Detecting Complex Cyber Attacks Using Decoys with Online Reinforcement Learning

Marcus Gutierrez
University of Texas at El Paso

DETECTING COMPLEX CYBER ATTACKS USING DECOYS

WITH ONLINE REINFORCEMENT LEARNING

MARCUS GUTIERREZ

Doctoral Program in Computer Science

to my

FAMILY

with love

DETECTING COMPLEX CYBER ATTACKS USING DECOYS WITH ONLINE REINFORCEMENT LEARNING

by

MARCUS GUTIERREZ

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

May 2023

Abstract

Most vulnerabilities discovered in cybersecurity can be associated with their own singular piece of software. I investigate complex vulnerabilities, which may require multiple software to be present. These complex vulnerabilities represent 16.6% of all documented vulnerabilities and are more dangerous on average than their simple vulnerability counterparts. In addition to this, because they often require multiple pieces of software to be present, they are harder to identify overall as specific combinations are needed for the vulnerability to appear.

I consider the motivating scenario where an attacker is repeatedly deploying exploits that use complex vulnerabilities into an Airport Wi-Fi. The network administrator attempts to detect these exploits through the use of cyber decoys. The network administrator chooses a fake device, which has no valuable data, to insert into the network in hopes that the attacker believes the decoy is a legitimate device for exploitation. However, for each interaction, the network administrator needs to make an active selection of which decoy device to insert into the network. I investigate this motivating scenario as a learning problem where the network administrator wishes to learn which decoy settings are best to capture the most exploits at a given time to learn about the most complex vulnerabilities.

Because I model that the network administrator is discovering new complex vulnerabilities over time, there is a lot of uncertainty in this problem. I resort to the Multi-Armed Bandit (MAB) reinforcement learning framework to address some of this uncertainty. The MAB problem epitomizes the exploration-exploitation dilemma. The network administrator wishes to capture the most exploits, so it makes sense to frequently use the decoy settings that have worked well in the past (exploitation), but there may be better-performing decoy settings that are insufficiently explored. Furthermore, the MAB framework emphasizes the decisions and actively ignores some complexities with the problem structure, as is the case with the adversarial MAB.

I first investigate how basic MAB solutions adapt to this adversarial cyber deception environment with complex vulnerabilities in a highly dynamic environment. Vulnerabilities overall are not static; new ways to exploit them get discovered, patches are developed, and new mitigation techniques are discovered. I demonstrate that basic MAB defender solutions are capable of adapting to a slow-learning attacker with an evolving arsenal of exploits. In this evolving exploits model, exploits decrease in detection value over time and the attacker gains new hidden exploits to use over the course of the interaction. Furthermore, due to the problem structure that arises from the inclusion of complex vulnerabilities, basic MAB solutions do not fully utilize the contextual information of the environment. Despite all this, I show that the basic MAB solutions do adapt and begin to approach optimal decision-making. With minor extensions to these algorithms, I also show that improvements can be made to better exploit the complicated problem structure formed by the complex vulnerabilities.

The evolving exploits work analyzed basic MAB strategies with a slow-learning attacker, however, this is not always an appropriate modeling of an attacker. In many cases, the attackers may be intelligent human hackers. In this case, it is important to understand how humans learn and make decisions in cyber adversarial environments. I look to a simplified adversarial interaction to better understand how humans will learn amidst various defenses. I detail a human experiment we ran with over 300 human participants that played a 50 round game versus 3 different simple defender algorithms. We found that human attackers quickly and efficiently discover static defenses, including fixed random defenses. The basic adaptive defender proved much harder for the human participants to attack.

If we wish to develop new defensive solutions and understand how they might perform against human attackers, it may be infeasible to repeatedly run an extensive experiment with human participants. Instead, it would be ideal to have a realistic model that acts in these cyber deception environments as a human would. We developed a cognitive model, from the Instanced-Based Learning (IBL) framework, that exhibits the same cognitive biases found in humans. With parameter tuning, we demonstrated that the IBL agent performed closely to the human participants in the experiment compared to other predictive models from reinforcement learning.

Once I demonstrated that basic MAB solutions are capable of adapting in these cyber adversarial environments with complex vulnerabilities and we have a cognitive model that performs as humans do, I finally address the problem structure that forms from complex vulnerabilities. I formalize the decoy feature selection problem with complex vulnerabilities and introduce a new MAB variant that directly addresses this problem structure formed by complex vulnerabilities, titled the Bipartite Edge Detection problem. I show that this Bipartite Edge Detection problem appears in multiple applications and I provide an initial solution with contextual MABs. I introduce a novel approach to modeling experts that directly addresses the Bipartite Edge Detection problem via the well-known contextual MAB algorithm: Exponentially-weighted algorithm for Exploration and Exploitation with Expert advice (EXP4). My version of EXP4 outperforms all the previously seen basic MAB solutions and an IBL defender and approaches optimal play relatively quickly. The IBL attacker proves a challenging adversary and no particular defensive algorithm did notably well against it. I leave room for future work to improve my version of EXP4, introduce new bandit algorithms, investigate IBL further for these problems, and implement this work in a realistic cyber environment.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern cybersecurity defense is highly dependent on reliable intrusion detection and the ability to identify and patch vulnerabilities. Often, attackers exploit multiple vulnerabilities found in the network to achieve their goals. Meanwhile, the cybersecurity defense community has put great effort into identifying as many vulnerabilities as possible to reduce the entry points of hackers and other cyber threats. Since April 2023, there are over 200,000 documented vulnerabilities according to the National Institute of Standards and Technology with the number of annual discoveries increasing every year [4].

The issue with discovering vulnerabilities in a system lies in the sheer amount of features involved. Applications, network protocols, operating systems, physical wiring, humans, browsers, and hardware are all possible vectors for attack and will each have their own set of vulnerabilities that are possible to exploit. Frequently, each interchangeable feature goes through its own individual, rigorous testing and hardening to remove or mitigate any known vulnerabilities that a cyber threat might exploit. When testing a system, often each individual feature will be treated as a separate unit where multiple vulnerabilities are associated with it (e.g., the microphone driver application has 3 known vulnerabilities), as is the case with unit testing.

More complex vulnerabilities may require unique combinations of features. For example, a new vulnerability might arise when a specific application runs on a specific operating system. I will refer to this class of vulnerabilities as *complex vulnerabilities* that require multiple features to be present (e.g., a vulnerability might require both a specific printer driver version *AND* a specific smart home hub application). Conversely, a single vulnerability may exist in multiple features where any equivalent feature will expose the vulnerability, such is the case with a vulnerability found in a dependency where any feature containing this dependency is vulnerable (i.e., application 1 *OR* application 2 *OR* application 3 all share the same vulnerable dependency, so any would be vulnerable). This can lead to complex vulnerabilities being described as intricate Boolean expressions with regard to their required features.

An exploit is a malicious utilization of a vulnerability, that usually has a payload or objective achieved once the vulnerability is exploited. There may be many different exploits utilizing the same vulnerability, but for this work, I focus on a 1-to-1 relationship and model exploits so that we can also express an exploit as a Boolean expression of features. With this terminology, an attack would be an instanced usage of an exploit, which would require a time component of when the exploit was used.

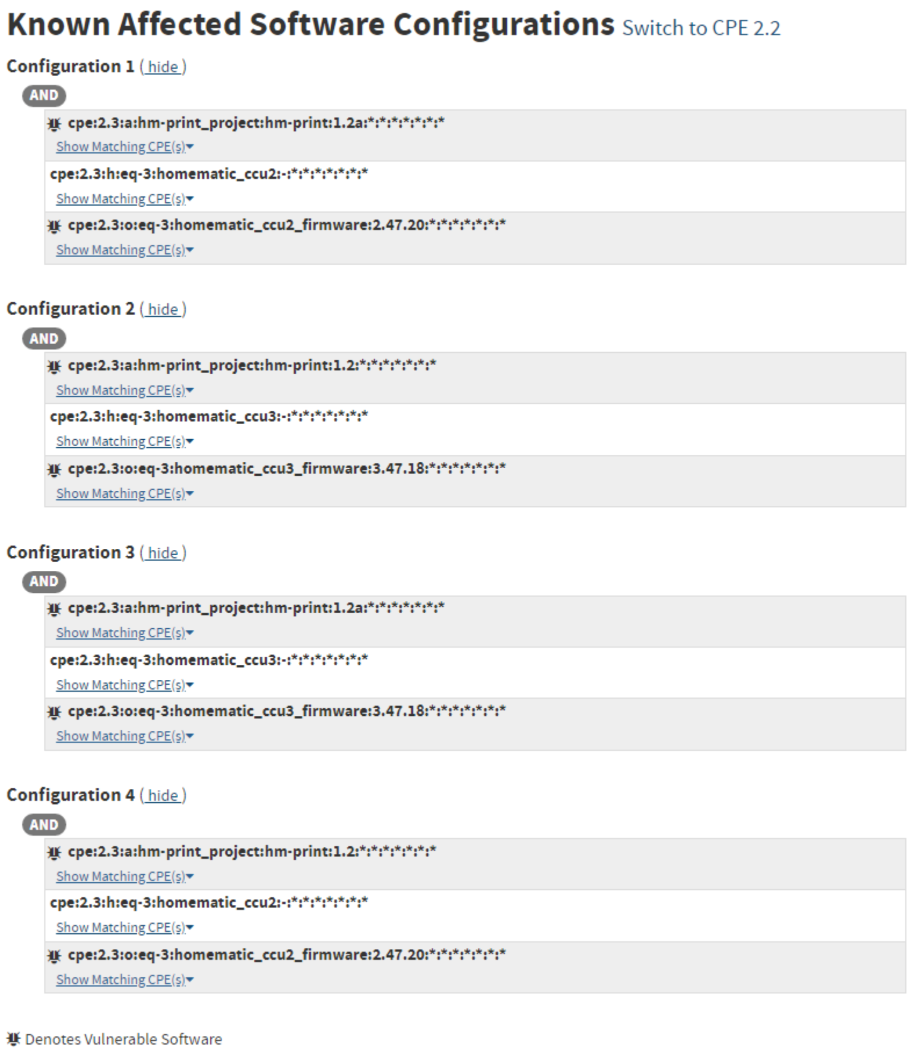
In this work, I will focus on detecting exploits deployed by attackers. An effective approach is through the use of cyber deceptive decoys. Network administrators can deploy decoy devices into their network, like an Airport Wi-Fi, to detect and monitor malicious activity. An attacker may enter the Airport Wi-Fi and target devices and critical infrastructure through the network then leave. If the network administrator can strategically select a decoy to place into the network and have the attacker interact with the decoy, she can log and analyze any potential exploit used, even previously unknown exploits. However, one of the primary problems I address in this dissertation is how to select which decoy to use. If a decoy is a single fake computer, the network administrator needs to decide which operating system it should run, which hardware, and which applications it should have. Each feature the network administrator decides on for the decoy opens a unique attack surface that an attacker can use to exploit.

There are far too many features and too many combinations to represent in a single decoy, so I instead consider that the network administrator has a collection of pre-configured settings for the decoy. I model this repeated interaction as a learning problem, where the network administrator is actively selecting decoy configurations to discover which is the best configuration at a given time. This allows the network administrator to discover new complex vulnerabilities and their associated exploits and learn the attacker's patterns. I will investigate how basic learning strategies perform for decoy configuration selection amid complex vulnerabilities. However, as the network administrator is learning, the attacker is learning as well. Attackers may be skilled hackers who are keenly aware of decoys and are learning the network administrator's favorite decoy configurations to avoid. I investigate how the defender's basic strategies and learning impact realistic human attackers and how we can extract that human attacker's decision-making for defender evaluation. I then address better learning strategies to consider learning with complex vulnerabilities, which pose interesting challenges for basic learning frameworks like the Multi-Armed Bandit.

1.0.1 Real World Complex Vulnerabilities

CVE-2019-18939 is a real-world example of a complex vulnerability [2]. This vulnerability allows for remote code execution when a specific combination of a printer add-on application, smart hub hardware, and smart hub firmware is present. The National Institute of Standards and Technology (NIST) catalogs each vulnerability as a Common Vulnerabilities & Exposures (CVE). This CVE has one of the highest severity scores of 9.8 (10.0 being the maximum for Common Vulnerability Scoring System version 3) and allows for remote code execution. For a system to be considered vulnerable to CVE-2019-18939, it requires 1 of the 4 possible configurations which each require 3 Common Product Enumerations (CPEs). CPEs are information technology systems, software, and packages as defined by NIST. Each configuration of CVE-2019-18939 requires a specific printer model, a specific gateway software version, and a specific firmware version. All 4 configurations for this vulnerability with their respective required features/CPEs are shown in Figure 1.0.1.

Although these complex vulnerabilities clearly exist as evidenced by CVE-2019-18939, it is not immediately obvious how common these classes of vulnerabilities are and if they are any more or less dangerous on average than simple vulnerabilities. To understand the prevalence of complex vulnerabilities, I searched NIST's entire National Vulnerability Database (NVD) API, ranging from CVE-1999-0001 to CVE-2023-29421. There are no ways to query



Known Affected Software Configurations Switch to CPE 2.2

Configuration 1 (hide)

AND

🐞 cpe:2.3:a:hm-print_project:hm-print:1.2a:*:*:*:*:*:*:*
Show Matching CPE(s)

cpe:2.3:h:eq-3:homematic_ccu2:-:*:*:*:*:*:*:*
Show Matching CPE(s)

🐞 cpe:2.3:o:eq-3:homematic_ccu2_firmware:2.47.20:*:*:*:*:*:*:*
Show Matching CPE(s)

Configuration 2 (hide)

AND

🐞 cpe:2.3:a:hm-print_project:hm-print:1.2:*:*:*:*:*:*:*
Show Matching CPE(s)

cpe:2.3:h:eq-3:homematic_ccu3:-:*:*:*:*:*:*:*
Show Matching CPE(s)

🐞 cpe:2.3:o:eq-3:homematic_ccu3_firmware:3.47.18:*:*:*:*:*:*:*
Show Matching CPE(s)

Configuration 3 (hide)

AND

🐞 cpe:2.3:a:hm-print_project:hm-print:1.2a:*:*:*:*:*:*:*
Show Matching CPE(s)

cpe:2.3:h:eq-3:homematic_ccu3:-:*:*:*:*:*:*:*
Show Matching CPE(s)

🐞 cpe:2.3:o:eq-3:homematic_ccu3_firmware:3.47.18:*:*:*:*:*:*:*
Show Matching CPE(s)

Configuration 4 (hide)

AND

🐞 cpe:2.3:a:hm-print_project:hm-print:1.2:*:*:*:*:*:*:*
Show Matching CPE(s)

cpe:2.3:h:eq-3:homematic_ccu2:-:*:*:*:*:*:*:*
Show Matching CPE(s)

🐞 cpe:2.3:o:eq-3:homematic_ccu2_firmware:2.47.20:*:*:*:*:*:*:*
Show Matching CPE(s)

🐞 Denotes Vulnerable Software

Figure 1.1: The 4 possible configurations of CVE-2019-18939 as described on the NVD website on April 2022.

the API for complex vulnerabilities directly. Instead, I downloaded every CVE exhaustively over the course of 2 weeks through the NVDLib Python library, which has built-in methods to query the API [5]. Each CVE was converted from an "nvdlib.classes.CVE" object into a Python JSON dict to be stored on my personal computer. Once downloaded, I could systematically crawl through all 211, 952 CVE files and evaluate the status of their complex vulnerability classification. I removed all CVEs that were rejected as official vulnerabilities or had no configuration section listed, leading to a total of $199,298$ eligible CVEs for classification.

In each “nvdlib.classes.CVE” object, there is a list of configurations (the majority of CVEs only contain a single configuration) and each configuration lists an “operator” field which is either an “OR” or an “AND.” This operator field describes the relationship of the CPEs listed in each configuration for the CVE. If a configuration for a CVE has multiple listed CPEs and lists an “OR” operator, this means any of the CPEs present will yield the same vulnerability. This is largely used for listing multiple versions of the same software, which will yield different CPEs, but when modeling can typically be considered the same application.

For example, CVE-2005-0179 is a Linux kernel vulnerability that allows for a denial of service attack via the “mlockall” call [1]. This CVE has only a single configuration with an “OR” operator field listed and 122 listed matching CPEs. However, these CPEs are listing various incremental versions of Linux kernel 2.4.x and 2.6.x. Depending on the modeling of the problem, it may be appropriate to consider this vulnerability a complex vulnerability as “OR” logic is definitely expressed, however depending on the scope, it may also make sense to combine all the listed CPEs into a single feature (e.g., the single feature would be “Linux Kernel 2.4.x-2.6.x”). For this initial search, I take the latter approach as I am considering a wide breadth of features for an adversarial cyber deception interaction. In the latter approach, CVE-2005-0179 would be considered a simple vulnerability.

For classifying complex vulnerabilities in the NVD, I consider two major criteria: 1) If multiple configurations are listed or 2) if a configuration uses the “AND” operator. The former can be exemplified by CVE-2021-20229, which is a vulnerability with PostgreSQL that allows for retrieving values of an entire table [3]. CVE-2021-20229 has 3 separate configurations that list separate CPEs for each configuration that include PostgreSQL, Redhat Linux, and Fedora. In general, when multiple configurations are listed, it implies *OR* logic but across different CPEs or features. The latter criterion that requires an “AND” operator can be demonstrated with CVE-2019-18939 which requires all the listed CPEs in each configuration to be present for the vulnerability to emerge.

Filtering with the described criteria above, there are 32,998 complex vulnerabilities which represent 16.6% of the total eligible vulnerability population as of April 6, 2023. An exhaustive list of all known complex vulnerabilities that meet the complex criteria can be found at: "https://github.com/MarcusGutierrez/complex-vulnerabilities" [69, 70]. In this list, I removed the "CVE-" part of the CVE ID to save space.

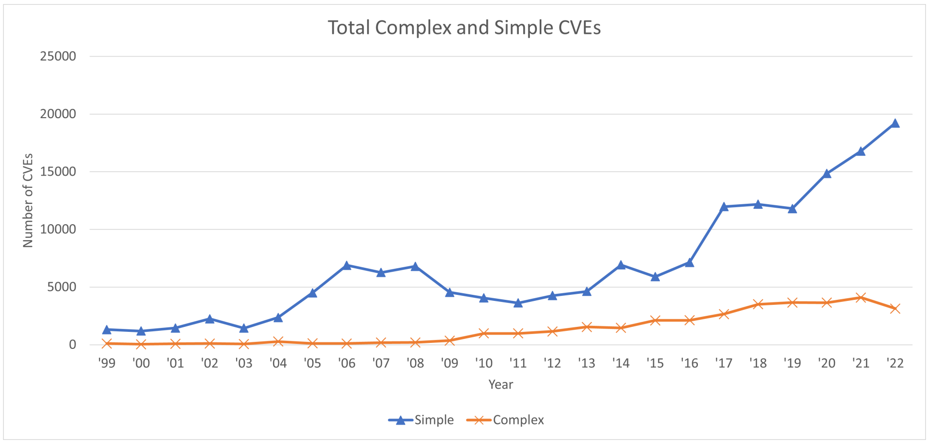


Figure 1.2: Total count of simple and complex vulnerabilities per year from 1999 to 2022. Data from the year 2023 is omitted due to the small data set as only January 2023 to March 2023 has been collected.

Investigating some aggregate statistics on the list of eligible CVEs, complex CVEs have an average CVSS value of 7.12 compared to the 6.8 average of the simple CVE list (out of a possible 10.0)[1]. The CVSS value also has a 3-level severity value (Low, Medium, and High). High-severity CVEs represent 45.2% of the total simple CVEs. Meanwhile, High-severity CVEs represent a total of 51.5% of the total complex CVE population. This demonstrates that on average, complex vulnerabilities are not only harder to find, but tend to be more severe than simple vulnerabilities. Figure 1.0.1 shows the total number of simple and complex CVEs per year. The total number of discovered CVEs seems to be increasing with each passing year as more software is developed, better testing tools are created, new

[1]Both the simple list and complex list of CVEs had a standard deviation of 1.86

testing techniques are developed, and adversaries are getting more sophisticated at finding complex vulnerabilities to exploit. While the total number of simple vulnerabilities seems to far outweigh the complex vulnerabilities, the complex vulnerabilities are still trending upwards each year.

Complex vulnerabilities represent a significant portion of the total vulnerability population and more are discovered each year. These vulnerabilities are inherently harder to find due to strict feature requirements but suggest higher severity on average than their simple counterparts.

1.0.2 Modeling Complex Vulnerabilities

We can describe the features required for a CVE to be present using a Boolean expression. When describing CVEs as Boolean expressions, a feature variable represents a Common Platform Enumeration (CPE) which could be hardware, software, or firmware that contains vulnerabilities and a feature f_k is **true** if the system has that feature present.

Consider as an example CVE-2019-18939; we can assign each Common Platform Enumeration (CPE) found in Figure 1.0.1 to a feature variable f where f_k represents the presence of a feature in a configuration. Features f_0 through f_5 represent the six unique CPEs described by CVE-2019-18939. Using features and Boolean expressions, we can verify if any software configuration is vulnerable to this vulnerability if the logical feature formulation presented in equation 1.1 is true:

$$(f_0 \wedge f_1 \wedge f_2) \vee (f_3 \wedge f_4 \wedge f_5) \vee (f_0 \wedge f_4 \wedge f_5) \vee (f_3 \wedge f_1 \wedge f_2) \tag{1.1}$$

At a high level, this shows that the vulnerability CVE-2019-18939 is present in a system if any of the four required configurations are present, represented with the logical $\vee$ connectors. Each of the 4 configurations requires 3 specific features to be present, represented by the logical $\wedge$ connectors.

Using this Boolean logical formulation, we can represent more complex vulnerabilities like CVE-2019-18939 and leverage strategic reasoning, like reinforcement learning, for vulnerability discovery. Reinforcement learning frameworks, such as the Multi-Armed Bandit, can quickly and efficiently discover and defend these vulnerabilities using configurable cyber decoys.

The obvious limitation to modeling features as Boolean variables is that not all features may be naturally modeled using binary values. Some non-Boolean examples include numerical features (e.g., response time) and probabilistically triggered features (e.g., the random ordering of packets). Temporal requirements may need to be met as well (e.g., some action is performed before another). All of these non-Boolean features can be seen in the descriptions of cyber attack graphs [80]. In attack graphs, there are temporal requirements such as privilege escalation that are required before other phases of a complex attack campaign can be fulfilled.

In the future, it may be fruitful to explore other logical paradigms outside of simple Boolean logic. For an initial approach, we can map some more nuanced feature requirements into a Boolean variable. For instance, consider an exploit that utilizes the response time f_i of a system. If it is above some threshold, the exploit can target the system. Instead of modeling feature f_i as the response time of a signal, we may simply model f_i to represent if the response time is above said threshold or not, yielding a binary result.

Discovering and identifying these complex vulnerabilities is much more challenging as the vulnerability search space increases exponentially with the increase of more features observed as analyzed by Kuhn et al [89]. I will investigate the challenges that arise when considering complex vulnerabilities and their respective Boolean expressions. I argue that this class of vulnerabilities deserves more attention since they represent a significant proportion of the total vulnerability population and on average are more severe. I then propose reinforcement learning solutions that provide a good baseline to handle these vulnerabilities.

1.0.3 Learning in Dynamic Environments

Identifying complex vulnerabilities proves to be a challenging problem as discussed previously, however, a simple static search over all of our devices and their applications is infeasible and unrealistic. Furthermore, we may not just want to learn about which vulnerabilities exist in our system, but also which ones are the most exploitable and desirable to attackers.

An approach we can take is to use cyber deception to learn about new vulnerabilities, exploits, and attack patterns. By allowing attackers the freedom to exploit a cyber decoy, like a honeypot, cyber defenders can learn about new vulnerabilities in their systems, attack preferences, and attack distributions. Meanwhile, we are monitoring the decoy by logging all interactions with it [126]. However, the defender must actively select which decoy configuration to present in the network. To address this, I formulate this interaction as a learning problem where the network defender is actively switching various decoy configurations, that all have different feature combinations, in hopes of detecting exploits from the attacker. However, vulnerabilities and their exploits are naturally dynamic. As time goes on, patches are developed and mitigation techniques are discovered to reduce the harm vulnerabilities may pose. In addition to this, new vulnerabilities are constantly being discovered (by both the defensive and offensive sides).

In chapter 4, I investigate how basic learning strategies will perform in a highly dynamic environment with complex vulnerabilities. The complex vulnerabilities and their associated exploits will decrease in value over time and new, hidden exploits are added to the attacker's arsenal. Since the vulnerabilities and the exploits that utilize them may be complex in their relationships to their required features, it is best if we make as few assumptions about the environment as possible. The Multi-Armed Bandit (MAB) framework excels at ignoring the potentially complex structure of an environment and instead solely focuses on the decisions available to the decision-maker.

I am addressing this complex vulnerability intrusion detection problem with MABs because of the nature of so much uncertainty. Solutions from game theory, such as with extensive form games, are inherently prospective and tend to model all possible future states of a given environment and action space. However, in cybersecurity, there are numerous unknowns. By utilizing MABs and focusing primarily on the defense configurations themselves, we can model scenarios where the exploits and their respective complex vulnerabilities are completely unknown to the defender and the defender must learn about new exploits over the course of an interaction. In addition, because I am modeling a repeated interaction with many decoy configurations for the defender and exploits for the attacker, the total state space is massive and the problem would be intractable in a game theoretic sense.

This translates to focusing on which defense configurations are best at a given time. However, there is a balance to strike in that we want to make as few assumptions as possible about the potential environment but still leverage any contextual information that is passed along. The primary issue arises when we do not detect an attack at all. If no attack is detected, the defender does not know which other exploit was deployed, simply that the chosen defense configuration did not detect an activity. In the MAB framework, this is known as bandit feedback in that you can only learn about the decision you chose. In chapter 4, I demonstrate that basic MAB solutions are capable at adapting to highly dynamic environments with exploits that change over time. I formalize the MAB problem in detail in section 3.1.

1.0.4 Modeling How Humans Learn

Every stage of cybersecurity, on both the offensive and defensive sides, has human components. Frequently, humans play the direct role of the attacker (in the case of hacking) and defender (IT personnel and network administrators). It is important to understand how humans make decisions in these cyber adversarial environments because any potential automated solution will likely interact with human actors. In the case of defensive cyber deception, we want to be able to demonstrate that any automated defensive solutions we propose can 1) perform well against human attackers and 2) outperform human defenders in the same decision-making. To address these concerns, we need a proper understanding of how humans learn.

Humans are notably poor at generating random defense strategies and should be easily improved upon [88]. However, to verify this, I wish to evaluate how effective human decision-makers might perform at selecting the best decoy configurations each round repeatedly and compare those results to bandit algorithms.

To model human-like defenders and attackers, I implement the Instanced-Based Learning cognitive model [67]. This cognitive model simulates human-like decision-making through the use of memory instances and the notion of similarity where it utilizes past experiences to inform lesser-known decisions based on how similar the decisions are. IBL also experiences the same recency and frequency biases that humans experience in decision-making where we as humans tend to prefer decisions that frequently yield good outcomes and that more recently yield good outcomes. In chapter 5, I address how quickly adaptive human-like attackers break down static defenses but tend to struggle with adaptive defenses. I compare the IBL model to human participants playing the same adversarial interaction to validate that the IBL model does perform human-like in these repeated adversarial settings. This result will be crucial when evaluating other defensive strategies as we wish to understand how they might perform versus human hackers.

1.0.5 Decoy Configuration Selection

I introduce and formalize the *Decoy Configuration Selection with Complex Vulnerabilities* (DCS-CV) problem in chapter 6. I explore this new problem space and the issues that arise when learning in an environment with complex vulnerabilities. Cyber decoys and, cyber deception in general, offer a unique approach to learning about novel exploits [109, 115, 134]. Since no legitimate actor should interact with the decoy as it produces no real work or holds any valuable data, any interaction at all is deemed suspicious and scrutinized. This built-in intrusion detection allows us to monitor attackers acting candidly inside our decoy objects, which could be real or emulated machines posted on a network. In this work, I investigate the case where a cyber defender is frequently altering their cyber decoy to present different features for an attacker to target. I leverage reinforcement learning to explore different feature configurations and exploit the best configurations at a given time.

Throughout this work, I assume the decoys and by extension, the defender, are able to recognize complex vulnerabilities when interacting with them. More specifically, I assume the defender can recognize unique exploits and their feature requirements. This does not mean the defender must necessarily understand the inner workings of every complex vulnerability and their respective exploits, but perhaps reads the honeypot logs and recognizes the exploit signatures and which features were needed for an attack. Utilizing tools like CVSS, a defender can also understand the basic characteristics of an exploit (e.g., does it require the internet, which privileges are required, etc.).

Decoys are particularly useful and uniquely capable of detecting zero-day exploits, which are previously unknown exploits. Network administrators can discover and learn about a previously unknown vulnerability if an attacker uses a zero-day exploit on a decoy. Numerous works have shown that zero-day exploits are classifiable through the use of learning in conjunction with decoy deception [52, 128, 12, 78, 109, 115, 134]. I take this notion one step further in assuming that the defender can recognize all exploits (i.e., simply being able to distinguish individual exploits from one another) and identifying which features are required for each feature, much like holding a database or catalog of the recognized exploits.

Where my work on evolving exploits in chapter 6 addresses simple bandit approaches to the complex vulnerability problem, in chapter 6 I directly address the complex vulnerability problem through a MAB solution with the introduction of a new MAB variant. In addition to an existing MAB algorithm, I demonstrate that contextual MAB solutions are capable of addressing the complex reward structures that form when considering complex vulnerabilities for a learning problem and make notable improvements over the basic strategies seen in chapter 6.

1.0.6 Research Questions

Throughout the rest of my dissertation, I investigate the role complex vulnerabilities play in repeated cyber deception. As I address my research questions, I primarily model and consider a rapidly changing cyber environment like inserting decoys into a public Airport Wi-Fi, however, there are numerous other applications that my model can address which I will discuss later. My dissertation will focus on answering the following questions:

Q1. How capable are standard Multi-Armed Bandit (MAB) solutions at adapting to attackers that exploit complex vulnerabilities that evolve over time?

In chapter 4, I model MAB defenders selecting from a static list of pre-configured feature combinations, called defenses, to detect exploits that evolve in value over time. Since each defense contains multiple features, we can choose to either model the decision-maker as choosing preset configurations as singular distinct entities (i.e., a non-combinatorial bandit problem) or as choosing feature combinations (i.e., a combinatorial bandit problem). I first investigate the non-combinatorial model with no contextual information. With this model formulation, classic bandit algorithms like Upper Confidence Bound (UCB) and Exponential-weight algorithm for Exploration and Exploitation (EXP3) ignore the inherent combinatorial structure of the problem.

However, vulnerabilities, complex or not, are notably dynamic elements. On initial discovery of a vulnerability, it may start as highly damaging to infrastructure with little to no mitigation techniques. However, as vulnerabilities are patched or network intrusion systems learn about a vulnerability, the exploits that utilize said vulnerability might wane in the value they provide to attackers [60]. Once older vulnerabilities essentially "die" off as their respective exploits are essentially useless, newer vulnerabilities are discovered, and the cycle continues.

I investigate the problem of the defender learning and adapting with a static set of honeypot configurations to an adaptive attacker that gains new attack actions over time and the utility of each exploit wanes with each successful detection, which models the gaining of knowledge and experience of a vulnerability and thus improving defense capabilities for that vulnerability. My results show that basic MAB solutions like UCB and EXP3 still learn and adapt in a highly dynamically changing environment even if they are not fully exploiting the inherent combinatorial structure of the problem. Regardless, they do still learn and approach zero-regret, the concept that a decision-maker chooses the optimal decision such that the difference in utility between the best decision and the selected decision.

Q2. How well do human attackers react to standard reinforcement learning algorithms in repeated adversarial interactions?

Considering the human-in-the-loop is critical when validating any adversarial models in cybersecurity [48]. Since I model a repeated adversarial interaction between attacker and defender engaging in cyber deception, it is important to understand how humans perform in these repeated adversarial learning environments. Oftentimes, humans play a critical role on both sides of cybersecurity, either on network defense or on the network intrusion side. It is therefore crucial to understand how humans learn and adapt to repeated cyber deception. In this work, I show how quickly human participants, who played the role of the hacker, adapt to 3 varying difficulties of defenders. The 3 defense algorithms are: 1) a purely static, unchanging defense 2) a fixed random distribution defense, and 3) a combinatorial MAB algorithm called Learning with Linear Rewards (LLR).

Naturally, the human participants performed the worst against the adaptive bandit algorithm, but the surprising takeaway was just how quickly the participants learned and adopted optimal strategies against the 2 other static defenses. This demonstrates the necessity to stay adaptive in these adversarial environments. It is not enough to simply deploy a fixed randomized defense policy, but instead to constantly learn the intruder's attack preferences and adjust accordingly.

Q3. How well will cognitive models evaluate and learn with complex vulnerabilities?

Once we have a good understanding of how quickly humans learn in repeated adversarial settings, we can investigate how they might perform when deploying exploits and defenses with complex vulnerabilities. I leave the collection of human participant data for future work, but I do implement IBL models that are capable of navigating the complicated relationships between exploits and decoy feature configurations. I manage this through the modeling of a similarity function. In essence, IBL does not fully understand the environment, but instead, we can model how similar decisions are from one another.

In the case of an IBL defender, we can model the full mapping between each defense D_i to each exploit E_i and form a truth table about which defenses are capable of detecting each exploit. The similarity between 2 defenses D_i and D_j where $i \neq j$ then is the proportion of the 2 truth tables matching values. Therefore, a similarity of 1.0 would be two defenses that share the exact same truth tables and capabilities of exploit detection whereas a similarity of 0.0 would be considered maximally dissimilar where they have opposite truth tables. This modeling of the similarity function assumes that the defender knows and understands all the exploits. If we wish to model a setting where the exploits are unknown, we can instead just model similarity as the proportion of matching the full truth table of features, where, if we assume some restrictions of exploits, we can reduce the size of the total necessary truth tables (e.g., no exploit will require all features to be present).

I also apply the same modeling for the IBL attacker, where an attacker compares the connections between exploits and defenses. Here, it makes more sense to model an attacker that has full knowledge of all the possible defense configurations and compares the similarities between 2 exploits, as this can form our "insider attacker" who has knowledge of the defender capabilities.

In comparison to even basic bandit algorithms that do not try to utilize the contextual information about connections between defenses to exploits, the IBL defenders struggle to learn fast enough. This result suggests that even basic reinforcement learning algorithms likely improve defenses over a human defender manually choosing defense configurations. The IBL attacker instead performs rather well and induces poor performance to the other defense algorithms, suggesting that more work can be done to defend against smart, adaptive attackers exploiting complex vulnerabilities.

Q4. How much can we improve over naïve non-combinatorial and linear reward solutions for adversarial interactions with complex vulnerabilities?

The 2 major approaches to model the defender deploying decoy configurations using MABs are to 1) model the bandit as selecting the configurations as separate entities or 2) model the bandit as selecting the features. I analyze the cases with 3 baseline bandit defenders for each model. In the simplest setting, we can consider the problem as a non-combinatorial problem where the individual arms on the bandit are the configurations (i.e., defenses) where each defense is independent of one another. The UCB defender takes this approach and ignores all potential contextual information about if any configurations share features with one another or if other defenses might be able to detect the same exploit in a given round. UCB is ultimately discovering the average expected reward of each configuration over time and finding the best configuration, but by ignoring the contextual information of the features, UCB learns slowly. The UCB defender also assumes that each configuration has some true static expected value to be learned.

EXP3 is a more adaptive non-combinatorial bandit that makes no assumptions about the structure of the rewards from each defense, allowing for a potential adversary to change the expected value of each configuration and EXP3 will still be able to adapt. The issue with EXP3 is that it is still ignoring any potential contextual information about the relationships between features, defenses, and exploits.

The other model assumes that the defender selects features and each defense is essentially a unique combination of features. This opens to the area of the combinatorial MAB problem. Arguably the most well-known combinatorial bandit algorithm is LLR. LLR builds on the principles of UCB, but tracks each arm/feature independently. A key assumption of LLR is that it assumes linear reward feedback. This means that the reward function at the end of each round can be modeled as a linear function or the summation of each selected arm in a given round. In this way, when we receive a reward each round, we can just add that reward to each feature separately. The issue with this modeling is that the assumption of a linear reward inherently breaks down when considering complex vulnerabilities. Because we are modeling logical $\wedge$ and $\vee$ vulnerabilities, the reward function cannot be expressed as a linear function [121]. If the defender were to detect an exploit:

$$E_i = f_0 \wedge f_1$$

The defender cannot reward features f_0 and f_1 independently as neither of them contributed to the reward separately. The LLR defender incorrectly takes this approach and, in doing so, incorrectly updates its beliefs about other defenses that have only one of the two features f_0 or f_1.

I introduce and formalize this MAB variant as the Bipartite Edge Detection problem later on. My solution to this variant is to resort to contextual non-combinatorial bandits. Specifically, I utilize EXP4, an algorithm modeled on top of EXP3 that also allows for expert advice. EXP4 has numerous experts and they impart advice in the form of the suggested probability of playing each defense. The EXP4 defender then still plays defenses each round but balances the consensus expert advice with the experiences of each defense. I introduce a way to model the experts for EXP4 that allows for the contextual information of the game, namely how each defense maps to each exploit. My implementation of EXP4 vastly outperforms all other defenders in a simple setting when measuring for cumulative expected regret, but still struggles when facing the IBL attacker, leaving room for modeling the experts in a more adversarial setting.

Chapter 2

Related Work

2.1 Cyber Deception

There are numerous approaches to applying strategic decision-making to cybersecurity and in particular cyber deception. Cyber deception involves misrepresenting information to a cyber actor [137, 81]. I use the definition of cyber deception as described by Yuill: "Planned actions taken to mislead and confuse attackers and to thereby cause them to take (or not take) specific actions that aid computer-security defense. [13, 141]." The most famous example of cyber deception is in the form of honeypots, first illustrated by Stoll in his book on cyber espionage [126]. However, since this original accounting, cyber deception has evolved to include elements that range from all forms of honey-things (e.g., honeynets, honey-tokens, honey-words) to endless files, value fuzzing, and more [84, 62, 95, 77, 142].

All of these techniques are primarily used on the cyber defense side, but cyber attackers have routinely engaged in cyber deception on their end, primarily exploiting the humans in-the-loop through social engineering and more. Cyber attackers also engage in cyber deception through the use of stealth [8, 93, 35, 118]. Frequently, attackers try to hide in a network by avoiding detecting and covering their tracks, such as cleaning log files after the attack. Often, hackers emphasize psychology and exploit the cognitive biases found in the humans in a network. Some newer work has started looking at exploiting the hacker's cognitive biases for cyber deception [137, 47, 57, 13].

2.1.1 Game Theory in Cyber Deception

Throughout this dissertation, I address problems in cybersecurity that lend themselves to learning, however, in critical cases, waiting for a learning algorithm to perform well might be too costly. In these cases, game theory provides a useful alternative to learning in cyber deceptive situations [113, 145, 144, 54]. Miah et al. study the case of two-sided deception using game theory where the defender chooses to alter features of the decoy systems and the real systems [107]. This work of two-sided deception games strongly resembles some of the work I address in this dissertation, primarily through the use of multi-feature decoy defenses. Notably, this work by Miah struggles with scaling, which is partly why I address multi-feature decoy deception with learning. Píbil et al. first introduced and analyzed the Honeypot Selection Game as a zero-sum extensive-form game which initially inspired the work found in this dissertation [114, 85]. This work was foundational for analyzing strategic honeypot selection but found similar limitations to Miah's work through the issues of scalability.

Anwar and Kamhoua present solutions to the problem of cyber deception by obfuscating attack graphs through Partially Observable Stochastic Games [15]. Later, I briefly describe modeling my presented complex vulnerabilities problem through the context of attack graphs with honeynets. Shlenkner et al. present the Cyber Deception Game, which is a zero-sum Stackelberg game, and the defender uses deceptive actions to thwart an attacker [119]. Tsemogne et al. take a slightly different approach by emphasizing honeypot placement in an Internet of Things network using One-Sided Partially Observable Stochastic Games [131]. In this work, they assume the attacker has perfect knowledge of the state of the network and demonstrated that their optimal defense strategy outperforms random defenses.

2.1.2 Reinforcement Learning in Cyber Deception

In recent years, there have been numerous advances in applying online reinforcement learning toward cyber deception as well. Huang et al. analyzed the effectiveness of using Markov Decision Processes (MDP) and Q-Learning for cyber deception in the use of sending attackers to a honeynet [79]. I primarily investigate the MAB problem, which is a special one-state case of the MDP problem. The same authors also investigated the usage of Q-Learning to robustly adapt cyber defenses when the attacker falsifies cost signals. Similarly, Wang et al. utilize Q-Learning for honeypot allocation [138].

Pauna and Bica leverage ϵ-Greedy for an adaptive SSH honeypot called RASSH [111]. They later updated their work on adaptive honeypots to leverage Q-Learning with QRASSH [112]. Dowling et al. utilize the State-Action-Reward-State-Action (SARSA) algorithm for solving MDPs for adaptive honeypots [50]. Walter et al. used Q-Learning and Deep Q-Learning for strategic decoy placement in the realistic "cyberbattlesim" environment [136]. Luo et al. utilize Q-Learning for intelligent honeypot interactions [98]. Du et al. also consider a game of decoy placement on attack graphs using MDPs, but solve their game using Proximal Policy Optimization (PPO) [51]. Similar to the limitations found with game theory, the authors ran into a scaling issue, allowing only for modeling smaller-scale interactions.

On the detection of attacker cyber deception, Venkatesan et al. use Bellman optimality to detect stealthy botnets avoiding detection [133]. Meanwhile, Sagha et al. use temporal difference learning for intrusion detection of a stealthy attacker in real time [117]. Xu and Xie also utilized temporal difference learning, but to analyze sequences of system calls for intrusion detection [139]. Servin resorted to Q-Learning and SARSA to detect denial of service attacks [122].

As seen, general MDPs and Q-Learning solutions have predominantly been used for adaptive defensive cyber deception or attacker cyber deception prevention. Few works have leveraged the MAB framework toward active cyber deception like mine. However, some researchers have leveraged bandits for cyber deception, such as Radoglou-Grammatikis et al. who developed TRUSTY, which utilizes the Thompson-Sampling MAB solution for honeypot intrusion detection in an industrial Internet of Things environment [116]. Many of the same authors then considered strategic honeypot deployment in dense software-defined networks using Q-Learning and ϵ-Greedy.

2.1.3 Human In-the-Loop

In recent years, numerous advances in the psychology of cybersecurity have been made. The general consensus is that it is ill-advised to ignore the human aspect of cybersecurity because humans have a hand at every level of cybersecurity. They are actively monitoring logs, developing malware, developing security software, building infrastructure, interacting with other users, etc. Both sides of cybersecurity maintain some human element that is ripe for exploitation. Rather a hacker pretends to be an IT technician for a social engineering attack or a hacker believes the honeypot they are interacting with holds truly valuable data, humans hold many cognitive biases that can be exploited.

Ferguson-Walter et al. presented data on 130 professional red had hackers interacting with decoys and other deception [55]. Overall, they found through self-reported surveys that defensive decoys can slow down and hinder human attackers to a degree, but the more information attackers have about the decoys, the less effective they are. In earlier work, Ferguson-Walter et al. ran the famous Tularosa study that demonstrated the exploitability of cognitive biases found in human attackers through the use of defensive cyber deception [56]. Johnson analyzes the impact of the sunk cost fallacy to delay attackers, where the defender throttles download speeds near the end of a malicious download to observe if the attacker will stay with the download, despite the drastically slowed down speeds [83]. In general, she found attackers exhibit exploitable cognitive biases like the sunk cost fallacy for cyber deception.

Another approach to the analysis of human decision-making is the concept of cognitive modeling. Many of the models presented in game theory and machine learning for cyber deception analyze unrealistic attackers, such as fully rational attackers. However, to fully test the effectiveness of these defense algorithms, we need to compare them with realistic attackers, which may include humans. However, it is infeasible to have humans play against these defense algorithms for the required amount of time to understand their effectiveness. This is where the notion of cognitive modeling comes into play. If we as a community can harness cognitive models that effectively make decisions as a human would, we can use these to validate the effectiveness of our defense algorithms.

For references to the field of cognitive modeling, I direct the reader to the notable works of Cleotilde Gonzalez, Noam Ben-Asher, Christian Lebiere, and Palvi Aggarwal to learn about the cognitive decision-making model: Instanced-Based Learning theory (IBL) [67, 66, 27]. Cranford et al. look into how these cognitive models perform under cyber deception to understand if they exhibit the same exploitable biases found in humans [47, 45, 46]. Taofeek et al. introduce a new cognitive model, differing from IBL, called the Cognitive Deception Model to analyze the effectiveness of humans to generate fake documents (i.e., to perform cyber deception) [129]. Aggarwal et al. analyzed the effects on human participants and cognitive models with various deception timings [9, 10]. In general, they found that later-stage deceptive actions resulted in fewer attacks from human participants and cognitive model attackers.

Chapter 3

Background

3.1 Multi-Armed Bandits

The Multi-Armed Bandit (MAB) problem epitomizes the exploration-exploitation dilemma in reinforcement learning [124, 34, 29]. When presented with optimizing multiple decisions, the decision-maker is forced to balance exploiting the best-observed options with exploring the lesser-observed decisions. The original formulation of the MAB problem comes from Thompson who wished to experimentally test two separate trials of medication [130]. Naturally, we wish to treat as many patients as possible, but you should explore both medications sufficiently. The term Multi-Armed Bandit originates from one-armed slot machines, called bandits, where the gambler would pull the arm of the bandit to play the slot machine. The goal of the decision-maker is to optimize utility (or in some formulations, minimize cost or loss). In the classical formulation, each decision is referred to as an arm on the bandit (hence the term, multi-armed bandit), however, throughout this dissertation, I will refer to bandit arms as **decoy configurations** or **defenses** as these are the decisions to optimize over for the defender in my models.

3.1.1 Definitions and Notations

In its original formulation, the gambler or decision-maker has a collection D of K total arms to sequentially choose from. Each arm is assumed to be independent of the other and has some unknown expected utility. In each round, the gambler chooses one arm k to play and observes the reward for arm k defined as $x_k \in [0, 1]$. In the simplest case, the problem may model that each arm on the bandit is i.i.d. from a probabilistic distribution like a Bernoulli or Gaussian function; however, there are other ways to model the reward structures of the arms. Due to the random or changing nature of the payoff of the arms, the gambler must simultaneously try to maximize her expected utility while learning as close to the true expected payoffs of each arm as possible.

3.1.2 Regret

Regret is the primary measure of the effectiveness of a MAB solution. Regret is strictly a measurement to perform in theory or in simulation and cannot realistically be applied in practice for the MAB problem because it requires the knowledge of the optimal decision. Regret is the utility difference between an omniscient oracle and a decision-maker in a given round. The basic form of regret for arm k in round t can be expressed as seen in equation 3.1 where $R_{k,t}$ is the regret of choosing arm k in round t and X_t^* is the maximum reward among all arms in D in round t.

$$R_{k,t} = x_t^* - x_{k,t} \tag{3.1}$$

However, regret is not always useful analytically due to the role of randomness in the MAB problem. For instance, if the decision-maker deterministically chose the arm with the highest expected payoff, but due to luck, received a low reward when polling the arm's distribution, this would lead to high regret despite having chosen the best decision statistically. For this reason, *expected regret* provides a more useful measurement for the effectiveness of a bandit algorithm with a lot of randomnesses and better captures the intention of the decision-maker. As seen in equation 3.2, expected regret of a chosen policy θ_t can be expressed as the difference between the expected reward of the optimal decision policy from the oracle, denoted by $\mathbb{E}[\theta_t^*]$, and the expected reward of policy θ_t in round t. Here a policy is a probability distribution over playing each arm on the bandit.

$$\mathbb{E}[R_{\theta_t}] = \mathbb{E}[\theta_t^*] - \mathbb{E}[\theta_t] \tag{3.2}$$

I also introduce *scaled expected regret*, which bounds the expected regret values from 0 (i.e., minimum regret or playing optimally) to 1 (i.e., maximum regret or playing the worst policy) as seen in equation 3.3. This in effect scales the proportion of how close to the optimal policy an algorithm is versus the worst policy. This is helpful when the differences between the maximum and minimum policies are less than 1 or when the utility can yield values higher than 1. In this formulation, Δ_t^* is the difference between expected values of the optimal policy $\mathbb{E}[\theta_t^*]$ from the expected value of the worst policy $\mathbb{E}[\theta_t^-]$ that yields the lowest expected value (i.e., choosing the worst performing arms in round t).

$$\mathbb{E}[R_{\theta_t}] = \frac{\mathbb{E}[\theta_t^*] - \mathbb{E}[\theta_t]}{\Delta_t^*} \tag{3.3}$$

These are not the only forms of regret, however, I only focus on these forms throughout this dissertation. Auer et al. covers the notion of *weak regret* as the worst-case regret when the worst-case scenario happens with regards to the randomness [19]. In the same text, Auer et al. also investigates strategic regret or the regret from a pool of options in relation to choosing experts instead of choosing arms in the contextual MAB problem.

Regardless of the specific form of regret modeled, its usage remains consistent: better algorithms approach zero-regret faster. An algorithm is said to approach *zero-regret* as it gains information about the arms and begins to play arms that aligns with how an oracle would play (i.e., playing arms that yield the highest expected rewards). The faster an algorithm converges to zero-regret, the better. Theoretical analysis suggests that this can be done in logarithmic time for the majority of MAB formulations and variants [36, 17].

3.1.3 Stochastic Bandits

The stochastic MAB problem is the most well-recognized and studied. It closely matches the original description of selecting slot machines, such that each arm holds a fixed random distribution where the goal of the decision-maker is to find the globally best arm to exploit for the maximum expected payoff (or minimum regret). In this problem, each arm is assumed to be i.i.d. from the same distribution (e.g., pulled from the same Gaussian distribution). The rewards are typically modeled between $[0, 1]$ but can also be $\sim B(\hat{x}_k)$ where each arm pulled yields a 0 or 1 from some Bernoulli distribution with probability $\hat{x}_k$ for arm k.

There are numerous algorithms that address the stochastic MAB problem ranging from ϵ-Greedy, Thompson Sampling, to the most famous: Upper Confidence Bounds (UCB). UCB is described later in algorithm 5 and directly addresses the exploration-exploitation dilemma with explicit terms for each. The applications for the stochastic MAB problem are far-reaching, ranging from clinical trials, web advertisements, and future branch prediction in state tree games [130, 65, 25, 20, 37, 120, 87, 44].

3.1.4 Adversarial Bandits

Stochastic bandits make a rather strict assumption about the expected rewards of each chosen arm. The Adversarial MAB variant relaxes this strict assumption by allowing for any arbitrary payoff structure of the arms, potentially even allowing an adversary to try to minimize the expected reward of the decision-maker. The most well-known adversarial bandit algorithm is the Exponentially-weighted algorithm for Exploration and Exploitation (EXP3), first described by Auer et al. and detailed in algorithm 6 [18]. Adversarial bandits are inherently more flexible and resilient to dynamic, complex environments and changing reward structures. Auer et al. demonstrated EXP3 approaches zero-regret logarithmically [19].

3.1.5 Contextual Bandits

Few real-world decisions are made in a vacuum and instead usually have some additional information that can inform the decision-maker. In the non-contextual MAB problem, the decision-maker starts with no information and can only learn from the results of her own actions. Contextual bandits assume the decision-maker has some additional context associated with each arm on the bandit. This can be in the form of experts that suggest decisions to the decision-maker or metadata about the arms (e.g., profile information in the case of web advertisement) [29, 143, 94, 97].

Later, I introduce an algorithm for the EXP3 with Expert advice (EXP4) algorithm that provides expert recommendations to EXP3. In the case of expert advice, we can collect any form of experts that recommend how each arm should be evaluated. In some expert advice algorithms, the decision-maker simply balances the expert advice with its own observations, whereas other algorithms are looking to find the best expert and simply follow the advice of the best expert, almost forming a meta-bandit problem. I use EXP4 and the contextual bandit approach as a whole to circumvent the inherent nonlinear combinatorial bandit problem that arises with deploying decoy configurations with complex vulnerabilities. Rather than having the bandit solve the nonlinear combinatorial bandit problem, I leverage the experts to consider the nonlinearity of complex vulnerabilities, which utilize *AND* and *OR* logic.

3.1.6 Linear Reward Bandits

The previously discussed MAB variants all assume a non-combinatorial problem where the decision-maker chooses a single arm each round. However, in numerous examples, multiple arms may be selected each round. In the case of web advertising, you may have multiple advertisements you can display to a single user or perhaps select a combination of advertisements for multiple users. In linear combinatorial bandits, the reward of a combination of selected arms (known as super arms), can be expressed as a linear function, summing up all the selected arms. From the original description of slot machines in a casino, we can imagine the case where each round you play a combination of slot machines then each round you sum up how much money each slot machine yielded. The problem slightly shifts from trying to find the single global optimal arm to play, but rather the optimal combination of arms to exploit. The majority of combinatorial bandits take the linear reward assumption [41, 39, 43, 68, 90, 91, 92, 96].

The notion of combinatorial bandits increases the search space exponentially as the decision-maker needs to consider the entire combinatorial decision space. However, in the case of linear combinatorial bandits, we can treat each arm as independent from the others and only need to keep track of the performance of each individual arm on the bandit as opposed to the unique combination of arms. The Learning with Linear Rewards algorithm, introduced by Gai et al., takes this approach by summing up the UCB values of each arm in a combination and selecting a combination that maximizes the total sum value [61]. I further describe LLR in algorithm 3.

When it comes to combinatorial bandits, there is also a dichotomy of observation of the reward: bandit feedback vs semi-bandit feedback. Bandit feedback only reveals the total reward of the super arm play but no indication of the role each arm played. Contrast this with semi-bandit feedback that provides full information on the individual reward of each played arm in the super arm.

3.1.7 Nonlinear Reward Bandits

The nonlinear combinatorial bandit problem considers the case where the reward feedback of the combination cannot be expressed as a linear summation over the reward of the arms. There are numerous possible non-linear functions formed from the rewards of the arms, such as the $max()$ function analyzed by Chen et al. [40]. The K-max problem looks at selecting K arms to form a super arm each round and the reward is the maximum value among the selected arms. Han et al. found through theory that there are no good regret bounds on the entire class of generalized nonlinear reward functions for combinatorial bandits [76]. However, for more specific subclasses of nonlinear functions, I believe there are areas for tight sublinear regret bounds, such as the case with the K-max bandit problem. The complex vulnerability problem I investigate throughout this dissertation falls into a subclass of nonlinear combinatorial bandits through the use of Boolean logic, namely logical

AND ($\wedge$) and logical OR ($\vee$) functions. Rather than directly applying an algorithm to this class of nonlinear combinatorial bandits, I take the approach of leveraging contextual noncombinatorial bandits where the bandit's experts address the nonlinearity.

3.2 Cyber Decoys

Cliff Stoll's seminal work introduced the world to honeypots as one of the first instances of cyber deception and decoys [126]. Since that time, the notion of cyber deceptive elements has expanded to include far-reaching aspects of cybersecurity [110, 100, 31]. Every single signal or piece of interactive information that an attacker interacts with is an opportunity for deceptive information. Cyber deception is particularly favored in that it can be layered on top of normal intrusion detection. Throwing a workstation with no valuable data into a network does not impede the firewalls or other intrusion detection systems.

3.2.1 Honeypots

The most well-known form of cyber decoys, honeypots, are typically individual computers or single services like an SSH honeypot. They typically fall into two major camps: low-interaction or high-interaction. Low-interaction honeypots are typically a façade and just collect basic information. An example of a low-interaction honeypot would be to just implement the login part of an SSH server. This low-interaction SSH honeypot does not actually provide the SSH service but instead is just logging who is attempting to interact with the SSH, at which times, and using which credentials. These low-interaction honeypots are great for their simplicity, and low resource requirements, and act as a good deterrence toward attackers. These types of honeypots are quickly "found out" by attackers as their interactions are limited.

Meanwhile, high-interaction honeypots allow for a full range of activities. In the case of an SSH honeypot, it may provide the actual SSH service but only allow interaction with fake data and files or it may mimic the entire SSH service while all the commands are simulated. High-interaction honeypots require more resources (sometimes setting aside an entire physical computer or server dedicated to the deception) but allow for the much more fine-grained data collection on the attacker's methods, capabilities, and intentions.

3.2.2 Honey Objects

Whereas honeypots typically cover a single service or computer, honey objects are much smaller scale and may include entries in a database, called honey tokens. For instance, if one inserts honey user data into a user table (i.e., a single row in a database) and someone attempts to log into that user's profile an alert can be set off and the network defenders know the database has likely been compromised. These tokens can extend to pointers in a frame as well [127, 16]. This concept can also extend to honey traffic, which is fake data being sent across an internal network to mislead any attackers who are able to sniff network traffic [14, 108]. The approach remains the same across all these different forms of defensive cyber deception: mislead and confuse the attacker about reality and monitor their interactions.

3.2.3 Honey Networks

Moving to a larger scale, honeypots and honey objects fail to capture larger-scale attacks like a data exfiltration attack which may require multiple steps, multiple exploits, and happen over a long period of time. To capture this behavior, one needs to move upwards in scale to represent an entire fake network [135, 103, 82, 53, 26]. With the rise of software-defined networking, it is now much more accessible to generate various networks with different configurations. One can now define an entire network of honeypots and change them as needed to allow attackers to interact with networks in different ways. Meanwhile, the honeypots could be real virtual machines passing along fake data and interacting in simulated ways to emulate a real, active network. If the deception is believable, it is possible that an attacker can spend longer periods of time in a network, performing multiple exploits to escalate their privileges and fulfill their true end goal. Meanwhile, the network administrators are offering no real data and monitoring all interactions.

Chapter 4

Adapting Honeypot Configurations to Detect Evolving Exploits[1]

In this chapter, I address my first research question by investigating the effectiveness of basic MAB algorithms in adapting to highly dynamic cyber adversarial environments. Specifically, the dynamicism stems from exploits that decrease in value as more are added over time. This work notably takes an initial look at complex vulnerabilities, which pose an inherent difficulty to address for combinatorial MAB solutions.

4.1 Online Learning for Deception in a Dynamic Threat Environment

One of the dominant characteristics of cybersecurity is the constant and quickly changing nature of the threat landscape, as new offensive and defensive techniques are developed and improved daily. When new vulnerabilities are discovered, cybersecurity experts try to detect them as quickly as possible and develop countermeasures. However, this takes significant time and it is not always possible to completely fix all vulnerabilities or to disseminate patches to all affected systems. A zero-day exploit is an undisclosed application vulnerability that can be exploited to negatively affect hardware, applications, data, or the network [28]. The time before zero-day attacks have been fixed, and patches have been

[1]This work was first published in the Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems [75]

applied is critical since there is often time for substantial damage. Most Intrusion Detection Systems (IDS) have detected these zero-day attacks because they rely on previously observed signatures or hard-coded rules, or imperfect methods for detecting other types of malicious behavior. Honeypots and other deception techniques provide one approach for improving the detection and tracking of this evolving space of vulnerabilities and exploits since they can provide high-quality alerts even for previously unknown types of attacks. However, optimizing the ability of honeypots to detect the changing threat landscape will also require methods to adapt the honeypot deployment strategy to track the current threat environment. We propose an approach for doing this using a method from online learning to adapt a deception strategy over time.

We model the problem of a network administrator selecting from a set of pre-configured, high-interaction honeypot profiles to detect ongoing attacks. Here, we consider an attack to be the full set of actions needed to exploit a specific system. This includes probing the network, the intrusion into the network, and the delivery of a malicious payload. Attacks utilize exploits to inflict harm to the integrity of a system, but exploits are not a static threat. New exploits are developed all the time and older exploits wane in utility as their vulnerabilities are patched. This aspect of exploits forms an evolutionary life cycle [104]. These dynamically changing exploits make the attacker's arsenal of attacks inherently dynamic. Defender solutions should be resilient and adapt to the constantly shifting capability of adversaries. We model these dynamically shifting exploits based on the vulnerability life cycles studied by Frei [60].

We model a repeated interaction wherein each round, a defender must select from a set of specific honeypot configurations that can detect some attacks by an adversary. Each honeypot configuration exposes a different attack surface to the attacker, so it can detect a different subset of possible attacks. There is potentially a vast number of honeypot configurations (protocols, server operating systems, application types, application versions, services offered, etc.). The defender's ultimate goal is to detect and learn from attacks that would harm the current network the defender is tasked with protecting. Each set of honeypot configurations (i.e., honeynet) exposes a different attack surface and provides different values in the detection capability depending on the current life cycles of the adversary's attacks and their potential to harm the given network. Exposing all attack surfaces would be infeasible, significantly harm the overall performance of the network, and would be impossible to maintain. So, the defender needs to reason strategically to make the best of use limited honeynet resources to track the current threat environment by exposing targeted attack surfaces.

Our primary contributions are two-fold. First, we present a realistic cybersecurity configuration that captures adaptive opponents learning from one another using honeypots and exploits in a constantly evolving threat environment. We then propose a solution method for the defender based on a method from the Multi-Armed Bandit (MAB) reinforcement learning framework. We demonstrate that by using standard MAB solutions, the defender is capable of adapting to the attacker regardless of the dynamically changing environment.

4.1.1 Model

We model the problem as a repeated game where in each round the defender selects a honeypot configuration from a list of K_d possibilities. Each configuration is represented by a vector of features that describe the configuration options; here we focus on binary descriptions of the state of a system (though this could be generalized). These features can describe any level of abstraction the user wishes from "port 22 is open" to "sanitizes user input."

We define D as the set of all honeypot configurations available to the defender in each round, where $D_i \in \{0,1\}^f$, such that $1 \leq i \leq f$ and f is the total number of existing honeypot configurations. A configuration contains honeypot j if $D_i^j = 1$ and does not contain honeypot j when $D_i^j = 0$ where $1 \leq i \leq K_d$ and $1 \leq j \leq f$. In each round of our repeated game, the defender selects a configuration D_i to add to the network. If the adversary chooses an attack that is compatible with the selected configuration, the defender can (possibly) detect the attack and gain a positive utility. Additionally, both the defender and adversary rely on some existing information to aid in their decision-making: the network.

The defender's ultimate goal is to protect its network from the adversary, while the attacker's goal is to deploy attacks that exploit vulnerabilities of the existing network. The defender could play a configuration that contains an HTTP server, but if the network does not have an HTTP server, this provides little value. W.L.O.G, the same can be said for the adversary's attacks. The network N is the set of all hosts on the network, such that $N_s \in \{0,1\}^f$ represents a single host on the network where $1 \leq s \leq K_n$ and K_n is the total number of hosts on the network. Because a host can be viewed as a server with one configuration, we note that $|N_s| = 1$. Also, note that we bound the set of total host configurations by f. If an attack were to target a host on the network and the set D was incapable of detecting such an attack (perhaps the attack is for the newest version of MySQL that is not represented in the set D), we view this as an issue with the developing of the configurations, which is external to the game we define here. Unlike the configurations in D, every host N_s has some value associated with it defined by a value function *HostVal(N_s).* These values will determine the utility given to the defender and adversary in each round. At this stage, we assume all aspects of the network are static, but will consider more dynamic networks in future work. Some dynamic networking techniques would include moving target defenses, that shift IP addresses in the network randomly or taking down servers temporarily for maintenance.

Lastly, we define the adversary's role in the game. The adversary starts with a set of attacks A. We define $A_u \int \{0,1\}^f$ to represent the required configuration features of a server for attack u where $1 \leq u \leq K_a$, such that K_a is the total number of attacks available to the adversary. Often attacks consist of multiple stages, meaning that the attack exploits several vulnerabilities in order to achieve some ultimate goal. In other words, an attack may require multiple servers in order to execute. For example, some complex attacks may require an Apache HTTP server v2.0 and a FileZilla FTP server v0.9.59 to fully execute. In order for the defender to successfully detect the adversary's selected attack, the chosen configuration needs to contain a honeypot with each required configuration feature of the attack. In other words, if $A_u \cdot D_i = |A_u|$, then the defender detects the attack. Similar to how we restrict hosts in the network N, note that we bound the required configuration features of attacks in A to f features, because an attack that bypasses all pre-defined configurations cannot be detected using this approach.

Not all attacks have the same purpose and level of severity. The National Institute of Standards and Technology uses the Common Vulnerability Scoring System (CVSS) to assess the severity, difficulty of implementation, and impact of exploits [105]. We assign a unique value function to each attack (which could be derived from CVSS or other data sources). We use functions because exploits inherently are not static. As an exploit becomes common, awareness and defenses rise to prevent said exploit from causing the same degree of harm it might have done during its zero-day infancy. To model this dynamic life cycle of an exploit, we restrict the value functions of each attack as monotonically non-increasing functions with respect to the number of honeypot detections from the defender. This captures the spirit of a zero-day attack, as an unknown zero-day attack will provide its most value in its infancy. As the number of detections increases for each attack, the value for future detections will decrease as potential patches for the vulnerability being exploited by the attack will arise. Every attack has a unique value function $AttackVal(A_u)$. For instance, an SQL Injection attack will provide high value while undetected for its severity but may be patched quickly in the system when revealed, leading to a low value for future detections. Meanwhile, a brute force password attack may provide little value to begin with, as little can be done to prevent it, but may only decrease slightly, maintaining consistent value in future detections.

The issue with these monotonically non-increasing value functions is that as time $t \to \infty$, the value of all attacks $AttackVal(A_u) \to 0$ and eventually become worthless to both the defender and adversary. To account for this, at the end of each round t, the adversary may gain a new attack with a random set of required features and a randomized attack value function with probability p. This creates a realistic look at dynamically evolving cyber-attacks. As the defender learns of attacks and detects them repeatedly, learning how to patch them, new attacks are being developed and added to the adversary's arsenal of attacks A, also increasing the total number of attacks $K_{a,t+1} = K_{a,t} + 1$ with probability p.

To recap, in each round t, the defender must select a configuration D_i and the adversary selects an attack A_u and both are rewarded a utility based on three factors: if D_i can detect A_u, the current value of the attack A_u, and the total value of all the hosts in the network N affected by the attack. At the end of each round, a new attack has a probability of being added to the adversary's collection of attacks. In this game, the defender must protect the hosts on the network and the various attacks employed by the attacker, particularly protecting from known, dangerous attacks, while still exploring the possibility of potentially, new, dangerous attacks. In essence, the defender must balance exploiting the configurations that defend from known dangerous attacks, while exploring configurations that may offer attack surfaces that detect new, unknown attacks. We consider these games as an instance of the Multi-Armed Bandit Problem (MAB) commonly studied in online reinforcement learning.

4.1.2 Honeypot Configuration Selection Methods

We now detail the adversary and the different defender solutions used in the experiments.

Adversarial Attacker

We model the attacker after Richard Klima's "Adversarial Attacker" which borrows techniques from fictitious play [86]. This agent determines each exploit's utility by balancing the exploit's current value in a round with past rewards of playing said utility. However, it transitions towards the exploit with this highest calculated utility smoothly with a learning rate λ. This style of agent provides a general decision-making strategy that balances potential reward with historical gain, but with a slow move rate, that mimics a collection of attackers. In cybersecurity, attacks typically happen in campaigns with the same type of attack. When a new vulnerability is discovered, many hackers and malware exploit the vulnerability with varying attacks until patches come out, leading to gradual shifts in vulnerability use instead of drastic and erratic shifts. Our Adversarial Attacker assesses each attack u's Utility U_u in time t as such:

$$U_u(t) = AttackVal(u, t-1) + c * \frac{AttackTotal(u, t-1)}{t}$$

$AttackVal(u, t)$ provides the current value of attack u, depending on its own generated monotonically decreasing value function and the number of defender detections. Attacks provide their highest value when they have no defender detections, otherwise known as zero-day status. c is a caution parameter that determines how important the history of the attack is. For our experiments, we used a caution value $c = 1.0$, such that the value of the attack and its past success is roughly worth the same. The $AttackTotal(u, t)$ provides the total rewards received from attack u.

Algorithm 1 Adversarial Attacker

```
Require: $\lambda, c$
Ensure: $P_0(u) = \frac{1}{K_a}$
1: for $t = 1, 2, 3, \cdots$ do
2:     for All Exploits $u = 1, 2, \cdots, K_a$ do
3:         Evaluate $U_u(t)$
4:         $M_{u,t} = 1 \iff u = \underset{u*}{argmax}(U_{u*}(t))$ else, $M_{u,t} = 0 \iff u \neq \underset{u*}{argmax}(U_{u*}(t))$
5:         $P_t(u) = (1 - \lambda) * P_{t-1}(u) + \lambda M_{u,t} U_u(t)$
6:     end for
7:     Play Attack $U \sim P_t$
8:     Observe Reward
9: end for
```

This attacker differs from the Adversarial Attacker described by Klima in that it is not deterministic. Instead, this agent uses a distribution that it updates according to its beliefs and randomly selects from this distribution each round to play an attack. Note that the learning rate λ affects how quickly the attacker shifts towards the best response attack. We implement this learning rate to simulate the collective knowledge and actions of hackers and malware. When hackers discover a new vulnerability, repeated exploits of the vulnerability from different hackers and malware may be launched. Similarly, if a new worm is released, it will jump from system to system and may attempt to breach a network multiple times from many of its affected hosts, even after defensive patches are released.

Naïve Defenses

The simplest defense follows the "set and forget" mentality. The defender chooses a honeynet configuration to play at the beginning of the game and purely plays this configuration for the remainder of the game. This limits the defender's possible attack surface coverage drastically and we aim to show that even simple configuration selection changes provides a stronger strategy and better coverage to detect attacks. For the remainder of the paper, we refer to this defense strategy as a Pure strategy.

Algorithm 2 Pure Defender

```
Ensure: Select random arm $i$ s.t. $1 \leq i \leq K_d$
1: for $t = 1, 2, 3, \cdots$ do
2:     Play Arm $i$
3: end for
```

Random defense strategies should improve over the pure strategy. A Uniform Random defense agent will equally distribute, but randomize its playing of each honeynet configuration, making it impossible to fully predict, but still exploit. Next is the Fixed Random agent that randomizes its distribution of random configuration selections. Similar to Uniform Random, this agent offers a random, but exploitable defense, as the attacker can determine the random distributions then minimize the expected detection.

Adaptive Defense Algorithms

MABs target a specific type of problem: the problem of balancing exploration vs exploitation. This translates naturally to the high-level problem we are tackling. Should the defender deploy honeynet configurations that detect well-known, dangerous exploits, or explore other configurations to try to detect new zero-day attacks? The two popular categories MAB solutions fall under are stochastic bandits and adversarial bandits.

In the stochastic setting, there are K_d arms to choose from, each with their own independent reward distributions on $[0, 1]$. We use the most notable stochastic solution, the Upper Confidence Bound (UCB) first described by Auer et. al. [17]. Though our defender plays against an adversary, the adversary remains somewhat controlled by the evolution of the exploits. As the defender detects exploits, the exploits lose value, eventually dying off while newly generated exploits with higher value, somewhat steering the adversary to prefer the new exploits. This may cause UCB to perform better than expected against the Adversarial Attacker.

The adversarial setting assumes that an opponent determines the expected rewards of each arm, potentially with some external randomization [32]. We utilize the Exponentially weighted algorithm for Exploration and Exploitation (EXP3) is the most famous adversarial bandit algorithm, described by Auer et. al. [18].

4.1.3 Experiments

Our initial experiment consisted of comparing all 5 defender strategies as they faced the Adversarial Attacker with a $\lambda = 0.25$ learning rate. We used a defender oracle that could see the attacker's exploit distribution before selecting a honeynet scenario in order to make the optimal decision. We subtract the oracle's expected reward each round from the expected reward of the defender agent in question to obtain the defender's regret. This was one metric of performance we used to evaluate the 5 solutions.

We use 4 general monotonically non-increasing value function types for the various

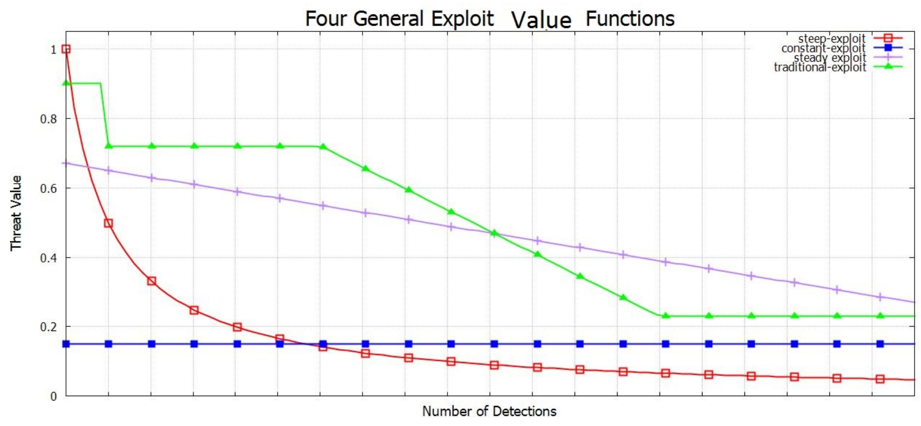


Figure 4.1: Example value functions for the 4 general exploit types: Steep, Traditional, Constant, and Steady.

attacks as seen in Figure 4.1.3. We start with a steep exploit, that starts with initially high value, but within the first few defender detections, drops drastically. This represents the simple to fix, but dangerous nonetheless exploits, such as an SQL Injection. We also utilize a constant low-value exploit classification, that includes long duration brute-force password attacks, which little can be done about them but are still worth detection. Our traditional exploit function type after the vulnerability lifecycle studied by Frei [60].

These types of exploits decrease in segments, depending on what stage of its lifecycle it is in. Initially, when an exploit is an undetected, zero-day exploit, it provides the most value. Upon detection, when network defenders become aware of its existence, but are unable to prevent it, it begins decreasing in attack value because with each additional detection, the defender grows closer to patching the vulnerability. Eventually, the defender patches the vulnerability and the exploit provides its lowest value where it turns into a constant exploit thereafter. Lastly, we show the steady exploit, which is a linearly decreasing value, that is a smoother version of the traditional exploit. Once low enough in value, the steady exploit, shifts towards a constant exploit, much like the traditional.

Rewards are calculated based on a combination of factors. We assign a random value to each network server in the game initialization. These values are static and normalized such that the total value of the servers on the network adds to 1.0. If the defender successfully detects the adversary's attack, the attacker receives no reward, such that $R_{A_u,t}^{detected} = 0$ in round t. However, if the defender's selected honeynet scenario D_i fails to detect the adversary's attack A_u, the attacker receives a reward $R_{A_u,t}$ such that,

$$R_{A_u,t}^{\neg detected} = \sum_{s \in N} AttackVal(u,t)(A_u \cdot N_s)HostVal(s)$$

The defender's rewards are opposite of the attacker, such that, $R_{D_i,t}^{detected} = R_{A_u,t}^{\neg detected}$ and $R_{D_i,t}^{\neg detected} = R_{A_u,t}^{detected}$. As mentioned earlier, we use the concept of an oracle to evaluate each defender. In each round, the oracle observes the adversary's strategy and plays the arm that maximizes its expected payoff. We compare this expected payoff to the defender's expected payoff. We define the oracle's expected utility as,

$$E[\theta] = \max_{\theta} \sum_{\theta \in D} \sum_{u \in A} X_t(\theta) * Y_t(u) * R_{D_\theta,t}^{detected}$$

and regret as,

$$Regret_t = E[\theta\ ] - \sum_{i \in D} \sum_{u \in A} X_t(i) * Y_t(u) * R_{D_i,t}^{detected}$$

where $X_t(i)$ is the defender's probability for playing honeynet scenario i in round t. In the deterministic algorithms Pure and UCB, only a single honeynet scenario will be given $P_t(i) = 1.0$. $Y_t(u)$ is the probability that the adversary selects attack u. The closer the defender is to fully playing the optimal honeynet scenario that will best detect the adversary's attacks, the closer the defender moves to 0 regret. We can now measure performance through the measure of regret, with lower total regret indicating a better agent.

In each match, the defender and attacker play for 20,000 rounds. We set the total number of possible configurations to $f = 50$. We set the number of honeynet scenarios to $K_d = 50$ where each scenario contains between 8 and 12 listeners. We feel this is well grounded, as a network defender hand-crafting 50 honeynet scenarios would provide a wide variety of attack surfaces. We have the attacker start with only $K_a = 10$ exploits, but each round the attacker has a 10% to gain a new exploit with its own, fresh value function, gaining zero-day status.

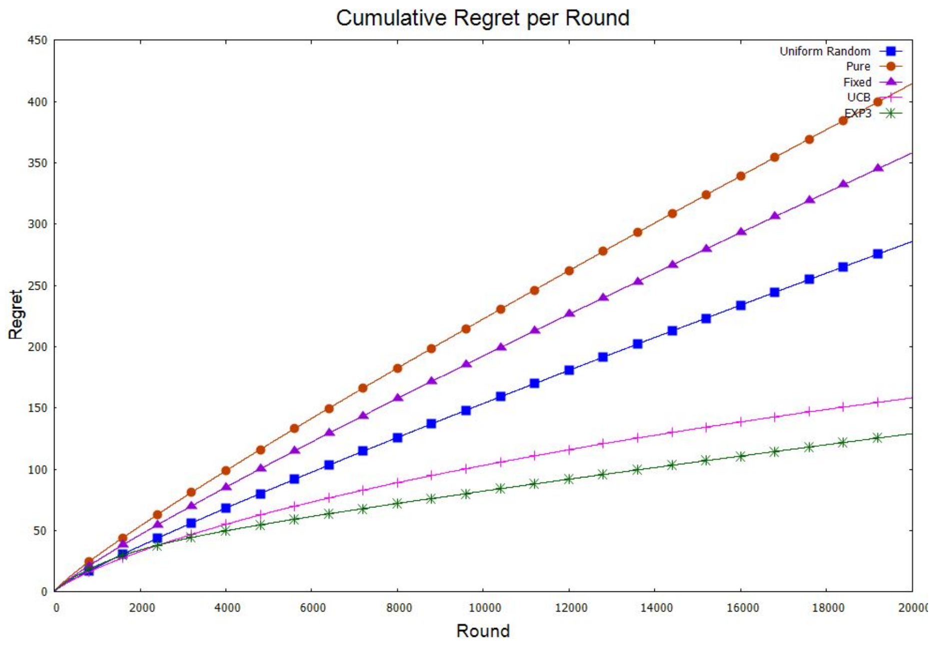


Figure 4.2: Cumulative Regret Over Time

As seen in Figure 4.1.3, the pure strategy performs the worst, followed by the Fixed Random, then Uniform Random strategies, with none of the defenders converging. This further suggests that the "set and forget" Pure strategy offers minimal defenses and even simple changes between honeynet scenarios can improve overall network security. UCB and EXP3 perform the best and seem to start converging by the end of the game, despite new exploits being created often. This implies that both strategies struck the right balance between exploration and exploitation, despite the numerous variables and uncertainty, providing a strong case for MABs as a solution to this problem.

One argument against using UCB, however, is the fact that is deterministic. Our Adversarial Attacker does not perform any pattern recognition or make any assumptions about the defender, but if an attacker had knowledge that the defender is employing UCB, they could exploit the defender and avoid detection 100% of the time. On the other

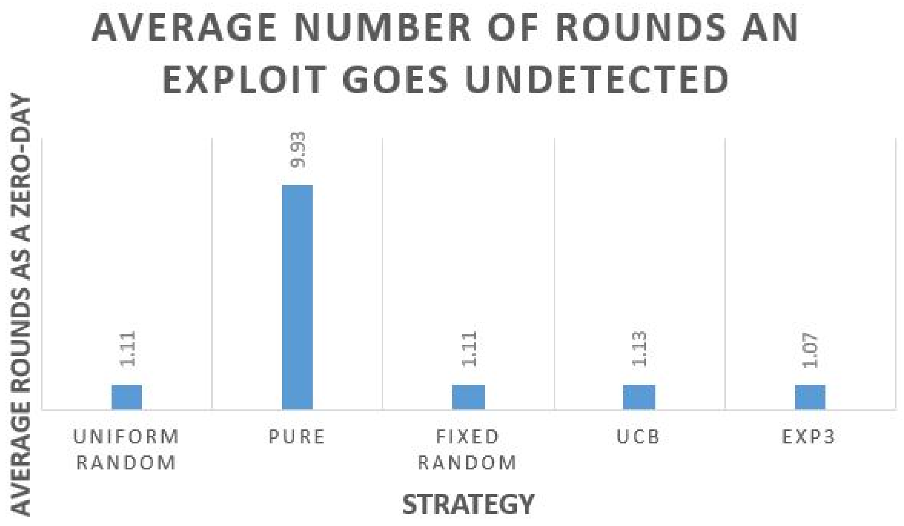


Figure 4.3: Average number of rounds new exploits go undetected (zero-day).

hand, EXP3 assumes that the opponent is actively trying to minimize the agent's expected rewards. To combat this, EXP3 uses computed distributions to randomly select an arm in each round.

In our second experiment, we keep the same parameters but increase the number of honeynet scenarios to $K_d = 100$ and the total number of honeypot configurations to $f = 100$. This increases the complexity and amount of decisions for the defender. As seen in Figure 4.1.3, the naïve strategies perform similarly, continuously being exploited by the Adversarial Attacker. UCB and EXP3 also perform only slightly worse than their experiment 1 counterparts. In the first 4000 rounds, EXP3 appears to perform worse than UCB, unlike in the first experiment 1. This is most likely caused by the increased number of scenarios and therefore, an increased amount of exploration.

Another performance metric we consider is the average number of rounds a zero-day goes undetected. This essentially measures exploration. As seen in Figure 4.1.3, nearly all defense algorithms average close to 1 round before detecting a never-before-seen zero-day attack. Pure strategy, on the other hand, averages close to 10 rounds before detecting zero-days. Though, this is just the way averaging works. In reality, if the chosen zero-day attack

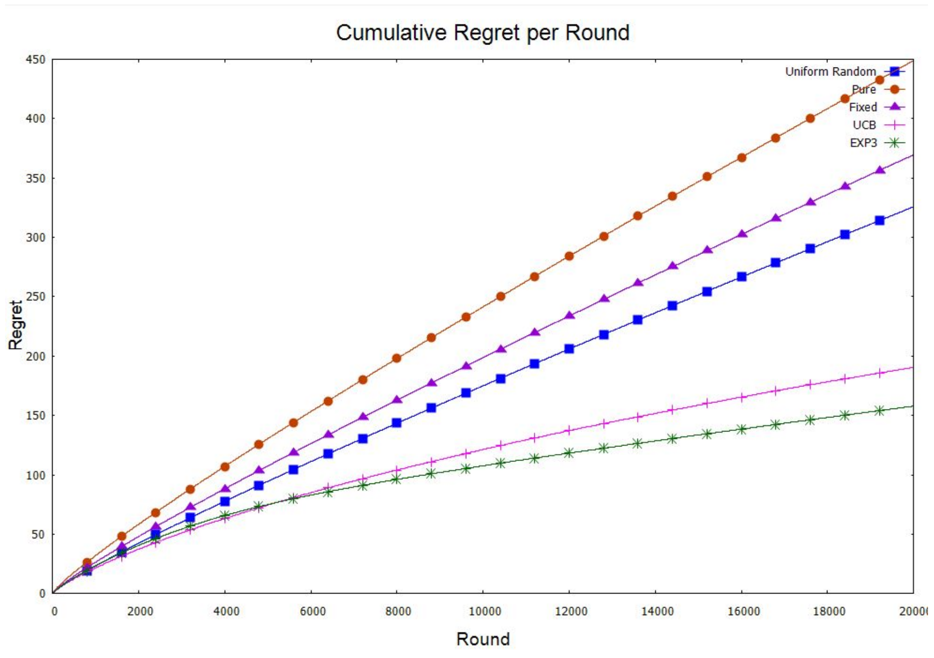


Figure 4.4: Second experiment, increasing the honeynet scenarios to $K_d = 100$ and honeypot configurations to $f = 100$.

bypasses Pure strategy's single selected scenario, it will always pass this scenario, maintaining its zero-day status. However, if the zero-day attack falls to the selected scenario, then it will be captured immediately, going 0 rounds undetected. Though trivial, Figure 4.1.3 exemplifies why changing honeynet scenarios to expose different attack surfaces is necessary for proper network security.

Measuring the average number of rounds an exploit goes undetected is also measuring exploration. This can be seen in Figure 4.1.3 where UCB has a higher averaged undetected rounds than Uniform Random and Fixed Random. This is because, as a deterministic algorithm developed for fixed random distributions, it exploits rather quickly after exploration. Compare this to EXP3 with the lowest average undetected rounds, because EXP3, as an adversarial setting algorithm, assumes the opponent is constantly learning from EXP3 and remains more pessimistic about the "optimal arm" than UCB does.

In the second experiment, we investigate how the attacker's learning rate impacts the

MAB defenders. In the first experiment, we found that UCB and EXP3 performed exceptionally well against the Adversarial Attacker with a $\lambda = 0.25$ learn rate. Intuitively, one might argue that the slow learning rate allowed for smooth transitions between selecting new exploits. In the second experiment, we run UCB and EXP3 10 times each against an Adversarial Attacker with learning rates $\lambda = 0.0$, $\lambda = 0.20$, $\lambda = 0.40$, $\lambda = 0.60$, $\lambda = 0.80$, and $\lambda = 1.0$ to determine how the attacker's design affects the defender's performance.

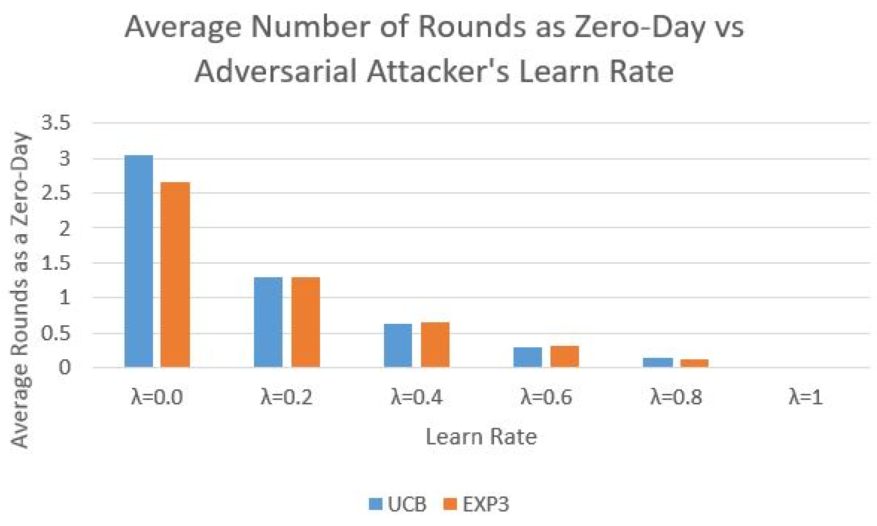


Figure 4.5: How the Adversarial Attacker's learning rate affects zero-day attacks.

As we see in Figure 4.1.3, the Adversarial Attacker's learn rate plays a crucial role in the performance of defender agents. Because the average number of rounds an attack goes undetected is a measurement of exploration, we can interpret why the learning rate negatively impacts the average rounds as a zero-day exploit. Because the Adversarial Attacker agent determines how quickly it moves towards the best decision, the bandits quickly learn what that best decision is. With a $\lambda = 0.0$ learn rate, the attacker never switches exploits and the bandits have trouble detecting this static attack since they explore frequently early on.

4.1.4 Discussion

This work illustrates a key point: Without fully understanding the complex relationships between features and vulnerabilities and how the environment may change online reinforcement learning demonstrates that defenses can still adapt. Furthermore, this work focused on the important AND-only complex vulnerabilities, which require multiple features to be present: a notably, nonlinear relationship w.r.t. the rewards of features.

This work also answers my first research question:

Q1. How capable are standard Multi-Armed Bandit (MAB) solutions at adapting to attackers that exploit complex vulnerabilities that evolve over time?

Despite the rapidly changing environment, non-combinatorial MAB solutions adapt slowly to the Evolving Exploits problem but still approach zero-regret over time. Because of the AND-only relationship among features in the complex vulnerabilities, which forms a non-linear reward function when considering a combinatorial bandit setting, standard linear-combinatorial bandits were incapable of addressing this issue directly. Instead, I utilized the non-combinatorial bandits, EXP3 and UCB, which inherently assume that there are no dependencies among the different honeypot configurations. Despite the fact that the attacker was adapting and had a rapidly changing arsenal of exploits, the basic non-combinatorial bandits were still capable of adapting and approached zero-regret. I also demonstrated a key point with slight modifications to EXP3 and UCB, there were notable improvements in their learning capabilities in this inherent nonlinear combinatorial bandit problem. This suggests that the nonlinear class of the "AND" function of rewards has room for improvement. Bandit algorithms that attempt to address this issue that arises from deploying decoy configurations amidst complex vulnerabilities, should also consider the nonlinear combinatorial bandit reward structure, which essentially forms a bipartite graph. In chapter 6, I introduce an algorithm that better directly exploits the bipartite structure formed from these nonlinear reward functions.

In this chapter, I only considered a rather slowly learning attacker that may not always represent the attack patterns seen in these kinds of cyber adversarial settings. Attackers may be individuals that are quick to adapt, like human hackers, rather than the model of a slowly moving attack community as a whole. In the next chapter, I investigate the effects of quickly adapting attackers and how this impacts simple defense strategies.

Chapter 5

Online Learning Algorithms for Deception Against Humans[1]

In the previous chapter, we looked at how capable basic learning strategies are at adapting to highly dynamic environments in a cyber deception interaction. Though we found that basic learning strategies are capable of adapting to simple, slowly learning attackers, not all attackers will adapt as slowly. Another important aspect of the success of cyber deception strategies is understanding whether defenses can be learned and exploited by an intelligent adversary. Of particular interest are human adversaries, who may be able to easily detect and exploit patterns in how deceptive objects are used, and avoid them to attack only real systems. Game theoretic models consider this problem by using randomized strategies to make it difficult for adversaries to exploit specific patterns in defensive strategies. However, adaptive strategies that learn to respond to opponents may also be hard to predict and have many of the same advantages of the game theoretic randomized strategies. Here, we consider both game theoretic and adaptive strategies against human opponents in a basic cyber deception game using honeypots.

With the popularity of autonomous systems, the question of how humans interact with these systems becomes increasingly important [64]. Humans are imperfect agents, but they are capable of learning and in some settings able to adapt to novel situations. Our ability to anticipate human behavior, to represent human decision-making computationally, and to use these predictions to improve autonomous agents is critical to making autonomous

[1]This work was first presented at the 41st Annual Meeting of the Cognitive Science Society (CogSci 2019), Montreal, QC [71]

systems more capable and secure.

We model a cybersecurity scenario as a repeated Multi-Armed Bandit task (MAB) where a human attacker plays against an automated defender. MAB tasks have been useful in the study of human decision-making, characterizing the common exploration-exploitation tradeoff (e.g., [125]). However, our goal is to determine how a human attacker is able to learn the defender's deception strategy and avoid honeypots based on previous experience. Here we take a simpler model from the previous chapter and consider the attacker's decision making. We show that online learning defenses provide similar (and in some cases, better) challenges as randomized defenses for human adversaries to learn.

In a standard MAB, a decision maker selects arms on a "slot machine" in each round and observes the outcome, typically with the value of each arm in the range $[0, 1]$. The adversarial MAB considers an adversary (i.e., the algorithmic defender) who has control over the rewards of each node. Here, we consider a variation of the MAB in which each node i has bounded support interval $\{-c_i^a, v_i - c_i^a\}$. This allows the MAB agent to make more informed decisions in earlier rounds. This maps naturally to an attacker who has probed the network prior to making an attack, and it relates to recent approaches to study learning and decision-making under contextual MAB, where information about rewards is provided.

5.0.1 Model

Cognitive modeling provides a unique opportunity to quickly and effectively test various strategies against human-like agents as it is infeasible to collect a sufficient amount of human trials every time one wishes to evaluate a new strategy in a cyber adversarial environment. However, we need to first understand how humans make decisions in these adversarial cyber adversarial environments.

For this study, we designed a model that focuses on the learning aspects of an adversarial cybersecurity interaction, motivated by honeypot deception. In this scenario, an attacker and defender compete over multiple resources (nodes) in the network belonging to the defender with the following parameters: v_i is the value of node i, c_i^a is the cost to attack node i, and c_i^d is the cost to defend node i. At the beginning of the interaction, each node is initialized with the non-negative parameters. At the beginning of each round, the defender spends some budget D to turn some subset of the nodes into honeypots, such that the total cost of defended nodes is $\leq D$.

Once the defender deploys honeypots, the attacker selects a node to attack or pass. If the attacker's chosen node i is undefended, the attacker receives the reward $v_i - c_i^a$, and the defender receives a reward of 0. On the other hand, if the attacker's chosen node i was a honeypot, the attacker receives the negative reward $-c_i^a$, and the defender receives the positive reward v_i[2]. At the end of the round, the interaction resets, and the process repeats each round.

The only feedback the attacker receives is the reward for her action. Therefore, the attacker can only partially and indirectly observe the defender's behavior. The defender observes the individual honeypot placements. So, if the defender captures the attacker with a honeypot, the defender knows which honeypot node was responsible for the capture. If the defender does not capture the attacker and there are more than 1 undefended nodes, the defender can never be certain about which node the attacker chose. This style of feedback is known as semi-bandit feedback in the MAB literature. Given our focus on the study of high-level decision-making, only general cognitive skills are needed from those humans playing the role of the attacker. Cybersecurity knowledge does not play a role in making decisions regarding "honeypot" configuration or realism [27].

We designed a repeated adversarial interaction with 6 arms (which we will refer to as nodes) to be played over 50 rounds as seen in Figure 5.1. We recruited 304 human participants on Amazon's Mechanical Turk [33]. Of the 304 participants, 130 reported female and 172 reported male with 2 participants reporting as other.

All participants were above the age of 18 and had a median age of 32. The experiment averaged roughly 10 minutes from start to finish and the participants were paid US $1.00

[2]We assume $v_i > c_i^a$ and $\sum_{i \in N} c_i^d > D$.

for completing the experiment. The participants were given a bonus payment proportional to their performance in the 50 round game, ranging from US $0 to an extra US $3.25. This bonus payment was intended to incentivize participants to play as best they could.

In a realistic cybersecurity environment, the domain knowledge of the attacker plays an important role as to which vulnerabilities to exploit and how to gain access to a system. When recruiting the participants in our study, we held no requirements or assumptions about the cybersecurity knowledge of the participants. To address this, we take the pessimistic assumption that if the participant (as the attacker) tries to attack a non-honeypot node, they perform a guaranteed successful attack. All that remains for the attacker is deliberate which node to target for an attack, which boils down to basic human cognition. Real-world expert hackers will share the same level of cognition with our participants, allowing the recruited participants to accurately represent real cyber attackers at the described level of abstraction [27].

5.0.2 Scenario

The defender has a budget $D = 40$ that limits the number of honeypot configurations (i.e., combinations of defended nodes). In each round, the participant attacks a node and receives either a positive reward $v_i - c_i^a$ or a negative reward $-c_i^a$ depending on the defender's action.

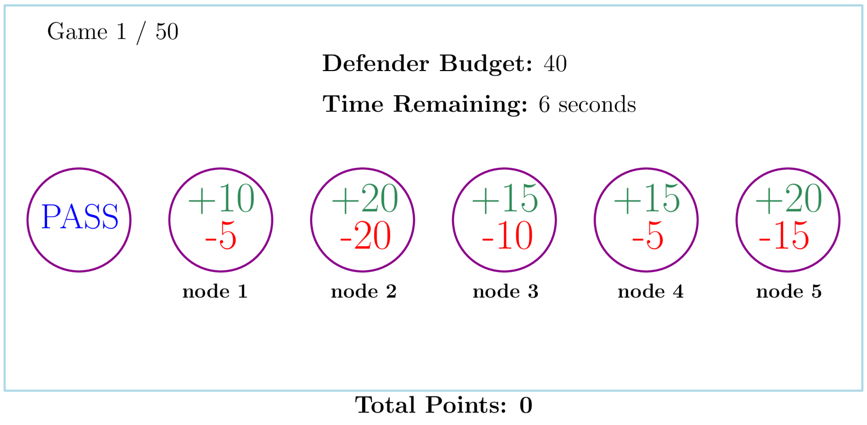


Figure 5.1: User interface for the Honeypot Intrusion Game.

The setup in Figure 5.1 was the same for every participant. For ease of the participant, we simplified the visible rewards on the nodes, where the reward $v_i - c_i^a$ for attacking a non-honeypot appears as the positive top number in the node. Meanwhile, the loss for attacking a honeypot $-c_i^a$ appears as the lower, negative number inside the node. Table 5.0.2 shows the actual parameters for each node.

We designed the nodes to fit a variety of risk-reward archetypes (e.g., low-risk/low-reward, high-risk/high-reward, low-risk/high-reward). The intuition is to allow for differences in strategies and learning. These differences in known parameters provide a noticeable difference from the traditional MAB. For instance, in the first round, the attacker is making an informed decision based on the attack costs and rewards.

Table 5.1: Node parameters for online human experiment.

	pass	node 1	node 2	node 3	node 4	node 5
v_i	0	15	40	25	20	35
c_i^a	0	5	20	10	5	15
c_i^d	0	10	20	15	15	20

5.0.3 Defenders

We deployed 3 different defenders to analyze the impact an online learning defense has on human learning as opposed to randomized and static defenses. We use the *Static Pure Defender*, the *Static Equilibrium Defender*, and the *Adaptive LLR Defender* that learns from its own action observations. Each defender creates a different level of learning complexity for the human participants. Here we are not investigating the best defense strategy versus humans. Instead, we are interested in analyzing the impact that varying levels of defense complexity have on human learning and decision-making as a whole.

Static Pure Defender: This defender employs a "set and forget," purely static defense that attempts to maximize its total value by assuming it must commit to a single, pure, non-stochastic strategy for a single round. This defender tries to spend its budget to protect the highest-valued nodes. For the scenario seen in Figure 5.1, the defender always defends nodes 2 and 5, leading to nodes 3 and 4 as optimal nodes for attacking. This defender acts as a baseline for human learning. With this defender, we investigate the upper bound on how quickly humans can learn a defense.

Static Equilibrium Defender: This defender plays over a fixed distribution of defenses (combinations of nodes to be honeypots). The defense is a Mixed Strategy Nash Equilibrium. It optimizes the defender's expected utility assuming only a single, non-repeated interaction against a fully rational attacker. An optimal strategy of the attacker in this equilibrium is to attack node 4.

Adaptive Learning with Linear Rewards Defender: The last defender, Learning with Linear Rewards (LLR) [61], provides a deterministic, yet adaptive defense as it tries to maximize its reward by balancing exploration and exploitation.

Algorithm 3 describes the LLR algorithm where $\mathcal{A}_a$ defines the set of all individual basic actions (nodes to defend). In the scenario from Figure 5.1, $\mathcal{A}_a$ is the set containing all 5 nodes. LLR uses a learning constant L, which we set to $L = 3$ for our scenario since this is the maximum number of nodes we can play in defense. LLR has an initialization phase for the first $N = 5$ rounds where it guarantees playing each node at least once. $\left(\hat{\theta}_i\right)_{1\times N}$ is that vector containing the mean observed reward $\hat{\theta}_i$ for all nodes i. $\left(m_i\right)_{1\times N}$ is the vector containing m_i, or number of times arm i has been played. After each round, these vectors are updated accordingly.

After the initialization phase, LLR solves the maximization problem in equation 5.1 and deterministically selects the subset of nodes that maximizes the equation each round until the end of the game. This deterministic nature of LLR indirectly adapts to the attacker's moves. It has no concept of an opponent, but instead is trying to balance between nodes with high observed means (i.e., have captured the attacker often in the past) and less

Algorithm 3 Learning with Linear Rewards (LLR)

```
1: //INITIALIZATION
2: If $\underset{a}{max}|\mathcal{A}_a|$ is known, let $L = \underset{a}{max}|\mathcal{A}_a|$; else, $L = N$
3: for $t = 1$ to $N$ do
4:     Play any action $a$ such that $t \in \mathcal{A}_a$
5:     Update $\big(\hat{\theta}_i\big)_{1\times N}$, $\big(m_i\big)_{1\times N}$ accordingly
6: end for
7: //MAIN LOOP
8: for $t = N + 1$ to $\infty$ do
9:     Play an action $a$ which solves the maximization problem
           $$a = \underset{a\in\mathcal{F}}{\operatorname{argmax}} \sum_{i\in\mathcal{A}_a} a_i\Big(\hat{\theta}_i + \sqrt{\frac{\big(L+1\big)\ln n}{m_i}}\Big), \tag{5.1}$$
10:    Update $\big(\hat{\theta}_i\big)_{1\times N}$, $\big(m_i\big)_{1\times N}$ accordingly
11: end for
```

frequently played nodes (which the attacker may move to in order to avoid capture). We say that LLR indirectly adapts to the attacker's actions.

In this scenario, the attacker can never fully exploit the deterministic strategy of the defender because of the partial observability aspect of the interaction. This defense leads to the optimal node(s) changing in each round as adaptive LLR learns.

5.0.4 Behavioral Results

Analysis of human data shows clear performance patterns among the 3 defenders. The pure defender predictably performed the worst, yielding an average score of 611.93 points to the human attackers, just over 100 points short of the maximum possible points achievable against the pure defender. Next, the equilibrium defender performed significantly better, yielding only an average of 247.81 points to the human attackers, a full 200 points short of the maximum expected points achievable by a human attacker. Finally, the LLR was the most resilient defender versus the human attackers with an average of 172.6 points yielded to the participants. Table 5.2 shows the aggregate data of the human attacker's performance.

Table 5.2: Aggregate data of participants' end-game attacker rewards.

	average	std. dev.	median	min	max
Pure	611.93	168.88	675	-375	750
Equ.	247.81	149.60	290	-185	570
LLR	172.6	123.02	160	-85	640

Figure 5.2 shows the cumulative utility frequencies among the participants. Visually, we can see the pure defender yielded many points to many participants with only a couple of outliers. The participant who received -375 points vs the static pure defender likely tried to speed through the game without trying to optimize the total reward.

LLR is designed to solve a combinatorial MAB problem that tries to optimize over a static, stochastic environment. Here, LLR does not take into account the attacker's

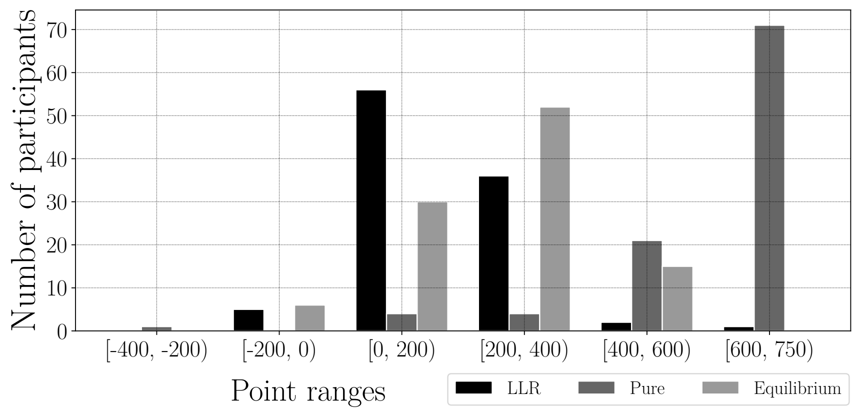


Figure 5.2: Frequencies of total cumulative utility ranges.

adaptive nature. We do not make any claims that LLR is a perfect way to respond to human decision-making. However, we note that LLR clearly outperforms the randomized and static defenders.

When analyzing the proportion of the human population that played optimally per round, we can see the differences in learning curves among the various defenders. The rightmost graph in Figure 5.3 shows the frequency of optimal decisions over the course of the 50 rounds.

Figure 5.3 details other measures that help describe the differences in human attacker performances. We can see that each defender case leads to drastic differences in switching (i.e., attacking different knows in consecutive rounds) and conditional switching (i.e., switching after triggering a honeypot or not). High switching suggests a more exploratory state, while low switching indicates a more exploitative state. We can see the noticeably higher switching in the LLR case as opposed to the other defenses. Combine this with the results seen in proportion of optimal play and it is clear that the human participants are struggling to adapt to the adaptive LLR defense.

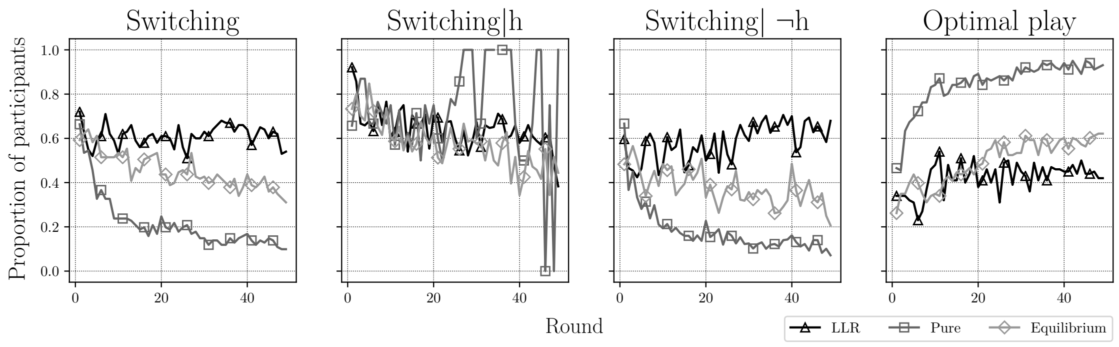


Figure 5.3: The proportions of participants switching nodes to attack or playing optimally over time. The high switching after triggering a honeypot seen in round 26 from participants facing the static pure defender is a small portion of the population as less than 10% of the population were triggering honeypots past round 25.

5.0.5 Discussion

We studied how humans learn in a novel version of an adversarial, contextual Multi-Armed Bandit scenario motivated by a real-world cybersecurity scenario where defenders use deceptive decoys and attackers must learn to avoid them. We evaluated three different types of defensive strategies and showed that an adaptive defensive strategy was clearly the strongest against human players, and the hardest for them to learn. We also made novel comparisons between predictive models to emulate how humans learn in this type of adversarial setting, comparing leading models from both the MAB literature and cognitive science. We find that the best models (Thompson Sampling and IBL) are able to predict human behavior quite effectively, but that human attackers use different strategies depending on the adversary they are up against, and the best predictor may depend on this context. There are many interesting opportunities to improve both types of models, especially in making personalized predictions for individuals and specialized contexts. However, the results so far have immediate practical implications for how we can design better strategies for deploying decoy systems to enhance cybersecurity. In particular, these systems must be adaptive to prevent attackers from easily learning the defensive strategy. The predictive models of an attacker learning we have developed will also allow us to develop defenses that actively mitigate the ability of attackers to learn the defensive strategy.

These results also demonstrate that it is not always enough to leverage algorithms with good theoretical bounds as cyber threats are not always stochastic or fully rational machines. Instead, the defender may come across suboptimal human hackers that are excellent at pattern recognition. Oftentimes, it is impossible to fully avoid the human-in-the-loop for cybersecurity, and any online learning in adversarial settings should consider human opponents as well.

This work also answers my second research question:

Q2. How well do human attackers react to standard reinforcement learning algorithms in repeated adversarial interactions?

Ultimately, this work found that human attackers quickly adapt to static defenses, even fixed randomized policies. Naturally, simple adaptive defense algorithms like LLR provide more resistance than static defenses, suggesting the need for adaptive defenses in adversarial cybersecurity settings. A "set and forget" fixed random policy can still be quickly optimized by human attackers. We can leverage cognitive models like IBL to simulate human behavior in these adversarial learning environments. For the decoy configuration for complex vulnerabilities problem, this suggests human attackers will demand adaptive defenses as they try to avoid decoy detection through their arsenal of complex exploits.

Chapter 6

Intrusion Detection with Complex Vulnerabilities

6.1 Motivating Scenario

The specific scenario I will emphasize for this chapter is using decoy devices to detect complex exploits deployed in an airport Wi-Fi. There are numerous dangers associated with accessing public Wi-Fi networks, but perhaps a person needs to send a critical email or log into a website to address a time-sensitive matter, so they take the risk [123, 42, 101]. Due to the nature of public Wi-Fi networks, a wide variety of devices with different vulnerabilities are constantly entering and exiting the network, changing the effective targets to attack. An attacker can enter an airport, log onto the Wi-Fi and deploy a network worm that targets all vulnerable devices and critical infrastructure, then leave the airport followed by a different attacker doing the same thing at a later time.

A network administrator can set up a decoy, like a honeypot, in the network to monitor all malicious activity [99]. If the network administrator simply sets up a honeypot or even a collection of honeypots and never changes them, the honeypots may miss out on critical exploits as it is impossible to represent all possible attack surfaces and whichever honeypots are present will yield a static attack surface for the attackers to exploit. A key assumption that we make is limited feedback. So, a network administrator can only see the result of the deployed decoy, not the results of the others that could have been deployed. For instance, if this network administrator sets up an Android phone honeypot, Windows 10 honeypot, and iPhone honeypot, they will still miss out on exploits targeting

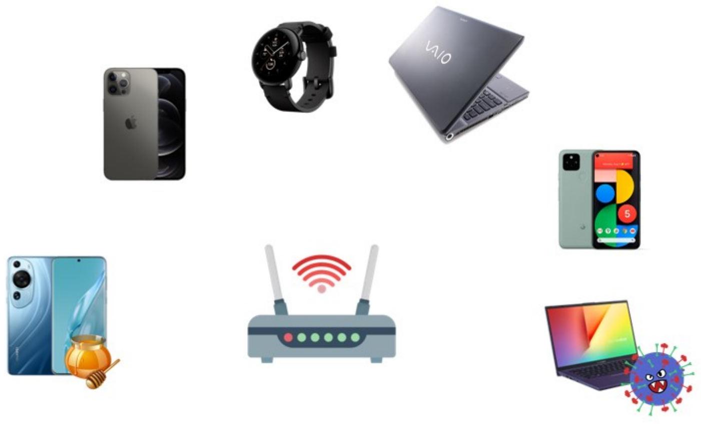


Figure 6.1: Many devices in an airport public Wi-Fi with a malicious attacker logged in, planning to exploit the devices on the network with a complex vulnerability (on the bottom right) and the decoy device placed by the defender trying to capture the attacker's exploit (bottom left)

only smart watches or Windows 11 systems. One strategy is to just add more honeypots to the network to increase the size of the attack surface, but if the exploits only target specific versions of specific web-facing applications, then you would need to add a honeypot for each version. Very quickly, the amount honeypots can grow out of hand, which could also end up constricting network performance. Instead, we can have a variety of pre-built decoy configurations that we switch between. This configuration switching can be handled with virtualization images or snapshots of builds and can easily and quickly be switched when needed [6, 59, 58]. I use the term decoy configuration instead of honeypot configuration because we can combine the usage of multiple honeypots in each configuration. Perhaps one configuration is actually the deployment of multiple mobile devices and laptops, exposing a unique attack surface among them. This is why I do not restrict the modeling to just honeypots and will instead use the term "decoy."

If we wish to detect the most malicious activity possible, the question then becomes, which configurations should we use at a given time? Another component of this problem, which I leave out of the scope of this work, is how often to switch between configurations.

For now, I only consider discrete decision-making in terms of “rounds” or “trials”, but when implementing this model in a real-world setting, one will need to justify the time duration of a round. Instead, With these discrete intervals the network administrator must repeatedly balance exploring lesser-known configurations with exploiting (in the decision-making sense) the best-performing configurations. Ultimately, the network administrator wishes to capture the most amount of attacks and the most valuable attacks possible, which ultimately provides fruitful information on how to patch these exploited vulnerabilities and on attacker preferences and trends. However, randomly selecting a defense configuration each round may improve upon a purely static defense as seen in chapters 6 and 4, strategic configuration selection can vastly improve over random selection to capture more attacks and adapt to attacker patterns. This problem naturally maps to a Multi-Armed Bandit problem where each round the the defender chooses which decoy configuration to deploy.

6.2 The Decoy Configuration Selection Problem

I introduce the novel Decoy Configuration Selection with Complex Vulnerabilities (DCS-CV) Problem. Each round, a defender selects one of many preset decoy configurations to place into the network. Each decoy configuration has a unique combination of binary feature variables that define which features are present in the decoy. The attacker has a collection of exploits that are represented by simple and complex vulnerabilities that are expressed as Boolean expressions containing logical AND ($\land$) and $\lor$ connectors. The exploits with simple vulnerabilities only require a single feature to be present. Each round, the attacker deploys one of their exploits into the network, attempting to exploit every device in the network, including the decoy.

If the decoy configuration’s representative binary features satisfy the logical Boolean expression of the attacker’s exploit, we say that the decoy detects or captures the exploit, because the decoy logs the full interaction and relays the full extent of the exploit to the defender via log files. When the defender successfully detects an exploit, the defender receives a positive reward and the attacker receives zero reward as we assume the honeypot alerts the network administrator to the intrusion. Meanwhile, if the defender's decoy configuration fails to satisfy the exploit's expression, the attacker does not engage with the decoy (and instead attacks the other devices on the network) and receives a positive reward while the defender receives no reward as the defender was completely unaware of the stealthy attack taking place to begin with. Both the attacker and defender have limited feedback in this model and if the attacker fails to exploit the decoy, they do not know which decoy configuration was chosen, only that they received no reward. Similarly, if the defender fails to detect the decoy, she only knows that her deployed configuration did not receive a reward but does have specific information as to which exploit was deployed. When a defender does capture an exploit, she learns about the exploit, and the Boolean expression is revealed. The knowledge of this exploit's feature requirement can inform the defender of which other decoy configurations were also capable to capture the exploit.

To model learning and the task of discovering new zero-day exploits, the defender starts the game with no knowledge of the attacker's exploits and must capture exploits to learn about them. I permit an expanding list of the attacker's arsenal of exploits allowing for the possibility of new ones being added as in my work on evolving exploits described in chapter 4.

The uniqueness of this problem formulation is the introduction of complex vulnerabilities and attackers deploying exploits that utilize these complex vulnerabilities. As mentioned in the introduction, I define a complex vulnerability as a vulnerability whose required features (i.e., the software, hardware, or firmware associated with it) can be expressed through a Boolean expression containing multiple features and using logical $\wedge$ or $\vee$ connectors.

I do not consider a logical "negation" connector ($\neg$) in this definition at this stage as combinatorial bandits do not consider this case (one cannot receive a positive reward from a slot machine they did not play). Furthermore, this is realistic as vulnerabilities such as CVEs described in the NVD do not model the negation of a feature (or CPE in the NVD).

All the vulnerabilities in the NVD simply state the affected platforms. If we do need to model the negation of a feature, we can view the feature as the negation. For example, if we wish to model that a feature f_k is off or not present, which models address space layout randomization is present in the system, we can just model f_k as “address space layout randomization is turned off” for a buffer overflow exploit.

6.3 New Bipartite Multi-Armed Bandit Variant

To address the Evolving Exploits problem and novel DCS-CV problem, we can simplify both problems and just consider the connections between defenses and exploits, which are formed if the defense satisfies the exploit's Boolean expression with features. A small example of this can be seen as a bipartite graph in Figure 6.3 where edges represent that the decoy configuration satisfies the connecting exploit's expression.

In each round, some nodes activate from the Exploit Set E, which activates their adjacent edges. In each round, the decision-maker must select a single node from the defender set D. If the decision-maker successfully selects a node from D that is adjacent to an activated edge, then they receive a positive reward. This Bipartite Edge Detection variant of the MAB naturally arises from the DCS-CV problem as a bipartite structure forms between the decoy configurations and exploits. A key difference from the normal non-combinatorial MAB problem is that the bipartite structure leads to limited information. The decision-maker not only sees the result of their chosen node but can observe all activated edges of an activated node if the activated node is detected. Through the decoy log files, the defender can know which other configurations (i.e. nodes in set D) were capable of detection (i.e., adjacent to the activated edges). This can naturally be a stochastic problem where nodes in E are randomly activated or through an adversary's actions.

When looking at the DCS-CV problem through the lens of bipartite graphs with MABs, we can see that each round, when the attacker selects an exploit, the reward essentially propagates through each adjacent edge of the exploit on the graph. The defender's goal, then is to try to pick any defense that is adjacent to the deployed exploit. By modeling this as a bipartite graph, we abstract the features and the potentially complex Boolean expressions into binary edges of a graph where a defense d_i either is able to detect exploit e_j (i.e., the edge (d_i, e_j) exists) or not (i.e., no adjacent edge). I leverage the idea of node adjacency to essentially feed the structure of the problem to the defender later in this chapter.

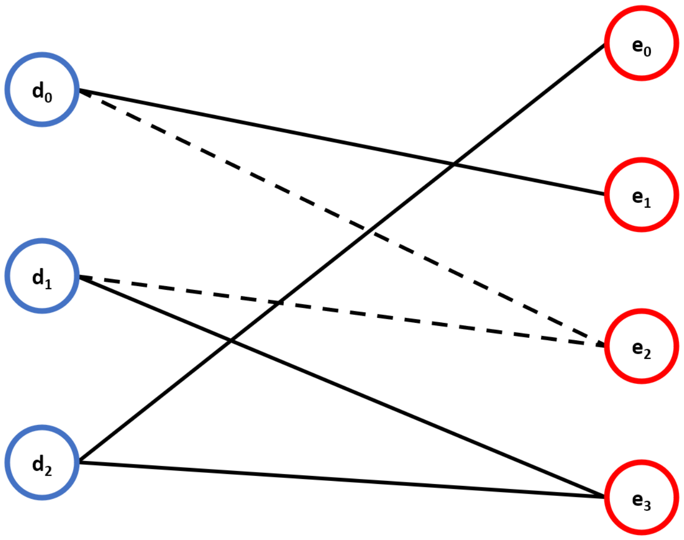


Figure 6.2: A simple bipartite graph of the connections between the Decision Set of nodes D and the Exploit Set of nodes E. In this round, the node e_2 is activated, and thus all adjacent edges (d_0, e_2) and (d_1, e_2) also activate. If the decision-maker selects nodes d_0 or d_1, she would receive a positive reward and observe all activated edges (d_0, e_2) and (d_1, e_2) to recognize that the other node would have also received a positive reward. If the decision-maker chooses d_3, she will receive no reward and only know that exploit nodes e_0 and e_3 did not activate this round.

Variable		Definition
T	:	number of rounds
N	:	number of features
F	:	set of features
f_k	:	feature k with a binary value 1 if the feature is present or 0 if not
D	:	set of all defenses available for selection by the defender
d_i	:	defense i, expressed as a collection feature values such that $d_i \in \{0,1\}^N$
$d_{i,k}$	:	value or presence of feature f_k in defense d_i (either 0 or 1)
E	:	set of all exploits available for selection by the attacker
e_i	:	exploit j, expressed as a Boolean expression of required features
$F(d_i, e_j)$	:	satisfaction function that returns 1 if defense d_i has all required features (i.e., satisfies the expression e_j) or 0 otherwise

6.4 Model

To formalize the problem, I model a 2-player asymmetric game where a defender and attacker have 2 separate action sets, D and E respectively, and repeatedly select a single action from their action sets over T rounds.

In each round, the defender selects from D with each defense d_i being represented as a unique collection of binary feature variables. Each defense d_i is defined by N total binary feature values (0 or 1) such that $d_i \in \{0,1\}^N$. $d_{i,k}$ represents the presence of feature f_k in defense d_i. If $d_{i,k}$ is 1, this can be interpreted as feature f_k is present in defense d_i.

The attacker selects exploits from E with each exploit e_j utilizing a complex vulnerability, which I express as a Boolean expression of features utilizing logical $\wedge$ and $\vee$ operators. An example of an exploit with a complex vulnerability might look like e_j below. This example suggests that exploit e_j can exploit any device that has the presence of either feature pair (f_0 and f_1) or (f_0 and f_2):

$$e_j = (f_0 \wedge f_1) \vee (f_0 \wedge f_2)$$

I assume the agents have access to some Boolean logical satisfaction checking function $F(d_i, e_j)$ that determines if d_i satisfies the Boolean expression for exploit e_j. If so, the defender receives a positive reward of 1 and observes the deployed exploit e_j and all its edges. Meanwhile, the attacker receives a reward of 0, observing the defense that detected it. If the defender does not detect the exploit, she receives a reward of 0 and does not observe the deployed exploit e_j. Meanwhile, the attacker receives a positive reward of 1 and does not observe the chosen defense d_i. W.L.O.G., we scale the rewards between 0 and 1 for calculation purposes. In the experiments below, I also assume that all exploits yield a reward of 1 for detecting them, but this does not need to be the case as all the decision-making algorithms are primarily focused on the average reward of each decision.

A key element of this model is the information observed by the attacker and defender. Both observe bandit feedback with some contextual information. That is to say that the decision-makers can only see the result of their chosen actions, however, in the event where the defender detects the attacker, both receive contextual information, specifically the chosen actions of the other. This means that a defender, when detecting an exploit, can see all activated edges associated with the exploit and should logically conclude other defenses that could have also detected the same exploit. Similarly, in the event that the defender does not detect any exploit, she should narrow down which exploits were not selected based on the understanding that any exploit that could have targeted the chosen defense would have yielded a positive reward.

6.4.1 Regret

To evaluate the defenders, I consider the notion of Cumulative Scaled Expected Regret (CSER) which is the summation of scaled expected regret over time as described in equation 3.3. This measure compares how close to optimal decision-making a defender does.

6.5 Attackers

In this section, I describe the 2 attackers that will appear in the experiments. The first is a fixed random attacker that acts as a baseline attacker and represents a community of attackers. The second is an IBL attacker that models a human attacker with knowledge of the game structure.

6.5.1 Fixed Random Attacker

This attacker acts as a stand-in for the hacker community as a whole. Considering the airport WiFi setting, this attacker might represent multiple worms randomly entering the network to attack all devices that are vulnerable to its vulnerability list. In this case, one can imagine each exploit e_j is a unique worm type that has a chance to enter the network and the attacker is the chance player that is deciding which worm tries to breach the network. It maintains a static distribution attack policy that is randomly generated. This attacker ignores all contextual information and rewards as it maintains a fixed randomized exploit policy.

Algorithm 4 Fixed Random Attacker

```
1: Initialize: weights $w$ for each exploit $w_j \sim U(1, 1000)$,
2: for $t = 1, 2, 3, \cdots, T$ do
3:     Play exploit $e_j \sim P(w)$
4: end for
```

6.5.2 IBL Insider Attacker

The primary ways we interact with an Instanced-Based Learning Model to feed it information about the game is by modeling the decisions, the rewards, any contextual information as attributes, and a similarity function. For modeling an attacker, the decisions are simply the exploits themselves. However, we need to address a core aspect of IBL similarity, which

Table 6.1: Truth table matching Figure 6.3 where a 1 indicates that defense d_i can detect exploit e_j

	d_0	d_1	d_2
e_0	0	0	1
e_1	1	0	0
e_2	1	1	0
e_3	0	1	1

is the connection between decisions. IBL relies on understanding how closely related decisions are from one another (0 being completely dissimilar and 1 being effectively the same decision like having the same Boolean expression), otherwise, it assumes each decision is independent of one another, much like the non-combinatorial stochastic bandit. To connect IBL's decisions to each other, we need to effectively pass the game structure to IBL, like the bipartite graph seen in Figure 6.3, we need to model a similarity function to tell IBL how exploits relate to one another. Instead of trying to teach IBL the understanding of Boolean algebra, we only need to tell IBL how similar exploits are to each other. The best way to handle this is to compare the truth tables of any 2 exploits. We could evaluate the entire 2^N truth table where we consider every combination of the N total features or even a restricted truth table of possible defenses such as $\binom{N}{n}$ if we know that every defense contains only n features such that $1 < n < N$. However, I instead investigate a more computationally inexpensive approach of feeding the truth tables of how each exploit interacts with each defense. Once you have a truth table to compare, you can see at each feature configuration modeled in the truth table, if the exploits match truth values.

By modeling the full truth table of only how exploits map to defenses, we effectively are passing the defender's full action set to the attacker. We only need to calculate the proportion of matching truth values between any 2 exploits. We can then end up with

Table 6.2: Similarity table matching Table 6.1 that tells how similar in terms of truth values 2 exploits are.

	e_0	e_1	e_2	e_3
e_0	1	$\frac{1}{3}$	0	$\frac{2}{3}$
e_1	$\frac{1}{3}$	1	$\frac{2}{3}$	0
e_2	0	$\frac{2}{3}$	1	$\frac{1}{3}$
e_3	$\frac{2}{3}$	0	$\frac{1}{3}$	1

a table like the one seen in Table 6.2. Once all these values are calculated, it turns into a simple lookup table to check the similarity between any 2 exploits. This tells IBL how much they can learn about other similar choices, thus exploiting the bipartite structure of the problem.

6.6 Defenders

In this section, I describe the 5 defenders that will appear in the experiments. These are combination of the agents described in chapters 6 and 4. I also describe an EXP4 defender, which is a well-known extension to EXP3 that allows for expert advice. For EXP4, I introduce a novel algorithm for modeling how the experts give advice to the bandit to capture the bipartite structure of the problem.

6.6.1 UCB Defender

Upper Confidence Bound (UCB) is perhaps the most well-known MAB algorithm, first introduced by Auer et al. [17]. The algorithm addresses the stochastic, non-combinatorial MAB problem. It assumes that each arm on the bandit is independent and identically distributed (i.i.d.). As seen in Algorithm 5, UCB has an initial exploration period where it must first play each arm once so that $m_i > 0, \forall i \in D$ where m_i is the number of times arm i has been played and $\hat{\theta}_i$ is the mean observed reward of arm i. Once the initial exploration loop is complete and there is some experience for each arm, UCB then solves maximization equation 6.1 to select an arm each round. Equation 6.1 notably sums between an exploration term (i.e., everything in the square root) and an exploitation term (i.e., $\hat{\theta}_i$).

Algorithm 5 Upper Confidence Bound (UCB)

```
1: **Initialization**
2: Set $L = |D|$
3: Randomly shuffle a queue containing every action $i \in D$
4: **for** $t = 1$ to $L$ **do**
5:     Pop and play next action $d_i$ from the queue
6:     Update $\hat{\theta}_i$, $m_i$ accordingly
7: **end for**
8: //MAIN LOOP
9: **for** $t = |D| + 1$ to $T$ **do**
10:    Play an action $i$ which solves the maximization problem
           $i = \underset{i \in D}{\text{argmax}} \left( \hat{\theta}_i + \sqrt{\frac{(L+1)\ln t}{m_i}} \right),$    (6.1)
11:    Update $\hat{\theta}_i$, $m_i$ accordingly
12: **end for**
```

For the DCS-CV setting, UCB considers each decoy configuration as i.i.d. arms. However, the defenses likely are dependent on each other as they may share features and connections to exploits. Overall, UCB will still learn and ultimately balance the mean observed reward $\hat{\theta}_i$ of each defense i, however, this translates to learning slowly as it ignores the dependencies between the defenses. This defender will act as a good baseline for performance and any defender should improve upon UCB.

6.6.2 EXP3 Defender

First described by Auer et al., The Exponential-weight algorithm for Exploration and Exploitation (EXP3) removes the assumption of how rewards are formed [18]. Whereas UCB assumes i.i.d. arms that will generate fixed random rewards, EXP3 makes no assumption about the forming of rewards and even allows for a potential adversary to change the rewards over time. However, EXP3 still operates non-combinatorially and assumes each arm is independent of one another. Similarly to UCB, EXP3 has explicit exploitation and exploration terms that are summed together as seen in equation 6.2.

Algorithm 6 Exponential-weight algorithm for Exploration and Exploitation (EXP3)

```
1: **Parameters:** Real $\gamma \in (0, 1]$
2: **Initialization:** $w_i(1) = 1$ for $i = 1, \cdots, |D|$
3: **for** $t = 1, 2, \cdots$ **do**
4:     Set
           $$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{|D|} w_j(t)} + \frac{\gamma}{|D|} \quad i = 1, \cdots, |K|. \tag{6.2}$$
5:     Draw $i_t$ randomly according the probabilities $p_1(t), \cdots, p_k(t)$.
6:     Receive reward $x_i(t) \in [0, 1]$.
7:     For $j = 1, \cdots, K$ set
           $$\hat{x}_j(t) = \begin{cases} \frac{x_j(t)}{p_j(t)} & \text{if } j = i_t \\ 0 & \text{otherwise,} \end{cases}$$
           $$w_j(t+1) = w_j(t)\, exp(\frac{\gamma \hat{x}_j}{|D|}). \tag{6.3}$$
8: **end for**
```

For the DCS-CV setting, EXP3 still learns quickly and should improve over UCB due to its lack of reward structure assumptions, which is closer to the truth, but it still does not make any combinatorial improvements by exploiting the structure of the Bipartite Edge Detection problem.

6.6.3 LLR Defender

Learning with Linear Rewards takes the approach of assuming a combinatorial stochastic bandit problem where the bandit selects multiple arms each round. The algorithm is described in more detail in Algorithm 3. For the problem of complex vulnerabilities, LLR views the decision of each round as selecting a combination of features instead of choosing large abstract configurations like the UCB Defender and EXP3 Defender do. In support of this, LLR assumes that each feature is independent of one other and stores data for each feature as opposed to each defense. If we were exclusively modeling non-complex vulnerabilities, or vulnerabilities that are only expressed with a single require feature, this form of data collection would work well. However, the issue lies in that LLR will incorrectly update its beliefs about each feature when confronted with a complex exploit. When the LLR defender detects an exploit, it will reward each feature that was necessary to detect the exploit. For example, let's consider the exploit discussed with the IBL Attacker:

$$e_j = (f_0 \wedge f_1) \vee (f_0 \wedge f_2)$$

If LLR Defender successfully detected exploit e_j by deploying a defense containing f_0 and f_2, it will erroneously update features f_0 and f_2 such that any defense with f_0 will be higher valued, even if those defenses do not contain f_1 or f_2. The further away an exploit gets in expression from a single feature, the more LLR will struggle. This baseline agent performs the worst due to its incorrect evaluations, however, for future work, I would like to investigate if any similar algorithms that focus on the features might perform better. Any feature-focused combinatorial algorithm would need to lift the feature independence assumption.

6.6.4 EXP4 Defender with Novel Expert Advice Algorithm

Introduced alongside EXP3 by Auer et al., EXP4 builds on top of EXP3 by allowing expert advice [18]. The experts impart advice vectors ξ_n to the bandit each round in the form of recommended probabilities to play each arm of the bandit. EXP4 makes no restriction on how the experts arrive at their recommended advice or even how many experts there should be, just simply that each expert's recommended probabilities sum to 1.0. Ideally, we can model the experts to consider the bipartite structure formed from the complex vulnerability problem. I introduce an effective way to generate the expert advice $\xi(t)$ found in step 4 of Algorithm 7.

Algorithm 7 Exponential-weight algorithm for Exploration and Exploitation with Expert advice (EXP4)

```
1: Parameters: Real $\gamma \in (0,1]$
2: Initialization: $w_i(1) = 1$ for $i = 1, \cdots, |D|$
3: for $t = 1, 2, \cdots$ do
4:     Get advice vectors $\xi^0(t), \cdots, \xi^{N-1}(t)$
5:     Set $W_t = \sum_{i=1}^{N} w_t(t)$ and for $j = 1, \cdots, |D|$ set
           $$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)}\xi_j^i(t) + \frac{\gamma}{|D|} \quad i = 1, \cdots, |D|. \qquad (6.4)$$
6:     Draw $i_t$ randomly according the probabilities $p_1(t), \cdots, p_d(t)$.
7:     Receive reward $x_i(t) \in [0,1]$.
8:     For $j = 1, \cdots, |D|$ set
           $$\hat{x}_j(t) = \begin{cases} \frac{x_j(t)}{p_j(t)} & \text{if } j = i_t \\ 0 & \text{otherwise,} \end{cases}$$
9:     For $i = 1, \cdots, N$ set
           $$\hat{y}_i(t) = \xi^i(t) \cdot \hat{x}(t) \qquad (6.5)$$
           $$w_i(t+1) = w_i(t)\, exp(\frac{\gamma \hat{y}_j}{|D|}). \qquad (6.6)$$
10: end for
```

Table 6.3: Expert advice for EXP4 for the set up in seen in figure 6.3 assuming that the exploits were discovered sequentially from e_0 to e_3. Expert 0 was created at the start of the game and assumes no exploit will be attacked.

	ξ^0	ξ^1	ξ^2	ξ^3	ξ^4
d_0	$\frac{1}{3}$	0	1	$\frac{1}{2}$	0
d_1	$\frac{1}{3}$	0	0	$\frac{1}{3}$	$\frac{1}{2}$
d_2	$\frac{1}{3}$	1	0	0	$\frac{1}{2}$

To generate the experts advice, EXP4 starts with a single expert that assumes no exploit will be deployed. Expert $\mathcal{E}_0$ always recommends the bandit play uniformly random among all the arms in Decision Set D. Each time an exploit is discovered, a new expert $\mathcal{E}$ is created and tied to that exploit for the rest of the game. This new expert assumes that its associated exploit will always attack in the next round. To address this, this new expert always recommends playing uniformly random over all of the adjacent nodes in D. Table 6.3 demonstrates the advice vectors ξ that would be sequentially created for the bipartite graph seen in figure 6.3.

There is the restrictive assumption that the defender is fully aware of the structure of the game as well and knows all exploits at the start of the game. However, we can easily modify how the experts pass information. We still need to assume that the defender knows how many exploits there are or at the least an upper-bound estimate. If we assume that the defender initially has no information about the exploits in the game and their relation to the defenses, then each expert will suggest a uniform distribution over all the defenses. However, once an exploit is discovered, we learn all its edges on the bipartite graph (i.e. which defenses can detect it) and the expert can start suggesting the usual advice, while the other experts, waiting to be assigned to an exploit, will continue to suggest a uniform random distribution over all the defenses. By modeling the bipartite graph this way, we can allow for the discovery of zero-day exploits and understand new relationships between features and exploits that may have been previously unknown. Once every exploit has been discovered, every expert should be assigned, providing the usual static advice for the rest of the game. This approach to expert advice is notably static as the experts always assume their associated exploit will always be deployed. There is room for improvement to allow the experts a little more understanding of the activation probability of their respective exploit nodes. This approach also passes off the Bipartite Edge Detection to the experts as opposed to the bandit directly solving the problem.

6.6.5 IBL Defender

I model the IBL Defender in much the same capacity as the IBL Attacker. This defender's decisions consist of the decoy configurations and follow the same reward feedback loop. Much like the IBL attacker, we need to address the IBL similarity. I model the defender's similarity such that the similarity between 2 defenses is a comparison of their truth tables with the attacker's exploits in the game.

6.7 Experiments

I developed a Python simulation to evaluate each defender in a rapid environment. The results in this section were all collected using this simulation. For experimental data, I randomly generated 5 action sets, which form the decision environment for the attacker and defender agents. Each action set contains the number of features, the defenses with each expressed as a vector of binary feature values, and each exploit, expressed as a Boolean satisfaction expression of feature variables. Each action set has 12 total features f_0 - f_{11}, 8 defenses d_0 - d_7 each containing between $4 - 7$ features such that $4 \leq |D| \leq 7$, and 16 total exploits e_0 - e_{15} where each Boolean expression only lists between 1 - 3 features, using both $\wedge$ and $\vee$ connectors. When generating the action sets, no defense is a repeat or a subset of another, and it is guaranteed that each defense has a unique configuration. Similarly, every

Table 6.4: adjacency table of action set 1

	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}
d_0	1	0	1	1	1	1	1	0	0	0	1	0	0	0	1	1
d_1	0	0	1	0	1	1	1	0	0	1	1	1	1	1	0	0
d_2	1	0	1	1	1	0	1	1	1	1	0	1	0	1	0	1
d_3	1	1	1	0	0	1	0	0	1	0	1	0	0	0	1	1
d_4	0	0	1	1	1	0	1	1	1	0	0	1	0	1	1	1
d_5	1	0	1	1	1	1	1	0	0	1	0	1	0	1	0	1
d_6	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0	0
d_7	0	1	1	0	1	0	0	0	1	1	1	1	0	1	0	0

exploit was generated as a randomized subset of a random defense such that every defense is guaranteed to connect to at least one exploit and every exploit is guaranteed to connect to at least one defense. The adjacency table for action set 1 can be seen in Table 6.4.

For each of the 5 defenders and 2 attackers, I ran a game for each defender-attacker pair 10 times for each action set where I collected the average CSER measurement over the 10 repeated matchups. The same relative results (defenders relative to each other) are held across all action sets and the results seen in figure 6.3 are representative of the results of all the action set experiments.

As I show in the results in figure 6.3, EXP4 with my Bipartite Edge Detection expert advice far outperforms the other representative defenders. Within 400 rounds, EXP4 approaches zero-regret versus a random fixed attacker which is not the case for the other algorithms. Over the short horizon of only 1000 rounds, it appears that EXP3 starts to trend toward zero-regret but maintains the highest standard deviation among all the defenders, suggesting instability in the defense algorithm. The only difference between EXP4 and EXP3 is the expert advice where the experts would pass along the suggested defenses based on the adjacent edges of their exploits. This suggests that the use of experts is a

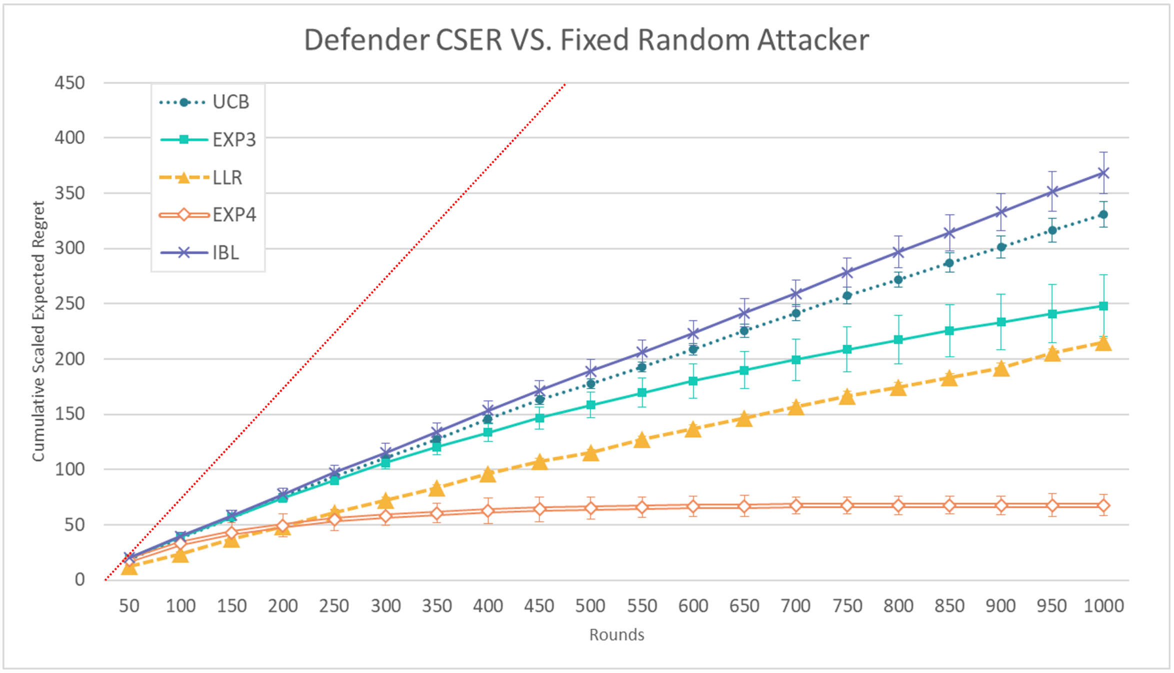


Figure 6.3: The 5 defenders versus a fixed random attacker in action set 1. The thin dotted red line represents the maximum regret line if the defenders were to systematically choose the worst defense node that yielded the maximum regret each round.

good way of addressing the Bipartite Edge Detection problem without using combinatorial nonlinear bandits.

Meanwhile, the 3 remaining defenders show no indication of approaching zero-regret with IBL performing the worst overall. Despite the IBL defender having access to the entire defense-exploit adjacency matrix for similarity, it performed the worst overall. I theorize this is due to the starvation of information. With little information (i.e., successful captures), IBL views zero-reward rounds as the average case and grows complacent.

Despite the success found in the first experiment with a fixed random attacker, the Insider Threat IBL attacker proves a much harder opponent, leading to roughly the same linear regret across all the defenders as seen in figure 6.4. More investigation is needed to fully assess this result as this attacker was operating on the same information as the IBL

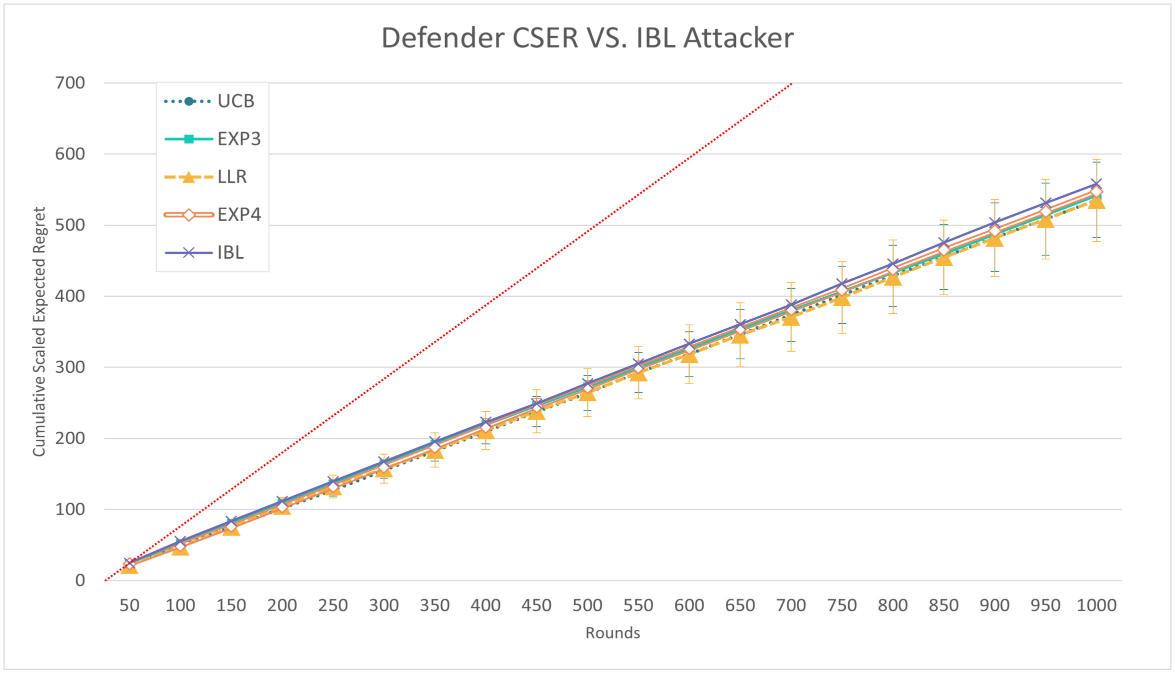


Figure 6.4: The 5 defenders versus an insider threat IBL attacker on action set 1. The thin dotted red line represents the maximum regret line if the defenders were to systematically choose the worst defense node that yielded the maximum regret each round.

defender. I theorize the cause of this result has to do with the inherent advantage held by the attacker in these action sets. When there are fewer adjacent edges to the exploit nodes in the action set (i.e., when the average amount of decoy configurations that can detect an exploit is low), the attacker holds an advantage as it is harder to capture. For the IBL defender, this leads to a starvation of information as grows complacent with getting zero reward each round. Meanwhile, the Insider Threat IBL attacker likely has a wealth of information on which exploits frequently yield positive rewards (i.e., avoid detection), leading to quick, exploitative learning.

6.8 Conclusion

I introduced the novel Decoy Configuration Selection with Complex Vulnerabilities problem and addressed this problem as a novel MAB problem with Bipartite Edge Detection. Although I did not introduce a direct combinatorial nonlinear bandit to solve the DCS-CV problem, I instead introduced a new algorithm to provide expert advice to the non-combinatorial EXP4 bandit on the bipartite structure that forms in the DCS-CV problem. In the stochastic setting, my version of EXP4 quickly adapted to the learning problem and reached zero-regret much faster than the other algorithms. This further suggests that the contextual bipartite structures formed in the DCS-CV problem can be exploited for better performance. However, when confronted with an adaptive, human-like attacker in IBL, EXP4 struggled significantly. I believe moving from the experts' static assumptions about their respective exploits to a more realistic accounting of exploit attack probabilities will improve EXP4's adversarial capabilities.

This work naturally answers my third research question:

Q3. How well will cognitive models evaluate and learn with complex vulnerabilities?

For the defensive side, the IBL cognitive model defender performs poorly because the defenders get "starved" of knowledge as it struggles to detect attacks. On the attacking side, the IBL attacker far outperforms the fixed random attacker because it receives much more information about positive rewards.

For my fourth research question:

Q4. How much can we improve over naïve non-combinatorial and linear reward solutions for adversarial interactions with complex vulnerabilities?

By passing along the information of the bipartite structure of the problem to experts, we can vastly improve the defensive capabilities of the MAB defenders for detecting complex vulnerabilities in a stochastic setting, even quickly converging towards zero-regret. However, much more needs to be done to improve the results of the defenders when confronted with adaptive learning attackers.

6.8.1 Applications

I focus on the DCS-CV problem with regard to intrusion detection in an airport Wi-Fi, but there are other applications where Bipartite Edge Detection appears.

The DCS-CV problem emphasizes the low-level detection of vulnerabilities, however, we can consider higher-level attack graphs as well. Attack graphs are typically generated through network analysis tools and demonstrate steps required for more complex attack campaigns like a denial of service attack [80]. Instead of modeling a decoy configuration with lower-level features like applications, we can consider our Decision Set D to represent various honeynet configurations and through attack graphs, the Exploit Set E consists of high-level attacks like denial of service attacks, distributed denial of service attacks, data exfiltration campaigns, and more. The problem then becomes selecting which honeynet configuration provides the best sandbox to discover the most about attacker intentions and capabilities.

Another possible setting is in software testing. Traditional software testing requires unit testing, which tests individual software modules or units for internal bugs. Less common, but still well-known is combinatorial software testing. In particular, pairwise testing involves combining pairs of software units together to discover bugs that might arise between multiple units. This form of combinatorial software testing is much more taxing with a much larger search space and likely yields diminishing results, however, numerous bugs can be discovered as software units are combined together [89].

As demonstrated by Kuhn et al., in a software application with large amounts of components or units, checking every possible combination of features can infeasible due to combinatorial explosion. However, only considering every pair or triplet drastically reduces the entire search space and typically covers about 90% all vulnerabilities found in applications. However, some vulnerability interactions may only arise with specific inputs and even software pairs may need to be tested multiple times as with combinatorial software fuzzing [102, 38, 140, 63, 30]. Formulating a problem of combinatorial software testing can be done by modeling it as a non-combinatorial Bipartite Edge Detection problem. In this case, the Decision Set D includes combinations of software units (perhaps some subset of pairs and triplets) and the Exploit Set E then forms the individual bugs. Notably, here, the bandit would have no knowledge of the exploit set E and would discover it over time. The decision-maker then strategically selects software unit combinations as its decisions and receives a reward if a crash occurs during runtime execution. Notably, MAB solutions typically try to "exploit" and pick the same best arm as frequently as possible, but this modeling would fall more under a pure exploration bandit problem where once an edge is successfully detected, the decision-maker may not want to dwell on the same configuration.

6.8.2 Discussion on Deployment of Model

All the work in this dissertation from the modeling to the simulation operates on a high-level abstraction. I would like to briefly discuss the elements that would need to be addressed to begin the implementation process for emulation or in the wild like in an emulation environment such as the Cybersecurity Deception Experimentation System (CDES) [7].

First is the notion of discrete intervals found in MAB problems. In my results, I present the finding that EXP4 quickly reached zero-regret by 400 rounds, but a natural question is: what is the timescale of a round? Ultimately, with any real-time system like cybersecurity, you may need to discretize time slots. Ultimately, for the DCS-CV model, it depends on how often a network administrator would want to change their decoy configuration. I imagine for the original example of a public Airport Wi-Fi, the timescale would operate on the order of hours as realistically a person might log on for an hour and then board their flight afterward. In a smallish setting, this would suggest a defender optimizing over a couple of weeks. When combined with the fact that cyber decoys are not the only forms of defense, but can be added on top of any existing defenses, I think this is reasonable. These bandits can also be slightly trained with a warm start prior to production, such as passing some basic game theoretic analysis for an initial training period.

I make a major assumption during my DCS-CV model that the defender has access to some accurate satisfaction function $F(d_i, e_j)$. In practice, this would translate to the honeypots having access to some classifier that can accurately detect and identify exploits and their required features. This classifier does not need to fully understand the exploits but rather only needs to detect if an exploit is the same or different from previously identified exploits and accurately determine the required Boolean expressions of features to determine the relationship to other decoy configurations. The construction of this classifier would likely spin off into an entire research project of its own.

Lastly, one of the most important driving forces in reinforcement learning is observation and reward. The observations should come from the classifiers which would likely stem from the decoy's log files. Through analysis of the log files, the decoy should recognize when attackers are interacting with the decoy and for how long. These can be used to model the reward. In my experiments, I simply modeled binary reward values (0 or 1), but these can be scaled to account for how much time was spent interacting with the decoy or if the exploits can be tied to specific CVEs, then the CVSS score could be a useful reward value.

Chapter 7

Future Work

Throughout my dissertation, I have addressed and answered all of my research questions. This work stands at the intersection of reinforcement learning, cognitive modeling, and cybersecurity. Naturally, each of these domains has avenues to continue down in future work. First, for reinforcement learning, there are areas for improvement in my EXP4 expert advice algorithm to allow for more dynamic modeling of the exploit probabilities. I believe a combinatorial nonlinear bandit could also be developed that addresses Boolean expressions directly, this would address the features directly rather than the macro view of the Bipartite Edge Detection. Furthermore, investigating other logical frameworks like temporal Boolean logic and fuzzy logic could prove useful for investigating more complex attacks as seen in attack graphs. Investigating more intelligent adaptive attackers, such as game theoretic attackers and bandit attackers.

For cognitive modeling, the agents can be improved upon through parameter tuning and more work can be done to investigate why the IBL attacker performs so strongly against the MAB defenders. These parameters can be fine-tuned to achieve the best performance versus the defenders or even better: to most realistically match human behavior. A good next step for validation would be to set up a playable version of this model for human participants (as attackers and defenders) and compare how they interact in this setting.

For contributions to cybersecurity, the natural approach is to implement the DCS-CV model in an emulation environment and evaluate these algorithms in a realistic environment like CDES. Many of the abstract elements would need to be addressed and hammered out. One of the major elements of the model would be to concretely identify what features and exploits would be modeled. One approach to do this would be to take a subset of

CVEs from the NVD and their respective CPEs as features and model a scenario that way. Another, potentially easier way to model the Bipartite Edge Detection problem would be to utilize attack graphs and model more complex attacks like denial of service attacks as exploits.

References

[1] Cve-2005-0179. `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-0179`. Accessed: April 28, 2023.

[2] Cve-2019-18939. `https://cve.mitre.org/cgi-bin/cvename.cgi?name=2019-18939`. Accessed: April 28, 2023.

[3] Cve-2021-20229. `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-20229`. Accessed: April 28, 2023.

[4] National vulnerability database. `https://nvd.nist.gov/`. Accessed: April 28, 2023.

[5] Nvdlib: Nist national vulnerability database api wrapper.

[6] Jaime C Acosta, Anjon Basak, Christopher Kiekintveld, and Charles Kamhoua. Lightweight on-demand honeypot deployment for cyber deception. In *Digital Forensics and Cyber Crime: 12th EAI International Conference, ICDF2C 2021, Virtual Event, Singapore, December 6-9, 2021, Proceedings*, pages 294–312. Springer, 2022.

[7] Jaime C Acosta, Anjon Basak, Christopher Kiekintveld, Nandi Leslie, and Charles Kamhoua. Cybersecurity deception experimentation system. In *2020 IEEE Secure Development (SecDev)*, pages 34–40. IEEE, 2020.

[8] Mario R Camana Acosta, Saeed Ahmed, Carla E Garcia, and Insoo Koo. Extremely randomized trees-based scheme for stealthy cyber-attack detection in smart grid networks. *IEEE access*, 8:19921–19933, 2020.

[9] Palvi Aggarwal, Cleotilde Gonzalez, and Varun Dutt. Cyber-security: role of deception in cyber-attack detection. In *Advances in Human Factors in Cybersecurity:*

Proceedings of the AHFE 2016 International Conference on Human Factors in Cybersecurity, July 27-31, 2016, Walt Disney World®, Florida, USA, pages 85–96. Springer, 2016.

[10] Palvi Aggarwal, Cleotilde Gonzalez, and Varun Dutt. Modeling the effects of amount and timing of deception in simulated network scenarios. In *2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, pages 1–7. IEEE, 2017.

[11] Palvi Aggarwal, Marcus Gutierrez, Christopher D Kiekintveld, Branislav Bošanskỳ, and Cleotilde Gonzalez. Evaluating adaptive deception strategies for cyber defense with human adversaries. *Game Theory and Machine Learning for Cyber Security*, pages 77–96, 2021.

[12] Ahmed AlEroud and George Karabatis. A contextual anomaly detection approach to discover zero-day attacks. In *2012 International Conference on Cyber Security*, pages 40–45. IEEE, 2012.

[13] Mohammed H Almeshekah and Eugene H Spafford. Cyber security deception. *Cyber Deception: Building the Scientific Foundation*, pages 23–50, 2016.

[14] Iffat Anjum, Mohammad Sujan Miah, Mu Zhu, Nazia Sharmin, Christopher Kiekintveld, William Enck, and Munindar P Singh. Optimizing vulnerability-driven honey traffic using game theory. *arXiv preprint arXiv:2002.09069*, 2020.

[15] Ahmed H Anwar and Charles Kamhoua. Game theory on attack graph for cyber deception. In *Decision and Game Theory for Security: 11th International Conference, GameSec 2020, College Park, MD, USA, October 28–30, 2020, Proceedings 11*, pages 445–456. Springer, 2020.

[16] Muhammad Kamran Asif and Yahya Subhi Al-Harthi. Intrusion detection system using honey token based encrypted pointers to mitigate cyber threats for critical

infrastructure networks. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1266–1270. IEEE, 2014.

[17] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[18] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 322–331. IEEE, 1995.

[19] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[20] Maryam Aziz, Emilie Kaufmann, and Marie-Karelle Riviere. On multi-armed bandit designs for dose-finding clinical trials. *The Journal of Machine Learning Research*, 22(1):686–723, 2021.

[21] Anjon Basak, Jakub Černỳ, Marcus Gutierrez, Shelby Curtis, Charles Kamhoua, Daniel Jones, Branislav Bošanskỳ, and Christopher Kiekintveld. An initial study of targeted personality models in the flipit game. In *Decision and Game Theory for Security: 9th International Conference, GameSec 2018, Seattle, WA, USA, October 29–31, 2018, Proceedings*, pages 623–636. Springer, 2018.

[22] Anjon Basak, Marcus Gutierrez, and Christopher Kiekintveld. Algorithms for subgame abstraction with applications to cyber defense. In *Decision and Game Theory for Security: 9th International Conference, GameSec 2018, Seattle, WA, USA, October 29–31, 2018, Proceedings 9*, pages 556–568. Springer, 2018.

[23] Anjon Basak, Charles Kamhoua, Sridhar Venkatesan, Marcus Gutierrez, Ahmed H Anwar, and Christopher Kiekintveld. Identifying stealthy attackers in a game theo-

retic framework using deception. In *Decision and Game Theory for Security: 10th International Conference, GameSec 2019, Stockholm, Sweden, October 30–November 1, 2019, Proceedings 10*, pages 21–32. Springer, 2019.

[24] Anjon Basak, Charles A Kamhoua, Sridhar Venkatesan, Marcus Gutierrez, Ahmed H Anwar, and Christopher D Kiekintveld. Scalable algorithms for identifying stealthy attackers in a game-theoretic framework using deception. *Game Theory and Machine Learning for Cyber Security*, pages 47–61, 2021.

[25] John Bather. Randomised allocation of treatments in sequential trials. *Advances in Applied Probability*, 12(1):174–182, 1980.

[26] Muhammet Baykara and RESUL DAŞ. Softswitch: A centralized honeypot-based security approach using software-defined switching for secure management of vlan networks. *Turkish Journal of Electrical Engineering and Computer Sciences*, 27(5):3309–3325, 2019.

[27] Noam Ben-Asher and Cleotilde Gonzalez. Effects of cyber security knowledge on attack detection. *Computers in Human Behavior*, 48:51–61, 2015.

[28] Leyla Bilge and Tudor Dumitras. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 833–844. ACM, 2012.

[29] Djallel Bouneffouf, Irina Rish, and Charu Aggarwal. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[30] Josip Bozic, Bernhard Garn, Ioannis Kapsalis, Dimitris Simos, Severin Winkler, and Franz Wotawa. Attack pattern-based combinatorial testing with constraints for web security testing. In *2015 IEEE International Conference on Software Quality, Reliability and Security*, pages 207–212. IEEE, 2015.

[31] Matthew L Bringer, Christopher A Chelmecki, and Hiroshi Fujinoki. A survey: Recent advances and future trends in honeypot research. *International Journal of Computer Network and Information Security*, 4(10):63, 2012.

[32] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

[33] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.

[34] Giuseppe Burtini, Jason Loeppky, and Ramon Lawrence. A survey of online experiment design with the stochastic multi-armed bandit. *arXiv preprint arXiv:1510.00757*, 2015.

[35] Lorena Cazorla, Cristina Alcaraz, and Javier Lopez. Cyber stealth attacks in critical information infrastructures. *IEEE Systems Journal*, 12(2):1778–1792, 2016.

[36] Nicolo Cesa-Bianchi and Paul Fischer. Finite-time regret bounds for the multiarmed bandit problem. In *ICML*, volume 98, pages 100–108. Citeseer, 1998.

[37] Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. *Advances in neural information processing systems*, 21, 2008.

[38] Jinfu Chen, Jingyi Chen, Saihua Cai, Haibo Chen, and Chi Zhang. A novel combinatorial testing approach with fuzzing strategy. *Journal of Software: Evolution and Process*, page e2537, 2023.

[39] Shouyuan Chen, Tian Lin, Irwin King, Michael R Lyu, and Wei Chen. Combinatorial pure exploration of multi-armed bandits. *Advances in neural information processing systems*, 27, 2014.

[40] Wei Chen, Wei Hu, Fu Li, Jian Li, Yu Liu, and Pinyan Lu. Combinatorial multi-armed bandit with general reward functions. *Advances in Neural Information Processing Systems*, 29, 2016.

[41] Wei Chen, Yajun Wang, Yang Yuan, and Qinshi Wang. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *The Journal of Machine Learning Research*, 17(1):1746–1778, 2016.

[42] Hoon S Choi, Darrell Carpenter, and Myung S Ko. Risk taking behaviors using public wi-fi™. *Information Systems Frontiers*, pages 1–18, 2021.

[43] Richard Combes, Mohammad Sadegh Talebi Mazraeh Shahi, Alexandre Proutiere, et al. Combinatorial bandits revisited. *Advances in neural information processing systems*, 28, 2015.

[44] Pierre-Arnaud Coquelin and Rémi Munos. Bandit algorithms for tree search. *arXiv preprint cs/0703062*, 2007.

[45] Edward Cranford, Cleotilde Gonzalez, Palvi Aggarwal, Sarah Cooney, Milind Tambe, and Christian Lebiere. Adaptive cyber deception: Cognitively informed signaling for cyber defense. 2020.

[46] Edward A Cranford, Cleotilde Gonzalez, Palvi Aggarwal, Sarah Cooney, Milind Tambe, and Christian Lebiere. Toward personalized deceptive signaling for cyber defense using cognitive models. *Topics in Cognitive Science*, 12(3):992–1011, 2020.

[47] Edward A Cranford, Cleotilde Gonzalez, Palvi Aggarwal, Milind Tambe, Sarah Cooney, and Christian Lebiere. Towards a cognitive theory of cyber deception. *Cognitive Science*, 45(7):e13013, 2021.

[48] Lorrie F Cranor. A framework for reasoning about the human in the loop. 2008.

[49] Shelby R Curtis, Anjon Basak, Jessica R Carre, Branislav Bošanskỳ, Jakub Černỳ, Noam Ben-Asher, Marcus Gutierrez, Daniel N Jones, and Christopher Kiekintveld. The dark triad and strategic resource control in a competitive computer game. *Personality and Individual Differences*, 168:110343, 2021.

[50] Seamus Dowling, Michael Schukat, and Enda Barrett. Improving adaptive honeypot functionality with efficient reinforcement learning parameters for automated malware. *Journal of Cyber Security Technology*, 2(2):75–91, 2018.

[51] Yinuo Du, Zimeng Song, Stephanie Milani, C Gonzales, and Fei Fang. Learning to play an adaptive cyber deception game. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems. Auckland, New Zealand*, volume 6, 2022.

[52] Patrick Duessel, Christian Gehl, Ulrich Flegel, Sven Dietrich, and Michael Meier. Detecting zero-day attacks using context-aware anomaly detection at the application-layer. *International Journal of Information Security*, 16:475–490, 2017.

[53] Hubert D'Cruze, Ping Wang, Raed Omar Sbeit, and Andrew Ray. A software-defined networking (sdn) approach to mitigating ddos attacks. In *Information Technology-New Generations: 14th International Conference on Information Technology*, pages 141–145. Springer, 2018.

[54] Fei Fang. Game theoretic models for cyber deception. In *Proceedings of the 8th ACM Workshop on Moving Target Defense*, pages 23–24, 2021.

[55] Kimberly Ferguson-Walter, Maxine Major, Chelsea K Johnson, and Daniel H Muhleman. Examining the efficacy of decoy-based and psychological cyber deception. In *USENIX Security Symposium*, pages 1127–1144, 2021.

[56] Kimberly Ferguson-Walter, Temmie Shade, Andrew Rogers, Michael Christopher Stefan Trumbo, Kevin S Nauer, Kristin Marie Divis, Aaron Jones, Angela Combs, and

Robert G Abbott. The tularosa study: An experimental design and implementation to quantify the effectiveness of cyber deception. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2018.

[57] Kimberly J Ferguson-Walter, Dana S LaFon, and TB Shade. Friend or faux: deception for cyber defense. *Journal of Information Warfare*, 16(2):28–42, 2017.

[58] Key Focus. Kfsensor overview, 2003.

[59] Daniel Fraunholz, Marc Zimmermann, and Hans D Schotten. An adaptive honeypot configuration, deployment and maintenance strategy. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 53–57. IEEE, 2017.

[60] Stefan Frei, Martin May, Ulrich Fiedler, and Bernhard Plattner. Large-scale vulnerability analysis. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, pages 131–138. ACM, 2006.

[61] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking (TON)*, 20(5):1466–1478, 2012.

[62] Nandan Garg and Daniel Grosu. Deception in honeynets: A game-theoretic analysis. In *2007 IEEE SMC Information Assurance and Security Workshop*, pages 107–113. IEEE, 2007.

[63] Bernhard Garn and Dimitris E Simos. Eris: A tool for combinatorial testing of the linux system call interface. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, pages 58–67. IEEE, 2014.

[64] Samuel J Gershman, Eric J Horvitz, and Joshua B Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278, 2015.

[65] JC Gittins. The two-armed bandit problem: variations on a conjecture by h. chernoff. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 287–291, 1975.

[66] Cleotilde Gonzalez and Varun Dutt. Instance-based learning: Integrating sampling and repeated decisions from experience. *Psychological review*, 118(4):523, 2011.

[67] Cleotilde Gonzalez, Javier F Lerch, and Christian Lebiere. Instance-based learning in dynamic decision making. *Cognitive Science*, 27(4):591–635, 2003.

[68] Aditya Gopalan, Shie Mannor, and Yishay Mansour. Thompson sampling for complex online problems. In *International conference on machine learning*, pages 100–108. PMLR, 2014.

[69] Marcus Gutierrez. Full list of complex cves from cve-1999-0001 to cve-2019-9923. `https://github.com/MarcusGutierrez/complex-vulnerabilities/blob/main/Complex-Vulnerabilities-List-CVE-1999-0003.pdf`, 2023.

[70] Marcus Gutierrez. Full list of complex cves from cve-2019-9924 to cve-2023-29421. `https://github.com/MarcusGutierrez/complex-vulnerabilities/blob/main/Complex-Vulnerabilities-List-CVE-2019-9924.pdf`, 2023.

[71] Marcus Gutierrez, Noam Ben-Asher, Efrat Aharonov, Branislav Bošanskỳ, Christopher Kiekintveld, and Cleotilde Gonzalez. Evaluating models of human adversarial behavior against defense algorithms in a contextual multi-armed bandit task.

[72] Marcus Gutierrez, Jakub Cernỳ, and Noam Ben-Asher. Evaluating models of human behavior in an adversarial multi-armed bandit problem.

[73] Marcus Gutierrez and Christopher Kiekintveld. Online learning methods for controlling dynamic cyber deception strategies. *Adaptive autonomous secure cyber systems*, pages 231–251, 2020.

[74] Marcus Paul Gutierrez and Christopher Kiekintveld. Bandits for cybersecurity: Adaptive intrusion detection using honeypots. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[75] Marcus Paul Gutierrez and Christopher Kiekintveld. Adapting honeypot configurations to detect evolving exploits. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[76] Yanjun Han, Yining Wang, and Xi Chen. Adversarial combinatorial bandits with general non-linear reward functions. In *International Conference on Machine Learning*, pages 4030–4039. PMLR, 2021.

[77] Kristin E Heckman, Michael J Walsh, Frank J Stech, Todd A O'boyle, Stephen R DiCato, and Audra F Herber. Active cyber defense with denial and deception: A cyber-wargame experiment. *computers & security*, 37:72–77, 2013.

[78] Hanan Hindy, Robert Atkinson, Christos Tachtatzis, Jean-Noël Colin, Ethan Bayne, and Xavier Bellekens. Utilising deep learning techniques for effective zero-day attack detection. *Electronics*, 9(10):1684, 2020.

[79] Yunhan Huang, Linan Huang, and Quanyan Zhu. Reinforcement learning for feedback-enabled cyber resilience. *Annual Reviews in Control*, 2022.

[80] Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical attack graph generation for network defense. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 121–130. IEEE, 2006.

[81] Sushil Jajodia, V Subrahmanian, Vipin Swarup, and Cliff Wang. *Cyber deception*, volume 6. Springer, 2016.

[82] Forough Ja'fari, Seyedakbar Mostafavi, Kiarash Mizanian, and Emad Jafari. An intelligent botnet blocking approach in software defined networks using honeypots. *Journal of Ambient Intelligence and Humanized Computing*, 12:2993–3016, 2021.

[83] Chelsea Kae Johnson. *Decision-Making Biases in Cybersecurity: Measuring the Impact of the Sunk Cost Fallacy to Delay Attacker Behavior*. PhD thesis, Arizona State University, 2022.

[84] Ari Juels and Ronald L Rivest. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 145–160, 2013.

[85] Christopher Kiekintveld, Viliam Lisỳ, and Radek Píbil. Game-theoretic foundations for the strategic use of honeypots in network security. In *Cyber Warfare*, pages 81–101. Springer, 2015.

[86] Richard Klíma, Viliam Lisỳ, and Christopher Kiekintveld. Combining online learning and equilibrium computation in security games. In *International Conference on Decision and Game Theory for Security*, pages 130–149. Springer, 2015.

[87] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings 17*, pages 282–293. Springer, 2006.

[88] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41:297–327, 2011.

[89] Rick Kuhn, Raghu Kacker, Yu Lei, and Justin Hunter. Combinatorial software testing. *Computer*, 42(8):94–96, 2009.

[90] Branislav Kveton, Zheng Wen, Azin Ashkan, Hoda Eydgahi, and Brian Eriksson. Matroid bandits: Fast combinatorial optimization with learning. *arXiv preprint arXiv:1403.5045*, 2014.

[91] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. Combinatorial cascading bandits. *Advances in Neural Information Processing Systems*, 28, 2015.

[92] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. Tight regret bounds for stochastic combinatorial semi-bandits. In *Artificial Intelligence and Statistics*, pages 535–543. PMLR, 2015.

[93] Cheolhyeon Kwon, Weiyi Liu, and Inseok Hwang. Security analysis for cyber-physical systems against stealthy deception attacks. In *2013 American control conference*, pages 3344–3349. IEEE, 2013.

[94] John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. *Advances in neural information processing systems*, 20(1):96–1, 2007.

[95] John Levine, Richard LaBella, Henry Owen, Didier Contis, and Brian Culver. The use of honeynets to detect exploited systems across large enterprise networks. In *IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop, 2003.*, pages 92–99. IEEE, 2003.

[96] Tian Lin, Bruno Abrahao, Robert Kleinberg, John Lui, and Wei Chen. Combinatorial partial monitoring game with linear feedback and its applications. In *International Conference on Machine Learning*, pages 901–909. PMLR, 2014.

[97] Tyler Lu, Dávid Pál, and Martin Pál. Contextual multi-armed bandits. In *Proceedings of the Thirteenth international conference on Artificial Intelligence and Statistics*, pages 485–492. JMLR Workshop and Conference Proceedings, 2010.

[98] Tongbo Luo, Zhaoyan Xu, Xing Jin, Yanhui Jia, and Xin Ouyang. Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices. *Black Hat*, 2017.

[99] David Maimon, Michael Becker, Sushant Patil, and Jonathan Katz. Self-protective behaviors over public wifi networks. In *The {LASER} workshop: Learning from authoritative security experiment results ({LASER} 2017)*, pages 69–76, 2017.

[100] Abhishek Mairh, Debabrat Barik, Kanchan Verma, and Debasish Jena. Honeypot in network security: a survey. In *Proceedings of the 2011 international conference on communication, computing & security*, pages 600–605. ACM, 2011.

[101] Dave Mancinelli. Public wi-fi: Friend or foe. 2014.

[102] Valentin JM Manès, Soomin Kim, and Sang Kil Cha. Ankou: Guiding grey-box fuzzing towards combinatorial difference. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 1024–1036, 2020.

[103] Iik Muhamad Malik Matin and Budi Rahardjo. A framework for collecting and analysis pe malware using modern honey network (mhn). In *2020 8th International Conference on Cyber and IT Service Management (CITSM)*, pages 1–5. IEEE, 2020.

[104] Miles A McQueen, Trevor A McQueen, Wayne F Boyer, and May R Chaffin. Empirical estimates and observations of 0day vulnerabilities. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, pages 1–12. IEEE, 2009.

[105] Peter Mell, Karen Ann Kent, and Sasha Romanosky. *The common vulnerability scoring system (CVSS) and its applicability to federal agency systems*. Citeseer, 2007.

[106] Mohammad Sujan Miah, Palvi Aggarwal, Marcus Gutierrez, Omkar Thakoor, Yinuo Du, Oscar Veliz, Kuldeep Singh, Christopher Kiekintveld, and Cleotilde Gonzalez. Diversifying deception: Game-theoretic models for two-sided deception and initial human studies. In *Cyber Deception: Techniques, Strategies, and Human Aspects*, pages 1–23. Springer, 2022.

[107] Mohammad Sujan Miah, Marcus Gutierrez, Oscar Veliz, Omkar Thakoor, and Christopher Kiekintveld. Concealing cyber-decoys using two-sided feature deception games. In *HICSS*, pages 1–10, 2020.

[108] Mohammad Sujan Miah, Mu Zhu, Alonso Granados, Nazia Sharmin, Iffat Anjum, Anthony Ortiz, Christopher Kiekintveld, William Enck, and Munindar P Singh. Optimizing honey traffic using game theory and adversarial learning. In *Cyber Deception: Techniques, Strategies, and Human Aspects*, pages 97–124. Springer, 2022.

[109] Constantin Musca, Emma Mirica, and Razvan Deaconescu. Detecting and analyzing zero-day attacks using honeypots. In *2013 19th international conference on control systems and computer science*, pages 543–548. IEEE, 2013.

[110] Marcin Nawrocki, Matthias Wählisch, Thomas C Schmidt, Christian Keil, and Jochen Schönfelder. A survey on honeypot software and data analysis. *arXiv preprint arXiv:1608.06249*, 2016.

[111] Adrian Pauna and Ion Bica. Rassh-reinforced adaptive ssh honeypot. In *2014 10th International Conference on Communications (COMM)*, pages 1–6. IEEE, 2014.

[112] Adrian Pauna, Andrei-Constantin Iacob, and Ion Bica. Qrassh-a self-adaptive ssh honeypot driven by q-learning. In *2018 international conference on communications (COMM)*, pages 441–446. IEEE, 2018.

[113] Jeffrey Pawlick, Quanyan Zhu, et al. *Game Theory for Cyber Deception*. Springer, 2021.

[114] Radek Píbil, Viliam Lisỳ, Christopher Kiekintveld, Branislav Bošanskỳ, and Michal Pěchouček. Game theoretic model of strategic honeypot selection in computer networks. In *International Conference on Decision and Game Theory for Security*, pages 201–220. Springer, 2012.

[115] Georgios Portokalidis, Asia Slowinska, and Herbert Bos. Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. *ACM SIGOPS Operating Systems Review*, 40(4):15–27, 2006.

[116] Panagiotis Radoglou-Grammatikis, Athanasios Liatifis, Elisavet Grigoriou, Theocharis Saoulidis, Antonios Sarigiannidis, Thomas Lagkas, and Panagiotis Sarigiannidis. Trusty: A solution for threat hunting using data analysis in critical infrastructures. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 485–490. IEEE, 2021.

[117] Hesam Sagha, Saeed Bagheri Shouraki, Hosein Khasteh, and Mahdi Dehghani. Real-time ids using reinforcement learning. In *2008 Second International Symposium on Intelligent Information Technology Application*, volume 2, pages 593–597. IEEE, 2008.

[118] Subham Sahoo, Sukumar Mishra, Jimmy Chih-Hsien Peng, and Tomislav Dragičević. A stealth cyber-attack detection strategy for dc microgrids. *IEEE Transactions on Power Electronics*, 34(8):8162–8174, 2018.

[119] Aaron Schlenker, Omkar Thakoor, Haifeng Xu, Fei Fang, Milind Tambe, Long Tran-Thanh, Phebe Vayanos, and Yevgeniy Vorobeychik. Deceiving cyber adversaries: A game theoretic approach. In *AAMAS'18: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 892–900. IFAAMAS, 2018.

[120] Eric M Schwartz, Eric T Bradlow, and Peter S Fader. Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science*, 36(4):500–522, 2017.

[121] Jennifer Seberry, Xian-Mo Zhang, and YL Zheng. Nonlinearity and propagation characteristics of balanced boolean functions. *Information and Computation*, 119(1):1–13, 1995.

[122] Arturo Servin and Daniel Kudenko. Multi-agent reinforcement learning for intrusion detection. In *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning: 5th, 6th, and 7th European Symposium, ALAMAS 2005-2007 on Adaptive and Learning Agents and Multi-Agent Systems, Revised Selected Papers*, pages 211–223. Springer, 2008.

[123] Ellie Shahin. Is wifi worth it: The hidden dangers of public wifi. *Cath. UJL & Tech*, 25:205, 2016.

[124] Aleksandrs Slivkins et al. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, 2019.

[125] Mark Steyvers, Michael D Lee, and Eric-Jan Wagenmakers. A bayesian analysis of human decision-making on bandit problems. *Journal of Mathematical Psychology*, 53(3):168–179, 2009.

[126] Cliff Stoll. *The cuckoo's egg: tracking a spy through the maze of computer espionage.* Doubleday, 1989.

[127] Dominic Storey. Catching flies with honey tokens. *Network Security*, 2009(11):15–18, 2009.

[128] Ruming Tang, Zheng Yang, Zeyan Li, Weibin Meng, Haixin Wang, Qi Li, Yongqian Sun, Dan Pei, Tao Wei, Yanfei Xu, et al. Zerowall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 2479–2488. IEEE, 2020.

[129] Olayiwola Tokunbo Taofeek, Moatsum Alawida, Abdulatif Alabdulatif, Abiodun Esther Omolara, and Oludare Isaac Abiodun. A cognitive deception model for generating fake documents to curb data exfiltration in networks during cyber-attacks. *IEEE Access*, 10:41457–41476, 2022.

[130] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[131] Olivier Tsemogne, Yezekael Hayel, Charles Kamhoua, and Gabriel Deugoué. Game-theoretic modeling of cyber deception against epidemic botnets in internet of things. *IEEE Internet of Things Journal*, 9(4):2678–2687, 2021.

[132] Oscar Veliz, Marcus Gutierrez, and Christopher Kiekintveld. Teaching automated strategic reasoning using capstone tournaments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[133] Sridhar Venkatesan, Massimiliano Albanese, Ankit Shah, Rajesh Ganesan, and Sushil Jajodia. Detecting stealthy botnets in a resource-constrained environment using reinforcement learning. In *MTD@ CCS*, pages 75–85, 2017.

[134] Alexander Vetterl and Richard Clayton. Honware: A virtual honeypot framework for capturing cpe and iot zero days. In *2019 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–13. IEEE, 2019.

[135] Hibatul Wafi, Andrew Fiade, Nashrul Hakiem, and Rizal Broer Bahaweres. Implementation of a modern security systems honeypot honey network on wireless networks. In *2017 International Young Engineers Forum (YEF-ECE)*, pages 91–96. IEEE, 2017.

[136] Erich Walter, Kimberly Ferguson-Walter, and Ahmad Ridley. Incorporating deception into cyberbattlesim for autonomous defense. *arXiv preprint arXiv:2108.13980*, 2021.

[137] Cliff Wang and Zhuo Lu. Cyber deception: Overview and the road ahead. *IEEE Security & Privacy*, 16(2):80–85, 2018.

[138] Shuo Wang, Qingqi Pei, Jianhua Wang, Guangming Tang, Yuchen Zhang, and Xiaohu Liu. An intelligent deployment policy for deception resources based on reinforcement learning. *IEEE Access*, 8:35792–35804, 2020.

[139] Xin Xu and Tao Xie. A reinforcement learning approach for host-based intrusion detection using sequences of system calls. In *International Conference on Intelligent Computing*, pages 995–1003. Springer, 2005.

[140] Jian Yang, Huanguo Zhang, and Jianming Fu. A fuzzing framework based on symbolic execution and combinatorial testing. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 2076–2080. IEEE, 2013.

[141] James Joseph Yuill et al. Defensive computer-security deception operations: Processes, principles and techniques. 2007.

[142] Ozgur Yurekten and Mehmet Demirci. Sdn-based cyber defense: A survey. *Future Generation Computer Systems*, 115:126–149, 2021.

[143] Li Zhou. A survey on contextual multi-armed bandits. *arXiv preprint arXiv:1508.03326*, 2015.

[144] Mu Zhu, Ahmed H Anwar, Zelin Wan, Jin-Hee Cho, Charles A Kamhoua, and Munindar P Singh. A survey of defensive deception: Approaches using game theory and machine learning. *IEEE Communications Surveys & Tutorials*, 23(4):2460–2493, 2021.

[145] Quanyan Zhu. Game theory for cyber deception: a tutorial. In *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, pages 1–3, 2019.

7.1 Publications

1. Mohammad Sujan Miah, Palvi Aggarwal, Marcus Gutierrez, Omkar Thakoor, Yinuo Du, Oscar Veliz, Kuldeep Singh, Christopher Kiekintveld, and Cleotilde Gonzalez. Diversifying deception: Game-theoretic models for two-sided deception and initial human studies. In *Cyber Deception: Techniques, Strategies, and Human Aspects*, pages 1–23. Springer, 2022

2. Palvi Aggarwal, Marcus Gutierrez, Christopher D Kiekintveld, Branislav Bošanskỳ, and Cleotilde Gonzalez. Evaluating adaptive deception strategies for cyber defense with human adversaries. *Game Theory and Machine Learning for Cyber Security*, pages 77–96, 2021

3. Anjon Basak, Charles A Kamhoua, Sridhar Venkatesan, Marcus Gutierrez, Ahmed H Anwar, and Christopher D Kiekintveld. Scalable algorithms for identifying stealthy attackers in a game-theoretic framework using deception. *Game Theory and Machine Learning for Cyber Security*, pages 47–61, 2021

4. Shelby R Curtis, Anjon Basak, Jessica R Carre, Branislav Bošanskỳ, Jakub Černỳ, Noam Ben-Asher, Marcus Gutierrez, Daniel N Jones, and Christopher Kiekintveld. The dark triad and strategic resource control in a competitive computer game. *Personality and Individual Differences*, 168:110343, 2021

5. Mohammad Sujan Miah, Marcus Gutierrez, Oscar Veliz, Omkar Thakoor, and Christopher Kiekintveld. Concealing cyber-decoys using two-sided feature deception games. In *HICSS*, pages 1–10, 2020

6. Marcus Gutierrez and Christopher Kiekintveld. Online learning methods for controlling dynamic cyber deception strategies. *Adaptive autonomous secure cyber systems*, pages 231–251, 2020

7. Marcus Gutierrez, Jakub Cernỳ, and Noam Ben-Asher. Evaluating models of human behavior in an adversarial multi-armed bandit problem

8. Anjon Basak, Charles Kamhoua, Sridhar Venkatesan, Marcus Gutierrez, Ahmed H Anwar, and Christopher Kiekintveld. Identifying stealthy attackers in a game theoretic framework using deception. In *Decision and Game Theory for Security: 10th International Conference, GameSec 2019, Stockholm, Sweden, October 30–November 1, 2019, Proceedings 10*, pages 21–32. Springer, 2019

9. Marcus Gutierrez, Jakub Cernỳ, and Noam Ben-Asher. Evaluating models of human behavior in an adversarial multi-armed bandit problem

10. Anjon Basak, Jakub Černỳ, Marcus Gutierrez, Shelby Curtis, Charles Kamhoua, Daniel Jones, Branislav Bošanskỳ, and Christopher Kiekintveld. An initial study of targeted personality models in the flipit game. In *Decision and Game Theory for Security: 9th International Conference, GameSec 2018, Seattle, WA, USA, October 29–31, 2018, Proceedings*, pages 623–636. Springer, 2018

11. Anjon Basak, Marcus Gutierrez, and Christopher Kiekintveld. Algorithms for subgame abstraction with applications to cyber defense. In *Decision and Game Theory for Security: 9th International Conference, GameSec 2018, Seattle, WA, USA, October 29–31, 2018, Proceedings 9*, pages 556–568. Springer, 2018

12. Marcus Paul Gutierrez and Christopher Kiekintveld. Adapting honeypot configurations to detect evolving exploits. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017

13. Marcus Paul Gutierrez and Christopher Kiekintveld. Bandits for cybersecurity: Adaptive intrusion detection using honeypots. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016

14. Oscar Veliz, Marcus Gutierrez, and Christopher Kiekintveld. Teaching automated strategic reasoning using capstone tournaments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016

Curriculum Vitae

Marcus P. Gutierrez is a Ph.D. graduate in Computer Science from the College of Engineering at The University of Texas at El Paso (UTEP). With a specialization in reinforcement learning and cyber deception, Marcus has made numerous contributions to cybersecurity by applying the Multi-Armed Bandit framework towards cyber deception.

During his doctoral studies, Marcus focused on detecting complex cyber attacks using decoys with online reinforcement learning. Under the guidance of Dr. Christopher Kiekintveld, he developed expertise in modeling adversarial network security games. Marcus designed and developed simulation systems using Java and Python, effectively utilizing the multi-armed bandit reinforcement learning framework to create robust defender solutions.

In addition to his research, Marcus also served as a Teaching Assistant for Computer Architecture. During the pandemic, he successfully transitioned the course to an online format, supporting approximately 30 students. Marcus took an active role in curriculum development, specifically creating five laboratory exercises that facilitated the practical application of teaching the RISC-V instruction set architecture. His dedication to enhancing the educational experience of students was evident in his contributions to the course.

Marcus's research has been recognized through presentations at prestigious workshops and conferences, including the International Workshop on Optimization in Multiagent Systems (OptMAS) and the Artificial Intelligence for Cyber Security (AICS) workshop. These platforms provided him with valuable opportunities to share his findings and engage with experts in the cybersecurity community.

Beyond his doctoral research, Marcus actively contributed to various projects. As a lead developer and game theoretic analyst for the Honeypot Behavioral Learning Project, he collaborated with researchers from Carnegie Mellon University and Czech Technical University. Marcus played a key role in developing a web application to study human behavioral learning in adversarial settings, resulting in valuable insights into cognitive and behavioral models.

Marcus's dedication to academic excellence and research is evident through his outstanding achievements. He received the CyberCorps®: Scholarship For Service program award, which reflects his commitment to cybersecurity education and practice. As a Ph.D. graduate, Marcus P. Gutierrez remains dedicated to advancing the field of cybersecurity. His passion for research, combined with his analytical and technical skills, positions him as a valuable contributor to addressing the evolving challenges of cyber threats and developing innovative defense strategies.