Cyber Deception in the Cloud: Automating Database Canarytoken Implants - ”Eleplanter”

Benjamin Rader
IUPUI
Purdue School of Engineering & Technology

1 Introduction & Literature Overview

Cloud security is, by far, one of the most consequential paradigm shifts in how we solve problems with information technology. In short, it allows for outsourcing layers of control or management of IT infrastructure or systems to third parties. The cloud service provider handles all of the problems with managing the systems and machines, while users utilize simplified systems to solve their problems. However, how the control and more often the visibility between the cloud service provider (CSP) and the user manifests will differ from implementation to implementation or CSP-to-CSP. In a literal sense, this interface of control and visibility depends on the CSP's implementation and the layer of the cloud service. There is not a concrete consensus on what these layers are but they generally go (from least control to most): applications (SaaS), data (PaaS), runtime (PaaS, FaaS, CaaS), middleware (PaaS, FaaS), virtualization (IaaS), OS (IaaS), server, storage, and networking. Explaining each of these layers is outside of the scope of this paper, but it is important to understand how ”cloud” affects the goals of cybersecurity. Each instance of a type of cloud service (SaaS, PaaS, IaaS) has a sort of ”border” of control or visibility between the user and the cloud service provider. For IaaS, that border exists at the level of virtualization or machines. The defender's dilemma sums up the risk surfaces that arise at these borders. Being a defender means covering all entry points or prioritizing certain ones. The attackers only need one way in, but the defender must cover the surfaces which the attacker will inevitably try to exploit [13].

1.1 Stats - Integration with Cloud is Hard

As stated previously, the user of cloud services will have a different amount of control based on what they are managing. This becomes extremely cost-effective for engineers, system architects, product makers, and developers because they don't have to worry about switching and routing to set up a quick function-as-a-service. They don't need to think even for a second about port forwarding. For example, can simplify their architecture when they go to cloud and leave behind there reverse-proxied machines on-prem. However, this merely shifts the issue from a wide surface of visibility and forces it to be obtained through more abstract means. In other words, abstraction can be an enemy to innovation, progress, and solving IT problems. In the case of most cloud service providers, the only answer is to create an abstract interface for visibility and control between the user and the provider. In the most literal sense this is done with APIs (application programming interfaces) and integrations. These abstractions must exist though because the cloud service provider must be able to automate and optimize the monumental task of managing huge amounts of infrastructure, being flexible to customer needs, and all while being cost effective. However, this abstraction plays out differently from cloud vendor to cloud vendor. What this means for security is that there is a need for skills and problem-solving abilities around integrations and APIs with cloud service providers. This is why we see practitioners getting whole certificates for single cloud service providers, because the CSP has optimized and abstracted their systems so much that you must spend weeks, if not months, to learn how to utilize their interface between those layers of control with cloud services.

The question becomes whether I am telling a sensible truth or whether I am speculating from my background in security which is quite limited. My experiences and conversations with experts (by consensus) have led me to conclude that cybersecurity practitioners do not spend enough time contemplating these relationships in the cloud and how to handle them. In terms of technical skills, it seems apparent that ”data engineering” which helps with integration tasks is considerably more important than even networking fundamentals when solving cybersecurity challenges in the cloud. Many practitioners hold that visibility is more significant than setting up security controls early on for prioritization benefits [6].

A large-scale survey from the renowned SANS Institute in 2022 on Cloud Security was conducted over several hundred high-profile engineers, analysts, and CISOs in cybersecurity. The survey deduced that the two largest challenges for incident response in the cloud revolve around visibility and correlating data: 1) Lack of real-time visibility into events and communications involved in an incident (48.5%), and 2) Difficulty correlating cloud and on-prem data (40.7%). Additionally, the majority of these businesses utilized CSP APIs for important security controls: IAM (58%), Config MGMT (53.6%), Logging (51%), Encryption and Data Protection (45.9%) [24]. According to the survey,

most orgs are trying to connect SIEMs (security information and event management systems) to cloud APIs, but it is difficult, costly, or requires rare minds or lots of certifications and classes. This creates slowdowns in increasing organizations' security maturity because most are stuck trying to switch to the cloud and utilize the information from CSPs. The skillsets needed to solve these problems are, at best, hit-and-miss. From the looks of it, the problems are only becoming more difficult.

1.2 Welcome to Cyber Deception

1.2.1 How To Avoid Integration and Abstraction in the Cloud

Cloud security is hard. Without bright minds at the forefront of innovation in organizations, most businesses will have a hard time evolving from on-prem to hybrid environments. So, where can an organization turn next in the context of the cloud environment and the various layers and control and visibility based on service? The answer is simple...avoid integration and manual intervention in operating these interfaces. For example, instead of utilizing the API for a CSP when working with IaaS (infrastructure as a service), opt-in for an agent-based solution or use an ”image” or machine that has security and logging implemented already. Usually, systems like this can be found on cloud marketplaces or the like. Use a data engineering solution that specializes in aggregation or integration before immediately turning to the CSP's specific implementation. Most of the time, the problem will not require building custom integration servers to get around the API. Security professionals merely need to widen their net and terminology when exploring solutions.

Having said all this, cyber deception is the epitome of flexible detection and alerting. The case with cloud security is abstraction. To get around abstraction, we need a process that never changes: 1) plant this agent or entity that detects, 2) have it detect something, and 3) send the alert to a SIEM. To do this in a way that extends to all layers of services in the cloud (e.g. SaaS, IaaS). In other words, we need solutions that are separate from the CSP APIs or integrations in terms of the entity itself (agent), ways to detect and prioritize risk (run code to send alert) in a way that is not CSP-specific, and in a way that is universal in terms of format so that any SIEM can use it.

1.2.2 Cyber Deception Explained with Knowledge Asymmetry

The defender's dilemma is a constant issue in cybersecurity. It is assumed that attackers always have an asymmetric advantage against defenders because of the simple fact of risk coverage. However, many, including myself, would argue that there is not an asymettric advantage for attackers, but rather for defenders. The issue is that the defender's true advantage is not often utilized because it requires visibility, communication, and thinking outside of the ”box.”This advantage can be called ”knowledge asymmetry.”

One could explain this by alluding to American movies. Often in action or adventure movies where there are external protagonists, the protagonist and the antagonist will have a final conflict. Often, these conflicts are centered around some sort of monument, landmark, a ”final stand”, or ”Alamo” as it were. The good guys (the protagonists) may have been defeated many times by the bad guys (the antagonists) in the past because they had been outgunned or outwitted. Often, the reasoning for their past losses is that the good guys were not thinking inventively or were not using their unique knowledge of what they are defending to their advantage. The ”final stand” of action and adventures movies such as these usually culminate with imaginative traps, deception, and using knowledge of the environment to the advantage of the defenders. This strategy is the essence of cyber deception.

From a philosophical or logical perspective cyber deception is a fusion of strategic knowledge asymmetry and the art of manipulating the perceived reality within a system, ultimately creating a formidable defense against maliciously-motivated threat actors. Defenders are exploiting their hopefully comprehensive understanding of a system's normal behavior and purpose to craft, maintain, and capitalize on false realities. False realities are simply perceptions of something that are false. This dynamic approach, which also lends itself to a game theory mindset, not only confounds attackers but also presents opportunities for detecting anomalies and unmasking entities which are acting upon these intentionally crafted false realities. In this intricate game of perception, the defender's proficiency in leveraging their system knowledge becomes the linchpin of their success, as they artfully blur the lines between reality and fabrication [1].

As I mentioned briefly, this sort of method lends itself to a game theoretical approach. In summary, a game involves two entities with conflicting interests - ”I want to get a point and you want to get a point, but only one can win.” The defenders in this case operate with incomplete information about the motives and vantage point of the attackers. However, it can be assumed they do not know about the functions of certain systems. The fact that visibility and collaboration is a difficult tasks in cloud security becomes a strength when deceiving attackers, because these weaknesses or challenges in visibility inform the defender of the sorts of knowledge that they can leverage over the attacker.

1.3 Cyber Deception Solution Landscape and Taxonomies

1.3.1 Game Theoretic Taxonomy for Cyber Deception

Most papers mentioning cyber deception are relatively new. although some of the technical solutions themselves have been around for a much longer time. Talk in this area of cybersecurity is usually niche, but it is also a debated topic which will be looked at in the next section of this paper. However, there only seem to be a few instances of a taxonomy being discussed. For instance, Pawlick discusses a game theoretic taxonomy for the groups by dividing them into perturbation, moving target defense, obfuscation, mixing, honey-x, and attacker engagement [20]. These categories describe cyber deception methods from the perspective of how they fit into a defender-attacker environment, but I think it is productive to also give a taxonomy for their implementation and how that fits into the cloud paradigm.

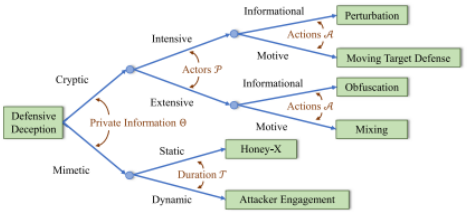

Fig. 1. Tree diagram of deception into "species." "The specific differences correspond to the game-theoretic notions of private information, actors, actions, and duration." Canary tokens fit into the Mimetic ¿ Static classification.

1.3.2 My Categories for Types of Cloud Cyber Deception Implementation

Cloud Cyber Deception Implementation Categories:

1) **Infra/CSP Specific**
   - Likely Examples: Honeypot Management, Deceptive Network Traffic
   - Usually requires a custom API
   - Could include CSP-specific integrations
2) **Integrated with CSP**
   - Likely Examples: Deceptive WAF, Honeypots
   - Can be agent-based or include hosting something in the cloud like with a honeypot
   - Can include premade integrations
3) **Portable / External**
   - Likely Examples: Deception Tokens (canary tokens), Decoy Data, Attribution or Beaconing Code
   - Data that is portable and separate from the systems it abides in
   - Can use external systems to activate alerts or portable code to do so
   - Can be thought of as tripwires

1.3.3 The Categories Explained

These categories synergize the core ideas of the cloud stack and implement them based on the varying layers of services in the cloud. Cyber deception technologies are not categorized well because of the novelty of the practice. The market will need to mature a bit more before we can get a consistent way to talk about the different solutions [6]. I am attempting to do so with these categories. Starting with Infra/CSP Specific, each one of theses three categories focus on the work that needs to be done in each system across that "boundary" of control which depends on the type of service (SaaS, IaaS, e.g.). For instance, with a CSP Specific Cyber Deception Method, you may be at the level of 'infrastructure as a service' where you manage everything above the virtualization of the machines. This virtualization aspect does not explicitly handle the networking aspects of say a VPC (virtual private cloud), so you may have to use the CSP-provided system for managing honeypots (fake vulnerable computers) and their networking. This differs from deploying a singular honeypot where you only have to manage that one box/machine. In that case, you can because you are using IaaS and can manage the machines. The less control that one has (at least with SaaS - software as a service) in the cloud, the more likely it is that one will have to rely on a custom cloud service provider (CSP) API. This goes to show how the simplicity of solving one problem with easy-to-deploy cloud technology will create new ones of abstraction and loss of straightforward customization. We have to rely more and more on custom interfaces to interact with the systems that we do not manage. It is a dilemma to say the least.

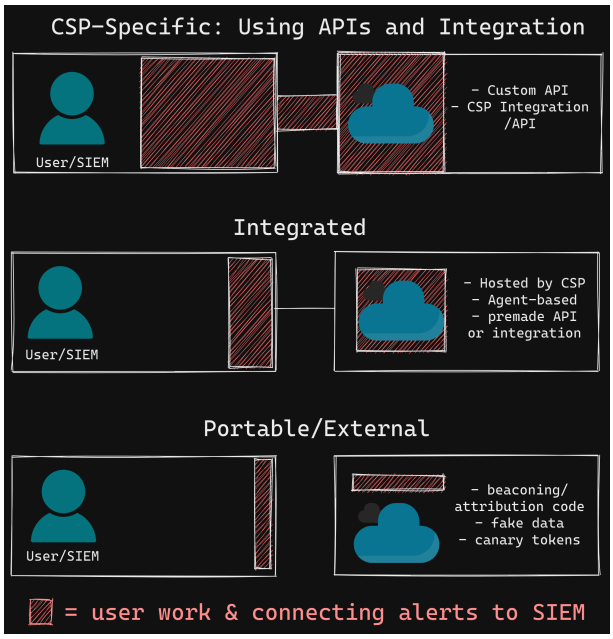

Fig. 2. These categories are designed to show the levels of implementation when it comes to using cyber deception methods in the cloud. It is important to note that some technologies and methods may belong to any number of categories in different contexts.

1.4 Expert Opinions on Cyber Deception

I thought it to be productive to look at expert practitioners and CISO (chief information security officer) opinions on cyber deception technology. Ironically, I owned a book that surveyed 54 cybersecurity practitioners with questions obtained from Twitter users in cyber. One of the questions was: "Has your organization implemented any deception technologies, and if so what effect has that had on the blue team's detection capabilities?" As I read through the answers, it became apparent that there was a lot of support but just as much uncertainty about the practical applications of the technology. To be thorough, I aggregated all of the answers by name, then labeled the answers with the below attributes:

- Degree of Support (1-10) (x-axis)
- Degree of Confidence (1-10) (value)
- Support x Confidence (y-axis)

The result showed the distribution of opinions with more weight going to confidence (darker) answers. Out of the 54 total respondents in the book, interestingly only 33 answered. This is an interesting finding because rarely did questions go unanswered or ignored by the respondents.

One possible reason for this was that most respondents were uncertain about the technology. Another is that these high-profile experts do not want attackers to potentially know about their methods.

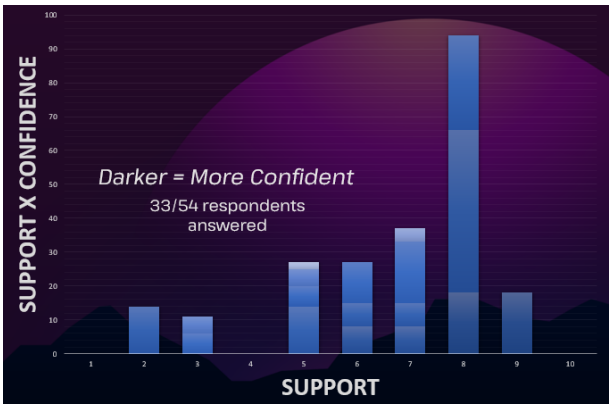

Fig. 3. Graph showing cybersecurity expert and practitioner support for cyber deception technologies. This is taken from my presentation. Darker colors mean that the answers were more confident which also equates to more weight/height.

The results were also fed through GPT-4 with extensive prompts to gather a somewhat objective look at the answers. The result shows that most of the dissenting opinions tended to have less confidence. Also, the most non-supporting answers were either ”No” or ”We should focus on these other controls more.” One supporting answer stated that deception tech gets ”the most bang for your buck.” Below are some takeaways and reasoning from both sides:

- Support
  - Early warning systems are cost effective
  - Helps learn adversary techniques
  - Tons of visibility
  - **LOW False Positive Rate**
- Opposition
  - Only for mature organizations
  - Time is better spent elsewhere
  - Not a mature market yet
  - **HIGH False Positive Rate**

You’ll notice conflicting statements of ”LOW” and ”HIGH” ”False Positive Rates.” Essentially, one side is saying that Cyber Deception alerts are hard to prioritize and produce lots of noise and the other side is saying the exact opposite.

1.4.1 An Explanation for HIGH vs LOW False Positive Rates with Cyber Deception Technology

The reasoning for the conflicting conclusions about false positive rates with cyber deception technology resides precisely in implementation and directly relates to the implementation categories put forth in this paper. For example, canary files can be used in file systems to detect ransomware [22]. Ransomware is a simple and quite effective method of holding an entity’s data for ransom by encrypting it. Canary files can be used as a tripwire for when these ransomware programs start encrypting files on a system. The second the file is encrypted, privileged system calls and actions like encrypting data are locked down. However, say that the implementer of these canary files puts them right on the desktop. This might cause the suspicious user to manipulate or delete them which would result in lots of false positives and headaches for business operations. However, if the file had been placed somewhere that only a hacker would ever go, then it would only activate when there is surely an incident happening.

The false positives or lack thereof are dependent on the type of cyber deception technology and the implementation methods and context.

2 Implementation

2.1 Portable/External Deception Technology

This is the category that seemingly has the best cost-effectiveness and general applicability out of all of the 3 implementation categories devised in this paper. The portability aspect refers to how the callback, beaconing, or alerting functionality is achieved. In order to do security, we need visibility. Typically, visibility is handled with a SIEM. With the CSP-specific implementation or the integrated cases, this means utilizing an integration or API to forward alerts or information into the SIEM. Every different detection case will be handled differently and some will be more manual than others even though every one of them are solutions to detect or prevent incidents. On the contrary, portable deception tech does not rely on the abstractions of the CSP service type or their interface. Instead, external or portable means for delivering information or alerts are relied upon.

I hypothesize there are two types of portable deception technology:

1) **Portable Alerting Instruction** - This type of deception tech beacons, calls back, or sends a message over the internet once the portable and hidden or obfuscated code is run. Usually, the alerting instruction is unintentionally activated by the threat actor or by a system the defenders predict would use the instructions. Example: Honeybadger - implants code in Excel to run and get the location of the device which opened it [5].
2) **3rd Party Reliant Fake Data** - This type of deception technology involves mimicry just like with the portable alerting code. However, the system, process, or components that allow for direct alerting to the defender exists due to external or 3rd party systems that result in the alerting functionality. Example: Fake data or fake credit card info - the threat actor may go to use the fake data in some sort of system that allows them to benefit, except this data is designed to be watched for use which causes an alert to be sent to the defenders.

2.2 3rd Party Reliant Fake Data

I believe this is a strong solution for deception technology. The benefit with 3rd party reliant data is that the defenders

do not have to rely on some alerting or beaconing code to run in order for it to work. Instead, the data itself is portable in extremely small amounts. This data could amount to a string of numbers that are subsequently utilized in some external system. The goal here is to find anomalies. The only problem with this category is that the use cases have lots of nuances and the types of data that can be used are currently limited, unexplored, or require building infrastructure to get them working which defeats the purpose. The only polished example of this type of technoloy would be canarytokens[.]org.

2.3 My Tech Stack and Implementation - "Eleplanter"

Eleplanter is a Python program that utilizes APIs and AI to achieve automated canary token implantation with Postgres databases. Eleplanter capitalizes on portable 3rd party reliant fake data.

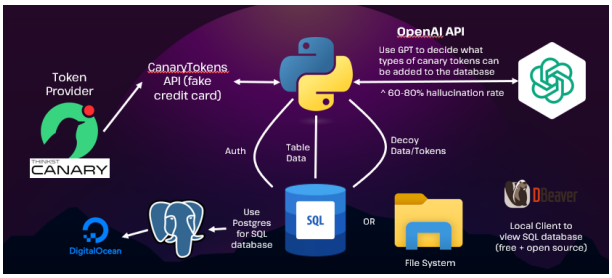

Fig. 4. Tech stack and architecture for Eleplanter. Taken from my presentation.

Outline of the components:

- Token Provider (Canarytokens)
  - Thinkst Canary is the most polished canary token provider on the web. However, they limit API use to organizations that pay at least $5k yearly. Therefore, I generated manual canary tokens at canarytokens.org, then put the relevant data into a canaries.json file.
  - There are limited types of "3rd Party Reliant" Canary tokens available:
    - **email** - unique email address that will alert you when emails are sent to it
    - **http** - fake URL that activates an alert when visited
    - **dns** -a fake unique resolvable domain that alerts the user when resolved
    - **aws_key** - fake aws_access_key_id ,aws_secret_access_key, and associated region which alerts the user when the API key is used
    - **credit_card** - fake credit card info which sends an alert when used
    - **fast_redirect** - fake URL that activates an alert when visited, but also redirects the user who activates it
- Python Program
  - This program runs locally and utilizes various configuration JSON files (credentials.json, canaries.json, and canaries_schema.json) to 1) connect to the database and relevant APIs, 2) identify suitable tables and columns to apply canaries to, and 3) update the database with canary implanted data.
  - The program utilizes libraries, including but not limited to faker, psycopg2, tqdm, colorama, colorama, os, pathlib, random, re (regex), csv, and json.
- OpenAI API
  - The "text-davinci-003" model is used with crafted prompts to help with various tasks in the program: 1) look at column and row data to determine the potential for canary use, 2) craft fake row data, and 3) utilize canary components to implant organic instances of the canarytoken values.
- PostgreSQL Database (hosted on Digital Ocean)
  - Postgres was used out of familiarity. A cheap Postgres instance was rented on Digital Ocean due to the simplicity of setup.
- DBeaver
  - Functions as a local GUI for testing with my remote Postgres database. It's open source and free as well.

3 Evaluation

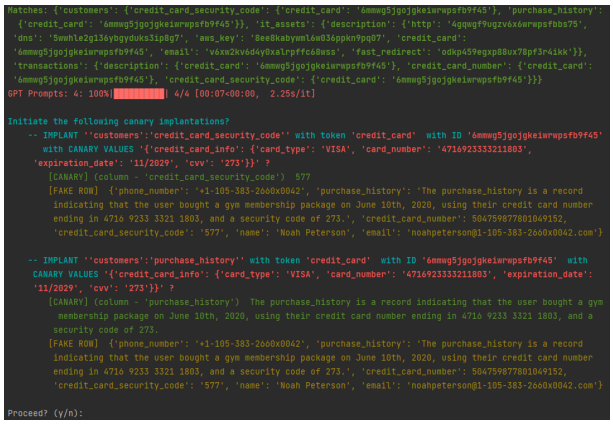

Fig. 5. Part of code that asks for user to approve or deny implantation for instances of canary tokens for a fake row. The "Matches" are every instance of matches with a table:column combination.

Most of the functionality of this program revolves around using the GPT 3.5 (text-davinci-003) model from OpenAI as a programmatic function in two ways: 1) to return a "yes" or "no" for if a column can be implanted with a canary token (if so, it adds the mapping to a csv for later use), and 2) to generate fake data for rows and fake instances that involve canarytoken values.

Frankly, GPT is not made for high-risk scenarios where you must rely on a particularly formatted output or cases where the result must be confident or accurate [11], [12], [18]. However, this case of canary token implantation is not

a high-risk case. However, there are times when improperly formatted data or GPT hallucinations could implant data into a table that is used for another program. This could inadvertently cause the other program to have errors if it doesn't expect a certain format of input generated by a GPT. To fix part of this I implemented a few type checks that parse the GPT response when certain data types are necessary. For instance, an integer column may need to be implanted and GPT sends back a number with some text after it. In this case, the program will pull out the longest number in the text and use that.

Figure 5 shows an instance of a complete hallucination. The first implant involves a "CANARY" value of "577." However, the "CANARY VALUES" include a cvv code of 273. In this case, GPT hallucinated and used one of the example values mentioned in the prompt instead. This was fixed with a longer and more costly prompt. However, this also resulted in thousands more "tokens" (word/sub words) which results in more costs and computation. This is where the idea of "prompt engineering" comes in handy where we need to have a balance between the length of the prompt and the desired result.

In terms of runtime, the intial run of the program to map out potential tables involves hundreds of calls using the OpenAI API and around 5 minutes of runtime for a 4 table database. Doing so once only costs around $1.50 or so.

4 Conclusion

Eleplanter is an effective way to process tabular databases efficiently and implant organically crafted instances of fake rows which utilize canary tokens. These canary tokens function as tripwires for "actions on objectives" so that the defender can know the second that the attacker utilizes data that they have obtained. One may argue that it is too late once this happens, but the fact of the matter is most organizations take almost a year to realize a breach has happened. With a canary token, the organization knows the breach has occurred the instance that attackers attempt to utilize the data they've obtained [14]. Moreover, in the case of database canary implantation, the incident responders can go to the exact system, database, and table where it occurred. Most organizations cannot afford to hold detailed system logs for long periods of time. With this sort of early warning system, organizations can mobilize forensics efforts and stop the bleeding. Cloud services present numerous security problems mostly due to their necessary abstractions and the complexities of interfacing with the user at various levels of service. This makes visibility difficult and subsequently makes detection a heavy lift for most organizations which either requires expensive vendor solutions or unicorn cybersecurity engineers. Having the ability to deploy or sprinkle canary tokens throughout high-risk systems can allow for a backstop and safety net in systems that organizations do not have the resources to interface with. Most of the time, cloud security is hard because it requires custom APIs and integrations. We should not ignore this problem or say we need to simplify the cloud and go yell at developers for how bad they are at designing systems. Rather, we need to innovate and create solutions that can be universally applied and repurposed for various cloud service provider implementations. I hypothesize that portable deception technology that is reliant on 3rd parties for alerting functionality is a flexible way to apply tripwires throughout an organization's tabular databases, and I have shown how that is done using my custom "Eleplanter" program. Some improvements could include making fake row generation more stable, improving prompts, utilizing the Canarytokens API, and attempting to make a serverless version of Eleplanter.

References

[1] *Towards self-adaptive cyber deception for defense.*, 2021.

[2] Rakshit Agrawal, Jack W Stokes, Lukas Rist, Ryan Littlefield, Xun Fan, Ken Hollis, Zane Coppedge, Noah Chesterman, and Christian Seifert. Long-term study of honeypots in a public cloud. *IEEE Xplore*, page 1–4, 06 2022.

[3] Hamad Almohannadi, Irfan Awan, Al Hamar, Andrea Cullen, Jules Pagan Disso, and Lorna Armitage. Cyber threat intelligence from honeypot data using elasticsearch. *IEEE Xplore*, page 900–906, 05 2018.

[4] Goyal S B, Pradeep Bedi, Shailesh Kumar, Jugnesh Kumar, and Nasrin Rabiei Karahroudi. Application of deep learning in honeypot network for cloud intrusion detection. *Proceedings of International Conference on Computational Intelligence and Data Engineering*, pages 251–266, 2022.

[5] BHIS. Getting started with tracking hackers with honeybadger, 04 2020.

[6] Marcus J Carey and Jennifer Jin. *Tribe of Hackers Blue Team*. John Wiley Sons, 08 2020.

[7] Tanmoy Chakraborty, Sushil Jajodia, Jonathan Katz, Antonio Picariello, Giancarlo Sperli, and Subrahmanian V S. A fake online repository generation engine for cyber deception. *IEEE Transactions on Dependable and Secure Computing*, 18:518–533, 03 2021.

[8] Jin-Hee Cho, Dilli P Sharma, Hooman Alavizadeh, Seunghyun Yoon, Noam Ben-Asher, Terrence J Moore, Dong Seong Kim, Hyuk Lim, and Frederica F Nelson. Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys Tutorials*, 22:709–745, 2020.

[9] Brian Corcoran. A comparative study of domestic laws constraining private sector active defense measures in cyberspace. *Harvard National Security Journal*, 11:1, 2020.

[10] Hai Jin, Zhi Li, Deqing Zou, and Bin Yuan. Dseom: A framework for dynamic security evaluation and optimization of mtd in container-based cloud. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2019.

[11] Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. Gpt-4 passes the bar exam, 03 2023.

[12] Anis Koubaa. Gpt-4 vs. gpt-3.5: A concise showdown. *Preprints.org*, 03 2023.

[13] Martin C Libicki, Lillian Ablon, and Tim Webb. *The defender's dilemma: Charting a course toward cybersecurity*. Rand Corporation, 2015.

[14] Ponemon Institute LLC. Cost of a data breach study, 2022.

[15] Stefan Machmeier. Honeypot implementation in a cloud environment, 01 2023.

[16] Nitin Naik and Paul Jenkins. Discovering hackers by stealth: Predicting fingerprinting attacks on honeypot systems. *IEEE Xplore*, page 1–8, 10 2018.

[17] Eric Nunes, Paulo Shakarian, Gerardo I Simari, and Andrew Ruef. *Artificial intelligence tools for cyber attribution*. Springer International Publishing, 2018.

[18] OpenAI. Gpt-4, 03 2023.

[19] Kyungmin Park, Samuel Woo, Daesung Moon, and Hoon Choi. Secure cyber deception architecture and decoy injection to mitigate the insider threat. *Symmetry*, 10:14, 01 2018.

[20] Jeffrey Pawlick, Edward Colbert, and Quanyan Zhu. A game-theoretic taxonomy and survey of defensive deception for cybersecurity and privacy. *ACM Computing Surveys*, 52:1–28, 08 2019.

[21] Sampsa Rauti. Towards cyber attribution by deception. *Hybrid Intelligent Systems*, pages 419–428, 08 2020.

[22] Abdul Rahim Saleh, Gihad Al-Nemera, Saif Al-Otaibi, Rashid Tahir, and Mohammed Alkhatib. Making honey files sweeter: Sentryfs – a service-oriented smart ransomware solution, 08 2021.

[23] Sailik Sengupta, Ankur Chowdhary, Abdulhakim Sabur, Adel Alshamrani, Dijiang Huang, and Subbarao Kambhampati. A survey of moving target defenses for network security. *IEEE Communications Surveys Tutorials*, 22:1909–1941, 2020.
[24] Dave Shackleford. Sans 2022 cloud security survey — sans institute, 03 2022.
[25] S Sivamohan, Sridhar S S, and S Krishnaveni. Efficient multi-platform honeypot for capturing real-time cyber attacks. *Intelligent Data Communication Technologies and Internet of Things*, pages 291–308, 2022.
[26] Alexander Washofsky. Deploying and analyzing containerized honeypots in the cloud with t-pot, 09 2021.
[27] DONNIE WENDT. Addressing both sides of the cybersecurity equation, 09 2019.
[28] Ben Whitham. Minimising paradoxes when employing honeyfiles to combat data theft in military networks. *IEEE Xplore*, page 1–6, 11 2016.
[29] Quanyan Zhu. The doctrine of cyber effect: An ethics framework for defensive cyber deception, 02 2023.