Challenges and Solutions in Using Virtual Testbeds to Study Hacker Cognitive Constraints

Sean Guarino, David Kelle, Curt Wu

Charles River Analytics Inc.
Cambridge, MA
{sguarino, dkelle, cwu}@cra.com

Max Slocum, Michael Sieffert

Assured Information Security
Rome, NY
{slocumm, sieffertm}@ainfosec.com

K. Raghav Bhat, Robert Gutzwiller
Arizona State University
Mesa, AZ
{kbhat4, rgutzwil}@asu.edu

Michelle Neisser
SimSpace Corporation
Boston, MA
michelle.neisser@simspace.com

ABSTRACT

Gaining a better understanding of the cognitive constraints of cyberattackers can drive the development of effective proactive cyber defenses. Studying hacker behavior, however, requires the development of highly realistic and well-instrumented cyber testbeds. Here, we describe key simulation challenges that we experienced while developing testbeds to execute five human subjects studies analyzing the ability to exploit cognitive biases of cyber adversaries, and the solutions we developed to address these problems. Specific challenges addressed include the unique experimental requirements for testbed configuration and management across numerous human subjects, requirements for generating content and variation that support realism and avoid training effects in within-subject studies, requirements to ensure participants can complete studies in the time constraints provided, and requirements to effectively instrument the testbed to better understand hacker behaviors. For example, one area of challenge that we confronted was supporting a trade-off between the need for an open-world, realistic cyberattack context, and the need to guide attackers to complete specific tasks so that we could collect pertinent behavior and performance data. Addressing these challenges allowed us to streamline our development process, to more rapidly construct testbeds for our later and future studies. Specifically, we describe the following: (1) how we configured sets of testbeds to ensure experimental consistency across subjects as they executed each trial twice; (2) how we used large language models (LLMs) to generate files and other content within each testbed, populating the testbeds with realistic, differentiated artifacts rather than using existing, recognizable datasets; (3) how we used a variety of methods to manage the time and progress of attackers across a range of skill levels; and (4) our development of a data collection pipeline to enable the analysis of both behavioral indicators of susceptibility and outcome-based performance metrics.

ABOUT THE AUTHORS

Sean Guarino is a Principal Scientist and Director of Training Technology at Charles River Analytics. His research interests include cognitive modeling, intelligent adaptive training, and immersive training technologies. Mr. Guarino has been leading efforts investigating proactive cyber defense, with a focus on exploiting human attackers, for more than 10 years. Mr. Guarino is Principal Investigator for Charles River's work in this effort, and he has led the design and execution of the studies our team has executed. He holds an MS in Psychology from Harvard University and a BS in Computer Science from Cornell University.

David Kelle, Scientist, led the development of the experimental testbeds in this effort. His research interests include cybersecurity for critical infrastructure. He has an extensive background in researching cyberattack tools. Mr. Kelle received his BS from the University of Oklahoma, an MS from The Florida State University, and is finishing his PhD at Northeastern University.

Curt Wu, Chief Engineer, led the engineering team. Mr. Wu received an MS in Computer Science from Boston University, an MS in Mechanical Engineering from Stanford University, and a BS in Mechanical Engineering from the Massachusetts Institute of Technology.

Max Slocum is a Senior Software Engineer at Assured Information Security specializing in red-team operations and tool development. With a unique blend of development expertise and adversarial thinking, his most recent focus has been on increasing the degree of automation used by cyber defenders to identify and mitigate vulnerabilities.

Michael Sieffert, Chief Engineer at Assured Information Security, leads several ongoing research and development efforts focused broadly on cybersecurity. His recent research related to this area of study has examined the application of deception in cyber contexts to benefit cyber defense. He has a history of low-level reverse engineering and development of stealthy execution and covert communications, and he enjoys looking at the problem of cyberattacks from new perspectives.

Raghav Bhat is a PhD student in Human Systems Engineering at Arizona State University, where he studies decision-making, adversarial cognition, and human-AI teaming in cybersecurity. His research has contributed to several Department of Defense (DoD)–funded academic projects focused on operator performance and human-systems integration. He has been managing the execution of each of the studies in this work.

Robert S. Gutzwiller, Associate Professor of Human Systems Engineering at Arizona State University, obtained his PhD in cognitive psychology in 2014 from Colorado State University. He drove pioneering research in oppositional human factors, exploring the exploitation of cyberattacker cognitive biases to reduce their performance. He has guided the design of the studies we have been running in this work. He is also an avid cyclist, racing both on and off the road.

Michelle Neisser is VP of Customer Success at SimSpace, the leading global cyber range provider. She began her career as a software developer before transitioning into R&D, Professional Services, and Customer Success/Support leadership roles for enterprise grade networking and security software products.

Challenges and Solutions in Using Virtual Testbeds to Study Hacker Cognitive Constraints

Sean Guarino, David Kelle, Curt Wu
Charles River Analytics Inc.
Cambridge, MA
{sguarino, dkelle, cwu}@cra.com

Max Slocum, Michael Sieffert
Assured Information Security
Rome, NY
{slocumm, sieffertm}@ainfosec.com

K. Raghav Bhat, Robert Gutzwiller
Arizona State University
Mesa, AZ
{kbhat4, rgutzwil}@asu.edu

Michelle Neisser
SimSpace Corporation
Boston, MA
michelle.neisser@simspace.com

INTRODUCTION

Cyberattackers often evade detection and achieve nefarious goals by exploiting human cognitive vulnerabilities, such as designing phishing attempts to exploit users' preexisting beliefs (effectively leveraging confirmation bias; Nickerson, 1998; Berthet et al., 2024). However, outside of recent, limited cyber psychology research (Ferguson-Walter et al., 2021a; Ferguson-Walter et al., 2021b; Gutzwiller et al., 2018; Gutzwiller et al., 2019; Gutzwiller et al., 2023), cyber defense rarely takes advantage of the attacker's cognitive vulnerabilities, missing opportunities to delay or prevent attackers from reaching their goals. A big part of the reason cyber defenders do not exploit these cognitive vulnerabilities is because there is only a limited understanding of how they might impact an attacker executing different components of the cyberattack chain. While there has been a host of psychological research investigating a range of cognitive biases in laboratory settings (Pohl, 2022; Gigerenzer, 2003; Krämer, 2014), little work has been done to understand how these biases might be exploited in operational cyber defense contexts.

A key challenge in researching how cognitive biases and vulnerabilities impact cyberattackers is finding an effective experimental setting for this work. Using actual network environments to study attacker behavior cannot scale effectively to run empirical research on attacker populations, where it is essential to observe the behavior of numerous attackers across both control and treatment conditions. One option is to study behavior in capture the flag (CTF) exercises; however, these exercises fail to measure the cognitive behavior and performance of attackers in a useful way to understanding how cognitive constraints impact behaviors (Ferguson-Walter et al., 2019). The performance measures they do gather tend to be focused on the success or failure of the exercise, failing to capture the more subtle but similarly important indicators of time wasted, cognitive effort, detectability, or even good measures of progress toward goals. Furthermore, CTF exercises tend to be focused on particular aspects of the attack chain that do not necessarily reflect real-world cyberattacks, with a focus on specific, known avenues of attack, and a setting in which participants incur little to no risk. Many existing cyber testbed environments provide a potential setting for performing scalable human subjects research (HSR), but currently they are not designed to streamline HSR processes or collect relevant behavioral data. Instead, existing tools are designed to support training and rehearsal exercises, often focused on training and understanding cyber defender behaviors, with limited insight into what attackers are doing. Even those testbeds that support cyberattacker training are designed for focused exercise execution and training assessment, not for cognitive manipulation and assessment of the attackers.

In our current work, we have adapted and applied a high-fidelity cyber testbed designed originally to support training—the SimSpace Cyber Range Platform—to execute a series of HSR studies investigating, measuring, and manipulating the cognitive vulnerabilities of cyberattackers. In doing so, we encountered and overcame several challenges to adapt the testbed environment to support the unique requirements of executing high-fidelity HSR with expert hacker communities. These challenges include range configuration for consistent experiences across participants and experimental conditions, content generation suitable for necessary experimental fidelity, guidance tools to ensure that participants progress in the attack to the point needed to test empirical impacts, and data collection challenges for empirical analysis of attacker behaviors. A key factor driving our solutions was the need to effectively leverage a limited population of expert hackers; because of the difficulty in recruiting these populations, we needed all participants to be effective in our studies, to produce the behavioral data we needed to collect, and to do so in a within-subjects study in which they had to perform each task in both control and treatment conditions.

Here, we describe our work in this area. First, we describe the requirements that hacker HSR creates for cyber testbeds and how those requirements compare with the state of the art in cyber testbed environments. We then describe how we have addressed specific challenges when adapting SimSpace to support hacker HSR needs, followed by some results and impacts of our work in the ongoing effort. Finally, we present conclusions addressing how to adapt cyber testbeds to support HSR, as well as future recommendations for how to improve empirical outcomes in these environments for future studies.

BACKGROUND

High-fidelity cyber ranges, also known as cyber testbeds, are primarily used for training cybersecurity personnel. Typical use cases include individual training and assessments, classroom-based training, team training events including team assessments and blue/red competitive exercises, and individual competitions such as Capture-The-Flag events. These types of training activities can be supported with a low-to-moderate level of realism in the test environment. Cyber testbeds are also suitable for research and development activities that are risky to conduct in production environments. This can include threat analysis and development of responses including detection engineering and mitigation techniques. These use cases require a higher level of realism in the testbed than typical training activities.

The specialized use case of studying hacker behavior requires a sophisticated and realistic cyber testbed environment, but it poses a different type of realism challenge. In addition to presenting a high-fidelity network environment with a realistic topology and security tools, the testbed must be seamlessly interactive with the hacker in a way that leads him to behave in a similar way as he would in a production network. Because of these needs, there are key differences that must be addressed to effectively leverage these testbeds, many of which are driven by differences in scenario objectives and in target audiences. We highlight several of these differences.

First, like training, HSR has a goal to achieve a consistent experience across the audience; for training, this is done to ensure that all trainees can achieve the same learning, whereas for HSR this is done to control the environment and ensure that measured effects on behavior are related to the intended manipulations. However, while training exercises can, in most cases, carefully control the approaches that the target audience should use, this is less controllable in a study of hackers, where participants coming with varied levels of experience may pursue different paths to exploit the system. Furthermore, while training exercises can be repeated, in HSR, if a participant fails on a first attempt, the participant cannot attempt again, as the exposure can impact study results. This is particularly disruptive, as expert hacker participants are difficult to recruit. This creates a significant challenge in designing scenarios that support consistent first-time experiences across the research participants to assure they complete the study, execute desired behavior, experience targeted manipulations, and are measured in a way that their resulting behavior can be directly linked to those manipulations.

Second, HSR studying cyberattackers brings some significant content needs to ensure that the experience fits the study objectives. While there is a need for realistic user content in both training and HSR applications, the specific needs differ based on the objectives of the testbed. In training applications, we might care about differences in user or traffic data that enable differentiation of target and noise behavior (e.g., allowing a trainee attacker to rapidly pick out which traffic belongs to actual defenders and which is autogenerated). In HSR, the focus is on providing realism around the manipulations that are being made, so that those manipulations are not automatically differentiated from everything else. In addition, in a within-subject, repeated study environment, where participants will need to execute many sessions, this content generation must be differentiated enough to prevent transfer or training effects across sessions; each session needs to be a unique experience, with minimal or no impact from previous sessions.

Third, related to the previous item, both training and HSR applications have a vested interest in ensuring participants can progress in the exercise. In training, this is often done with scenario stages, moving trainees on to a new stage when ready to progress. Ultimately, there is inherent control in training applications, where there are clear training objectives, and the exercise can be scheduled in terms of those objectives. This is less clear, however, in HSR, where the goal is to assess attacker behavior in a realistic attack setting—which often means the attacker having significant freedom in choosing how to attack. Stages in the scenario can lead to discontinuity for the attacker. Nevertheless, attackers will come in with widely varying skill levels, preferences, and capabilities, and it is essential to be able to move participants through the scenario to the parts where we need them to act to effectively measure behavioral outcomes related to the HSR objectives. In the context of our studies, this often meant we needed them to rapidly get to an early attack stage where we were measuring behavioral indicators of bias vulnerability, and then get to a later attack stage where we were (or were not, in control conditions) applying a cognitive manipulation to measure its impact on behavior.

Finally, while data collection is essential in both training and HSR contexts, the collection needs and methods can be widely different. In training, collection is oriented around assessing performance according to training metrics (e.g., did they achieve specific training objectives). These are often targeted objectives that can be measured in straightforward ways (e.g., did a defender prevent an attacker from gaining access to specific data or did an attacker gain such access). Furthermore, external sources—such as white cell and/or observer-controllers—are often used to collect specific details describing trainee performance. In HSR studying attacker cognition and behavior, however, we need to assess the attacker's behavior and approach, not just the final outcome, often with limited direct insight into what the attacker is doing or their motivation and reasoning for doing it. Furthermore, cognitive cues that are collected across a host of behavioral data need to be interpreted and analyzed to understand their relationships to human cognition. To rapidly execute these studies, this collection needs to be done automatically, without a human proctor able to observe detailed behavior. Testbeds do not generally have built-in tools to collect behavioral data on attackers, creating a significant need for enhancing data collection to support HSR.

CHALLENGE SOLUTIONS

Configuring for Consistency

A staple of experimentation is orientation around an independent variable to determine if and to what degree manipulation of that variable has effects on different, possibly several, dependent variables. Typically, this manifests in controlling, or holding steady, as much of the environment as possible and varying only whether subjects are exposed to the independent variable or not. When studying attacker behavior, this can go beyond the environment itself, to the overall experience of the attacker (e.g., ensuring that the attacker effectively arrives at the point of manipulation). In our experiments, the independent variable is exposure to a certain intervention in the cyber testbed that is intended to exploit attacker cognitive vulnerabilities. In any given run, a member of the population may or may not be exposed to the intervention (in our studies, we were using a within-subject approach, so each member of the population experienced one version of the scenario where they were exposed to an intervention and another where they were not). In our experiments, this was a simple dichotomy, meaning the treatment was either present or it was not, and there was no additional scale or degree to which the treatment was presented.

In our experiments, we introduce the treatment (or remove the treatment, for the control condition) at distinct points in the expected, or nominal, attack sequence. When aggregated, our experiments target several steps along the attacker kill chain, which cover early preparatory steps taken by an attacker, such as reconnaissance and weaponization, to latter stages such as establishing persistence and exfiltrating documents. In fact, even within individual experiments, in some cases, we introduce the treatment at different points in that sequence. It is our belief that if and when operationalized, cyber defenders will look to target all stages of the attack rather than maintaining a narrow aperture on activity post compromise.

The treatment in each experiment is closely tied to specific cognitive biases being evaluated for their tendency to evince or exploit a cognitive vulnerability. As an example, for the base rate neglect bias, in which we humans tend to neglect the base rate of some occurrence or event, the treatment condition is the presence of an extremely dated and out-of-place vulnerability found on an otherwise modern and professionally administered system. The same system in the control configuration did not have this particular vulnerability. In both configurations, the system also had a vulnerability that was more appropriate in terms of recency and expectedness. The presence of this latter vulnerability enabled the complete attack progression regardless of configuration.

A key factor in our specific approach to HSR was the use of a within-subject study design, where each participant had to experience both a control and treatment condition for each cognitive bias. We could not simply have the participant execute the same scenario twice, because they would learn from the first experience, and that would impact performance on the second. We therefore focused on developing parallel versions of each scenario that focused on producing similar, but different, experiences across the entire attack chain, removing all information that could be used directly from one scenario to the other, while keeping the fundamental behavioral measurements the same in each version of the scenario. Then, for each version of our scenario, we generated a control and a treatment condition, for a total of four versions of each scenario. To support consistency in this development process within the testbed environment, we did the majority of the scenario development in a first version of the scenario, completing all scenario refinement and debugging before cloning it and refining it into the other three versions, where we introduced the specific situational differences that enabled repetition.

To assist in configuration and deployment, the team used a combination of methods. We depended on existing integrations of Puppet assisted in the initial deployment of the systems, but elected to use a custom Python actuation tool for many scenario-related tasks. Throughout scenario development there were often many tasks that required more dynamic execution throughout three primary stages: scenario execution, range cleanup, and range deployment.

Due to overlap between these actions, we reused this actuation capability to regularly perform tasks including during scenario execution. Some tasks that may occur during scenarios included collecting network and datetime information, collecting file hashes to identify changes, and even acting as an offensive hunt team to "detect" attackers within the network. Furthermore, results of a given task may be output in JSON and aggregated to a Splunk HTTP Event Collector (HEC), where results are later queried during the data collection process. Alternatively, prior to deploying scenarios we needed to clean up development artifacts, which include tasks such as clearing the Splunk index, clearing Windows Event Collector (WEC) logs, clearing Security Onion (SO) logs, removing browser profiles, or even wiping a user's ".ssh" directory to ensure hosts appear as new and novel discoveries to attackers. Finally in our third use case, deploying the range, we used various capabilities to assist in placing files on hosts, configuring local user accounts, enabling SSH, and more.

The actuation tool used SSH to move and execute bash scripts against Linux hosts while also using WMI and PowerShell scripts against Windows-based hosts. For large file movement to Windows hosts, we further transferred files using an HTTP server instance, which hosted the files for WMI to pull from. We did this instead of executing large base54-encoded WMI commands due to the enforced length limit in command execution. To configure the tool, we used a JSON file structure that indicated the host, credentials, scripts, and arguments to the scripts for execution. Using this approach, we were able to create templates for common tasks such as cleaning the range for scenario use.

Content Generation

To ensure a realistic and "lived in" feel for a scenario, cyber exercises traditionally rely on populating the host filesystems with documents that are either manually created or drawn from existing datasets. Manual document creation can be time consuming and reliance on datasets can limit content type to what is already available. To efficiently align data and content generation for a particular document to match the desired target files for a scenario, we developed an LLM-based content generation tool. This helps with making documents appear consistent for experiments, which in turn increases believability for participants.

Using Ollama, we were able to host LLMs on commodity hardware such as existing CPU/RAM or alternatively commercial off-the-shelf (COTS) GPU components. For document generation we used the Meta Llama 3.1 8B Q4 model. This model, while staying relatively small, allowed for scenario-based content results due to the quantization. Breaking the process into several steps, we developed a pipeline for document creation. This pipeline includes steps such as crafting prompts for LLM requests, validating the results from the models, and finally acting on the results by creating a set of documents.

To validate output from models we developed and used a set of JSON validation schemas. This allowed us to request specific formatted data fields required for document creation from the model. In the event results were invalid for a given schema, LLMs were prompted again using the existing context. This method is used for specifying attributes about files such as size, content type, and file type, as well as supplemental information regarding the artificial source of the document (e.g., a company or other user). This information aids in producing various types of files. For example, we produced noise (generic), target (sensitive proprietary project or similar information), and distractor (supporting target information) document types. Detailed content for the files was produced by the LLMs based on initial content descriptions also defined by the LLM and provided supplemental information. Finally, we created documents of particular types based on the files extension defined by the previous results such as plaintext file formats, PDF, Word, Excel, and PowerPoint.

While LLMs may yield high-fidelity results, there were several challenges we encountered when using this technique. One example is that the LLM output may yield results that contain fill-in-the-blank style or placeholder content. Similarly, when attempting to generate baseline scenario content, the LLM might use generic labels such as “data 1,” “data 2,” etc. These types of issues were somewhat unpredictable, but were mitigated through JSON schemas that gave better, situation-specific content; prompt engineering to improve outcomes; and refined context through examples.

Ensuring Experiment Progression

A key limiting factor in HSR involving expert hackers is the limits on the amount of time that populations will commit to studies. Expert participants in this domain are difficult to find, expensive to commit to studies, and have limited availability to support those studies. For this reason, we wanted to focus the hackers on the parts of the scenarios most relevant to the behavioral analysis that was the subject of the studies. It was important that participants avoided any impediments to their progression throughout the scenario. In particular, in many cases, we were less concerned with observing success in executing a targeted exploit (which can be a difficult, time-consuming task, taking even experienced hackers hours to achieve), and more concerned with how they acted during their attack while pursuing the exploit and after successfully executing it. In our specific HSR, we were particularly focused on early activity, where we were measuring behaviors indicative of susceptibility to a target cognitive vulnerability, and then later activity, where attackers were trying to perform a task while being manipulated (or not being manipulated in the control case) by a treatment intended to exploit the cognitive vulnerability. While our specific objectives were focused on cognitive constraints and manipulations, any HSR studying hacker behavior will need to move the attacker population through the attack chain to ensure they reach the parts of the scenario chain that are the subject of the study and data collection.

To this end, we implemented a help and guidance framework running on the participants’ local Kali machine that provided hints at regular time intervals, to keep the scenario moving forward. In our framework, this was done with simple startup scripts with timers set to appropriate intervals in the scenario. Time limits give well-performing experts plenty of time to complete the task, while also leaving enough remaining time in the scenario to ensure that the participant can progress in the parts of the scenario that are manipulated in our studies. The hint scripts create pop-ups in the operating system notification bar that are written to contextually appear as message notifications from teammates (e.g., incoming intelligence from the participant’s cyberattack team). Additionally, to allow participants to access the contents of the hints after dismissing the notification, the text of the hint message is output to a file in a directory named “Inbox” on the desktop and the participant is informed to expect messages from teammates in this way in their initial scenario briefing.

While this initial hint framework was implemented using simplistic, existing tools on the Kali box, developing a feature within the cyber range to support this kind of guidance directly would ultimately be more beneficial, as it could be done in a more salient way (e.g., outside of the Kali box, in the testbed environment). In future work, we would recommend a deeper integration between this support system and the overarching cyber range to facilitate more salient feedback and guidance.

Data Collection and Extraction

To process the experimental data and execute our analytical tools, we designed a multistep pipeline (see Figure 1) that simplified implementation and maximized the reusability of each component. This pipeline design enabled each component to narrow its dependencies to one or more components in the previous step. Early on, we defined the interfaces between each step, and this interface remained nearly unchanged throughout our experiments. As a result, components were compact and easy to implement and test, and components from later experiments were able to build on existing components implemented for previous ones.

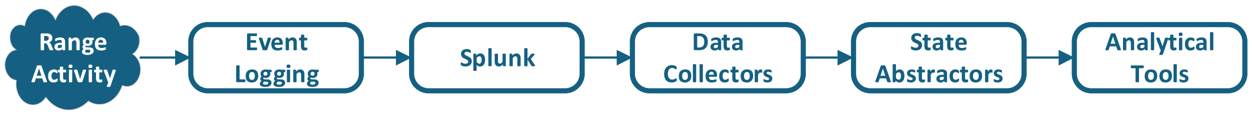

Figure 1: Data collection and extraction pipeline

During a scenario, the participant interacts with the testbed, which is implemented in the SimSpace platform and consists of an emulated public Internet with three separate IP spaces: (1) a "red" network accessible to the participant, (2) a "blue" network consisting of a small corporate network containing the scenario targets, and (3) a "gray" network representing the broader Internet. The blue network is implemented using Active Directory (AD) and configured with varying numbers of subnets, workstations, and servers, depending on the requirements of the scenario.

Event logging uses common, commercially available network- and host-based security logging tools installed within the blue corporate network. Network-based logging was implemented using a Security Onion virtual machine (VM) monitoring network traffic on the blue domain using port mirroring of desired interfaces. Suricata and Zeek are configured in Security Onion to analyze the captured network stream and log their outputs to files. Both tools are useful for capturing participant network behavior during a scenario. Zeek provides a more detailed event logging of network traffic, whereas Suricata provides higher-level detection based on a configured alert ruleset. Additionally, host-based logging is primarily performed by Windows event logging. Domain Groups and associated Group Policy Objects in the blue domain were used to organize both access control and auditing policies to enable the specific Windows event-logging functions required for each scenario. For example, non-built-in commands run via cmd.exe were captured by enabling Windows "Audit Process Creation," which logs Windows event ID 4688. These Windows events are logged locally on each host in the domain and should be centralized to be more easily ingested into Security Information and Event Management (SIEM) tools such as Splunk, a commonly used SIEM software solution. This is done by configuring Windows Event Forwarding for all domain computers along with a corresponding Windows Event Collector (WEC), which receives the source-initiated event logs.

The next step in the data collection and extraction pipeline is aggregation of all event and alert data by Splunk. By integrating with commercially available SIEM software, we ensure that our approach is extensible to security infrastructure commonly used by organizations. To aggregate the host and network logging data, Splunk universal forwarders are configured on the WEC server as well as on Security Onion. This approach is scalable to as many different logging sources as required. Splunk aggregates and indexes all collected streams of events in near–real time and provides several ways to access events stored in the indexes. We primarily use two of these methods for directly querying the indexes. First, the Splunk web portal allows developers to enter search terms and easily filter and visualize event data, and second the Splunk Search Processing Language (SPL) provides a command line interface that can be queried by automated scripts.

After data aggregation by Splunk, the *Data Collector* pipeline step acts as a wrapper for the SIEM repository and enables extraction of the specific event data needed by the state abstractors. This enables flexibility in the choice of logging and event aggregation solutions as the data collectors can be easily tailored to different APIs. Data collectors are organized by their event type with each data collector configuration specifying query parameters such as event ID, alert type, IP address, hostname, etc. It should be noted that while the data collectors perform basic filtering, they do not perform any advanced processing, as their function is primarily as a wrapper for the event data. A central data collection manager runs data collection via Splunk SPL using each of the required data collector configurations, allowing straightforward reconfiguration by adding or exchanging new data collectors.

Once the data collectors have collected all of the available data across different channels within the network, we next focused on using that data to extract specific state information about the attacker. Each *State Abstractor* aggregated concrete data from one or more data collectors to interpret a behavioral state that was meaningful within the context of attacker activity. Converting generic data into domain-relevant states enabled our analytical tools to reason about attacker behavior and performance, which simplified their design and implementation. For example, we implemented a state abstractor that inferred the extent of the attacker's activities around Defender Search. It processed data from three general-purpose data collectors—Command Line, PowerShell, and File Access—to count specific actions that potentially indicated relevant searching activities.

Finally, *Sensors* and *Performance Measures* interpreted the data coming off state collectors in terms of the behavioral cues associated with attacker cognitive vulnerabilities and behavioral metrics for assessing attacker performance, respectively. For example, the analytical tools were able to infer the attacker's Carefulness level by reasoning about the Defender Search state (based on the state assessor), rather than directly processing low-level events like File Access. This layering enabled reuse of components at multiple levels and easier configuration of data collection across our HSR enterprise. Using this approach, across our five experiments, we were able to collect 24 sets of behavioral activities that reflected whether attackers were exhibiting particular cognitive vulnerabilities, and we were able to demonstrate performance impacts across a number of key attack metrics, such as *rate of attack success*, *time wasted*, *progress towards goal*, and *detectability*.

In collecting this level of data across testbeds, one critical issue that we experienced was the impact of time synchronization on data collection. Specifically, to effectively interpret attacker behavior, it is essential to be able to not simply measure behavior, but measure it in relation to key events in the scenario—when the attacker starts to attack, when key stage shifts happen, and when the time limit for the scenario ends. In our early studies, we did not necessarily collect these key times effectively, and we had to recreate the timeline events based on behavioral data. This created complex issues when bugs were introduced into timing (e.g.,, at one point in our study execution, a change in daylight savings lead to some of our data collectors being off by an hour because the underlying hosts did not update their times effectively). As we progressed, introducing more careful management of time constraints, and clear markers for these shifts in scenario state, have been essential improvements in our data collection.

TESTBED EXPERIMENTATION AND RESULTS

Using the configuration and content generation methods described above, we were able to successfully adapt the SimSpace environment to run five HSR studies assessing the ability to exploit cognitive vulnerabilities in expert hackers. These studies, summarized in Table 1, provided targeted scenarios to elicit and study the degree of cognitive bias observed in hackers, and the impact of experimental manipulations in increasing those biases. In each study, 34 participants executed two attack sessions in which they performed a full-scope attack chain, infiltrating a multilayered network, exploring the target nodes they were able to uncover, finding information pertaining to their mission objectives, and attempting to exfiltrate it. Across these studies, we collected more than 340 hours of behavioral hacking data from our participants.

Because of limited populations of experts, we were running a within-subjects study, requiring each participant to perform both a control and experimental condition for each study. We therefore developed four detailed, but related, testbeds for each study, including two variations of the scenario, and a control and an experimental condition within each variation. In developing these four testbeds, our goal was to have enough similarity across them that we were testing the same cognitive behaviors, but enough differentiation that attackers were not benefiting from learning from the previous versions of the scenario.

To address this need, we applied both our configuration management approaches described above and our automated content generation to ensure the development of testbed content that supported these requirements. Using these methods, once scenario design was complete, we were able to rapidly develop each scenarios, with the majority of testbed development tasks completed in two–four-week development cycles (using a team of two developers). We found that these approaches were largely successful in avoiding ordering effects, with no ordering effects in Studies 1, 3, or 4. In Study 2, we saw a small but statistically significant difference in one of our five performance outcomes based on ordering of variants; however, other performance metrics were not impacted. Study 5 is ongoing at this time, and any ordering impacts will be included in our presentation at I/ITSEC.

Table 1: Summary of HSR studies that were executed on SimSpace testbed, including key testbed characteristics and study objectives; note, outcomes of studies are reported in separate publications

Study	Study Objective	Testbed Characteristics
Study 1	Investigated detection and exploits of loss aversion and endowment effect in attackers (Knetsch & Sinden, 1984; Tversky & Kahneman, 1991)	• Attack node provided immediate access to a staging node, which sensed risk-taking behavior and carefulness around defender activity • Attacker objective was to gain access to deeper nodes, exfiltrate data • Intervention: defender agents threaten attacker gains • Versioning leverages different network structure, content
Study 2	Investigated detection and exploits of representativeness bias and base rate neglect in attackers (Kahneman & Tversky, 1972; Bar-Hillel, 1980)	• Network employed two exploitable services, one valid and one out of date and out of place in an otherwise up-to-date network • Attacker objective was to exfiltrate target data; target data was on valid service • Intervention: agent emulates user behavior on out-of-date server • Versioning leverages two additional exploitable services
Study 3	Investigated detection and exploits of confirmation bias (Nickerson, 1998; Gutzwiller et al., 2019; Gutzwiller, Rheem, & Major, 2023; Berthet et al., 2024)	• Network included four target nodes with a targeted type of data that needed to be exfiltrated • Attacker objective is to infiltrate nodes and exfiltrate target data • Intervention: target data obfuscated with fake data of target type • Versioning adapts to different type of data
Study 4	Investigated detection and exploits of anchoring bias (Tversky & Kahneman, 1975; Furnham & Boo, 2011)	• Network included three layers that had to be infiltrated, with target files in deepest layer • Attacker objective is to exfiltrate target data from deepest layer • Intervention: anchor establish presence of honeypots; misleading indicators make high-value targets look like honeypots • Versioning provides different instances of network structure and content
Study 5 (In progress)	Investigated detection and exploits of asymmetric dominance (Huber et al., 1982; Padamwar & Dawra, 2024)	• Network includes web servers that need to be exploited to exfiltrate data • Attacker objective is to exfiltrate data from web servers, then establish persistence elsewhere in the network • Augmented data collection with web server to collect decisions • Versioning provides different instances of network structure and content

A central focus of our testbed adaptation and development, as mentioned above, was to ensure that participants would get to the points of the scenario in which we could effectively test our sensors and behavioral manipulations. To achieve this objective, we augmented the testbed with pop-up guidance targeted at helping the attackers to move forward in the attack process. In some cases, this guidance would prod them toward targets of interest; in others, it would give them the specific information they need to move on, to ensure they reached the stage of the attack where the manipulation is made, and data is collected.

Within our studies, while we observed a wide variation in performance, with many participants failing to achieve the full scope of attack behavior targeted within the scenario, our manipulations were largely successful in moving participants through the attack chain to ensure successful data collection, with all but a handful of participants across the study reaching the target points in each scenario. For each participant across each study, we were able to collect behavioral data reflecting bias susceptibility in the early stages of the study. While many participants were able to move through the attack chain without issue, for those who struggled with the exploits that would move them on, we were able to provide hints that successfully moved them to parts of the testbed where they could be impacted by our experimental conditions (or lack of those conditions, in control cases). However, while attackers were able to move onto the stages of the attack where we could collect data, we did observe that many attackers ultimately failed in the attack objectives (e.g., exfiltrating data). Fortunately, in many cases, we were able to explore alternate behavioral indicators to assess their progress toward those objectives, rather than the objectives themselves.

Finally, using the data harness described above, we designed and developed 37 behavioral sensors, each designed to assess susceptibility for one of the cognitive biases described in Table 1 above. Each of these sensors were designed to translate a cognitive construct or moderator related to the cognitive bias into observable behavior in the cyber environment. The majority of these sensors were successful, aligning or correlating with psychometric established measures for the cognitive biases. In those few cases where the sensors were unsuccessful, it was largely due to the data scarcity; that is, the sensor was trying to measure behavior that did not show up in the collected data, primarily because the specific commands that were included in the sensor were hidden within command line interactions (e.g., two of our sensors for confirmation bias were attempting to analyze the search commands used by the attacker;

however, such commands do not show up in logs, because they are embedded within the command line interface and are not tracked separately).

In addition to collecting these behavioral sensors, we also collected a wide variety of performance metrics across these studies, on average looking at five performance factors in each study. We specifically explored indicators for (1) rate of attack success (e.g., looking at the percentage of target files exfiltrated), (2) progress towards goal (e.g., looking at the number of commands executed that are related to achieving the target objective), (3) wasted time (e.g., looking at time spent on decoy servers or the number of distractor files exfiltrated), (4) detectability (e.g., looking at the number of commands on known decoy servers, or the noise generated by exfiltrating irrelevant data), and (5) cognitive effort (e.g., looking at the time spent searching for a fake defender). In three of our four initial studies, we observed multiple significant performance impacts based on the experimental manipulation (these results will be reported in separate publications on each study). Overall, our test harness has proven successful in both collecting a wide range of behavioral data and interpreting that data based on related cognitive constructs for susceptibility and performance impact constructs.

CONCLUSIONS AND FUTURE RECOMMENDATIONS

Overall, in this work, we successfully applied a cyber testbed environment to study the behavior and constraints of experienced hackers. We identified several critical challenges in applying cyber testbeds to human subjects research (HSR), and we developed a variety of solutions in our environment to address those challenges, including a configuration management framework to ensure that we are able to reuse as much content as possible across versions of testbed scenarios, tooling for automated content generation suitable to HSR, guidance systems to ensure that HSR stays on track, and a data collection and analysis framework for assessing human behavior and performance within studies of cyberattacker behavior. Our framework is designed to support high-fidelity studies of attacker behavior, where attackers progress through an attack chain in realistic scenarios with targeted mission objectives, rather than the capture-the-flag (CTF) events explored in previous studies. Our solutions have streamlined the development of cyber testbeds for HSR, and provided essential experience and tooling needed to support future studies in this domain. Results of our studies and future work will be reported separately in other publications in cyber and behavioral conferences and journals.

As we continue our research on studies of hacker behaviors and cognitive vulnerabilities, there are several that we will explore further. First and foremost, we are looking at ways to automate and streamline the data collection pipeline further, with two objectives in mind. First, to streamline data collection in similar studies going forward, we recommend developing a framework within the testbed to support automated data collection. In our current framework, proctors need to go in and activate the scripts to collect and export the data; this could readily be automated to execute based on one of several events, including the completion of scenario objectives, the attacker logging out of the testbed, and various other activities. Automating this process would significantly streamline study execution and ensure more consistency in the data collection process. Beyond this, in future work, we will streamline this analysis framework to provide real-time behavior analysis, supporting proactive and dynamic defenses that revise the interventions based on the behavior of the attacker, continuing to exploit the attackers' underlying cognitive vulnerabilities. Beyond these updates to our existing data management framework, testbeds themselves can greatly benefit from more advanced and automated APIs that provide accessibility to behavioral data; our data pipeline provides an initial framework for this extraction and could readily be integrated into SimSpace itself, or into other similar cyber ranges. A key focus, as we continue studying cyber adversary behavior and cognitive constraints, will be to develop this advanced data harness for extracting adversary behavioral data from cyber ranges.

ACKNOWLEDG~~E~~MENTS

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via N66001-24-C-4501. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the US Government. The US Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation therein.

REFERENCES

Bar-Hillel, M. (1980). The base-rate fallacy in probability judgments. *Acta Psychologica*, *44*(3), 211–233. http://doi.org/10.1016/0001-6918(80)90046-3

Berthet, V., Teovanović, P., & de Gardelle, V. (2024). A common factor underlying individual differences in confirmation bias. *Scientific Reports*, *14*(1), 27795. http://doi.org/10.1038/s41598-024-78053-7

Ferguson-Walter, K., Gutzwiller, R., Scott, D., & Johnson, C. (2021). Oppositional human factors in cybersecurity: A preliminary analysis of affective states. In *36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)* (pp. 153–158). Melbourne, Australia. http://doi.org/10.1109/ASEW52652.2021.00040

Ferguson-Walter, K. J., Major, M. M., Johnson, C., & Muhleman, D. (2021). Examining the efficacy of decoy-based and psychological cyber deception. In *Proceedings of the 30th USENIX Security Symposium*. Retrieved from https://www.usenix.org/system/files/sec21-ferguson-walter.pdf

Ferguson-Walter, K., Major, M., Van Bruggen, D., Fugate, S., & Gutzwiller, R. (2019). The world (of CTF) is not enough data: Lessons learned from a cyber deception experiment. In *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)* (pp. 346–353). http://doi.org/10.1109/CIC48465.2019.00048

Furnham, A., & Boo, H. C. (2011). A literature review of the anchoring effect. *Journal of Socio-Economics*, *40*(1), 35–42. http://doi.org/10.1016/j.socec.2010.10.008

Gigerenzer, G. (2003). *Calculated Risks: How to Know When Numbers Deceive You*. Simon & Schuster. Retrieved from https://www.simonandschuster.com/books/Calculated-Risks/Gerd-Gigerenzer/9780743254236

Gutzwiller, R., Ferguson-Walter, K., Fugate, S., & Rogers, A. (2018). "Oh, look, a butterfly!" A framework for distracting attackers to improve cyber defense. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *62*(1), 272–276. http://doi.org/10.1177/1541931218621063

Gutzwiller, R., Rheem, H., & Major, M. (2023). Exploratory analysis of decision-making biases of professional red teamers in a cyber-attack dataset. *Journal of Cognitive Engineering and Decision Making*, *18*(1), 37–51. http://doi.org/https://doi.org/10.1177/15553434231217787

Gutzwiller, R. S., Ferguson-Walter, K. J., & Fugate, S. J. (2019). Are cyber attackers thinking fast and slow? Exploratory analysis reveals evidence of decision-making biases in red teamers. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *63*(1), 427–431. http://doi.org/10.1177/1071181319631096

Gutzwiller, R. S., Gilbert, M., Drescher, T. J., Ferguson-Walter, K. J., Mikanda, N., Johnson, C. J., & Scott, D. D. (2023). Frustration, confusion, surprise, confidence, and self-doubt: Cyber operators' affects during a realistic experiment. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *67*(1), 233–239. http://doi.org/10.1177/21695067231192883

Huber, J., Payne, J. W., & Puto, C. (1982). Adding asymmetrically dominated alternatives: Violations of regularity and the similarity hypothesis. *Journal of Consumer Research*, *9*(1), 90–98. http://doi.org/10.1086/208899

Kahneman, D., & Tversky, A. (1972). Subjective probability: A judgment of representativeness. *Cognitive Psychology*, *3*(3), 430–454. http://doi.org/10.1016/0010-0285(72)90016-3

Knetsch, J. L., & Sinden, J. A. (1984). Willingness to pay and compensation demanded: Experimental evidence of an unexpected disparity in measures of value. *The Quarterly Journal of Economics*, *99*(3), 507–521. http://doi.org/10.2307/1885962

Krämer, W. (2014). Kahneman, D. (2011): Thinking, Fast and Slow. *Statistical Papers*, *55*(3), 915. http://doi.org/10.1007/s00362-013-0533-y

Nickerson, R. (1998). Confirmation bias: A Ubiquitous Phenomenon in Many Guises. *Review of General Psychology*, *2*(2), 175–220. http://doi.org/https://doi.org/10.1037/1089-2680.2.2.175

Padamwar, P., & Dawra, J. (2024). An integrative review of the decoy effect on choice behavior. *Psychology & Marketing*, *41*(11), 2657–2676. http://doi.org/https://doi.org/10.1002/mar.22076

Pohl, R. F. (Ed.). (2022). *Cognitive Illusions: Intriguing Phenomena in Thinking, Judgment, and Memory* (3rd ed.). London: Routledge. http://doi.org/10.4324/9781003154730

Tversky, A., & Kahneman, D. (1975). Judgment under uncertainty: Heuristics and biases. In D. Wendt & C. Vlek (Eds.), *Utility, Probability, and Human Decision Making* (pp. 141–162). Dordrecht: Springer Netherlands. http://doi.org/10.1007/978-94-010-1834-0_8

Tversky, A., & Kahneman, D. (1991). Loss aversion in riskless choice: A reference-dependent model. *The Quarterly Journal of Economics*, *106*(4), 1039–1061. http://doi.org/10.2307/2937956