Hunting High or Low: Evaluating the Effectiveness of High-Interaction and Low-Interaction Honeypots

Yekta Kocaogullar[1], Orcun Cetin[1], Budi Arief[2], Calvin Brierley[2], Jamie Pont[2], and Julio Hernandez-Castro[2]

[1] Sabanci University, Turkey,
{ykocaogullar, orcun.cetin}@sabanciuniv.edu
[2] University of Kent, UK,
{b.arief, c.r.brierley, j.pont, j.c.hernandez-castro}@kent.ac.uk

Abstract.
Background. Honeypots are cybersecurity mechanisms that are set up as decoys in networks to lure and monitor attackers trying to compromise vulnerable systems. Two commonly used honeypot designs are high-interaction and low-interaction honeypots, which differ in the amount of interplay that the attackers are allowed to do. So far, the effectiveness of high-interaction and low-interaction honeypots has been understudied, making it difficult for security teams to choose between different honeypot technologies.
Aim. The aim of this paper is to compare the effectiveness of high-interaction and low-interaction honeypots through real-world data.
Method. We deployed multiple Elasticsearch honeypot implementations to collect data: a closed-source high-interaction honeypot developed by the authors, and three types of open-source low-interaction honeypots (namely Elastichoney, Delilah and Elasticpot). The collected data came from 48 instances of high-interaction honeypots and 111 instances of low-interaction honeypots, over a period of 14 days.
Results. We found that low-interaction honeypots captured only a fraction of the attacks that high-interaction honeypots can catch. On the other hand, low-interaction honeypots are simpler, more efficient to run due to their low usage of resources, and easier to deploy. In our dataset, high-interaction honeypots captured 76.12% of the total attack packets and attracted 70.61% of the unique attacker IPs. In comparison, low-interaction honeypots performed a lot worse in collecting attack data; they only managed to capture 23.88% of the total attack packets and attracted 29.39% of the unique attacker IPs.
Conclusions. In this paper, we present an experiment that evaluated and compared the effectiveness of high-interaction and low-interaction honeypots in terms of the amount and the type of information collected from attacks targeting them. It follows from our findings that it would be wiser to either concentrate solely on using high-interaction honeypots, or to increase the effectiveness of low-interaction ones by automatically changing each static value during deployment and/or by increasing the mimicking capabilities of low-interaction honeypots.

Keywords: Security · honeypot · high-interaction · low-interaction · decision-making · comparative study

1 Introduction

Honeypots are one of those essential cybersecurity systems that can be useful in detecting network compromises, while learning the behaviour of the attackers at the same time. These systems are set up to mimic vulnerable assets that could be infiltrated by attackers. A honeypot typically has monitoring and logging capabilities that would allow security researchers (the honeypot operators) to gather information about attackers' behaviour, including the tools and exploits used by these attackers. In other words, a honeypot can be thought of as a "digital network bait" which is used to discover potential attackers – both insiders and remote [17] and to gather evidence about them.

Honeypots can serve as a valuable tool, as long as attackers interact with them [14]. In the simplest form, there are two different categories of honeypot interaction: high and low. High-interaction honeypots offer a fully functional decoy system that can be compromised by the attackers [20]. This allows high-interaction honeypots to collect more information regarding attackers' behaviour, as well as their attack tools [17]. However, high-interaction honeypots are harder to develop – and typically more resource-intensive – in comparison to low-interaction honeypots.

On the other hand, low-interaction honeypots provide attackers with minimal access – for example, they do not let attackers to access the operating system but instead they provide some minimalist implementation of a limited number of Internet protocols and services [14]. This limitation – while reducing the possibility of the honeypot getting completely compromised by the intruder – also restricts the honeypot's ability to emulate the full functionality of a vulnerable system [17]. Low-interaction honeypots are also usually easier to develop, deploy and maintain, as they require less computational resources.

Challenges and motivation. Nowadays, some attackers are becoming increasingly aware of the presence of honeypots. As they try to navigate their way through their victim's devices, those attackers would keep an eye for honeypot-like features on the systems they hit, in order to avoid being monitored – or even worse, captured [30]. They would also like to avoid wasting their time and effort on not-real systems, and naturally, they do not want to give away valuable and incriminating information about themselves.

At the same time, security community knows remarkably little about which type of honeypot would attract more attackers, and how much of the attackers' data can be captured by using either the high- or low-interaction honeypots.

This means that any claims within the security community that high-interaction honeypots are outperforming low-interaction honeypots currently lack numerical evidence. Furthermore, the extent of such a proposed performance difference is currently unexplored. These observations provide the key challenges and motivation for the research we present in this paper.

Most importantly – and in relation to the socio-technical aspects of security – this gap in knowledge makes it difficult for practitioners (with varying degrees of expertise) to make decision whilst trying to build a secure system. This is especially true for real-world systems, in which diverse sets of needs would come up, due to different budgets, environments and demands. Furthermore, not knowing enough about how high- and low-interaction honeypots would perform in the real-world setting may lead to a false sense of security by the users of such systems. As a consequence, this false sense of security may increase the possibility of errors due to the human element.

To address the challenges above, this paper compares the effectiveness of high-interaction and low-interaction honeypots in terms of attracting attacks in a real-world setting. We conducted an experiment in which we contrasted network traffic data captured by one custom high-interaction honeypot group and three popular low-interaction ones, within a 14-day period. We deemed this 14-day observation period would be sufficient, since first and foremost, we were only interested in the initial and short term effects, rather than the long term and historic data of honeypot operations (the latter is a valuable research in its own right, but it is beyond the scope of the research we wanted to focus on here).

Contributions. The main contributions of our paper are:

- we analyse and compare the efficacy of high- and low-interaction honeypots based on the data we collected from our experiment;
- we provide some insights into the geographical spread of potential attackers;
- finally, we come up with a set of recommendations that may help security researchers in choosing the type of honeypots suitable for their work.

The rest of the paper is organised as follows. Section 2 provides some background information regarding the honeypot systems we used in our study, as well as our rationale in using Elasticsearch for this study. Section 3 focuses on our deployment methodology and our rule set for evaluating incoming attacks. Section 4 presents the results by (a) comparing the performances of high- and low-interaction honeypots, (b) comparing the three types of low-interaction honeypots we used with each other, and (c) analysing the geographical locations of the attackers. Section 5 outlines and briefly discusses related work, while Section 6 concludes our paper and provides several ideas for future research.

2 Elasticsearch Honeypots

Elasticsearch is a NoSQL database that can store, search, and analyse large amounts of data [22]. It currently has eight common vulnerabilities and exploits (CVEs) listed on the Exploit Database [5]. While six of these CVEs are from 2014 and 2015 and are currently legacy problems affecting older versions, two of them are relatively recent, from July 2021 [5].

In 2021, Paganini scanned 334,013 servers that used port 9200 and discovered that 9,202 were running instances of Elasticsearch and 5,740 were accessible

without any authorisation [23]. Elasticsearch servers are heavily targeted by cyber criminals to steal and ransom victims' data. In another recent example, an attacker wiped and defaced more than 15,000 Elasticsearch servers and tried to pin the blame on Night Lion Security, a US cyber-security firm [10]. These vulnerabilities and recent attacks are the main reasons behind our decision to focus on Elasticsearch honeypots.

In order to respond to these threats, security community has designed various open-source honeypot implementations. Most of these are low-interaction, and their source code is available on GitHub. These honeypots usually provide an easy to set up and easy to use approach with ready-made default configurations. In addition to this, since low-interaction honeypots are less resource hungry, simply cloning a low-interaction Elasticsearch honeypot from GitHub and following the provided setup instructions is a cheap and effortless way of setting up honeypots. However, these low-interaction honeypots have several limitations, most notably the lack of useful information about the attackers' behaviour that can be collected.

2.1 Designing High-Interaction Elasticsearch Honeypot

In order to avoid the shortcomings of low-interaction honeypot implementations (and the lack of readily available high-interaction honeypots), we decided to construct our own high-interaction Elasticsearch honeypot, based on the latest version of the Elasticsearch services which we deployed into a Docker image. This allows our honeypot to return exactly the same responses that any compromised Elasticsearch servers would do. Moreover, in order to increase the decoy capabilities of our honeypot, we created random (yet believable) business and organisation, and related datasets to fill the indices of the Elasticsearch servers with realistic data. Moreover, index names were randomly selected from a list of 16 business-related database tables. Random indices data creation and name selection were performed every time a honeypot was deployed, to reduce the risk of potential attackers recognising "already seen" data.

Lastly, a `tcpdump` script was placed outside of the Docker image to record Internet traffic coming from all the visiting and attacking IP addresses. These traffic data were stored in packet capture (or `pcap`) files. These files were regularly parsed by automated scripts using the rule set mentioned in Section 3.2.

2.2 Low-Interaction Elasticsearch Honeypots

In comparison to high-interaction honeypots, low-interaction honeypots only allow attackers a limited access to the system [9, 27, 30].

This lack of freedom means that low-interaction honeypots use fewer resources than high-interaction honeypots. However, whilst easy to deploy and maintain, low-interaction honeypots do not behave like a real production system, and hence typically do not collect so much valuable data [27].

There are many open-source low-interaction Elasticsearch honeypots readily available on the Internet. For the purpose of our research, we selected widely-used low-interaction honeypots in the wild, taking into account their ease of deployment, as well as the clarity and availability of their documentation. These selection criteria would allow for the deployment of the chosen honeypots without demanding high technical knowledge or specific configurations.

For these reasons, we selected all three Elasticsearch honeypots (only the ones that are listed as database honeypots) that are being listed in popular honeypot lists [1, 16, 32]. These three honeypots are:

- *Elastichoney* is an open-source low-interaction Elasticsearch honeypot. It was "designed to catch attackers exploiting RCE vulnerabilities in Elasticsearch" [13]. It takes requests in the `/`, `/_search` and `/_nodes` endpoints and mimics an Elasticsearch database by replying with a JSON response that emulates what an actual vulnerable Elasticsearch instance would send [24]. Elastichoney keeps track of the attacks it receives and writes them into a log file. We deployed it with the default configuration in order to standardize our deployment process and capture the real world scenarios as more systems being deployed with default configurations rather than specific configurations.
- *Delilah* is another open-source low-interaction Elasticsearch honeypot that aims to catch attackers who are using the Elasticsearch Groovy vulnerability (CVE-2015-1427) [15, 26]. Delilah mimics a vulnerable Elasticsearch instance and "detects and identifies attack commands, reconnaissance attempts, and download commands (specifically `wget` and `curl`)" [26]. Delilah is capable of downloading files an attacker is trying to insert into the system and it can also send email notifications to the managers of the system for real-time analysis.
- *Elasticpot* is an open-source low-interaction Elasticsearch honeypot that mimics a vulnerable Elasticsearch instance and stores the logs of incoming attacks [8]. It does not use Docker by default; however, deploying the honeypot into a Docker container is an option. This honeypot responds to an attacker by mimicking an old version of a vulnerable Elasticsearch server. However, it is possible to change the responses it gives to attackers if the honeypot owner would like the system to behave in a certain way (for instance, if they would like to focus on certain types of interaction), by editing the configuration file.

3 Methodology

We carried out a study design process to prepare the set up to be used for running our honeypots, as well as the strategy for collecting the data, and the rule set for examining the traffic behaviour.

3.1 Honeypot Deployment

Before starting the data collection, it is necessary to make critical decisions regarding the deployment process, which includes parameters such as deployment

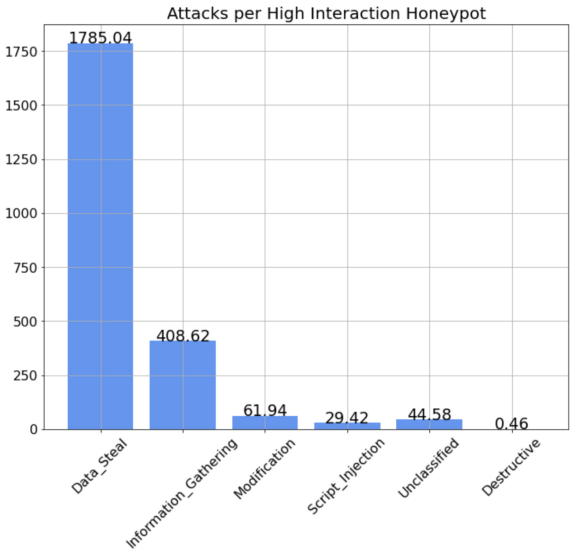

Fig. 1: The number of attacks for each of the six types of attack behaviour (high-interaction honeypot)

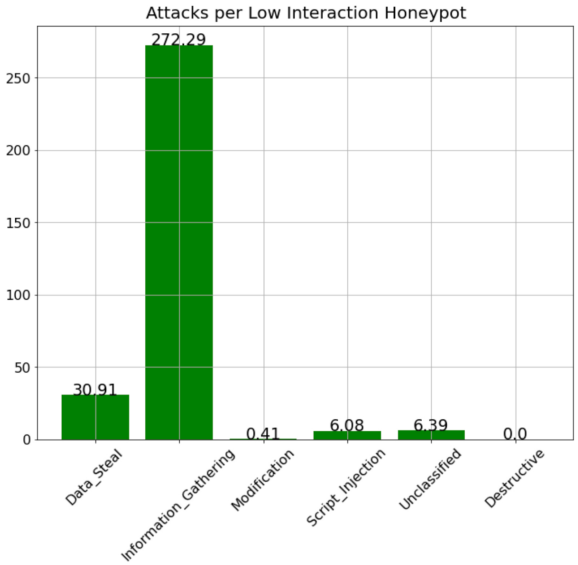

Fig. 2: The number of attacks for each of the six types of attack behaviour (low-interaction honeypot)

locations, cloud providers and the deployment period. In order to minimise the chance of attackers identifying our honeypots collectively, the deployment process was randomised.

The deployment of our 48 high-interaction and 111 low-interaction honeypots was deployed consecutively, over a period of 14 days in order to understand initial and short term effects of fidelity and degree of honeypot interaction. Using this deployment strategy – instead of simultaneously deploying all our honeypots at once – ensured that the likelihood of attackers finding any link between our honeypots would be low.

We randomised the location of both high- and low-interaction honeypots to avoid any region-specific trends. The randomly selected regions for deployment were Europe, US, as well as some parts of Asia, Australia and Africa.

Furthermore, to avoid any cloud-provider-specific issue, we selected them randomly from a list of companies such as Alibaba, Amazon, Azure, Digital Ocean, Google Cloud, Hetzner, Linode and OVH. We selected well-known and widely-used cloud providers to include in this set in order to have our honeypots hosted at IP addresses that are commonly scanned and attacked by potential intruders. Additionally, the number of honeypots that we deployed was also randomised for the same reason (to avoid identification of all our honeypots as a collective). We restricted the randomisation of the number of instances in order to avoid widely different numbers of instances for each honeypot type. This process resulted in 48 high-interaction honeypots, 35 Elastichoney honeypots, 38 Delilah honeypots and 38 Elasticpot honeypots. We also conducted a "per honeypot type" analysis in Figure 1 and Figure 2 to mitigate the risk of having a different number of instances affecting the results.

3.2 Evaluation and Rule Set

We manually classified six different types of traffic behaviour towards our honeypots. We explain these below.

- *Data Steal.* We have classified data steal attacks as attempts to send a request to endpoints where reading critical information is possible. For example, a request sent to the `/_search` endpoint may access critical information inside the honeypot; thus, we classified such a request as a "data steal" attempt.
- *Information Gathering.* Information gathering consists of reconnaissance attempts, i.e. activities that are attempting to acquire information about the clusters and nodes of the system, rather than attempting to access the content inside of our honeypot. They normally precede an attack. So, requests that we classify as "information gathering" try to gain details about the system, rather than the information inside.
- *Data Modification.* This type of traffic behaviour includes `POST` and `PUT` requests that try to write data inside the honeypot, or attempt to alter existing data.
- *Script Injection.* Script injection attacks are associated with requests that abuse the known vulnerabilities of Elasticsearch to run malicious code or conduct a code injection into the server which allow attackers to gain full control of the the compromised system.

- *Destructive.* We classified as "destructive" traffic behaviour requests that attempt to remove existing data in the server. These requests usually include the keyword `DELETE`.
- *Unclassified.* This traffic category is a catch-all type group, and consists of requests that do not fall easily into any of the other five categories above. For example, a request meant to target another service can fall into this category. Only 0.5% of the total packets we collected belong to this category.

4 Results

In this section, we discuss the efficacy of high- and low-interaction honeypots. First, we evaluated how the honeypots performed, in terms of their handling upon receiving attacks. After that, we analysed and evaluated the impact of honeypot locations. Lastly, we reported the possible origins of these attacks.

4.1 Efficacy of High-Interaction Honeypots

Honeypots are valuable tools in network security as long as they capture attacks. In this section of our paper, we analyse how good high-interaction honeypots are at capturing attacks compared to low-interaction honeypots.

First, we would like to determine whether using high-interaction honeypots would attract more attacks than low-interaction honeypots. Table 1 provides a statistical summary of the packets received for each type of attack, in relation to each type of honeypot. This table shows that the high-interaction honeypot received more packets than any of the low-interaction honeypots. For instance, high-interaction honeypots captured 1412 (67.66%) script injection attempts, compared to 253 (12.12%), 207 (9.92%) and 215 (10.30%) script injection attempts for those that belonged to Elastichoney, Delilah and Elasticpot, respectively. Similarly, high-interaction honeypots received 96% of the data steal attempts and 98% of data modification attempts. Moreover, only high-interaction honeypots received attacks designed to remove data from the database (this is typically associated with an effort to demand a ransom from the database owners).

We also investigated the number of unique IP addresses captured by the honeypots. Table 2 displays the number of unique IP addresses captured by each honeypot group, per attack type. Some IP addresses are involved in multiple attack types. For example, attackers often gather information about the database before performing data steal or modification attempts from the same IP. As shown in Table 2, high-interaction honeypots captured fewer unique IP addresses for data steal and information gathering attacks in comparison to two of the low-interaction honeypots. However, high-interaction honeypots captured significantly higher numbers of unique IP addresses for more involved attacks such as data modification, script injection and data removal. One way to interpret these results is that high-interaction honeypots received more attacks from more sophisticated attackers.

Table 1: A statistical summary of the types of attack behaviour, according to each honeypot group

Honeypots	#	# Packet captures	#Unique IP addresses	Data steal	Information Gathering	Modification	Script injection	Destructive
High-interaction	48	111843 (76.12%)	1578	85682 (96.15%)	19614 (39.36%)	2973 (98.51%)	1412 (67.66%)	22 (100%)
Elastichoney	35	10300 (7.01%)	1571	1109 (1.24%)	8711 (17.48%)	28 (0.93%)	253 (12.12%)	0
Delilah	38	12841 (8.74%)	1649	1486 (1.67%)	10911 (21.89%)	17 (0.56%)	207 (9.92%)	0
Elasticpot	38	11943 (8.13%)	1428	836 (0.94%)	10602 (21.27%)	0	215 (10.30%)	0
Total	159	146927 (100%)	3276 (100%)	89113 (100%)	49838 (100%)	3018 (100%)	2087 (100%)	22 (100%)

Table 2: The number of unique IP addresses based on the types of attack behaviour, according to each honeypot group

Honeypots	#	# Packet captures	# Unique IP addresses	Data steal	Information Gathering	Modification	Script injection	Destructive
High-interaction	48	111843 (70.61%)	1578	338	1353	31	102	4
Elastichoney	35	10300 (6.50%)	1571	379	1375	11	84	0
Delilah	38	12841 (8.11%)	1649	411	1430	5	73	0
Elasticpot	38	23419 (14.78%)	1428	303	1224	0	67	0

Table 3: A statistical summary of some attack behaviour of IP addresses captured by the high-interaction honeypot, when they also hit low-interaction honeypot(s)

Categories	# of IPs captured by HIH	Elastichoney			Delilah			Elasticpot		
		Seen IPs	Used for the same purpose	Used for other purpose	Seen IPs	Used for the same purpose	Used for other purpose	Seen IPs	Used for the same purpose	Used for other purpose
Data steal	338	70	34	36	87	50	37	68	35	33
Modification	31	3	0	3	4	0	4	1	0	1
Script injection	102	21	21	0	22	22	0	24	24	0
Destructive	4	3	0	3	2	0	2	3	0	3

Working under the assumption that high-interaction honeypots are much harder to spot by attackers, we focused on studying if there was any evidence pointing towards attackers being able to detect and avoid low-interaction ones.

To get a better sense of whether attacker have shown any avoidance behaviour towards low-interaction honeypots, we investigated if IP addresses captured by high-interaction honeypots were also observed in low-interaction honeypots.

Table 3 shows how attackers that were captured by high-interaction honeypots behaved when they encountered low-interaction honeypots. Many attacking IP addresses captured by high-interaction honeypots were not seen at all by low-interaction honeypots. For example, only a handful of IP addresses engaging in modification attacks on high-interaction honeypots were also seen by low-interaction honeypots. This might be because these attackers use up-to-date blacklists that can be derived from using self-written scripts or publicly available honeypot detection tools to ignore low-interaction honeypots [25, 4, 3, 28]. They only scan the homepage and two other pages (namely `/_search` and `/_all/_mapping`), which

provide information about the database. By using the returned answers, these attackers might have detected and ignored the low-teraction honeypots.

The destructive attack category shows a similar picture, whereby known IP addresses are not used for gathering information about the database rather than deleting instances. Additionally, at most 24% of attacks in the script injection category was captured by single low-interaction honeypots. The rest of the attackers mıght have used a blacklist or some other method to avoid low-interaction honeypots.

Interestingly, all the IP addresses captured in low-interaction honeypots were involved in script injection. This demonstrates that one quarter of attackers

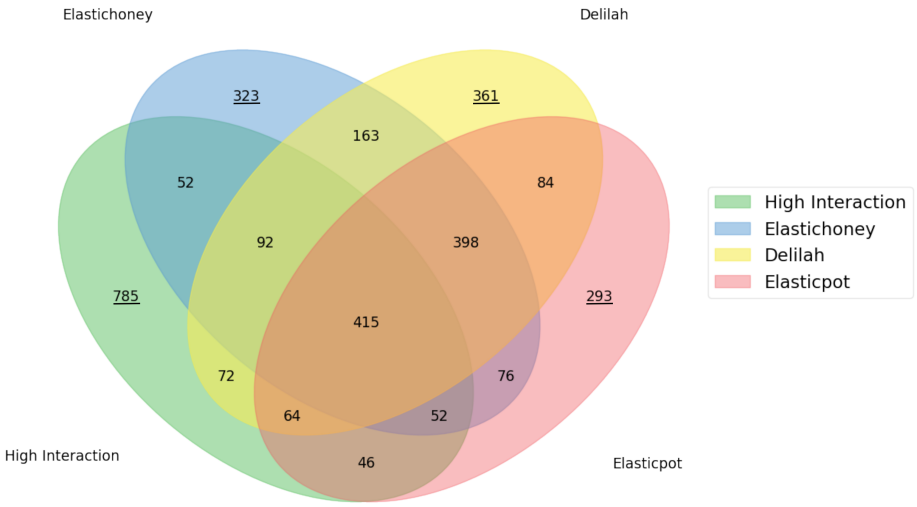

Fig. 3: A Venn diagram showing the distribution of the unique IP addresses captured for each honeypot group

involved in script injection were not involved in any kind of low-interaction honeypot avoidance. Similarly, in data steal category, at most 25% of the attacks reported in high-interaction honeypots were captured by single low-interaction honeypots. More than half of these attackers were observed using queries for stealing data on low-interaction honeypots, as they did on high-interaction ones.

Our results show that attackers that conducted destructive and modification attacks clearly engaged in avoidance of low-interaction honeypots. Data stealing attackers display a somewhat similar behaviour. On the other hand, attackers involved in script injection did not exhibit this honeypot-evading behaviour.

Still, high-interaction honeypots captured five times more script injection attacks. This might be because script injection attacks only use blacklists to ignore low-interaction honeypots but no response-based avoidance is used. On the other hand, other attacker types seem to be using both response-based

Furthermore, Figure 3 shows the overlap of captured IP addresses for each honeypot group. We can see that a total of 415 IP addresses were captured by *all* honeypot groups. On the other hand, 785 IP addresses were only captured by high-interaction honeypots, while Elastichoney only captured 323 IP addresses, and Delilah only captured 361 IP addresses. Lastly, 293 IP addresses were only captured by Elasticpot which was less than all other honeypot groups. The total number of IP addresses exclusively attacking each honeypot group is shown as underlined in Figure 3.

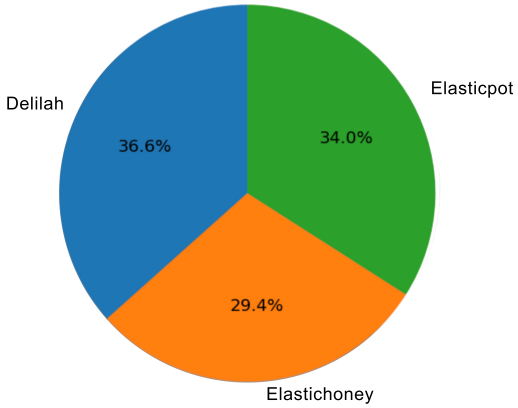

Fig. 4: The percentages of attacks captured by low-interaction honeypots for each of the three implementations used

4.2 Efficacy of Different Low-Interaction Honeypots

In our study, the low-interaction honeypots (Elastichoney, Delilah, and Elasticpot) all performed badly in comparison to our high-interaction honeypot, both collectively and individually. Furthermore, the performance of these low-interaction honeypots did not differ notably, which suggests that the characteristics of any low-interaction honeypot would not lead to a meaningful difference in a real-world scenario.

Figure 4 presents the breakdown of the percentages of network traffic received by low-interaction honeypots based on their implementation. This figure shows that 36.6% of the total malicious network traffic were received by Delilah, compared to 34% and 29.4% for those received by Elasticpot and Elastichoney implementations respectively. These discrepancies in performances are most likely due to attackers' ability to identify low-interaction honeypots.

To understand whether our honeypots can be identified by external resources, we queried our IP addresses in the Shodan search engine [29]. Typically, Shodan can identify low-interaction honeypots by just scanning their metadata. Our search results found that Shodan identified none of the Delilah honeypots. However, all the Elastichoney and Elasticpot honeypots were correctly identified by Shodan. This success of detecting honeypots by Shodan might be the reason behind the slightly better overall performance of Delilah honeypots compared to the other low-interaction honeypots.

4.3 Analysing the Attackers

This subsection focuses on the source locations of the attacks that our honeypots had attracted. We used Maxmind GeoIP services to identify the origins of the attacks [2].

Table 4: The top-10 offending countries from which the IP addresses of the traffic to our honeypots came from

Total		High-Interaction		Low-Interaction	
Rank	Country	#	Country	#	Country	#
1	United States	2054	United States	978	United States	1569
2	China	316	China	152	China	240
3	United Kingdom	112	United Kingdom	53	United Kingdom	79
4	Germany	84	Netherlands	51	Germany	59
5	Netherlands	76	Germany	35	Netherlands	57
6	Canada	57	Canada	25	Canada	36
7	Singapore	51	Singapore	25	France	34
8	India	44	Lithuania	23	Singapore	31
9	Russian Federation	39	Russian Federation	20	Russian Federation	31
10	France	35	India	18	India	30

Table 4 shows the top-10 countries in terms of the number of attacking IP addresses per honeypot type. Our results cover IP addresses captured by both high-interaction and low-interaction honeypots. IP addresses were located in 60 different countries. The top-10 countries were United States, China, United Kingdom, Germany, Netherlands, Canada, Singapore, India, Russia and France. These 10 countries account for more than 91,4% of the total number of unique IP addresses. Moreover, United States alone accounts for more than 66.4% of the total number of attacking IP addresses. Typically, countries with larger hosting industries are present in this top-10 list.

The results from the Maxmind data do not show a clear difference in the top-10 countries in terms of the number of attacking IP addresses for different honeypot groups. These top-10 countries are very similar for both high-interaction and low-interaction honeypot groups. In fact, the top-3 countries were the same for both high- and low-interaction honeypots. Nine countries appeared in the top-10 of both groups, albeit at slightly different positions. The only major difference is that Lithuania appeared in the high-interaction list (at number 8) but not in the low-interaction list, while France appeared in the low-interaction honeypot (at number 7), but not in the high-interaction list.

5 Related Work

As far as we know, there is no prior research comparing the performances of high- and low-interaction honeypots through real-world data. This section will look at prior research under two subsections: Honeypot Detection and Comparison of Honeypots.

The studies that focus on detecting low-interaction honeypots are related to this research as the performance differences between high- and low-interaction honeypots are likely due to – at least in part – low-interaction honeypots getting detected more easily. However, these studies do not typically include high-interaction honeypots into their analysis. The scarce prior research in comparing

honeypots is also related to our present research. However, previous work tends to compare low-interaction honeypots with each other, rather than drawing a comparison between high- and low-interaction honeypots [12, 18, 21].

5.1 Honeypot Detection

The ability to avoid detection is crucial for the performance of a honeypot. There are many attempted methods for detecting low-interaction honeypots. One such approach proposed by Aguirre-Anaya et al. is done by fingerprinting the low-interaction honeypots' static features [6]. In this method, using some features of a honeypot, such as "communication protocols, network services or specific environments", a fingerprint is obtained. Then, using this fingerprint, it is possible to distinguish a honeypot from a real system.

In addition to this, Morishita et al. discussed the detection of 14 open-source low-interaction honeypots [18]. This research is significant for our study, because we have used open-source low-interaction honeypots as well. In particular, Morishita et al. created 20 simple signatures to detect 19,208 honeypots across 637 autonomous systems. Furthermore, they found that low-interaction honeypots that use default configurations are more susceptible to getting identified.

Mukkamala et al. used some additional methods for honeypot identification [19]. In their research, detecting honeypots at the network level is explored, and they argue that, by looking at the network features of a system, low-interaction honeypots may be identified. Among other methods, fingerprinting is also discussed as a viable tool for identification in this paper.

In another paper that focuses on network-level detection, Defibaugh-Chavez et al. also argued that it is possible to identify low-interaction honeypots just by looking at their network features [11].

5.2 Comparison of Honeypots

Several studies have looked into comparing low-interaction honeypots. A paper by Abubakar Zakari et al. presents a comparative analysis among five widely used low-interaction honeypots, namely Honeyahole, Honeywall, Honeyd, Honeytrap, and Nepenthes [31]. Their study focused on literature analysis rather than real-world data. Through their work, they showed that almost all honeypots in question lack robustness or intelligence, and these limitations play negatively into their effectiveness.

In another work that evaluated honeypot technologies, we can see a comparison between open-source honeypots and commercial honeypot tools [21]. However, this study did not focus on real-world data based on network traffic, but instead compared their various features, such as services offered and platform support.

Finally, Alata et al. focused on the behaviour of attackers who have succeeded in entering the system [7]. Since this study used both high- and low-interaction honeypots, it is the one most closely related to our paper here. Alata et al.

made an observation that their low-interaction honeypots have performed worse compared to high-interaction honeypots in terms of capturing attacks. However, since their experiment focused more on the attack behaviour and weak password attacks, their study did not offer anything more than an observation regarding the comparison of high- and low-interaction honeypots. Furthermore, the authors noted that this discrepancy was due to low-interaction honeypots not having the `ssh` service open, which means a lower number of attacks were being received.

6 Conclusion

Honeypots are heavily used by cybersecurity teams to collect indicators of compromise and other intelligence regarding cybercriminals. Some teams prefer to use high-interaction honeypots, which are designed so that they fully emulate a vulnerable system. Others use low-interaction honeypots that emulate only a few basic elements of a vulnerable system. There are good reasons to chose the latter, particularly because they consume less resources, are more readily available, and easier to deploy. Moreover, there are many open-source low-interaction honeypot projects to choose from, while there are very few high-interaction honeypots available freely.

In this paper, we present our study regarding the impact of the degree of interaction and fidelity in terms of capturing intelligence from attackers. For that purpose, we compared attack data collected by a high-interaction Elasticsearch honeypot with those collected by a group of three different low-interaction open-source honeypot projects.

We found a clear evidence that the high-interaction approach leads to gaining more volume and more pertinent intelligence from the attackers. Moreover, the difference is so significant that no combination of any of the three low-interaction honeypots can match the quantity or quality of the evidence gathered by the high-interaction one.

We recommend practitioners to use high-interaction honeypots where possible and limit the use of low-interaction ones to systems that, due to a chronic lack of resources, cannot afford to run the costlier high-interaction type.

There are several areas that merit further investigation. First, it would be interesting to explore if there is a link between the initial feature being hit by an attacker, and how long the attacker is likely to spend interacting with that honeypot. Second, investigating the possible use of blacklists (of already identified high-interaction and low-interaction honeypots) among attackers might reveal other strategies for improving the effectiveness of these honeypots. Finally, it would be valuable to explore ways to reduce the cost and the system requirements of high-interaction honeypots, while improving their usability and detection-avoidance rates.

References

1. Curated list of awesome lists. https://project-awesome.org/paralax/awesome-honeypots.

2. GeoIP and GeoLite. https://dev.maxmind.com/geoip?lang=en.
3. Honeypot or not? https://honeyscore.shodan.io/.
4. nessus. https://www.tenable.com/products/nessus.
5. _. Offensive security's Exploit Database Archive. https://www.exploit-db.com/.
6. Eleazar Aguirre-Anaya, Gina Gallegos-Garcia, Nicolás Solano Luna, and Luis Alfonso Villa Vargas. A new procedure to detect low interaction honeypots. *International Journal of Electrical and Computer Engineering (IJECE)*, 4(6), 2014.
7. Eric Alata, Vincent Nicomette, Mohamed Kaâniche, Marc Dacier, and Matthieu Herrb. Lessons learned from the deployment of a high-interaction honeypot. In *2006 Sixth European Dependable Computing Conference*, pages 39–46. IEEE, 2006.
8. Bontchev. Bontchev/elasticpot: An elasticsearch honeypot. https://github.com/bontchev/elasticpot.
9. Ronald M Campbell, Keshnee Padayachee, and Themba Masombuka. A survey of honeypot research: Trends and opportunities. In *2015 10th international conference for internet technology and secured transactions (ICITST)*, pages 208–212. IEEE, 2015.
10. Catalin Cimpanu. A hacker has wiped, defaced more than 15,000 elasticsearch servers, April 2020.
11. P Defibaugh-Chavez, R Veeraghattam, M Kannappa, S Mukkamala, and AH Sung. Network based detection of virtual environments and low interaction honeypots. In *Proceedings of the 2006 IEEE SMC, workshop on information assurance*, pages 283–289, 2006.
12. Wenjun Fan, Zhihui Du, David Fernández, and Victor A Villagra. Enabling an anatomic view to investigate honeypot systems: A survey. *IEEE Systems Journal*, 12(4):3906–3919, 2017.
13. Jordan-Wright. Jordan-Wright/elastichoney: A simple elasticsearch honeypot. https://github.com/jordan-wright/elastichoney.
14. Navneet Kambow and Lavleen Kaur Passi. Honeypots: The need of network security. *International Journal of Computer Science and Information Technologies*, 5(5):6098–6101, 2014.
15. Xiphos Research Ltd. Elasticsearch - remote code execution. https://www.exploit-db.com/exploits/36337, Mar 2015.
16. "Maciej". Collection of honeypots. https://iceburn.medium.com/collection-of-honeypots-a7ec6e446163, Sep 2019.
17. Iyatiti Mokube and Michele Adams. Honeypots: concepts, approaches, and challenges. In *Proceedings of the 45th annual southeast regional conference*, pages 321–326, 2007.
18. Shun Morishita, Takuya Hoizumi, Wataru Ueno, Rui Tanabe, Carlos Gañán, Michel JG van Eeten, Katsunari Yoshioka, and Tsutomu Matsumoto. Detect me if you... oh wait. an internet-wide view of self-revealing honeypots. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 134–143. IEEE, 2019.
19. S Mukkamala, K Yendrapalli, R Basnet, MK Shankarapani, and AH Sung. Detection of virtual environments and low interaction honeypots. In *2007 IEEE SMC Information Assurance and Security Workshop*, pages 92–98. IEEE, 2007.
20. Ruslan E Mushtakov and Dmitry S Silnov. New approach to detect suspicious activity using http-proxy honeypots. In *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 600–605. IEEE, 2017.

21. Bharti Nagpal, Nanhay Singh, Naresh Chauhan, and Pratima Sharma. Catch: Comparison and analysis of tools covering honeypots. In *2015 International Conference on Advances in Computer Engineering and Applications*, pages 783–786. IEEE, 2015.
22. Elastic NV. What is elasticsearch? https://www.elastic.co/what-is/elasticsearch.
23. Pierluigi Paganini. Data breaches tracker monitor unsecured Elasticsearch Servers Online. https://securityaffairs.co/wordpress/115698/security/data-breaches-tracker-unsecured-elasticsearch.html, Mar 2021.
24. Jordan Wright's Picture, Jordan WrightSecurity Researcher, and San Antonio. Introducing Elastichoney - an Elasticsearch Honeypot. https://jordan-wright.com/blog/2015/03/23/introducing-elastichoney-an-elasticsearch-honeypot/, Mar 2015.
25. Salvatore Sanfilippo et al. Hping-active network security tool. http://www.hping.org/, 2008.
26. SecurityTW. SecurityTW/Delilah. https://github.com/SecurityTW/delilah.
27. Christian Seifert, Ian Welch, and Peter Komisarczuk. Taxonomy of honeypots. 2006.
28. Send-Safe. Send-safe honeypot hunter download. https://send-safe-honeypot-hunter.apponic.com/, Jul 2022.
29. "Shodan". Shodan Search Engine. https://www.shodan.io.
30. Michail Tsikerdekis, Sherali Zeadally, Amy Schlesener, and Nicolas Sklavos. Approaches for preventing honeypot detection and compromise. In *2018 Global Information Infrastructure and Networking Symposium (GIIS)*, pages 1–6. IEEE, 2018.
31. Abubakar Zakari, Abdulmalik Ahmad Lawan, and Girish Bekaroo. Towards improving the security of low-interaction honeypots: Insights from a comparative analysis. In *Int'l Conf. on Emerging Trends in Electrical, Electronic and Communications Engineering*, pages 314–321. Springer, 2016.
32. Zion3R. Collection of awesome honeypots. https://www.kitploit.com/2015/12/collection-of-awesome-honeypots.html, Dec 2015.