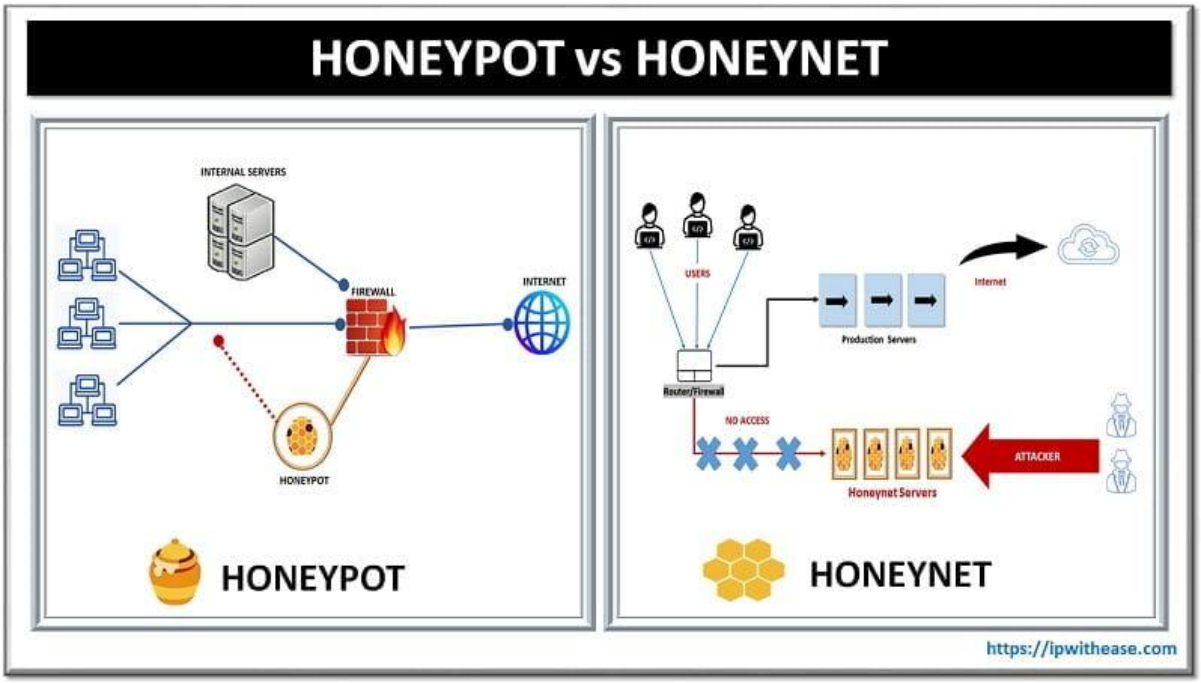


Figure 3: Honeypot Vs Honeynet Diagram (Bhardwaj, 2025)

4.2.3 Cloud deployed Honeypots

A cloud deployed honeypot is a honeypot that is deployed in a cloud environment, such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Computing (GCP). They still fall under the previous interaction classifications but are designed and implemented to attract attackers targeting cloud infrastructure rather than an individual’s or organisation’s network. These types of honeypots are incredibly important, as more and more companies are moving their workloads and services away from in house and into a cloud environment. Deploying a honeypot to the cloud is similar to deploying it onto your network and is done for similar reasons; as such, you could have production cloud based honeypots and research cloud based honeypots.

One such research deployment is documented in the analysis paper “A Comparative Analysis of Honeypots on Different Cloud Platforms”. In it, the authors deployed a wide range of low and medium interaction honeypots on various popular cloud platforms; these included Google Cloud, Microsoft Azure, and Amazon Web Services (AWS). The deployed honeypots were configured to expose commonly targeted services such as SSH, Telnet, and HTTP.

Their results showed the cloud honeypots receiving 1,168,996 attacks in the span of 21 days, with the Google Cloud honeypots receiving the most attacks; Microsoft Azure and AWS followed close behind. This work utilised honeypots to show the level of malicious activity that is consistently targeting cloud environments. Work such as this is incredibly valuable as organisations move to the cloud.
These findings also demonstrate the effectiveness of cloud deployed honeypots at capturing large amounts of real world attack traffic. Cloud services allow for honeypots to be exposed to many more attackers who are actively scanning these cloud environments looking for weaknesses such as exposed APIs or weak credentials. A honeypot deployed in one of these environments is well positioned to collect massive amounts of data on current attack trends against cloud environments. Cloud environments also enable users to deploy honeypots across multiple regions, allowing researchers to gather data on how attacks vary based on the location and local time of a system.
The analysis paper also demonstrates the flexibility and scalability that cloud environments enable for honeypot deployment. The authors deployed ten different honeypot systems across the previously mentioned three different cloud environments and in three geographic regions (North America, Europe, and Asia). Attempting to achieve this level of distribution using on premiss infrastructure would be extremely difficult, as it would require local infrastructure in each region. Cloud environments allow researchers to deploy globally distributed honeypots with relative ease, enabling large scale experiments such as the analysis paper.

There are still limitations and challenges with this approach. A large one being the operation costs associated with these deployments, researchers must pay cloud environments for compute power, storage, and data transfers. Also, researchers are at the will of the cloud environment operators'/owners' terms of service; operators may restrict adversarial research, as it can impose greater risk to the platform than regular use cases. Users deploying honeypots to the cloud must ensure they are familiar with the terms of service presented by the environment operations and must strive to stay compliant. Despite these issues, cloud deployed honeypots remain one of the most effective methods for researching attacker activity at scale, along with insights into specific cloud related attacks.
(Kelly et al. 2021, Brown et al. 2012)

4.3 Limitations, Challenges and Disadvantages of Honeypots

While honeypots are powerful and valuable security tools, they are not without their drawbacks. Any individual, business, or organisation considering the development and deployment of a honeypot must carefully consider the risks and challenges that come with their use. Whether it be potential fingerprinting, adversary takeover, or the legal and ethical issues of their use.

When implementing honeypots for either research or production purposes, there are several areas where they may fall short.
Attacker fingerprinting and honeypot discovery are critically dangerous possibilities that must be kept in mind. If an attacker can identify that they are interacting with a honeypot and

not a real system, then the effectiveness of the honeypot is completely diminished. This can occur due to simple mistakes, such as the misspelling of a word in a scripted response (Nisa, 2024) or system information not matching what the attacker would expect (such as little to no CPU usage).

If an attacker can identify that they are interacting with a honeypot, then they may attempt to utilise the honeypot system to perform real attacks on the live system. This risk is amplified if the honeypot that is discovered is a high interaction honeypot due to the real operating system that is utilised.
Another issue with honeypots is that they are only effective when attackers actually interact with them. Honeypots can be rendered completely ineffective if a skilled attacker can identify and avoid the system.

Along with the technical disadvantages that honeypots present, there are also legal and ethical issues with their use.

A significant legal issue that is consistently raised during honeypot deployments is the risk of entrapment. Entrapment refers to situations where a government or a government's law enforcement attempts to persuade or encourage an individual to commit a criminal act that they would have otherwise been unlikely to commit.
However, entrapment is likely to be an overstated issue when deploying honeypots. This is due to the entrapment criminal defence relying on a government or government agency actively encouraging criminal behaviour. In private honeypot deployments from individuals or organisations, entrapment has no bearing, as the government is not involved and the honeypot is being used for defence purposes and not for criminal prosecution. It should also be noted that entrapment as a criminal defence can vary by place. In Ireland an entrapment legal defence is not explicitly recognised as a valid defence, while in the United States there have been several cases where this defence has been used.
There are also issues involving data collection and privacy. Honeypots by their very nature collect large amounts of data; this can potentially be the main goal of a honeypot depending on the reasons for its deployment. Due to this data collection, the individual or organisation behind the deployment of a honeypot must remain in compliance with any and all data collection laws that apply in the location of its deployment. If deploying a honeypot within the European Union, the individual or organisation must be familiar with the General Data Protection Regulation (GDPR) and also the Data Protection Act 2018. Under these items of legislation, explicit permission for the collection of data is typically required. This presents an issue for honeypot deployment as by design these systems aim to collect data on users without their knowledge. In cases such as these, the individual or organisation deploying the honeypot must clearly define the purpose of the honeypot system, along with the data collection methods, the type of data being collected, the use of the data being collected and more. The individual or organisation behind the honeypot deployment must also ensure that the data being collected is the least amount possible to achieve their uses. As previously mentioned, the legislation around privacy and data collection will vary from place to place. In the United States, for example, legislation such as the Wiretap Act and the constitutional 4th Amendment may interfere with the deployment of honeypots; this, however, is in regard to a

government or government agency honeypot and will have limited impact on a privately deployed honeypot.
(Mokube, et al. 2007; Nisa, 2024; Naeem, 2021)

4.4 Honeypot Fingerprinting

As previously mentioned, the value of a honeypot relies nearly exclusively on its ability to be indistinguishable from a real system to a potential attacker. If a honeypot system were to be identified by an attacker, the system, at best, would lose nearly all its value, with the attacker being able to simply evade the honeypot, and at worse, the attacker could leverage the honeypot system to perform an attack on the real system the honeypot is attached to and protecting.

Honeypot fingerprinting is the process of revealing that the system being interacted with is in fact a honeypot. There are two main techniques deployed when attempting to fingerprint a honeypot: active probing and passive probing.
Active probing involves an attacker creating specific probes or messages and using them to directly query the suspected honeypot. The responses to these probes are then analysed by an attacker in order to gather as much data on the systems as possible. By analysing the header fields of these responses, information such as the systems operating system can be discovered. Other system specific identifiers include the Initial Congestion Window (ICW) and Retransmission Time Out (RTO). This technique is primarily based on both the TCP and ICMP protocols. Popular tools for active probing include Nmap, Metasploit, and Hydra.
Active probing is not without issues; the paper “Evaluation of Fingerprinting Techniques and a Windows-based Dynamic Honeypot” lists several issues with active probing:

- Nmap and similar tools need to keep requesting answers from the honeypot client to work in real time. As such, if the honeypot has a firewall in place, the targeted honeypot may not be able to respond.
- Active fingerprinting can be blocked by network protection tools or could set off an intrusion detection system.
- Active fingerprinting can generate large volumes of data, causing the targeted system to accidentally be forcibly shutdown.

Passive probing, on the other hand, works differently than active probing in order to achieve the same result. Passive probing attempts to obtain similar data to active probing, but instead of sending active probes to the target system, it involves “sniffing” the target systems network for packets transmitted by the honeypot. This data, similarly to active probing, will be analysed to determine information on the system, such as operating system information that may give away the honeypot, similarly to the active probing. A form of passive probing is metascan based fingerprinting. This form of passive probing leverages a systems known Ip address and an internet mass scan engine to determine information such as the hosting provider and the internet service provider that leased the Ip address. This information can then be analysed to determine if the associated system is real or a honeypot. For example, a

discovered vulnerable system with an Ip address allotted to a university or known research centre would highly suggest the system is a research honeypot. These mass scan engines can also reveal information such as system geographic location and information on if the Ip address is likely assigned to an autonomous system. All this information will assist in the fingerprinting of the honeypot.

The paper “Gotta Catch ’em All: A Multistage Framework for Honeypot Fingerprinting” conducted research into honeypot fingerprinting and made attempts to fingerprint several leading commercially available honeypots. They utilised both active probing and passive probing. Their active probing pipeline consisted of seven probing stages and serves as a very well put together flow guide on how active probing can identify a honeypot.
Below are the stages outlined in the paper in order to provide context on how this fingerprinting could occur in a real world scenario.

- Portscan: This is the beginning of the active probing. In the paper the authors purposely scan for ports that are specific to services commonly emulated by honeypots. This is still viable for an attacker to perform, as these are also the services that are most commonly attacked.
- Banner Check: The results of this port scan are then processed via banner check. Static banners that are provided by default and hard coded in certain honeypot implementations can instantly fingerprint a honeypot system.
- HTTP Static Response: Next a honeypot that's mimicking a web service can be checked for static HTTP responses; these static responses can then be compared to known honeypot responses, and the honeypot can be fingerprinted.
- SSL/TLS Certificate Check: At this stage, you can check for certification specific attributes to know values from honeypots. Even though there is a fingerprint change for each certificate, certain attributes, such as issuer and provider, will remain static, thus fingerprinting the honeypot.
- Protocol Handshake: Many honeypots will fail to offer a typical range of protocol options for handshake and will instead rely on a limited number of protocols in simplified forms due to either design constrains of limitations on the libraries used.
- Library Dependency Check: Many low and medium interaction honeypots are developed with the use of external libraries. Since the libraries offer limited emulation capabilities and some are not well maintained, tools have been developed to probe honeypots using these libraries to return a static response.
- Static Command Response: Due to the nature of honeypots developers must include static responses or disconnect communication when an unexpected but correct command is issued by the attacker.

A similar pipeline was compiled for their passive probing attempts.

- Mass Scan Engines: As previously discussed, mass scan engines such as Shodan and Censys Search continually scan the public internet in order to catalogue systems with open ports and exposed services. An attacker can query these databases for devices running commonly emulated honeypot services. The attacker can then take the returned Ip addresses and ports to continue further fingerprint analysis.
- Keyword Search: Mass scan engines can also store metadata on exposed systems such as banners, protocol negotiation information, geographic location, and hosting provider information. Attackers can perform keyword searches against these databases for values that are statically set by honeypots.
- Cloud Hosting Check: Attackers can check if a service is being hosted in a cloud environment or not by examining the Ip address of a potential honeypot and comparing it to known Ip cloud ranges. Attackers can then see if the service being hosted on the cloud makes logical sense for it to be there. For example, honeypots that emulate ICS/SCADA systems are industrial control systems that would almost never be hosted on a cloud. If it makes no logical sense for the service to be cloud hosted, it is likely to be a honeypot.

Both these pipelines showcase how active and passive probing can fingerprint a honeypot before any attacker even begins to attack the system.

There are, of course, limitations to both active and passive probing.
As active probing relies on the honeypot system producing consistent static results, if the honeypot were to have the capability to produce dynamic outputs, then the effectiveness of these probes would be severely diminished. Output should also be modified by the user deploying the honeypot system, as default honeypot responses can be cross referenced by attackers, as previously mentioned. Additionally, when developing a honeypot, it is important to not become overly reliant on publicly available libraries and also to ensure that the libraries that are being used are regularly maintained and up to date.
Passive probing can be diminished through the rotation of Ip addresses to avoid Ip identification by mass scan engines. Users should also be mindful when deploying honeypots to a cloud environment that the service being mimicked is a realistic service that is appropriate to be hosted in the cloud.
(Srinivasa et al. 2023, Mohammadzadeh et al. 2013)

Along with probe based fingerprinting, an attacker may also be able to fingerprint a honeypot system based on behavioural fingerprinting.
Behavioural fingerprinting can occur when an attacker connects to a honeypot system; it relies on an attacker being knowledgeable on what the real system the honeypot is attempting to mimic should appear and behave like. This will include items such as timing delays, response speed, lack of system load, and no background activity. It can encompass anything that a knowledgeable attacker may notice during their attack.

4.5 AI Honeypots

As previously discussed, while honeypots have significant utility in both research and production environments, they are not without significant limitations. Many research honeypot deployments often required high interaction honeypots in order to collect detailed data on attacker behaviour and tactics. However, these honeypot systems are expensive to both deploy and maintain; they also carry substantial risk due to the possibility of detection and subsequent compromise. Conversely, many production honeypots tend to be low or medium interaction level honeypots in order to minimise the potential risks that are associated with high interaction honeypots. While safer, their imitated functionalities can make them easier to be identified by attackers, resulting in short engagement times and reducing the potential research value.

AI driven honeypots attempt to address several of these issues through one solution, provide believable and realistic outputs without the implementation of an operating system. Through leveraging generative AI models to produce dynamic, realistic, and contextually aware responses to attacker input, AI driven honeypots can present a realistic and convincing environment without requiring a real operating system behind the honeypot. This approach has the potential to increase attacker engagement beyond what traditional medium interaction level honeypots can offer, while also avoiding the risks associated with high interaction honeypots.
(Sladic et al. 2024, Lewis et al. 2024, Sezgin et al. 2024)

4.5.1 Issues with Traditional Honeypots

AI driven honeypots attempt to solve several issues from both medium interaction honeypots and high interaction honeypots. Low interaction honeypots are generally unsuitable for AI enhancement due to their use cases typically consisting of the measurement of bot activity against a system rather than to engage human attackers.

High Interaction Honeypot
High interaction level honeypots offer the highest level of realism to the attacker, allowing for deep engagement and a greater increase in the level of behavioural data that could potentially be collected. However, this realism is not without its costs. First are the deployment and operational costs; due to attackers interacting with a fully functional operating system, these honeypot deployments require strong virtualisation, continuous monitoring, and infallible containment mechanisms to prevent discovery and takeover. If these honeypots were to be fingerprinted, then there is a possibility that a skilled attacker could compromise the operating system behind the honeypot and utilise it for attacks against the real system or outside of it. Because of these issues high level honeypots are typically resigned to research environments.

Medium Interaction Honeypot
Medium interaction level honeypots are safer and cheaper than high interaction honeypots to deploy and run but offer a lower level of realism to an attacker. Medium interaction honeypot system behaviour is largely scripted and emulated; as such, skilled attackers can often

recognise when they are interacting with them through observing items such as inconsistencies in provided system responses, missing system files, static banners, incorrect error messages, etc. Once detected, the attacker will typically fully abandon the system. This results in minimal engagement, reducing the honeypots' ability to collect behavioural and tactical data. It also enables the attacker to discover that honeypots have been deployed, allowing them to maintain high vigilance if proceeding with attempts to gain access to the targeted system.

Honeypot Fingerprinting
Honeypot fingerprinting techniques have advanced alongside honeypot technologies. Tools such as Nmap, Xprob, and mass scan engines (i.e. Shodan or Censys) make it easier for attackers to identify systems as likely honeypots through items such as mismatched protocol negotiations, cloud hosting inconsistencies and abnormal TCP/Ip signatures. As fingerprinting techniques and technologies become more sophisticated, traditional honeypots become increasingly ineffective.

New Approach
These issues and limitations highlight the need for a new approach to honeypot design. A design that can provide the necessary realism of high interaction honeypot, with the lost risks and costs of a medium interaction honeypot. AI enabled honeypots provide a promising solution to this problem.
(Sladic et al. 2024, Lewis et al. 2024, Sezgin et al. 2024)

4.5.2 Examples of AI Honeypots

In recent years researchers have begun exploring the use of artificial intelligence within honeypot systems in order to address the long standing limitations of traditional honeypots that were previously discussed. There have been several notable AI driven honeypot implementations and proof of concept systems presented by researchers. These implementations demonstrate how generative AI can increase the realism, reduce detections, and extend attacker engagements.

LLM in the Shell: Generative Honeypots
This research paper, written by a group of researchers from the “Czech Technical University in Prague” and “Conicet Mendoza” in Argentina, documents the creation and implementation of a shell emulation honeypot with responses generated by a Large Language Model.
The resulting honeypot system was named shelLM The goal of shelLM was to successfully emulate a “credible and realistic Linux shell that is engaging to attackers and, therefore, may delay the realization that they are not interacting with a real Linux shell” (Sladic et al. 2024). ShelLM was developed in Python and designed to “work seamlessly with a typical SSH setup”. The researchers made use of prompt engineering techniques to guide the behaviour of their honeypot; a cloud based LLM was provided with an initial prompt at the beginning of each session. This initial prompt actually contained two different prompt sections first a personality prompt that instructed the LLM to “be (i) precise, (ii) realistic, (iii) not to disclose its internal details, (iv) describes the expected behaviour of a Linux shell, and (v) provides examples of desired output under certain situations”.

The second part, or the “thinking process prompt”, instructed the LLM to “’think step-by-step’ following the Chain of Thought (CoT) prompt technique, in combination with the few-shot technique, and repeated orders to enforce this behaviour”. Session management was broken down into four main steps. (1) The user would input a command, (2) The LLM prompt would be created, the prompt would include the initial prompt and the command entered, (3) The LLM would process the prompt and generate a response, (4) The reposes would be returned and displayed to the user. ShelLM was tested by a focus group of 12 cybersecurity experts, each of whom prompted the shelLM honeypot with commands ranging from common to complex. The group was then asked if the output was what they would expect it to be. Results of this testing were very positive, concluding that shelLM 90% of the time the response was an expected output.

Hypnotic Honey: Adaptive Honeypots Managed by Local Large Language Models

This research paper was produced by two researchers from Montvieux. Similarly to the first paper examined, the researchers developed and produced a honeypot system where the system responses are generated by an LLM This was done for the same reasons as the previous paper discussed as well as to find a solution to the previously stated issues with traditional honeypots. Hypnotic Honey, however, had a few key differences from shelLM. First, Hypnotic Honey was built from the ground up to make use of a local LLM response engine instead of a cloud based one. Its interaction flow remained similar to that of shelLM, an attacker would interact with an emulated terminal where they would issue a command; this command would be parsed and sent to the LLM, the LLM response would be processed and returned back to the attacker’s screen. Hypnotic Honey also consisted of several integration levels; the set integration level would determine the level of LLM involvement with the honeypot. MAX integration would allow the LLM to handle every input fully. High would allow the LLM to handle every input but would split multiple commands apart that were inputted on the same line. Medium would allow for some of the commands to be processed outside of the LLM with a pre scripted response. The hypnotic honeypot was evaluated using automated adversaries, such as The Caldera platform from the MITRE Corporation. The developers of Hypnotic Honey concluded that LLMs can be utilised to convincingly emulate a shell environment capable of extending attacker engagements; they also provided work showcasing that a hybrid model that consists of LLM output and pre scripted output can be more stable than pure LLM response.

5.0 Potential Technologies

For this project there are several potential technologies that could be used to develop an AI driven honeypot, these include which Large Language Model would best suit the needs of the project, SSH emulation technologies, prebuilt honeypot framework technologies, logging technologies, programming languages, and deployment environments. This section evaluates a range of technologies to assist in determining which would be best suited to the development of the system.

5.1 Large Language Models

The Large Language Model (LLM) will be a core component of the honeypot system; it will be responsible for generating the realistic responses to the attacker input. There are two main options that would be appropriate for this use case. First are locally hosted open source models such as Llama 3 or Mistral. Local models offer significant advantages for a honeypot use case, such as offline operations, full data privacy, and greater control over model behaviour. Also, the risk of inadvertently transmitting the attacker input externally is reduced through the use of a local model. Local based model use is seen in the Hypnotic Honey system. The second option is a cloud hosted API mode such as OpenAI's GPT models. Cloud models are not reliant on local hardware and, as such, can provide higher performance than locally based models; they also can be more straightforward and simpler to implement. However, cloud modes can introduce latency issues as the data must be transported off site and the response transported back to the system. There also may be privacy concerns similarly to other cloud hosted applications. A cloud hosted model was used in the shelLM honeypot.

5.2 SSH Emulation Technologies

The conceptual design of the AI honeypot system for this project is an SSH interface honeypot. As such, attackers will require an SSH interface to interact with. There are several Python based SSH libraries that can help support this functionality. One such library is Paramiko. Paramiko provides a straightforward implementation of custom SSH servers and is a popular choice for security researchers. The Paramiko library was used in the HoneyPy project.

5.3 Pre Built Frameworks

Prebuilt frameworks such as HoneyPy, Cowrie, or Dionaea offer prebuilt honeypot functionalities that could assist in reducing development time. HoneyPy in particular is a simple open source medium interaction SSH honeypot that has potential to be modified to work with an AI response engine relatively easily. Cowrie is also an option; it supports SSH and command emulation, filesystem emulation, and logging. However, implementing an AI

response engine into a complex framework such as this may increase the difficulty of the project. However, cowrie may be a good system to use for generating comparison data between how it performs and how the project honeypot performs.

5.4 Deployment Environment

It is of the upmost importance that this project system be run in a fully controlled and isolated environment due to safety concerns. Virtual machines provide strong containment and also allow for the honeypot to be reset quickly if the need arises. Docker may also be a viable choice, as it provides fast setup and easier portability. Cloud deployment was looked into, but the costs associated with cloud deployment were significant. The honeypot should also never be fully exposed to the public facing internet and only used as a proof of concept system.

5.5 Other Technologies

Additional tools that may be used in this project include SQLite for log storage and configuration presets, logging frameworks such as Python's logging module, and further Python libraries such as Pandas for assistance with summary report generation.

6.0 Conclusion

This research document has examined the evolution, deployment models, issues, limitations, and fingerprinting techniques that are associated with traditional honeypots. These findings showcase that while honeypots are incredibly important tool for both security research and security operations, they suffer from several weaknesses that limit their effectiveness in these tasks. Specifically, traditional honeypots remain highly vulnerable to detection and fingerprinting methods employed by attackers, including both probe based fingerprinting and behavioural fingerprinting. Once detected, the consequences can be severe: at best the attacker will immediately disengage from the honeypot system, thus reducing the quantity of behavioural data collected, while also alerting them to the existence of honeypots on the network. At worst, in instances where high interaction honeypots have been deployed, an attacker may attempt to perform a system takeover of the honeypot and utilise its resources for attacks.

This research highlights that these issues often occur due to the limited realism offered by medium interaction honeypots along with the significant risk associated with high interaction honeypots. This research shows a clear gap in existing honeypot technology, in so that there is currently no honeypot system that can provide both the realism of high interaction honeypots and the security of medium interaction honeypots.

Recent developments in AI driven honeypots, specifically those leveraging Large Language Modules, present a promising solution. LLMs can be utilised to generate dynamic, contextually aware, and realistic responses that can prevent system behavioural fingerprinting performed by attackers. Existing AI honeypot proof of concepts have demonstrated the ability of LLMs to deceive security professionals into believing that the system they are interacting with are real systems and not honeypot systems. As such AI driven honeypots have the potential to allow for greater data to be gathered on attackers' behaviours and tactics, as well as increase the effectiveness of defence operation deployments.

The information gathered in this research report has helped establish the strong foundation necessary to begin designing the project's practical AI driven honeypot system. This research not only highlights the limitations and issues of traditional honeypots, it also showcases the ability of LLMs to assist with overcoming these limitations. This research will help guide the development, implementation and evaluation of the AI driven honeypot that is to be built for this project.

7.0 References

Sezgin, A. and Boyacı, A. (2025) *DecoyPot: A large language model-driven web API honeypot for realistic attacker engagement*, *ScienceDirect*. Available at: https://www.sciencedirect.com/science/article/pii/S0167404825001476 (Accessed: 17 November 2025).

Román, J.J., Almenares-Mendoza, F. and Sánchez-Macián, A. (2025) *Design and development of an intelligent LLM-based LDAP Honeypot.* Available at: https://www.arxiv.org/pdf/2509.16682 (Accessed: 17 November 2025).

Franco, J. *et al.* (2021) *A survey of honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems*. Available at: https://arxiv.org/pdf/2108.02287 (Accessed: 17 November 2025).

Childs, L. and Wood, M. (2024) *Hypnotic honey: Adaptive honeypots managed by local large language models.* Available at: https://ieeexplore.ieee.org/abstract/document/10941449/ (Accessed: 17 November 2025).

Moric, Z., Dakic, V. and Regvart, D. (2025) *Advancing Cybersecurity with honeypots and deception strategies*. Available at: https://www.researchgate.net/publication/388666014_Advancing_Cybersecurity_with_Honeypots_and_Deception_Strategies (Accessed: 19 November 2025).

Srinivasa, S., Pedersen, J.M. and Vasilomanolakis, E. (2021) *Gotta catch 'em all: A multistage framework for honeypot fingerprinting*, *arXiv.org*. Available at: https://arxiv.org/abs/2109.10652 (Accessed: 19 November 2025).

Moric, Z. *et al.* (2024) *Honeypots in cybersecurity: Their analysis, Evaluation and importance*, *Preprints.org*. Available at: https://www.preprints.org/manuscript/202408.0946 (Accessed: 19 November 2025).

Anon (no date) *Cowrie/Cowrie: Cowrie SSH/telnet honeypot*, *GitHub*. Available at: https://github.com/cowrie/cowrie (Accessed: 10 November 2025).

Collins, G. (2024) *Build a Honeypot to Trap Hackers & Bots in Python (Crash Course)*, *YouTube*. Available at: https://www.youtube.com/watch?v=gDjDxS55890 (Accessed: 10 November 2025).

Collins, G. (no date) *Collinsmc23/SSH_HONEYPY: A basic SSH honeypot to capture IP addresses, usernames, passwords, and commands.*, *GitHub*. Available at: https://github.com/collinsmc23/ssh_honeypy (Accessed: 10 November 2025).

Titarmare, N., Hargule, N. and Gupta, A. (2019) *An overview of honeypot systems*, *ResearchGate*. Available at: https://www.researchgate.net/publication/332113726_An_Overview_of_Honeypot_Systems (Accessed: 18 November 2025).

Agarwal, S. (2025) *What is honeypot?*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/blogs/what-is-honeypot/ (Accessed: 12 November 2025).

Vaideeswaran , N. (2025) *What is a honeypot in cybersecurity?*, *CrowdStrike*. Available at: https://www.crowdstrike.com/en-us/cybersecurity-101/exposure-management/honeypots/ (Accessed: 12 November 2025).

Mahmoud, R.V. and Pedersen, J.M. (2019) *Deploying a University Honeypot: A case study*, *ResearchGate*. Available at: https://www.researchgate.net/publication/342304643_Deploying_a_University_Honeypot_A_case_study (Accessed: 24 November 2025).

Dodson, M., Vingaard, M. and Beresford, A.R. (2020) *Using Global Honeypot Networks to detect targeted ICS attacks*, *CyCon*. Available at: https://ccdcoe.org/uploads/2020/05/CyCon_2020_15_Dodson_Beresford_Vingaard.pdf (Accessed: 25 November 2025).

Agrawal, R. et al. (2023) *Long-term study of honeypots in a public cloud, Microsoft.* Available at: https://www.microsoft.com/en-us/research/wp-content/uploads/2022/11/AgrawalHoneyPotDsn2022.pdf (Accessed: 25 November 2025).

Bodnar, D. (2023) *What is a honeypot and how does it work?*, *Norton*. Available at: https://us-stage.norton.com/blog/iot/what-is-a-honeypot (Accessed: 26 November 2025).

Anon (2020) *Production vs research honeypots: What's the difference?*, *Logix Consulting Managed IT Support Services Seattle*. Available at: https://logixconsulting.com/2020/06/22/production-vs-research-honeypots-whats-the-difference/ (Accessed: 26 November 2025).

Beres, I. and Hurley-Smith, D. (2022) *Dynamic honeypot deployment in the cloud*, *ResearchGate*. Available at: https://www.researchgate.net/publication/360541079_Dynamic_honeypot_deployment_in_the_cloud?channel=doi&linkId=627cde42b1ad9f66c8b82ef8&showFulltext=true (Accessed: 26 November 2025).

Mohammadzadeh, H., Mansoori, M. and Welch, I. (2013) *Evaluation of Fingerprinting Techniques and a Windows-based Dynamic Honeypot*. Available at: https://www.researchgate.net/publication/262273650_Evaluation_of_fingerprinting_techniques_and_a_windows-based_dynamic_honeypot (Accessed: 29 November 2025).

Niclas, I. *et al.* (2023) *A survey of contemporary open-source honeypots, frameworks, and tools*, *ScienceDirect*. Available at: https://www.sciencedirect.com/science/article/pii/S108480452300156X (Accessed: 25 November 2025).

Nisa, Z. (2024) *Honeypots: Concepts, Approaches, and Challenges*, *ResearchGate*. Available at: https://www.researchgate.net/publication/378164367_Honeypots_Concepts_Types_and_Challenges (Accessed: 29 November 2025).

Anon (2016) *Entrapment - definition, examples, cases, processes*, *Legal Dictionary*. Available at: https://legaldictionary.net/entrapment/ (Accessed: 25 November 2025).

Roxana, M. (2023) *The defence of entrapment in Ireland*, *Academia.edu*. Available at: https://www.academia.edu/101331263/The_defence_of_Entrapment_in_Ireland (Accessed: 25 November 2025).

Naeem, A.A.N. (2021) *Honeypots: Concepts, Approaches and Challenges*, *HAL open science*. Available at: https://hal.science/hal-03324407/document (Accessed: 28 November 2025).

Chamoli, A. (2025) *Honeypot vs Honeynet*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/ethical-hacking/honeypot-vs-honeynet/ (Accessed: 26 November 2025).

Lackner, P. (2021) *How to Mock a Bear: Honeypot, Honeynet, Honeywall & Honeytoken: A Survey*. Available at: https://www.scitepress.org/PublishedPapers/2021/104000/104000.pdf (Accessed: 26 November 2025).

Adeel, M. *et al.* (2005) *Honeynets: An architectural overview*, *IEEE Xplore*. Available at: https://ieeexplore.ieee.org/abstract/document/4382883 (Accessed: 27 November 2025).

Anon (no date) *About Us*, *The Honeynet Project*. Available at: https://www.honeynet.org/about/ (Accessed: 29 November 2025).

Bhardwaj, R. (2025) *Honeypot vs honeynet: Complete guide*, *IP With Ease*. Available at: https://ipwithease.com/honeypot-vs-honeynet-complete-guide/ (Accessed: 29 November 2025).

Kelly, C. *et al.* (2021) *A comparative analysis of honeypots on different cloud platforms*, *MDPI*. Available at: https://www.mdpi.com/1424-8220/21/7/2433 (Accessed: 01 December 2025).

Brown, S. et al. (2012) *Honeypots in the cloud.* Available at: https://pages.cs.wisc.edu/~sbrown/downloads/honeypots-in-the-cloud.pdf (Accessed: 01 December 2025).

Sladi, M. *et al.* (2024) *LLM in the Shell: Generative Honeypots*. Available at: https://arxiv.org/pdf/2309.00155 (Accessed: 01 December 2025).