Game-Based Testing for Active Cyberdefense and Cyberdeception

Neil C. Rowe (contact)
U.S. Naval Postgraduate School
Monterey, CA 93943 USA
ncrowe@nps.edu

Justin J. Green
U.S. Naval Postgraduate School
Monterey, CA 93943 USA
justinjgreen87@gmail.com

Andrew S. Benn
U.S. Naval Postgraduate School
Monterey, CA 93943 USA
andrewbenn12@gmail.com

Sasha K. Drew
U.S. Naval Postgraduate School
Monterey, CA 93943 USA
drews24cyan@gmail.com

Charles W. Heinen
U.S. Naval Postgraduate School
Monterey, CA 93943 USA
heinencw1991@gmail.com

Robert E. Bixler
SoarTech, Inc., 3600 Green Ct.
Ann Arbor, MI 48105 USA
robert.bixler@soartech.com

Angela Tan
U.S. Naval Postgraduate School
Monterey, CA 93943 USA
atan9611@g.ucla.edu

Armon C. Barton
U.S. Naval Postgraduate School
Monterey, CA 93943 USA
armon.barton@nps.edu

Abstract—Active defenses are becoming popular for cybersecurity, offering more protections beyond access control and signature analysis. Cyberdeception is a particularly useful kind of active defense. We tested two tools for testing active defenses for simulated cyberattacks: a commercial tool CCAT from Soar Technology Inc., and a tool Decepgame we wrote. Both tools simulated a game of alternating moves between attacker and defender, where each possible move had costs and benefits. CCAT first plays random games to learn the best tactics for attacker and defender in a variety of situations, then tests on new situations; Decepgame uses reinforcement learning to incrementally improve its choices from experience. Experiments with CCAT then tested the best active defenses to six types of attacks with semi-random choices, and showed that the defense was effective. Experiments with Decepgame simulated attacks based on the MITRE ATT&CK taxonomy of cyberattacks, and showed gradual improvements over time without needing training runs. Both products showed realistic behavior but required configuration time, and their actions were often unsurprising.

Keywords—cybersecurity, wargames, attack, defense, deception, optimality, planning, testing, reinforcement learning, training

This paper appeared as a chapter in *Cybersecurity – Cyber Defense, Privacy, and Cyberwarfare*, De Gruyter, 2024, ISBN: 978-3-11-143641-8.

I. Introduction

Active defenses are becoming more common tools for improving cybersecurity. They include dynamic modification of systems to confuse cyberattacks as well as automated tracking of attacks and attackers. They also include deliberate defensive deception, for which many methods are available (Rowe and Rrushi, 2016). Deception has advantages of flexibility and unexpectedness. Most theories of ethics permit occasional deception to reduce harms, and damage to computer systems and theft of valuable data often are harms worse than that of occasional defensive deception. Deception has long been considered acceptable in warfare (Clark and Mitchell, 2019).

In the last ten years, commercial products have become available for active defenses, especially those with deception as an option. Some are not very good; one product we reviewed recently provided decoy processes that were easily detectable by their unusual traffic. Better products are available, but a question is how well they will fool attackers. Most cyberattacks are automated and do not pay much attention to their targets, so it is hard to deceive them. However, manual cyberattacks on high-value targets do occur, and the most important of these are "automated persistent threats" or APTs. These are state-sponsored "information operations" that slowly probe a system to find its weaknesses, then steal data or do sabotage. Deception can be especially effective in defense against them.

Wargames can help plan and analyze a broad range of military activities. Cybersecurity is like a war between attackers and defenders, and this is a particularly good analogy when state-sponsored specialists are involved. Cybersecurity wargames can model operations using procedures, data, and rules at far lower cost than exercises or red-teaming, and can handle situations with more options than possible for human comprehension. Surprises that occur in wargames prepare users better for real conflict.

This work evaluated two wargaming tools for planning of active defenses for local-area networks, both including cyberdeception methods. Parts of our work were previously reported in (Green et al., 2023). The tools are only planners and do not actively defend systems, but the planners are highly portable as software and attacks evolve.

II. Previous Work

A. Military cyberspace operations

Cyberspace is now a domain of warfare along with land, water, air, and space. It is a complex and highly connected environment that challenges traditional military notions of operations. Superiority in weapons and resources tends to be less important than strategy and tactics, benefitting a weaker adversary and supporting asymmetric warfare.

Warfare in cyberspace can be offensive or defensive (Joint Chiefs of Staff, 2018). The MITRE ATT&CK framework provides a good set of tactic types for both offense and defense (MITRE, 2024). Offense would often seem to have an advantage because defense must prepare for wide range of tactics. However, the number of effective offensive tactics is often limited because the most effective ones require exploiting flaws in software, and modern software is well-written with few flaws. A few general offensive techniques such as denial of service are well-known, and defenses can be preplanned for them. In many cases, offense in cyberwarfare also prefers attack targets based on military, economic, political, or religious goals that can be predicted in advance. Thus preplanning of defenses is often effective.

B. Active defenses in cyberspace and cyberdeception

Many military defenses are based on setting lines of fortification that impede an adversary attack. However, a straightforward application of this principle to cyber defense has been discovered to be limited in value because there are many ways to circumvent obstacles in cyberspace. Hence "active defenses" have become popular, where the defender seeks to provide a "moving target" that is harder for an adversary to attack because it changes over time. A popular moving-target defense is randomly changing the names and addresses of resources on a system often to prevent the frequent precompiled and automated attacks (Sun et al, 2019). Another example of an active defense is blocking network addresses providing unusual amounts of activity that could be denial-of-service attempts.

The work reported here also used a another kind of active defense, defensive deception in cyberspace ('cyberdeception"). Deception has a long history in warfare (Latimer, 2003). It is helpful for defense whenever adversaries have a strong advantage in surprise, and it works well when an adversary's knowledge of its target is limited. Both these conditions often apply in cyberspace (Rowe and Rrushi, 2016). Deception is justified by most ethical systems when the consequences of failing to deceive can be disastrous, as is often the case in warfare.

We can classify users as adversaries when they show enough suspicious behavior to exceed a threshold. Then we can deceive them using a wide variety of tactics. Lies, camouflage, and decoys are effective types of deception in cyberspace (Rowe and Rrushi, 2016). More specific examples of useful cyberdeceptions are concealment of high-value targets and promotion of low-value targets on a site, imposition of time-wasting delays and procedures on an attacker, and providing incorrect data to confuse attackers (Gartzke and Lindsay, 2016). Cyberdeceptions can give defenders more time to analyze and thwart attacks (Park & Kim, 2019), something very valuable for defense in warfare. Planning using methods of artificial intelligence such as deep learning can develop complex deceptions that are difficult to recognize (Matthew, 2020).

Lies, camouflage, and decoys are often implemented with honeypots, otherwise useless network nodes that can confuse attackers with false data, or entice and trap them into revealing their methods. Honeypots provide more useful data than regular

network monitoring since they have no other purpose than data collection. The research described here simulated both honeypots and false data.

C. Game analysis

Cyberwarfare can be modeled as a partial-knowledge probabilistic game in which an attacker and a defender alternate moves. Possible moves can be drawn from the MITRE ATT&CK taxonomy of offensive and defensive actions (MITRE, 2024). Then agents optimize their moves by looking ahead to the other agent's possible responses to their tactics, where attackers received points for achieving a set of goals and lost points for taking too much time; defender points were the negative of attacker points. Games were used to plan a defense using lightweight decoys while concealing the true targets in (Major et al., 2019).

Previous work has built a variety of attack models. One approach focuses on the effects of responses to cyberattacks (Cayirci and Ghergherehchi, 2011). Another approach visualizes cyberattack steps as a graph, though this requires manual analysis (Liu et al., 2012). Tests using costs and incentives can estimate numeric parameters of game models (Liu, Zang, and Yu, 2005). Artificial intelligence with deep learning can be used to refine offensive and defensive models (Najada, 2018), as can topological analysis of simulation outputs (Swarup, 2019). The Malicious Activity Simulation Tool is a scalable architecture for training with offensive models (Swiatocha, 2018).

Attack modeling should include simulated attacker exploration of a network to find launching points, or "lateral movement" (Bai et al., 2019). A defender can detect this and station useless decoys to receive attacks (Amin et al., 2020). Lateral movement can sometimes be predicted by exploring with random walks (Wilkens et al., 2019). Honeypots are useful for detecting lateral movement since users other than administrators have no legitimate reason to visit them.

Mixing real and fake data in network information can mislead adversaries (Jajodia et al., 2016). Game-theoretic analysis can also rate deceptions (Wang and Lu, 2018), attacker and defender tactics (Fang et al., 2018), and the degree of concealment of targets (Miah et al., 2020).

III. The CCAT Tool, An Active-Defense Planner

A weakness of much previous work is that it tested research prototypes of limited scope and robustness. Our work described here tested two tools with diverse sets of tactics.

We first tested a proprietary product from Soar Technology Inc. (SoarTech), the Cyberspace Course of Action Tool (CCAT) (SoarTech, 2017). It simulates cyber attacks and active defenses within a computer network, and learns the best tactics for both attacker and defender (the "agents") by running a game many times with random choices. In collaboration with SoarTech, we designed plausible scenarios, agent (player) actions, deceptions, and evaluation metrics. We tested eight scenarios including two controls. Further details beyond those below are in (Green et al., 2023).

A. Experiment design

The CCAT alternates moves of attackers and defenders in a simulated network with nodes and assets. A special kind of reinforcement learning was used to train agent behavior from results of games. Actions taken in games with better results were more likely to chosen in future games by the same agent. "Better" meant lower cost and higher benefits for themselves, as well as higher cost and lower benefits for their opponent.

In our experiments, we simulated a typical military network with mission-critical assets. The attacker goals were to obtain a file "PII.txt" and destroy a file "sysconfig.conf" file on the file server. PII.txt simulated a military roster with sensitive personal data; "sysconfig.conf" simulated a configuration file for a critical service. The defender goals were to delay the attacker as much as possible and increase their costs.

Choices made by the attacker and defender were based on a "double oracle" game model with deep reinforcement learning (Wright et al., 2019) over a series of games. Before testing, both attacker and defender agents did a million training runs for each experiment in which they chose random actions to rate tactics. The optimal strategy for an agent in this game is a Nash equilibrium; and the strategies closest to it for both attacker and defender were used in subsequent testing.

B. Scenario components and testing

Figure 1 shows our simulated test network. It has three subnetworks (DMZ, Servers, and Users) which contain commonly seen network devices. Servers in the front DMZ support Web, mail, and applications in the backend servers. The network also included simulated software (POR) servers, domain controllers, database servers, file servers, and a router for the application and mail servers.

We ran eight experiments in which the attacker and defender agents competed against each other.

- Experiment 1 was a control experiment without any deception.
- Experiment 2 used two decoy servers and two decoy files.
- Experiment 3 camouflaged the PII.txt and sysconfig.conf files by renaming them.
- Experiment 4 combined Experiments 2 and 3, and also allowed the defender additional decoy and camouflage.
- Experiment 5 was another control experiment without deception, testing how much the simulation software had been improved since the earlier experiments.

- Experiment 6 delayed the attacker by disabling links and attachments that were sent in a phishing email to make it ineffective.
- Experiment 7 staged a denial-of-service attack against one part of the network to distract defenders from attacks on local servers or Web pages. The defender could blacklist the address, mitigate the denial, or install additional sensors for the attacks.
- Experiment 8 had an attacker spoof a trusted network address in packets. The defender had to recognize the spoofing and block its traffic.

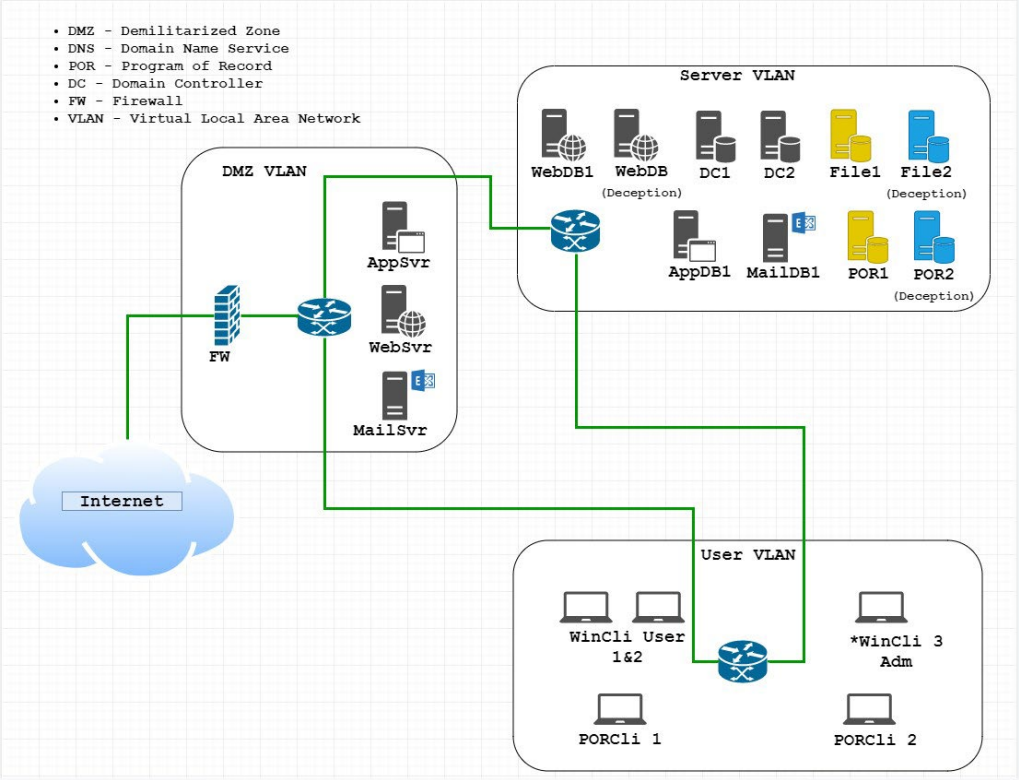

Figure 1: Map of the simulated network on which the CCAT game is played.

C. *Possible actions*

Figure 2 is the core of the defender flowchart, showing the response options based on the attack experienced. The flowchart was a collaborative effort between us and SoarTech using parts of the MITRE ATT&CK framework (MITRE, 2020). Possible defender actions are the squares, and "security conditions" are the circles, conditions that become true after a successful action. Security conditions visible to one agent are hidden from the other. The probability of a successful action by a defender depends on its complexity, its plausibility, and the sophistication of the attacker. Semi-random choices of actions are made based on training runs. "Semi-random" means that actions are rated by benefit to defender – cost to defender – benefit to attacker + cost to attacker, and then the larger the rating, the more likely the action is to be selected.

The defender tries to recognize attacker actions and thwart the attacker's objectives. At the start of the game, the defender can choose among general-purpose actions like file monitoring, process monitoring, and auditing to look for suspicious activity, actions that help the defender recognize suspicious events. If they find any, they can choose from follow-on actions like trying to stop the attack, stopping the exfiltration of data, or preventing a server being destroyed. Later actions include resetting a password, tightening access to servers, and blocking connections. They also include deceptions such as creating decoy servers and changing host names. Last resorts when a defender is in trouble are reimaging secondary storage and blocking all external traffic. Other options are "Enforce Robots.txt" to impede the attacker's Web crawling and "Block IP Addresses" to prevent internal and external traffic from reaching the attacker.

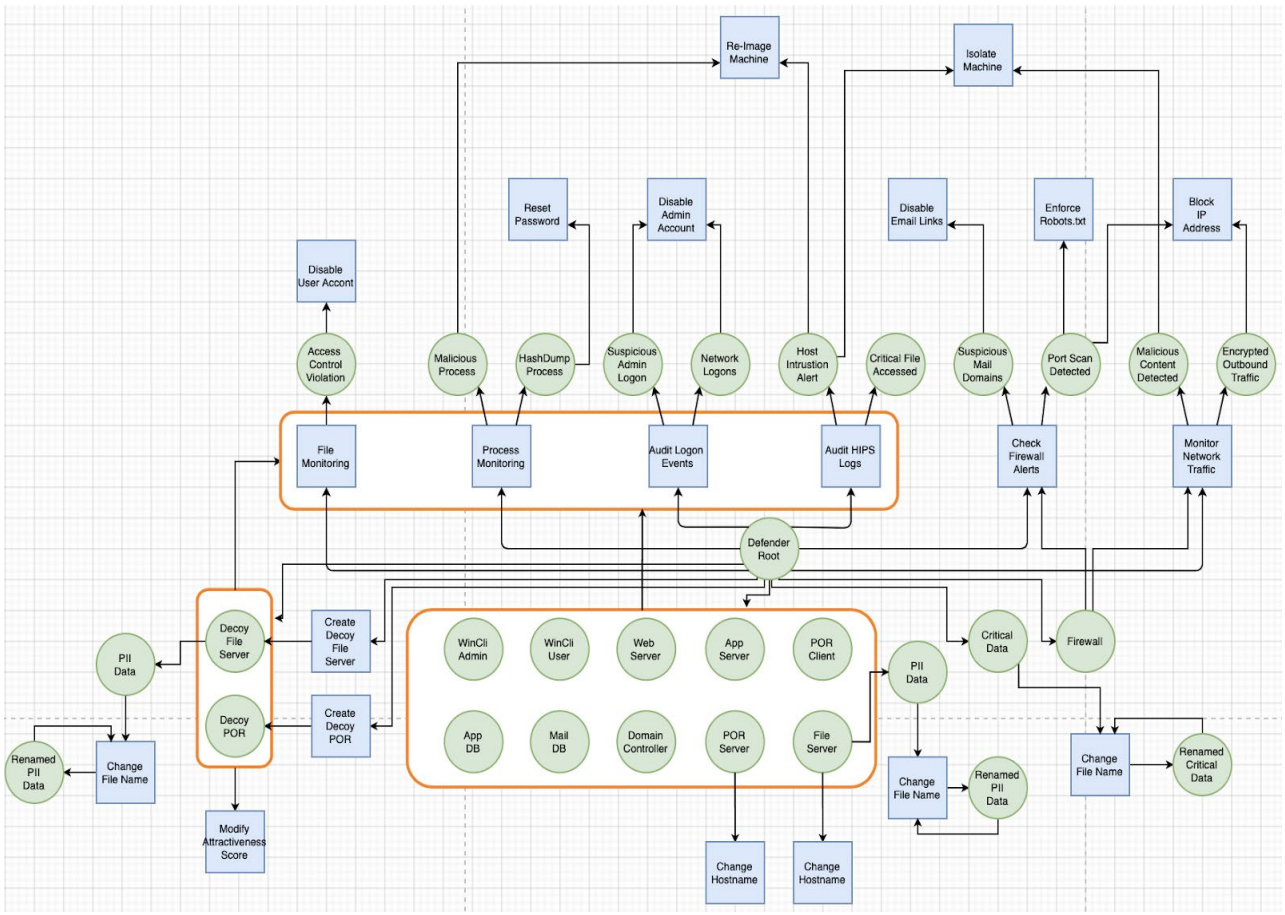

Figure 2: Dataflow diagram of the defender's main options in the CCAT game.

An attacker flowchart was also created to represent standard cyberattacks using tactics from the offensive taxonomy of the MITRE ATT&CK framework (MITRE, 2020). Because attackers have more options than defenders, it is more complex than the defender flowchart. Its middle part in shown in Figure 3. Note that it includes actions done both inside and outside the defender network. Rating of an attacker action is the benefit to attacker – cost to attacker – benefit to defender + cost to defender.

Attackers begin with preparations such as detecting decoys or maintaining IP addresses. Then they map the network, at least in part, and choose an attack based on the map. Options include spear phishing, exploiting servers, and collecting data from Web servers. If any of these actions succeed, a security condition is added to their knowledge. For example, if spear phishing succeeds, a security condition for a victimized user becomes available.

Next, the attacker can explore local networks, their users, their processes, and their local data. They can collect useful information like user names, personally identifiable information, and network connections, using password and key cracking methods as necessary. Subsequent options are privilege escalation and lateral movement, such as discovering processes, interfering with them, and search for secrets. The two possible final actions are exfiltrating data and destroying a server.

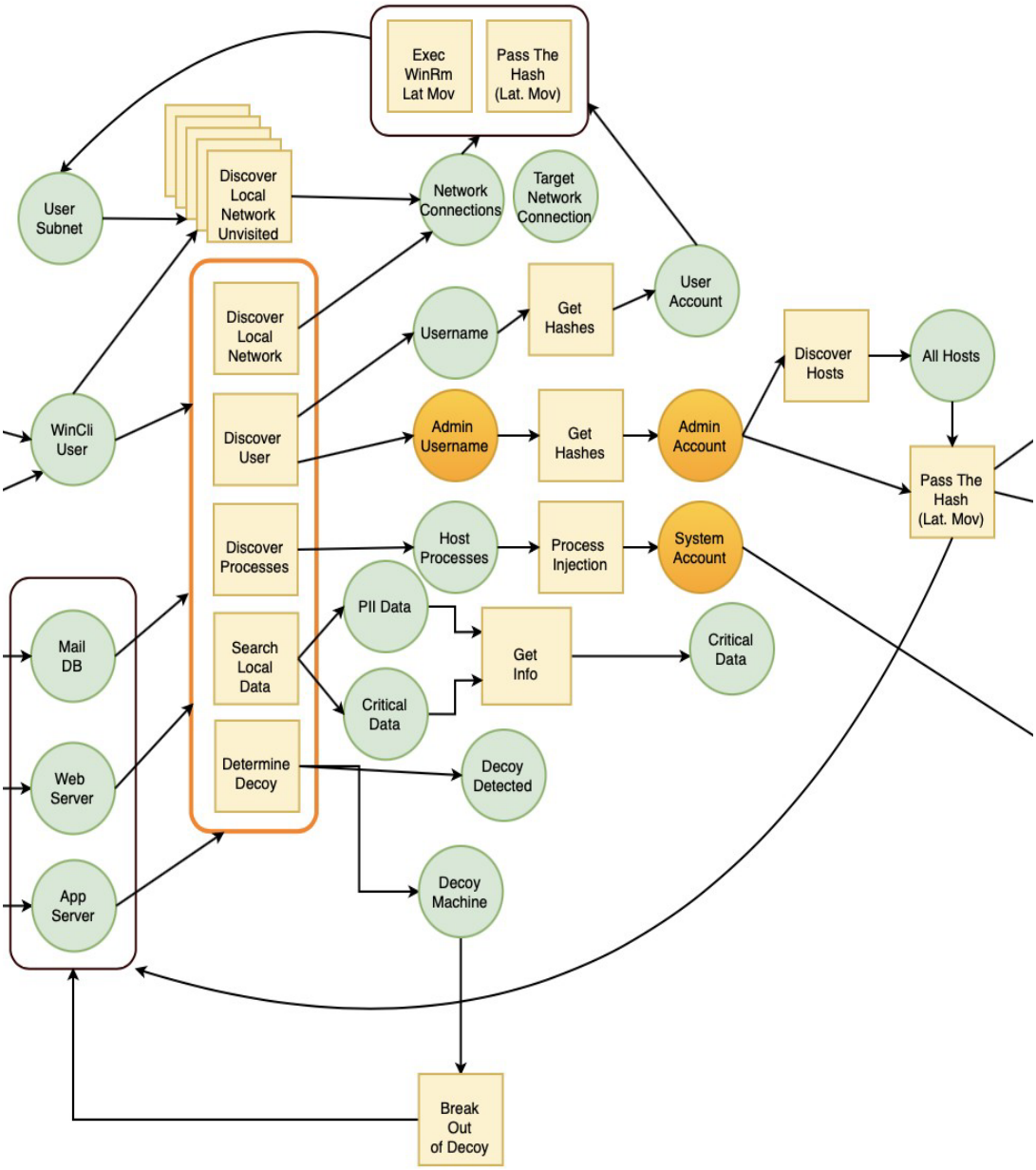

Figure 3: Dataflow diagram of the attacker's options in the CCAT game, middle part.

D. Costs and benefits of options

Costs (including "losses") and benefits ("rewards") are assigned to each action. They were carefully set with the help of SoarTech to give realism to the game. Costs are assigned to the number of steps (turns) required for an action and rewards are assigned to achievement of goals. The cost and reward table for the attacker is in Table 1.

Table 1: Costs and rewards for the attacker in the baseline CCAT game.

Attack action	Cost	Benefit	Attack action	Cost	Benefit
Connect to network	5	0	Map network	5	0
Send spear-phishing link	10	0	Send spear-phishing attachment	20	0
Send delayed-effect spear phishing	15	0	Exploit public-facing mail	15	0
Crawl Web pages	10	0	Exploit public-facing applications	15	0
Spoof internal IP decoy	15	0	Discover location network	10	0
Discover user	10	0	Discover processes	20	0
Search local data host	10	0	Get user hash	25	0
Get administrator hash	35	0	Inject process	35	0
Get information	20	0	Pass the hash	30	0
Execute lateral movement	20	0	Discover hosts	15	0
Search local data file server	10	0	Search local data critical server	15	0
Exfiltrate encrypted data	35	100-300	Exfiltrate command and control	25	100-300
Destroy critical service	25	900	Denial of service from decoy	50	1050
Corrupt critical service	75	1200			

Additional actions for specialized scenarios were also assigned. As an example, in the denial-of-service experiment 7 with actions shown in Table 2, the attacker will get a large benefit if they can gain critical-system access, but defender's benefit for stopping the first attack is low. A small benefit is given the defender when they stop the attack. However, denial of service impedes security activities by the defender, which adds to the defender's cost.

Table 2: Example defender and attacker actions (with defender cost) in a denial-of-service camouflaging attack in CCAT.

Defender action	Cost	Linked attacker action
Audit logon events	15	Execute lateral move, pass the hash
Monitor network traffic	25	Map network, exfiltrate data, spear phish, distributed denial of service, spoof internal IP
Check firewall alerts	15	Map network, exfiltrate data, spear phish, distributed denial of service, spoof internal IP, crawl Web pages
Audit HIPS logs	25	Critical service corruption, critical service destruction
Monitor files	20	Search local data
Monitor processes	30	Inject process, get hash
Isolate machine	60	Exfiltrate data
Block IP address	20	Map network, exfiltrate data, spear phish, spoof internal IP
Enforce robots.txt	20	Crawl Web pages
Disable email links	25	Give spear phishing link
Disable administrator account	50	Pass the hash
Disable user account	50	Execute lateral move, pass the hash, search local data, get hashes
Reset local password	20	Get hashes
Block denial of service	50	Distributed denial of service
Reimage machine	90	Inject process

E. Training the models

The CCAT agents are trained by playing random games, separately for each experiment. Training used a CentOS 7 core server with an i7-7820X processor at 3.60GHz, 8 cores, 128 gigabyte random access memory, and two Nvidia Tesla

V100 32 gigabyte graphical processing units. Training took around three weeks in parallel on a cloud service, and did a million runs each for attacker and defender in each experiment, for 16 million training runs altogether. The tactics whose results were closest to the Nash equilibrium were chosen for final testing.

F. Average final scores of the CCAT experiments

Table 3 shows average results on 100 runs of the CCAT game after training. A separate control experiment was done in experiment 5 since experiments 5-8 were done a year later than experiments 1-4. The more negative the score, the better the attacker did. Deception increased the attacker's costs compared to the control in all but the third experiment, and defender costs increased too but not as much as attacker costs. Therefore, we judge the deceptions a success except in experiment 3.

Table 3: Key CCAT results in scores and step counts.

Experiment	Average defender score	Average attacker score	Average attacker exfiltration steps	Average attacker corruption steps
1: First control, no deception	-3850	-1534	25.3	32.6
2: Decoys	-1427	-1861	26.3	33.5
3: Camouflage	-4158	-1658	25.1	32.8
4: Decoys and camouflage	-1475	-1803	25.4	32.9
5: Second control, no deception	-3944	-1551	-	25.7
6: Delays	-3636	-1682	-	26.0
7: Distributed denial of service	-3310	-1740	-	29.3
8: Spoofing	-3908	-1668	-	25.2

The maximum number of turns per player in a game was 40; this was reasonable, as most cybersecurity policies would prevent an attacker from exploring a system that long. We did not see any significant effects of deception on the average number of timesteps, but deception required different and more complex steps for the attacker than the defender. Most actions occurred with approximately the same frequency with deception, but exceptions were the decoy-related actions when decoys occurred and with the "Check File" action.

IV. DECEPGAME, A PLANNER AGAINST ADVANCED PERSISTENT THREATS

A weakness of the SoarTech approach is that it took much time to train the agents, as for example, three weeks on experiments 5-8. This appeared necessary to ensure we had sufficiently explored the combinations of options. However, this may be overkill for both attacker and defender; usually the best options in cyber operations are well known in advance (Rowe & Rrushi, 2016). Furthermore, an important principle of deception planning is to avoid wasting effort in designing details that deceivees will not notice, and with so many options, an attacker cannot notice many of them.

One way to reduce training time is to measure the rate of improvement of the best solution found. If a better solution is not found in a certain maximum time, we can assume we have likely found the best one. However, this varies considerably with the number of options. Another approach is to count the number of distinct plans that an attacker could follow, and estimate the predicted time to a likely best solution.

Another approach to use reinforcement learning to incrementally improve performance during game playing rather than in advance (Benn & Benn, 2023). This can incrementally improve choice probabilities based on the differences between outcomes and average outcomes and the degree of correlation of the action choice with the outcome. This does not require training runs, and can give useful results with only small numbers of runs against real attackers. It does not provide optimal solutions, but deception is a psychological effect that does not require high precision.

A. Game design

Tactics of advanced persistent threats (APTs) can be modeled for a multi-stage game (Zhu and Rass, 2018). Detection of an APT is a critical goal for a defender. APTs use so many tactics to remain stealthy and persistent that they are best identified by automated machine learning. APT actions can more easily be distinguished from benign traffic during reconnaissance and establishing a foothold. Otherwise, the interactions between the APT and defender provide clues to the APT's goals, tactics, techniques, and procedures.

We followed these ideas in constructing a program Decepgame in the programming language Python to do a version of the SoarTech implementation, with a kernel of 348 lines of code. Our game design has the standard components of players, actions for each player, preferences among actions for each player, and moves made sequentially by alternate players (Osborne, 2004). Our first experiments had an initial state of 30 facts, and an action core set of 66 attacker actions and 80 defender actions to which APT-specific actions were later added. Each action was defined by 9 parameters: player, action name, new state, new time, new total cost for player, new total benefit for player, success probability for the action, the previous benefit for the state, and the previous state; the last three were used when the action fails according to the success probability. Each player state was defined with six parameters: the player, the last action name, the current state, the current time, the current total player cost, and the current total player benefit.

Our players were a simulated APT attacker and a simulated cybersecurity group defending a local-area network. We encoded the actions for each player from APT tactics and possible defender actions listed in the MITRE ATT&CK enterprise framework (MITRE, 2023). Preferences for actions for each player were determined by action costs, benefits, probabilities of success, and duration required, which we specified. Cost is mainly determined by duration, but additional factors such as intellectual difficulty were also included.

1) Player generation

For realistic testing, we modeled a diverse group of APTs including APT39/Remix Kitten, Lazarus Group, MuddyWater, Sandworm Team, Turla, Wizard Spider, APT31/Zirconium, and a composite case APT X for generic exfiltration of data (Table 4).

Table 4: Advanced persistent threats (APTs) providing action examples for Decepgame.

APT	Suspected attribution	Goals	Targets	Preferred tactics
APT39/ Remix Kitten (Hawley et al., 2019)	Iran	Sensitive data exfiltration, political repression	Political targets, including foreign dissidents	Connecting to victim machines remotely for persistence and lateral movement, data exfiltration uses zip files, custom tools
Lazarus Group (Park, 2021)	North Korea	Data exfiltration, intelligence gathering, sabotage, financial gain	Financial institutions, government agencies, entertainment industry	Holding critical data hostage in exchange for Bitcoin ransom
MuddyWater (Avertium, 2022)	Iran	Intelligence gathering, financial gain, sensitive data exfiltration	Government and private sector defense, energy, government, and telecommunications industries	Using open-source tools
Sandworm Team (Cunningham, 2020)	Russia	Cyber sabotage	Critical infrastructure, particularly energy-related	Long-term persistence
Turla (Faou, 2019)	Russia	Intelligence gathering and cyber sabotage	Geopolitical adversaries, international organizations	Custom tools and malware, obscuring destination of exfiltrated data
Wizard Spider (DiMaggio, 2021)	Russia	Financial gain	Banking institutions	TrickBot Trojan ransomware, exploiting wake-on-LAN capability to spread
APT31/ Zirconium (Soesanto, 2021)	China	Intelligence gathering	High-level US and international community election campaign personnel	Collecting data about Web browsing (Fonseca et al., 2005), repurposing exploits from other APTs
APT X/Use Case	Generic	Sensitive data exfiltration	Cloud-service providers	Trusted third party compromise

Table 5 shows our estimates of attribute values of these APTs used in the simulation, based on historical observations:

- Resources: How well-resourced the APT is on a scale from 1-3 (1 = non-state, 2 = state-sponsored, 3 = state). Considerations are available tools, available skills, intelligence support, and financing.
- Stealth: How much the APT tried to remain undetected on a scale from 1-3 (1 = low concern, 2 = medium concern, 3 = high concern). Considerations are reputation, potential of retaliation, and possible sanctions and political fallout associated with being attributed.
- Preference weight: The APT's historical preference for their known attack methods in the preferred tactics set (1: no preference, 0.8: moderate preference, and 0.6: strong preference) obtained from the MITRE ATT&CK data. Attackers with strong preferences repeat the same techniques.

Table 5: APT attributes used in testing Decepgame.

Advanced persistent threat	Resources	Stealth	Preference weight
APT39/Remix Kitten (RK)	3	1	1
Lazarus Group (LG)	2	2	0.8
MuddyWater (MW)	2	3	0.6
Sandworm Team (SWT)	3	1	0.8
Turla (Tur)	2	2	0.6
Wizard Spider (WS)	1	2	0.6
APT31/Zirconium (Zirc)	2	3	0.8
APT X/Use Case (UC)	3	3	1

The defender could be a commercial or public entity such as a cloud-service provider, a military-network administrator, or a critical-infrastructure cybersecurity team. The defender attributes were:

- Resources available: How well-resourced a defender is on a scale from 1-3 (1 = few resources available for cybersecurity, 2 = medium amount of resources, 3 = many resources).
- Confidentiality, integrity, and availability importance: The defender's tolerance for risk for each information-security principle (Fenrich, 2008). For example, one organization may be concerned about data compromise (higher concern for confidentiality), whereas another may only be concerned about maintaining operations (higher concern for availability). We used values of 3, 2, or 1, with 3 being the least concern, and 1 being the highest concern. The multiplier applied for highest priority is 1.25, the second is 1.15, and the third is 1.
- Initial security state: The conditions before the game begins, specified as either hardening attributes or vulnerabilities. Hardening attributes represent defender preparation such as security policies and installed defense systems; examples are firewalls and strong passwords. Vulnerabilities are flaws in the target system such as an unobservant user and software bugs. Our experiments used four security states:
    - Very low security: 10 vulnerabilities, 0 hardening
    - Low security: 10 vulnerabilities, 6 hardening
    - Medium security: 5 vulnerabilities, 13 hardening
    - High security: 0 vulnerabilities, 19 hardening

2) Action profiles and player specification generation

For our experiments, 65 action profiles were generated for the attacker and 78 for the defender; additional defender profiles were modified to fit our framework. We primarily focused on techniques in which the attacker seeks technical instead of social information about a system, because it can be difficult to convincingly model the latter. Each action profile had these attributes:

- Phase. The APT lifecycle phase in the DAPT2020 dataset in which an action is done. The dataset distinguishes 14 phases of an attack, reflecting larger-scale planning than with the CCAT focus on tactical planning. We chose the four earliest stages to study, since APTs are easier to detect then.
- Action Name. The action in either the MITRE ATT&CK Framework v13 (MITRE, 2023), in a repository of cybersecurity attacks and countermeasures (Kaloroumakis & Smith, 2021), in the Nmap Reference Guide (Nmap Project, n.d.), in a survey on adversarial reconnaissance techniques (Roy et al., 2022), or in a review of attack vectors and countermeasures (Ullah et al., 2018). If actions had different names in the sources, we generalized them.
- Action Class. These were:

  - Attacker, Phase 1 (reconnaissance): scanning, sniffing/spoofing/observing, host-based reconnaissance, third-party reconnaissance, and human-based reconnaissance.
  - Attacker, Phase 2 (foothold establishment): system-based actions, human-based actions, and execution.
  - Attacker, Phase 3 (lateral movement): evasion, privilege escalation, and obtaining persistence.
  - Attacker, Phase 4 (data exfiltration): active and passive.
  - Defender, all phases: moving-target defense, cyber deception, deception, and hardening.
- Cost. The cost to execute an action was estimated based on its technical sophistication (S), resources available to a player (R), the "noise level" of an action based on observations (N), a player's concern for remaining stealthy (C), and a weight on the action. Noise level is defined as the degree of generation of statistical anomalies or artifacts that indicate compromise to a defender (Hare & Diehl, 2020). For attacker action, the weight was the preference-weight attribute representing an APT's observed historical preference for it, or for a defender, a cost adjustment for the action class (hardening and detection were given a 1, moving-target defenses a 2, and cyber deception a 3). The defender weights for hardening and detection were assumed to follow an organization security policy and therefore have low costs; cyber deception required credibility and interaction to deceive adversaries. Putting it together:

  $\text{Cost} = \left(\left(\frac{S_{action}}{R}\right) + (N * C)\right) * W$ where $N_{defender} = 0$, $C_{defender} = 0$, $W_{hardening} = 1$, $W_{detection} = 1$, $W_{mtd} = 2$, and $W_{deception} = 3$.
- Probability of success. Values were high (default 95%), medium (45%), low (15%), or very low (1%). Complex actions or those requiring defender negligence had lower probabilities of success.
- Benefit. This is the value of successful completion of the action. For example, a high benefit was assigned to an attacker successfully exfiltrating data, and to a defender successfully luring an APT to a honeypot to study it.
- Duration. Part of the cost of an action is proportional to its duration because we want to delay attacks as much as possible. We weighted duration by 0.5 and added the result to the cost.
- Preconditions. These are necessary conditions for an action to occur, expressed as facts.
- Negative preconditions. These are conditions that must be false for an action to occur.
- Postconditions. These conditions become true (if not already) after a successful action performance. Postconditions can be negative to represent facts that become false after the action. For our model, benefits on a scale from 1-5 were tied to postconditions.
- Support goal. These were the general purpose of an action for the player, which for attackers were discovery, network observation, user redirection, cryptosystem analysis, vulnerability enumeration, footprinting, obtaining credentials, initial compromise, execution of a payload, evading of defenses, obtaining of persistence, data exfiltration, and cleanup. Defender support goals were denying, delaying, deterring (the three categories of active cyber defense), and detecting.

Visualizations like Figure 4 were produced to help understand possible paths between phases of the game.

B. Implementation of the APT game

The player profiles were inputs to the game-modeling program Decepgame. It runs games with the attacker and defender profiles from a specified initial state. Other parameters in the game program that can be set are:

- Maximum number of moves. This is the number of moves played per game by alternating players with the attacker starting. Values used were from 50-300 moves per game.
- Number of games. This is the number of games that the attacker and defender play, updating the reinforcement on actions after each game. For our experiments, this was set to 100.
- Mean (m) and slope (s) of the sigmoid logistic function used for calculating reinforcement. This maps from an average score over games using a particular action to the probability of selecting that action in a future game.

With variations in priority ordering of confidentiality, integrity, and availability, and the initial environment security states of high, medium, low, and very low, 72 kinds of defenders are possible, and we tested 13 representative combinations, labeled as A-M. Games were played for each defender against each attacker specification in Table 4, while keeping the sigmoid parameters fixed. We varied the number of maximum moves per game from 50 to 300 with increments of 5. We then looked for indications that reinforcement learning helped increase defender performance across the 13 kinds of defenders.

Decepgame scores each action over the series of games, and chooses actions having better scores with higher probabilities. Higher scores are better for the defender, and lower scores are better for the attacker. Decepgame applies a logistic function $f(x, m, c) = 1/(1 + e^{-c(x-m)})$ to scores for each action; m and c were set to the average score and its standard deviation over all occurrences of actions. The logistic function provides weights on selection of actions, for all actions that satisfy the preconditions in the current game state, and then Decepgame randomly chooses an action for both attacker and defender using these weighted likelihoods by normalizing them to probabilities. This acts as reinforcement, since actions that lead to good outcomes are increasingly chosen over time.

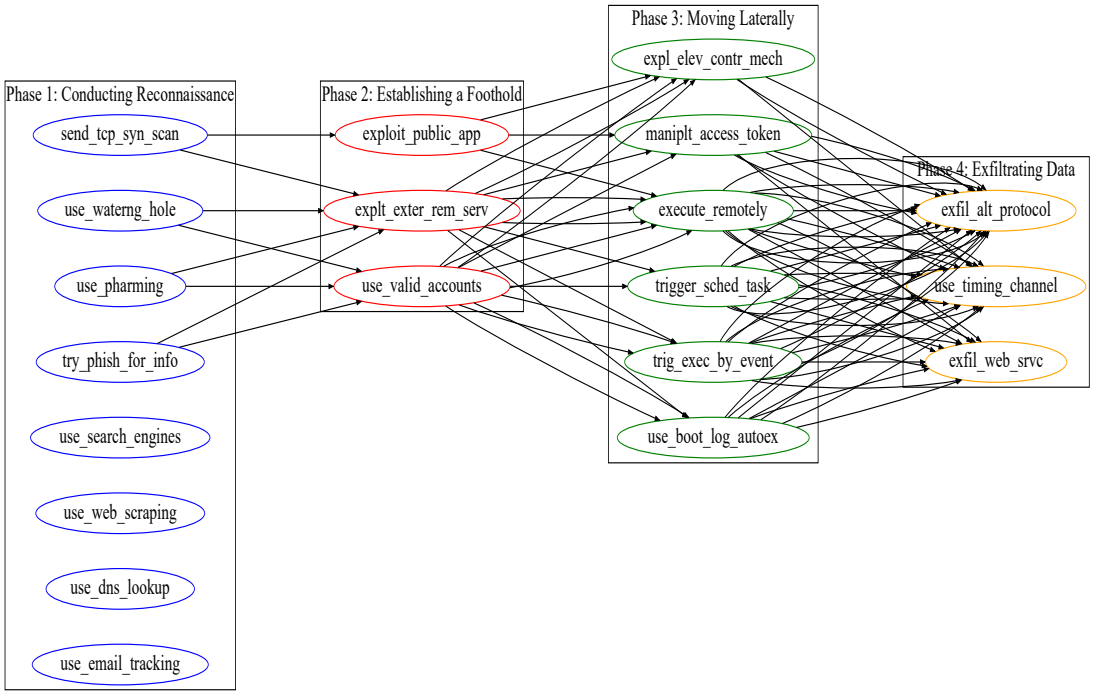

Figure 4: Partial subset of APT X's attack profile, represented as a directed graph of its preferred actions over the first four phases.

Probabilistic selection is important so that the players do not become too predictable, but the actions observed as better in previous games should be selected more often. Actions may fail with a specified probability, in which case they incur costs but no benefits. The final score for a game is defined by an evaluation function which was net cost to the

attacker minus net cost to the defender, where net cost is defined as the cost of all the actions minus all the benefits obtained.

C. *Results of the Decepgame APT simulation*

5,304 tests were done of 104 attacker-defender pairs. Each test played 100 games, with 50-300 allowed turns per game in increments of 5 turns. Statistics were collected for the average final scores (Table 6). Column headings are the APT abbreviations used in Table 4. In general, the final scores initially increased with number of allowed turns as learning occurred, but a maximum score average occurred where the attacker started to overcome the defenses. Statistics were also collected for the average number of turns it took to achieve this maximum score (Table 7).

Table 6: Average final scores for APT/Defender pairs in Decepgame.

APT / Defender	RK	LG	MW	SWT	Tur	WS	Zirc	UC	Def Avg Score
A	-2.47	32.73	153.72	-29.93	18.98	83.26	173.11	189.63	77.38
B	41.63	149.51	279.75	38.28	114.08	179.95	287.05	236.57	165.85
C	58.16	194.13	247.86	2.70	156.54	174.21	301.05	296.47	178.89
D	3.22	38.69	113.30	-28.80	20.63	85.48	170.22	117.33	65.01
E	44.95	144.51	247.56	43.79	100.09	169.47	232.07	235.78	152.28
F	20.58	136.68	257.82	29.80	110.03	188.66	301.35	292.14	167.13
G	-60.32	14.86	126.63	-37.10	4.23	53.85	109.07	82.98	36.78
H	6.60	94.04	204.70	-38.81	72.91	87.96	269.60	190.17	110.90
I	52.45	100.36	293.36	33.03	81.97	137.06	338.82	213.17	156.28
J	-55.20	30.24	239.55	-34.18	63.09	162.08	347.47	331.48	135.57
K	18.50	36.03	326.91	-9.01	94.36	41.14	446.78	260.50	151.90
L	-34.28	93.12	240.82	-0.17	45.04	56.54	195.06	209.92	100.76
M	26.30	-6.70	54.90	-55.15	26.40	56.21	118.34	137.17	44.68
Average score	9.24	81.40	214.38	-6.58	69.87	113.53	253.08	214.87	

Table 7: Average number of turns in Decepgame at which the defender maximizes their score over 5304 tests, with 100 games 50-300 turns per game, incremented by 5.

APT / Defender	RK	LG	MW	SWT	Tur	WS	Zirc	UC	Avg Turn
A	290	230	255	130	285	295	280	275	255
B	235	285	230	120	265	280	280	180	234
C	235	265	300	145	250	285	285	275	255
D	260	240	290	215	280	220	280	125	239
E	250	275	275	240	275	270	250	260	262
F	290	250	215	230	260	235	280	300	258
G	270	225	150	180	230	140	215	250	208
H	260	230	300	275	285	245	290	255	268
I	255	225	295	175	270	145	295	245	238
J	255	285	300	195	280	250	295	300	270
K	245	235	295	195	240	250	285	280	253
L	275	285	300	240	180	220	295	295	261
M	275	135	250	205	285	270	270	285	247

Figures 12 and 13 plot final scores for two defenders with different initial security states and resources, playing against the MuddyWater APT. Figure 5 shows games for this APT against defender F, a medium-security defender with high resources. The horizontal axis is the average number of turns per game over 100 games, and the vertical axis indicates final scores of the games. Darker dots indicate positive final game scores. The trend line is a third-degree polynomial. Attacker-defender interactions with a higher security state environment generally follow the trendline in

this figure, with final scores increasing as the defender learned how to counter the attack, but having a maximum when the attacker learning started to compensate.

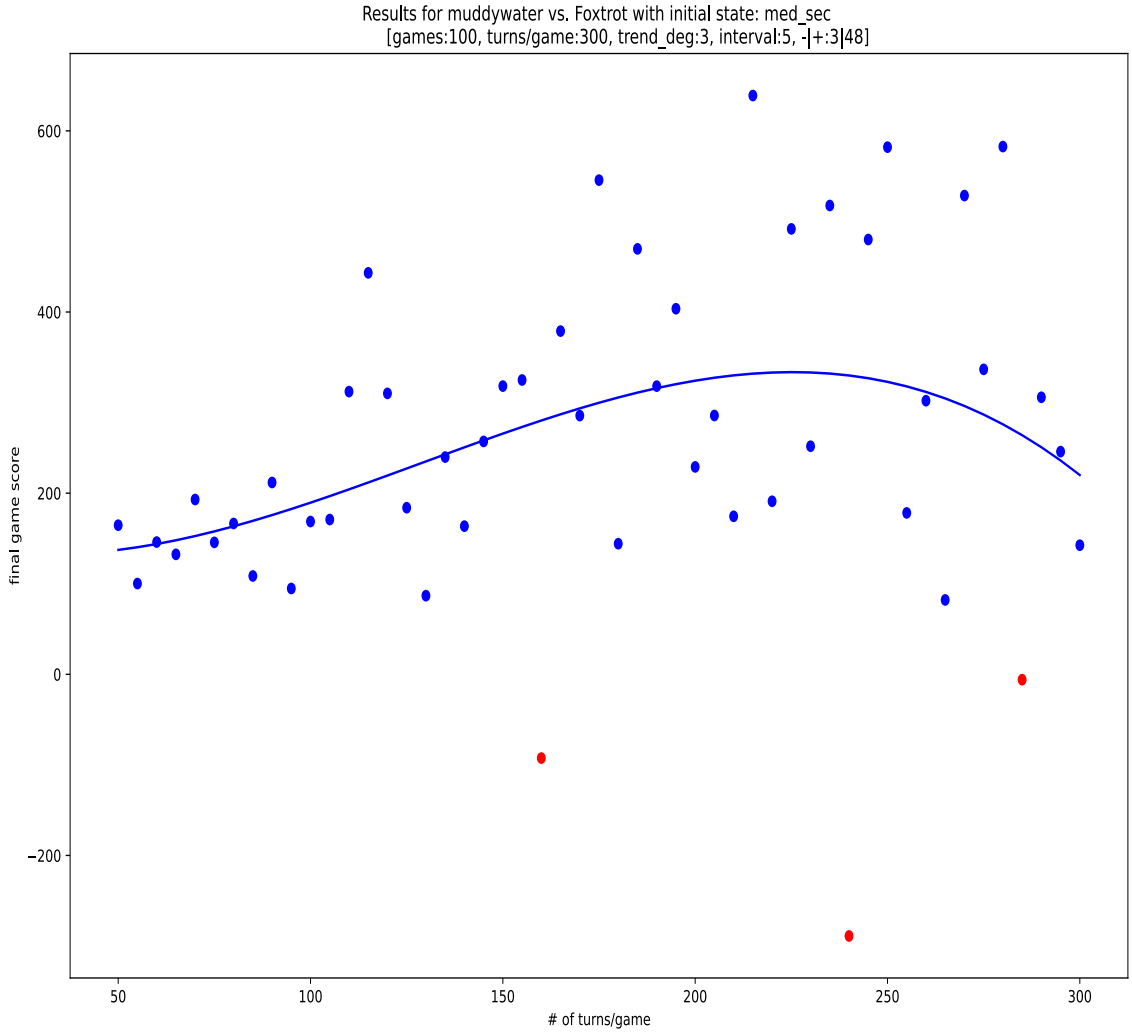

Figure 5: Data summary for tests of the MuddyWater APT versus defender F in Decepgame.

Figure 6 summarizes the attacker-defender interactions of the MuddyWater APT versus the M defender, a very-low security defender with the least resources. Results in a lower-security environment generally follow this trendline, where the maximum defender score occurs early, with a drop as games lengthened.

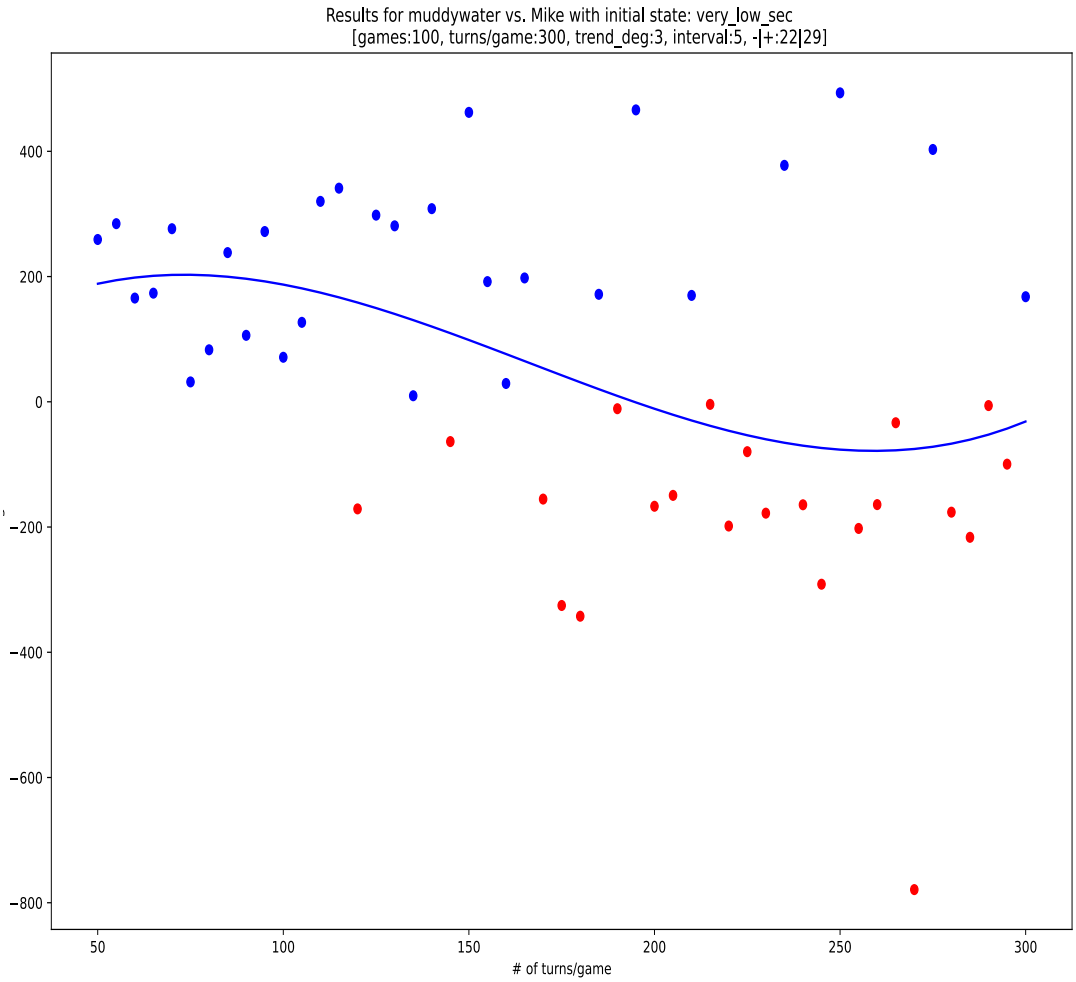

Figure 6: Data summary for tests of the MuddyWater APT versus the M defender in Decepgame.

Statistics were also collected on the defender actions overall across all games by phase (Table 8) over 5,304 tests with 8 attackers, 13 defenders, and 100 games per test, 50-300 turns per game. The phases were reconnaissance, establishing a foothold, and lateral movement. Some actions scored well relative to other actions in the same phase but did not score well when compared to actions in other phases. The game sequence likely affected this, as the later phases are not reached as often as early phases, so more data was collected about actions in earlier phases.

Table 8: Highest-scoring and lowest-scoring defender actions in Decepgame runs.

Action	Action Class	Action Score	Phase
Deploy honeynet	Deception	30817.7	1
Use high interaction client honeypots	Deception	29459.2	2
Create decoy public release	Deception	29190.8	1
Deploy honeytokens	Deception	29117.4	2
Create decoy file	Deception	29077.9	1
Use shadow honeypots	Deception	28980	3
Harden system configuration permanently	Hardening	28905.1	3
Create decoy website	Deception	28903.3	1
Check exception handling pointer	Deception	28799.1	2
Create decoy session credential	Deception	28770.3	1
Isolate kernel process	Hardening	5002.72	2
Do authorization event thresholding	Hardening	4944.9	1
Use biometric authentication	Hardening	4899.64	1
Do authentication event thresholding	Hardening	4825.74	1
Do process segmentation execution prevention	Hardening	4794.01	2

D. Discussion of the Decepgame results

Reinforcement learning improved defender scores, since defender scores improved as turns increased. The attacker learned as well, as the attacker sometimes minimized the game score in the second third of the interaction (after 150-250 turns). Both the attacker and defender could improve as the games lengthened, but the scale of these improvements was unpredictable.

The average total scores showed that some APTs were easier to defend against based on their resource and stealth levels (Table 9). Attacker resource levels also have a statistically significant effect on the final total game scores (Table 10). This reflects the differences in sponsorship associated with these actors, as those that were state-sponsored typically performed better than others. We also observed that the attacker's concern for stealth had a statistically significant effect on the final total game scores (Table 11). However, APT X is an exception. Although it has a high resource value assignment, it also has a high concern-for-stealth value. This imposed a higher cost when the attacker took less-stealthy actions, increasing the defending player's final score. This was seen in other results, with players assigned a higher concern-for-stealth value incurring higher costs, which gave lower final scores.

Table 9: Average scores for particular kinds of APTs in Decepgame.

APT test group from Table 4	APT Group Attribute	Average Final Score
RK, ST, UC	Resource level = 3	72.51
LG, MW, Tur, Zirc	Resource level = 2	154.68
WS	Resource level = 1	113.53
MW, Zirc, UC	Stealth level = 3	227.44
LG, Tur, WS	Stealth level = 2	88.27
RK, ST	Stealth level = 1	1.33

Table 10: Effect of the APT resources attribute in Decepgame.

APT groups compared by resources attribute	P-value in a T-test of significance	Average final scores
Low resources: WS Med resources: LG, MW, TUR, ZIRC	0.00783	Low: 113.5 Med: 154.7
Med resources: LG, MW, TUR, ZIRC High resources: RK, SWT, UC	0.03120	Med 154.7 High: 72.5
High resources: RK, SWT, UC Low resources: WS	0.00072	High: 72.5 Low: 113.5

Table 11: Effect of the APT stealth attribute in Decepgame.

APT Groups Compared by Stealth Attribute	P-value	Avg Final Scores
Low stealth: RK, SWT Med stealth: LG, TUR, WS	2.39E-07	Low: 1.3 Med: 88.3
Med stealth: LG, TUR, WS High stealth: MW, ZIRC, UC	7.80E-15	Med: 88.3 High: 227.4
High stealth: MW, ZIRC, UC Low stealth: RK, SWT	4.07E-08	High: 227.4 Low: 1.3

Across all 5,304 games, those with turn limits greater than 200 offered the best opportunities for the defender to maximize their score against the attacker. Therefore, lengthening the game helped the defender, but only up to a point. As to real time, each action during a turn has varying associated duration (e.g. a security policy can be implemented quickly, but a virtualized network requires much setup time).

Despite lacking a direct conversion to real time, lengthening the game's turns helps defend against APTs since APTs tend to persist even when they know they are engaging with an active defender. The benefit has risks though, as the score difference between attacker and defender becomes wider as the games progress; while the defender may perform well in these games, this added length affords attackers the opportunity to perform their most unusual actions in later phases, actions for which the defenders are not prepared. In general, we saw that an attacker with sufficient turns and time will eventually outperform the defender.

Defender attributes showed inconsistent trends. Defenders with high resource attribute values did not take fewer turns to maximize their score. The confidentiality-integrity-availability ordering had no statistically significant effect on average number of turns to maximize defender score.

The defender had 23 deception, 7 detection, 41 hardening, and 7 moving-target actions available in the game (Table 12). The top 10 highest-scoring defensive actions included 9 deception actions (29.5% of the total actions), 1 hardening action (52.5% of the total actions), no moving-target defenses (9% of the total actions), and no detection actions (9% of the total actions), so deception was very effective in this simulation over many varied situations. The table also shows that the highest-scoring defensive techniques varied by phase. We saw it was most effective for the defender to deceive early in the APT lifecycle, particularly during reconnaissance, and that active cyber defense is less effective later in the lifecycle when an APT is already inside a target network. Phases 3 and 4 did best with hardening techniques, consistent with the defense-in-depth strategy (Mughal, 2018).

Table 12: Classes of the highest-scoring defensive technique by phase in Decepgame.

Phase	Cyber deception	Hardening	Detection	Moving-target defense
Reconnaissance	100%	0%	0%	0%
Establishing a foothold	40%	20%	20%	20%
Lateral movement	20%	80%	0%	0%
Data exfiltration	0%	100%	0%	0%

When comparing defenders of varying resource levels without varying other parameters, defenders with more resources scored better against APT attackers as expected (Figure 7). This confirms that increased spending on cybersecurity can reduce cyber risks (Asen et al., 2019).

Figure 7: Average score versus resources available in Decepgame.

When comparing defenders of differing priorities of confidentiality, integrity, and availability, the order confidentiality-integrity-availability had an average final score of 141.9, integrity-confidentiality-availability had an score of 121.3, and availability-integrity-confidentiality had a score of 87.2.. This supports the well-known result that a high emphasis on keeping systems available often conflicts with security since systems must be available to be attacked. This reflects a common dilemma in cybersecurity as priorities on confidentiality, integrity, and availability can conflict with those of business operations. The initial defender security state also affected the final score as expected. The average final score was 171.9 with a High initial security state, 126.5 with Medium, 101.3 with a Low, and 108.2 with a Very Low.

V. Precalculating Defender Tactics from Attacker Variables

The Decepgame experiments still required some time, although they were considerably faster than the experiments with the SOAR CCAT. Our current work is exploring a way to reduce game analysis further for the defender by precomputing best strategies in generic situations. If a defensive situation can be summarized with a limited set of parameters, we can analyze it in advance and cache its results (perhaps parameterized) for guidance in similar situations in the future. Decision trees are a good way to summarize these analyses. Figure 8 shows an example for a network that permits downloading, with parameters defined in Table 13. The parameters can be estimated by reinforcement learning from experience with attackers. Then the defender should select the leaf node with the highest value of its formula based on the parameters. More detailed analysis of similar applications to the defense of industrial control systems is in (Rowe, 2024).

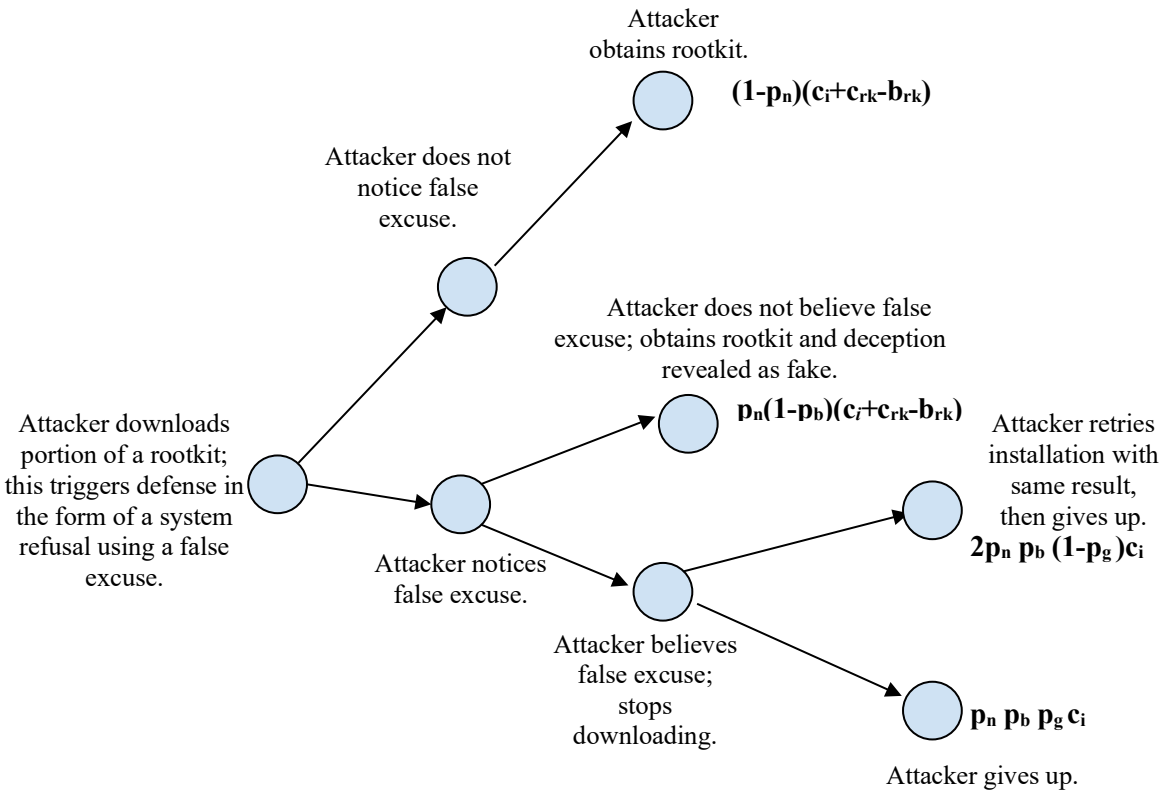

Figure 8: Example generic decision tree for considering the option for the defense of offering false excuse of "file corrupted" for failing to complete a download.

Table 13: Definitions of parameters in Figure 8.

Variable	Description
p_n	Probability that the attacker notices the false excuse
p_b	Probability that the attacker believes the false excuse
p_l	Probability that the attacker logs off
p_r	Probability that the attacker retries the installation
p_g	Probability that the attacker gives up and disconnects.
c_i	Initial duration cost of downloading part of the rootkit
c_{rk}	Duration cost of downloading the rest of the rootkit
b_{rk}	Benefit to the attacker of obtaining the full rootkit

VI. Conclusions

Two sets of experiments reported here, on the CCAT and Decepgame game simulations, suggest that active and deceptive tactics in the cyberspace domain can be systematically planned in a cost-effective way. The CCAT tool did considerable pre-planning of tactics and could find subtle interactions of tactics that enabled better defenses; the Decepgame tool avoided preplanning in favor of reinforcement learning, but could better address new tactics never encountered before. The tactics found for both games can increase the time to mount an attack and its likelihood of success. Wargaming worked well as a technique to explore and rate deception options. The tactics found by both sides were nonetheless predictable in our experiments due to the limited number of options, and this could be exploited by an adversary.

ACKNOWLEDGEMENTS

This work was supported by the U.S. Defense Intelligence Agency and the U.S. Department of Energy. Statements are those of the authors and do not represent the U.S. Government.

REFERENCES

Asen, A., Bohmayr, W., Deutscher, S., González, M., & Mkrtchian, D. (2019). Are you spending enough on cybersecurity?. https://www.bcg.com/publications/2019/are-you-spending-enough-cybersecurity.

Avertium (2022, April 13). An in-depth look at Iranian APT "MuddyWater". https://explore.avertium.com/resource/in-depth-look-at-iranian-apt-muddywater

Amin, M. A. R. A., Shetty, S., Njilla, L. L., Tosh, D. K., & Kamhoua, C. A. (2020). Dynamic cyber deception using partially observable Monte-Carlo planning framework. In Modeling and Design of Secure Internet of Things (pp. 331–355). IEEE. https://doi.org/10.1002/9781119593386.ch14

Bai, T., Bian, H., Daya, A., Salahuddin, M. A., Limam, N., & Boutaba, R. (2019). A machine learning approach for RDP-based lateral movement detection. 2019 IEEE 44th Conference on Local Computer Networks (LCN), 242–245. https://doi.org/10.1109/LCN44214.2019.8990853

Benn, A. & Benn, S. (2023, September). Application of game theory for active cyber defense against advanced persistent threats. M.S. thesis, Naval Postgraduate School.

Cayirci, E., & Ghergherehchi, R. (2011, December). Modeling cyber attacks and their effects on decision process. In Proceedings of the 2011 Winter Simulation Conference (WSC) (pp. 2627–2636). IEEE.

Clark, R., & Mitchell, W. (2019). Deception: Counterdeception and counterintelligence. Thousand Oaks, CA, US: Sage.

Cunningham, C. (2020). A Russian Federation information warfare primer. Henry M. Jackson School of International Studies. Washington University.

Drew, S. and Heinen, C. (2022, March). Testing deception with a commercial tool simulating cyberspace, M.S. thesis in Cyber Systems and Operations.

DiMaggio, J. (2021). Ransom mafia. Analysis of the world's first ransomware cartel. Analyst white paper.

Faou, M. (2019). Turla lightneuron: One email away from remote code execution. ESET Research white paper.

Fenrich, K. (2008). Securing your control system: the "CIA triad" is a widely used benchmark for evaluating information system security effectiveness. Power Engineering, 112(2), 44+. https://link.gale.com/apps/doc/A177028777/AONE?u=anon~e527e3a2&sid=googleScholar&xid=8cc52b63

Fonseca, F., Pinto, R., & Meira, W. (2005, October). Increasing user's privacy control through flexible web bug detection. In Third Latin American Web Congress (pp. 8-pp). IEEE.

Gartzke, E., & Lindsay, J. R. (2015). Weaving tangled webs: Offense, defense, and deception in cyberspace. Security Studies, 24(2), 316–348. https://doi.org/10.1080/09636412.2015.1038188

Green, J. (2020). The fifth masquerade: An integration experiment of military deception theory and the emergent cyber domain [Master's thesis, Naval Postgraduate School]. NPS Archive: Calhoun. http://hdl.handle.net/10945/66078

Green, J., Drew, S., Heinen, C., Bixler, R., Rowe, N., & Barton, A. (2023, July). Evaluating a planning product for active cyberdefense and cyberdeception. In Proceedings of the 22nd International Conference on Security and Management, Las Vegas, NV, US.

Hare, F., & Diehl, W. (2020). Noisy operations on the silent battlefield: Preparing for adversary use of unintrusive precision cyber weapons. The Cyber Defense Review, 5(1), 153–168. https://www.jstor.org/stable/26902668

Hawley, S., Read, B., Brafman-Kittner, C., Fraser, N., Thompson, A., Rozhansky, Y., & Yashar, S. (2019, January 29). APT39: An Iranian cyber espionage group focused on personal information. https://www.mandiant.com/resources/blog/apt39-iranian-cyber-espionage-group-focused-on-personal-information1

Jajodia, S., Subrahmanian, V., Swarup, V., & Wang, C. (2016). Cyber deception (Vol. 6). Springer.

Joint Chiefs of Staff (2017). Military deception. Publication JP 3-13.4. https://jdeis.js.mil/jdeis/new_pubs/jp3_13_4.pdf

Joint Chiefs of Staff (2018). Cyberspace operations. Publication JP 3-12. https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp3_12.pdf

Kaloroumakis, P., & Smith, M. (2021). Toward a knowledge graph of cybersecurity countermeasures. The MITRE Corporation.

Latimer, J. (2003). Deception in war: Art bluff value deceit most thrilling episodes cunning mil hist from the trojan. Abrams.

Liu, P., Zang, W., & Yu, M. (2005). Incentive-based modeling and inference of attacker intent, objectives, and strategies. ACM Transactions on Information and System Security (TISSEC), 8(1), 78–118.

Liu, Z., Li, S., He, J., Xie, D., & Deng, Z. (2012, December). Complex network security analysis based on attack graph model. In

2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control (pp. 183–186). IEEE.

Major, M., Fugate, S., Mauger, J., & Ferguson-Walter, K. (2019). Creating cyber deception games. 2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI), Los Angeles, CA, USA 102–111. https://doi: 10.1109/CogMI48466.2019.00023.

Matthew, A.(2020). Automation in cyber-deception evaluation with deep learning. www.researchgate.net/publication/ 340061861_Automation_in_ Cyber-deception_Evaluation_with_Deep_Learning.

Miah, M. S., Gutierrez, M., Veliz, O., Thakoor, O., & Kiekintveld, C. (2020, January). Concealing cyber-decoys using two-sided feature deception games. Hawaii Intl. Conf. on Systems Sciences.

MITRE (2024) MITRE ATT&CK. https://attack.mitre.org/. Retrieved December 1, 2024.

Mughal, A. (2018). The art of cybersecurity: Defense in depth strategy for robust protection. International Journal of Intelligent Automation and Computing, 1(1), 1-20.

Najada, H. A., Mahgoub, I., & Mohammed, I. (2018). Cyber intrusion prediction and taxonomy system using deep learning and distributed big data processing. 2018 IEEE Symposium Series on Computational Intelligence (SSCI), 631–638. https://doi.org/10.1109/SSCI.2018.8628685

Nmap Project. (n.d.). Nmap reference guide (7.92 ed.). https://nmap.org/book/

Osborne, M. (2004). An introduction to game theory (Vol. 3, No. 3). Oxford University Press.

Park, J. (2021). The Lazarus group: The cybercrime syndicate financing the North Korean state. Harvard International Review, 42(2), 34-39.

Park, Y., & Stolfo, S. (2012). Software decoys for insider threat. Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, 93–94.

Rowe, N. C., & Rrushi, J. (2016). Introduction to cyberdeception. Berlin: Springer International Publishing.

Rowe, N. (2024). Designing deceptions for protecting industrial-control systems. Chapter in book of Springer Nature, in press.

Roy, S., Sharmin, N., Acosta, J., Kiekintveld, C., & Laszka, A. (2022). Survey and taxonomy of adversarial reconnaissance techniques. ACM Computing Surveys, 55(6), 1–38. https://doi.org/10.1145/3538704

SoarTech Inc. (2017). https://soartech.com/cyberspace/.

Soesanto, S. (2021). The 19th of July: Divided or united in cyberspace? From the EU and NATO to Five Eyes and Japan. Royal Institute of Spain.

Sun, J., Liu, S., & Sun, K. (2019). A scalable high fidelity decoy framework against sophisticated cyberattacks. Proceedings of the 6th ACM Workshop on Moving Target Defense, 37–46. https://doi.org/10.1145/3338468.3356826

Swarup, S., & Rezazadegan, R. (2019). Generating an agent taxonomy using topological data analysis. Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2204–2205.

Swiatocha, T. (2018). Attack graphs for modeling and simulating sophisticated cyberattacks [Thesis, Monterey, CA; Naval Postgraduate School]. https://calhoun.nps.edu/handle/10945/59599

Ullah, F., Edwards, M., Ramdhany, R., Chitchyan, R., Babar, M., & Rashid, A. (2018). Data exfiltration: A review of external attack vectors and countermeasures. Journal of Network and Computer Applications, 101, 18–54. https://doi.org/10.1016/j.jnca.2017.10.016

Wang, C., & Lu, Z. (2018). Cyber deception: Overview and the road ahead. IEEE Security & Privacy, 16(2), 80–85.

Wilkens, F., Haas, S., Kaaser, D., Kling, P., & Fischer, M. (2019). Towards efficient reconstruction of attacker lateral movement. Proceedings of the 14th International Conference on Availability, Reliability and Security, 1–9.

Wright, M., Wang, Y., & Wellman, M. (2019, June). Iterated deep reinforcement learning in games: History-aware training for improved stability. In EC (pp. 617– 636).

Zhu, Q., & Rass, S. (2018). On multi-phase and multi-stage game-theoretic modeling of advanced persistent threats. IEEE Access, 6, 13958–13971. https://doi.org/10.1109/access.2018.2814481