Adaptive Experimental Design for Intrusion Data Collection

Kate Highnam[1,2], Zach Hanif[3], Ellie Van Vogt[1], Sonali Parbhoo[1], Sergio Maffeis[1], and Nicholas R. Jennings[4]

[1]Imperial College London
[2]The Alan Turing Institute
[3]Independent Researcher
[4]Loughborough University

Abstract

Intrusion research frequently collects data on attack techniques currently employed and their potential symptoms. This includes deploying honeypots, logging events from existing devices, employing a red team for a sample attack campaign, or simulating system activity. However, these observational studies do not clearly discern the cause-and-effect relationships between the design of the environment and the data recorded. Neglecting such relationships increases the chance of drawing biased conclusions due to unconsidered factors, such as spurious correlations between features and errors in measurement or classification. In this paper, we present the theory and empirical data on methods that aim to discover such causal relationships efficiently. Our **adaptive design (AD)** is inspired by the clinical trial community: a variant of a randomized control trial (RCT) to measure how a particular "treatment" affects a population. To contrast our method with observational studies and RCT, we run the first controlled and adaptive honeypot deployment study, identifying the causal relationship between an `ssh` vulnerability and the rate of server exploitation. We demonstrate that our AD method decreases the total time needed to run the deployment by at least 33%, while still confidently stating the impact of our change in the environment. Compared to an analogous honeypot study with a control group, our AD requests 17% fewer honeypots while collecting 19% more attack recordings than an analogous honeypot study with a control group.

1 Introduction

Automated cyber intrusion attacks continuously scan and probe internet-connected systems [1, 2]. The state of the art in cyber intrusion defenses employ observational techniques augmented with automated statistical techniques, including temporal point processes and machine learning. This approach has been effective, but is susceptible to a variety of biases that might mislead or confuse such solutions from generalizing or learning quicker [3, 4].

Table 1: Mapping of healthcare terminology to security terminology for this paper.

Healthcare	**Security**
"trial"	"a study comparing honeypots with and without a vulnerability"
"study population"	"our Ubuntu honeypots with our host-based sensors"
"patient" or "participant"	"a honeypot"
"recruiting more subjects"	"starting more honeypots with specific characteristics"
"disease"	"attacker technique for exploit"
"intervention" or "treatment"	"corruption" or "the presence or insertion of a vulnerability"
"treated"	"corrupted"

We aim to limit the impact of potential bias by improving the datasets that train intrusion detection methods.

Intrusion datasets can be acquired from third party vendors or compiled by recording logs from existing, simulated, or newly deployed research infrastructure [5, 6, 7, 8, 9, 10]. Conventionally, these methods provide **observational data**, containing information on current attacks implemented against the given systems and how to observe the attacks in the given environment. However, even in large volumes, observational data has a high potential for bias due to uncontrolled characteristics, including possible spurious correlations between variables and outcomes or measurement error [11, 4, 12, 9, 13].

To limit potential erroneous conclusions by both statistical models and researchers, we explore intrusion data collection with a **control group**: a collection of systems studied that are not altered to compare with identical systems that have been altered. Our usage of control groups in an experimental study draws inspiration from clinical research, one of the oldest fields conducting control-based studies [14, 15]. In healthcare, a typical control group study randomly recruits a subset of a population to remain untreated (as the control) and treated (as the altered version). This is known as a **randomized-control trial** (RCT), the gold standard for clinical trial methods; its random assignment to groups minimizes the impact of researcher biases while evaluating causal relationships [16]. Our method is based on **adaptive design** (AD): a variant of RCT that adds pre-planned opportunities to modify aspects of an ongoing trial in response to data accumulated during the study, without invalidating its integrity [17, 18, 19]. RCT and AD both account for known conditions and unforeseen events (e.g., a pandemic or war) which might require the trial to end early by separating the trial into multiple stages to run interim analysis.

Unlike clinical trials with human patients, intrusion research aims to increase the occurrence of events of interest (i.e., intrusions or exploits). See Table 1 for some of the terminology from healthcare mapped to security as it is used to define our work. To demonstrate our intrusion-focused interventional methods, we use a **honeypot**, a common tool for recording intrusion data. A honeypot is an intentionally vulnerable system with covert monitoring that is used to both entice and observe attackers without their knowing [20].

Traditional honeypot deployments - or "vanilla" deployments as we will call them in this

paper - expose a large number of identical vulnerable systems for a particular (extended) length of time to collect intrusion data [21, 22, 23, 20, 24, 25]. While sufficiently large and long-lived vanilla deployments all but guarantee observations and can summarize the general state of automated threats, they carry several risks and costs that could be unacceptable. If a meaningful quantity of identical honeypots were left online, it would provide an opportunity for adversaries to identify the presence of the employed monitoring tools. This can hinder observations (i.e., bias the data) and render the tools useless (i.e., when adversaries stop acting after detecting active monitoring or debugging tools). Additionally, large scale deployments cost time and money, which absorbs budget, and can hinder or preclude timely observations.

In this paper, we present the first control-based deployment method for honeypots to optimize resource allocation and limit honeypot exposure. Our method is used in an exemplary study to determine the impact of an `ssh` vulnerability on cloud servers across the United States. When compared to the vanilla deployment method with the same setup, we find that AD can determine the impact of the vulnerability in 33% of the total trial duration, while limiting the likelihood of error. and requesting 17% fewer honeypots overall. With a control group, our AD collects 19% more attack recordings than the RCT trial.

Our contributions in this work are as follows:

- The first adaptive method for a control study in security, optimizing resource allocation and duration of the study based on the events seen in prior stages and error tolerance.
- The first interventional study using honeypots, demonstrating the effectiveness of our adaptive method and how it helps attribution of environment changes during data collection.

Although we showcase our method with honeypot deployments, it can be used for other control studies in security applications. For example, one could study the impact of a new spam email training on the rate of spam emails being opened, or on the removal of local file inclusion access on the exploitation rate of a web application hosting other vulnerabilities. Our AD strategy uses a new interpretation of clinical trial methodologies, encouraging infections rather than preventing them mid-trial. Additionally, our study ran using automated scripts, presenting the first fully-automated experimental study. This automation and cheaper application setting enables future inventors of new clinical trial methodologies a new venue to showcase their improvements, rather than run an expensive trial with patients.

This paper is structured as follows. Section 2 reviews control studies in security and provides a brief background on the healthcare-based methods that inspired this work. We then introduce our new AD method in Section 3 while contrasting it with the vanilla and RCT methods. In Section 4, we implement our method against the vanilla and RCT methods in a exemplary honeypot deployment. We conclude and consider how the method might not behave the same in other settings in Sections 5.

2 Background

An experimental study starts from a hypothesis on how a change or treatment will alter an aspect of a given environment [26]. The hypothesis is then tested, in the simplest form, by observing a control group that is unchanged and comparing it with another (ideally identical) group that is then changed. An RCT provides one of the strongest evidence on an intervention's impact due to its random allocation of participants to treatment arms (there might be multiple treatments available) or control arm (standard care or a placebo) [27, 28]. This process removes potential bias of unaccounted factors in the environment. Participants are then observed and their outcomes recorded.

Part of the rigor of RCTs is that all aspects of the trial conduct and (interim) analysis must be documented prior to the execution of the study. This prospective approach avoids the introduction of bias from investigators and statisticians mid-trial. An important part of this planning process is the sample size calculation. Using estimates and prior knowledge of interventions, trialists (those conducting the trial) can estimate the required number of participants that must be recruited in order to detect a significant difference in outcomes between the groups [29]. After approximating the needs and impacts of the study, the execution of the study should be justified.

For medicine, developing the justification to go from drug discovery to licensing can take an average of over 10 years [30]; recent advances in trial methodology have been able to improve the efficiency of trials in order to reduce this time [31]. The conduct and methodology of a clinical trial are highly regulated because of the direct involvement of patients [28]. However, this is not often a barrier to research in cyber security.

Security has several advantages in running experimental studies. Digital infrastructure is cheap with the advancement of cloud technologies [32]. Our honeypot study could have cost up to $2,000USD with 600 participants, compared to the millions of USD required in drug testing [33]. Digital resources can also be exactly copied as many times as needed, whereas biological studies must make strong assumptions about the similarity between patients. Security experimental studies can be quicker to complete if they consider the attacks that occur at a higher frequency and pace of development than a biological infection or disease.

In this section, we review previous experimental studies in security settings that consider or run control groups. We finish this section by briefly highlighting the other techniques developed to improve the classic experimental design that have spawned from the constrained medical setting.

2.1 Control Trials in Security

Our work is not the first to apply clinical methodology within security; prior works focus on the interaction of security and users of digital systems. For example, Simoiu et al. [34] survey user awareness of ransomware in the general U.S. population. Lin et al. [35] analyze spearphishing cyber attacks and its correlation with various human demographics. A common human-oriented security study involves antivirus software and how it is used by the lay

person [36, 37, 38]. These works implement an experimental study to find strong indications of how successful antivirus software can be based on human performance. Yen et al. [39] further extends this research area by incorporating the users job title and responsibilities to contextualize the impact of malware within a company. However, we circumvent the recruitment (and cost) of human involvement by focusing on how these methods can be applied to digital systems with automated, autonomous threats.

Few experimental studies have been published without humans in cyber security. Bošnjak et al. [40] prepare an experimental study to systematically evaluate defenses for shoulder surfing attacks after an extensive literature review. Gil et al. [41] approach this by using a case-control study to identify complex relationship of threats to a single host within a large network. Although it is called a "study," a case-control study filters and randomly selects data from purely observational studies for its patient population. There is no interaction with the data collection process. Causal relations can be learned from such data but there is no control for error or bias. We discuss how our method controls for error in Section 3.

These studies indicate a major challenge in security experimental studies: the need for human-interpretable interventions. Recording data in medical settings is relatively straightforward, e.g., heartbeats per minute or body temperature indicating there is a fever. Understanding how these indicate a particular disease is also fairly intuitive. But translating host-based logs to indications of unwanted activity in a system is immensely difficult, let alone stating the type of unwanted activity. Thus, mapping "symptoms" from sensor logs can be difficult unless we control our human-level interventions.

2.2 Advances in Control Trial Methodology

Randomization limits the impact of unknown external factors in influencing a participant's chance of receiving a treatment, we therefore expect the baseline characteristics of participants to be similar between studied groups. If there is a concern about some baseline characteristics that may be prognostic, then we can stratify randomization based on these variables without loss of statistical strength [42].

Traditional RCTs are known for their rigor and complete pre-specification of procedure. A common adaptation to an RCT is the inclusion of stopping rules for efficacy, safety, or futility [43]. If the trial has gathered enough evidence that an intervention is effective, or conversely that the intervention is harmful, then the study can cease, saving resources for a future study or a re-run of the same study with corrections. Similarly, interim checks would also catch if there is not enough evidence of an intervention's effect to reach a significant conclusion [44].

Contemporary approaches to running RCTs aim to make the process of evaluating an intervention faster and more efficient [31]. As mentioned, we implement one such method, adaptive design (AD). The principle of AD is that it permits certain aspects of a study to be modified intermittently based on available evidence.

Interim data used to inform stopping decisions can also be used to inform an updated sample

size calculation, or even to update randomization allocation proportions [19]. A well-known example of this is the REMAP-CAP study, which has many treatments available across multiple domains for treating community acquired pneumonia, including severe COVID-19, in intensive care unit settings [45]. Platform trials have also been used to successfully evaluate many different types of interventions simultaneously [46]. Monthly interim analyses are conducted and a Bayesian model is used to update randomization probabilities for new participants entering the trial, so that patients are randomized to treatments that are more likely to benefit them [47]. In the present study, we draw on the concept of response adaptive randomization to optimize the allocation of honeypots.

3 Adaptive Design for Security Applications

In this section, we define our methods for interventional data collection in security applications. This setting typically studies adversarial effects, i.e., encouraging intrusions as data are collected. We call the change or treatment (e.g., a drug or surgical procedure) made to a population a **corruption**[1]. We shall now enumerate the key terms to document prior to executing a study as we define our AD method.

The **population** considered in the experiment is assumed to be a device or contained system that is or is not corrupted. Deploying a copy of these devices or systems is the same as recruiting a patient into a study. The goal of the study is to achieve a set of objectives, evident from observing a particular **event of interest**. One can identify an event of interest through the recorded logs on the population; these events should be clear evidence that the corruption caused some change in system behavior. For example, if the corruption is a new login website and we are interested in its effect on attempted SQL injections, then events of interest should be a record of when an SQL injection occurs.

The **objectives** of our methods are always two fold:

1. Confirm evidence of corruption's impact within the population.
2. Maximize the recording of events of interest.

Returning to the SQL injection example, if we wanted to collect a diverse range of attacks rather than automated repeated uses of the same attack, the events of interest could be only recorded if not seen prior.

Before recording events and running the study, it is crucial to accurately define **endpoints** to anticipate possible errors or miscalculations. Similar to clinical trials, we recommend setting an endpoint bounding the trial resources by the given budget. We also recommend stopping the trial early if the adaptive design tries to deploy a group that is too small. If this occurs in the early stages of the trial, it can indicate the rates of allocation have converged to nothing conclusive. This should be followed with a manual review by human experts. While it might seem inconsequential, it is good practice to list obvious endpoints. This might

[1]We chose corruption to remove the benevolent intentions frequently affiliated with "treatment" from healthcare settings. A reminder that the translations for other clinical trial terms can be found in Table 1.

Method 1 Vanilla Observational Study

Input: Budget b, Trial Duration t (in hours)
Ensure: $b, t > 0$
1: $N = \text{GetNumToDeploy}(b, t)$
2: /* Start Trial */
3: $\text{Deploy}(\text{control} = 0, \text{corrupted} = N)$
4: $\text{Wait}(t)$
5: $L = \text{SaveLogs}()$
6: $\text{CleanUp}()$

Method 2 Randomized Control Trial

Input: Budget b, alpha α, beta β, Number of Stages s, Stage duration t (hours), proportions of interesting events happening in control p_1, proportions of interesting events happening in corrupted p_2
Ensure: $s, b, t > 0$; $0.0 < \alpha, \beta < 1.0$
1: $N_1, N_2 = \text{PowerAnalysis}(p_1, p_2, \alpha, \beta)$
2: **for** $i = 0; i < s; i{+}{=}1$ **do**
3: /* Start Stage of Trial: $N_1 == N_2$ */
4: $\text{Deploy}(\text{control} = N_1, \text{corrupted} = N_2)$
5: $\text{Wait}(t)$
6: $L = \text{SaveLogs}()$
7: $\text{CleanUp}()$
8: $N_{\text{total}} += N_1 + N_2$
9: **if** $\text{isEarlyStop}(L, b, N_{\text{total}})$ **then**
10: $i = s$
11: **end if**
12: **end for**

Method 3 Adaptive Design Study

Input: Budget b, alpha α, beta β, Number of Stages s, Stage duration t (hours), proportions of interesting events happening in control p_1, proportions of interesting events happening in corrupted p_2
Ensure: $s, b, t > 0$; $0.0 < \alpha, \beta < 1.0$
1: $N_1, N_2 = \text{PowerAnalysis}(p_1, p_2, \alpha, \beta)$
2: **for** $i = 0; i < s; i{+}{=}1$ **do**
3: /* Start Stage of Trial */
4: $\text{Deploy}(\text{control} = N_1, \text{corrupted} = N_2)$
5: $\text{Wait}(t)$
6: $L = \text{SaveLogs}()$
7: $\text{CleanUp}()$
8: $p_1, p_2 = \text{SurvivalAnalysis}(L)$
9: $N_1, N_2 = \text{PowerAnalysis}(p_1, p_2, \alpha, \beta)$
10: $N_{\text{total}} += N_1 + N_2$
11: **if** $\text{isEarlyStop}(L, b, N_{\text{total}})$ **then**
12: $i = s$
13: **end if**
14: **end for**

include recording an unexpected exploit technique or an overwhelming number of exploits that break data collection infrastructure. All of these details must be defined prior to the study execution to maintain the robustness of the trial.

3.1 Trial Methodologies

In this section, we review each method for comparison to our AD before using them in a honeypot study (Section 4). The pseudo code for the traditional observational study (vanilla), RCT, and our AD are presented in Methods 1, 2, and 3, respectively. See Appendix A for the definitions of the functions. The highlighting indicates the similar lines between the algorithms. Notably, the RCT and AD trials are split into s stages with early stopping - shown as the loops on line 2 in both Methods 2 and 3, highlighted in pink. Each stage deploys some proportion of control and corrupted systems, waits for the stage duration, saves the logs, cleans up the deployment and reviews the logs from the stage to see if an endpoint condition has been reached. The difference between the standard RCT and our AD is what occurs during the interim update.

The vanilla deployment (Method 1) takes a given budget b and the maximum trial duration[2] t to determine the maximum number of devices N that can be observed within this study - noted as `GetNumToDeploy(b, t)` on line 1. Then N altered devices are then deployed for observation during the "trial"; no control devices are present.

In contrast, the RCT (Method 2) and AD (Method 3) account for the risk of error into selecting how many devices to study using **power analysis**[3]. We pass four parameters into the power analysis equation from HECT [48]: the probability of committing a Type I error (α), the probability of committing a Type II error (β), and the rate of incidence for the control and corrupted groups. A Type I error means claiming an effect is present due to the corruption when it is not true. A Type II error means the study did not collect evidence of an effect when it is correct. The **power** of a study is the inverse of the likelihood of committing a Type II error ($1 - \beta$). The rate of incidence for the control and corrupted groups is initially determined by a pilot study or educated guess based on related reports. It is an approximation of the rate an event of interest should be observed within the stage. The power analysis equation returns the total number N_{total} of devices needed to deploy in each stage. We equally split this value for RCT and the initial stage of AD. After the first stage of AD, we use the updated rates from the prior stage to weight the split of N_{total}, adapting the allocation of resources mid-trial.

Based on the responses seen in the previous stage within an AD trial, trialists can use interim analysis to make pre-defined changes that will not invalidate or weaken the power of a study. Our AD updates the next population counts for control and corrupted. To not risk weakening the power of our study, we apply this update indirectly between stages through the assumed rates of incidence (p_1 and p_2).

From the logs we have a complete view into the events of interest for the study. Our AD method assumes that each participant will have at most one event of interest before terminating the system, removing it from the trial. In the case of honeypots, this would be to protect the honeypot from becoming a launchpad or providing free resources to attackers. To calculate the likelihood of an event of interest occurring, i.e., the rate of incidence within a group, we use a **Kaplan-Meier (KM) Function**, a popular approach for survival analysis within healthcare applications[49, 50].

During a stage, the KM function is updates the likelihood S upon every event of interest recorded. At $t = 0$, all participants are at risk and $S(t) = 1$. The remaining time steps update following this:

$$S_{t+1} = S_t \times \frac{N_t - D_{t+1}}{N_t} \tag{1}$$

where N_t is the current number of participants at risk and D_{t+1} is the number of participants that have seen events of interest since t. The difference $(N_t - D_{t+1})$ is not always one. If we know exactly when all participants see an event of interest, the KM function is calculated when an infection is recorded. In trials without this ability, a time interval must be set to

[2]We say in the methods that this is given in hours, but any time duration is works here.

[3]This calculation can be found in more detail in Appendix A.

check with the partipants (e.g., every hour or half-hour) to collect data and see if an event has been recorded.

4 Adaptive Design for Honeypot Deployments

We demonstrate the capabilities of our AD for intrusion data collection in a sample study using honeypots. Our method can be applied in other intrusion data collections, but we chose one to illustrate its specific capabilities. In this study we analyze the risk of an `ssh` vulnerability within misconfigured cloud servers. This scenario is based on a dataset used as a pilot study for our trials[4]. The dataset contains a variety of attacks via the `ssh` vulnerability, but we can only empirically infer how the presence of this vulnerability affects its likelihood of exploitation. Although the presence of an `ssh` vulnerability is well known to affect the rate of exploitation in a server, this study to emphasizes how our method provides evidence on a corruption's impact and the benefits of its adaptation.

This study includes Methods 1, 2, and 3 in separate trials, each attempting to collect evidence of the corruption's impact. Our budget restricts each trial to a maximum of 200 honeypots (approximately $650USD) over 12 hours. As recommended in Section 3, this threshold is noted as the first of our early stopping criteria. Based on the pilot study, we assume an initial rate of incidence in the control to be 0.01 and in the corrupted to be 0.4. We ensure the study only considers strong evidence by limiting the chance of error[5], setting $\alpha = 0.05$ and $\beta = 0.10$. The remaining details for our honeypot studies are summarized in our Study Synopsis (Section 4.1). We then review the results of the study in Section 4.2.

4.1 Study Synopsis

Following the guidance issued for clinical trials [51], we summarize the characteristics of our study below:

Study Duration : The maximum total duration per trial is 12 hours. This is applied across the three trial methodologies compared in this study

Objectives : (1) Determine if the corruption causes a significant increase in exploitation rate. (2) Maximize the exploitation rate for honeypots in the U.S. by region in time specified by the trial or stage.

Endpoints : (a) The maximum number of honeypots that can be recruited into the study is 200. (b) The total number of honeypots allocated in the corruption group is below 10 (indicating the event of interest is not recorded frequently enough to study in this duration) (c) The number of honeypots allocated is identical to the last stage of the AD trial, indicating strong evidence has been collected regarding the current rates of

[4]This citation is removed for anonymity.

[5]It is generally accepted in healthcare settings to set $\alpha = 0.05$ and power = 80% (meaning $\beta = 0.2$). We assume a power of 90% ($\beta = 10\%$) because we know there is a large difference between the control and the corrupted.

incidence.

Study Population : Cloud-based honeypots monitored with a kernel-level sensor recording all create, clone, and kill system calls. Each honeypot runs with 1 vCPU, 32GB memory, and Ubuntu 20.04. There are no additional programs or fake activity and no active connection between honeypots. They are all hosted by the same large-scale cloud provider within the U.S. that instantiates identical servers with unique, randomly assigned IP addresses upon request. The IP address ranges are based on the requested region; our study only considers four regions within the U.S.: `east-1`, `east-2`, `west-1`, and `west-2`.

Study Corruption and Control : The corruption is an `ssh` vulnerability that accepts any password for four fake user accounts mimicking IT support accounts on industrial infrastructure: `user`, `administrator`, `serv`, and `support`. This corruption is an exaggerated version of a common misconfiguration seen in cloud servers [2, 52, 53]. Control honeypots host the same user accounts but only accept "password" as the password. We chose the word "password" as the password based on evidence of attackers scanning for it in cloud provider networks [25].

Event of Interest : We record an event when a user login is seen in one of the four user accounts. Because we never login or generate fake calls to login, any user login seen is considered malicious.

Measuring Corruption Effect : This study assumes a binary state model to describe each honeypot as whether an intrusion has or has not occurred. The state of the honeypot is determined by real-time monitoring of the logs to deal with ethical issues that may arise from purposefully exposing compute to adversaries. An exploit is assumed to not have occurred until this event is seen.

To prevent providing free resources to the attacker or opportunities to launch further attacks, we terminate the instances upon recording an event of interest. Although this limits the data we acquire from the study, it satisfies our objectives in recording events of interest. This can be re-evaluated in alternative studies based on new objectives.

Each honeypot functions independently with no communication between the honeypots in the same trial. Their logs are aggregated on a central queuing system within their region (for their trial) and downloaded before termination. Because we use kernel-level sensors, our implementation can be easily extended for other objectives and vulnerabilities.

4.2 Results

The total number of honeypots deployed and attacks recorded is shown in Table 2. As expected, the AD trial deployed the fewest honeypots overall while recording more attacks than RCT. The AD trial recorded around 36% of the total attacks seen in the vanilla trial, which saw the highest number of intrusions. This is because the vanilla trial did not deploy any control honeypots, which had a small rate of incidence. However, by not including a control group, it does not account for potential bias in the corruption implementation,

Table 2: Comparison of honeypot deployment methods used in each 12-hour trial.

Method for Trial	Control	Corrupted	Total Deployed	Total Attacks Seen
Vanilla	0	140	140	**137**
RCT	72	72	144	42
AD	32	87	**119**	50

Table 3: The deployed honeypots by region and stage for the RCT and the AD trial. Region names are abbreviated to `e` for "east" and `w` for "west".

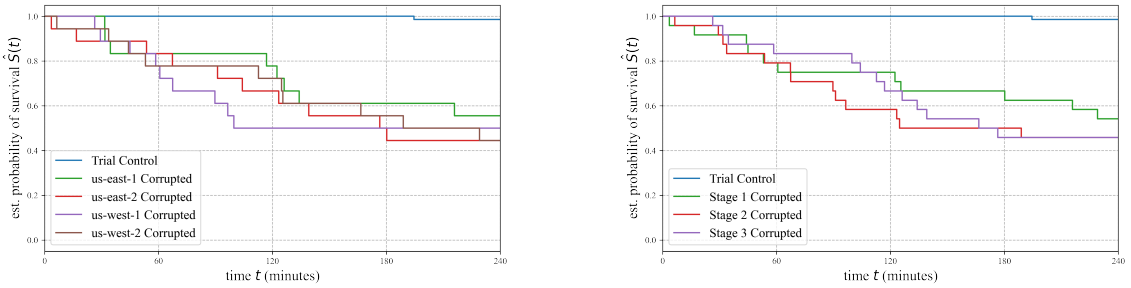
		Control				Corrupted				Total
		e-1	e-2	w-1	w-2	e-1	e-2	w-1	w-2	
Stage 1	RCT	6	6	6	6	6	6	6	6	48
	AD	6	6	6	6	6	6	6	6	48
Stage 2	RCT	6	6	6	6	6	6	6	6	**48**
	AD	4	4	0	0	8	12	8	16	52
Stage 3	RCT	6	6	6	6	6	6	6	6	48
	AD	0	0	0	0	2	5	8	4	**19**

preventing it from confidently identifying the causal relationship of the corruption's effect. Even so, the data collected could still be used with the rigorous documentation stating the assumptions made in the study. This enables other researchers to review it independently and determine if the study's findings are relevant for their environments.

From the trials with a control group, it is clear that the corruption causes an increase in exploitation rate. As can be seen in the trial summary in Table 3, more control honeypots were exploited than expected in the AD trial, causing a disparity between our initial assumption ($p_1 = 1\%$) and the results from the first stage (marginalized $p_1 = 15\%$). This caused our AD method to reallocate resources, requesting more honeypots to accommodate for error in the initial assumptions without triggering an endpoint or requiring the study to be reevaluated. After the second stage of the AD trial concluded with no exploits in the control group, the control arm was dropped, confirming that the corruption led to more infections.

We could have ended the AD trial after the first stage (saving 66% of the trial's budget) if Endpoint (b) included the control group in the group size minimum. This was not done because of the second objective to collect intrusion data. Even with the larger request in stage two, the AD trial requested fewer honeypots than both the vanilla trial and the RCT. The stages within the trial also limited the time a honeypot was online, preventing adversaries for extended time to develop a signature for our trap.

Although there was an exponential rate of exploit across the trials, we noticed signs of instability in the four-hour stages, where some regions were observed to have different exploitation rates. This was especially apparent by comparing their respective survival curves by the regions and stages within the RCT trial, shown in Figure 1a and Figure 1b. Around 60-120 minutes into the stage, the rate of infection in the regions diverges as though `us-west-1` was



(a) RCT by Region

(b) RCT by Stage

Figure 1: Comparison of the control and corrupted honeypots by 4-hour stage and region across the RCT trial.

hit first, then `us-east-2`, `us-west-2`, and `us-east-1`, respectively. From the IP addresses of the hosts, there is no obvious indication of sequential IP scanning. This instability is another result of this study so future work can note the impact of smaller time windows. Because this was unanticipated at the start of this study and not a prior listed early stopping criterion, future work will include it to provide an opportunity for the trialists to discuss whether the trial should continue.

5 Conclusions

Our work is the first to apply an adaptive experimental study in intrusion data collection and discuss the benefits of collecting counterfactual information with a control group. We provide general details on running an experimental study with necessary factors to document prior to conducting the study. Our AD method extends this by optimizing resource allocation based on events seen at every stage, ensuring the statistical confidence through power analysis based on updated exploitation likelihoods with the assumption that an event only occurs once per participant. Because the interventional data collected contains true relations between features known through experimentation, future statistical models trained with this data are given higher confidence in learning general trends. This method is especially applicable for security studies seeking to identify causal relations between a corruption and automated attacks in the wild.

We then implemented our method in a honeypot study, confirming that the corruption (an `ssh` vulnerability) increased the infection rate of misconfigured cloud servers. This study also found that while recording more intrusions in observational studies (i.e., in the vanilla trial), the presence of the control group (as in RCT and AD) enables us to identify the corruption effect. Our AD shows it is capable of confirming corruption effect than RCT, requiring only 33% of the total trial duration to conclude corruption effect and using 17% fewer honeypots to see 19% more attacks. Prior to conducting the study, we knew the corruption would increase infection rate because attackers were provided more options for password entry, including the control's "password" for the same user accounts. Had the difference due to the corruption been less apparent (e.g., in altering multiple points of entry or limiting sequences

of vulnerability exploits), our study would have taken more time and resources to collect evidence.

Future work should consider implementing multiple vulnerabilities to study the interaction of corruptions. For example, one could add vulnerable applications within the honeypots to either study the scanning and exploit of multiple existing programs or tracing the sequence of exploits from the `ssh` vulnerability to a vulnerable application. This would require introducing a new methodology that can simultaneously consider multiple treatment arms, such as REMAP-CAP [45]. By isolating causal relationships, we hope these data can assist in generalizing solutions, remove some bias in the data, and enable other improvements in the intrusion detection community.

References

[1] Spamhaus Malware Team. Spamhaus botnet threat update: Q4-2021. *Spamhaus News*, 01 2017.

[2] Matt Muir. Real-world detection evasion techniques in the cloud. Black Hat Europe, 2022.

[3] Sema K. Sgaier, Vincent Huang, and Grace Charles. The case for causal ai. *Stanford Social Innovation Review*, 18(3):50–55, 2020.

[4] Penelope P. Howards. An overview of confounding. part 1: the concept and how to address it. *Acta Obstetricia et Gynecologica Scandinavica*, 97(4):394–399, 2018.

[5] MIT Lincoln Labs. 1998 darpa intrusion detection evaluation dataset, 1998.

[6] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. IEEE, 2009.

[7] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374, 2012.

[8] Melissa J. M. Turcotte, Alexander D. Kent, and Curtis Hash. *Unified Host and Network Data Set*, chapter Chapter 1, pages 1–22. World Scientific, nov 2018.

[9] Seth T Hamman, Jack Mewhirter, Richard J Harknett, Jelena Vićić, and Philip White. Deciphering cyber operations. *The Cyber Defense Review*, 5(1):135–152, 2020.

[10] Alex Andrew, Sam Spillard, Joshua Collyer, and Neil Dhir. Developing optimal causal cyber-defence agents via cyber security simulation. *arXiv preprint arXiv:2207.12355*, 2022.

[11] Tudor Dumitras and Iulian Neamtiu. Experimental challenges in cyber security: a story of provenance and lineage for malware. In *4th Workshop on Cyber Security Experimentation and Test (CSET 11)*, 2011.

[12] Penelope P. Howards. An overview of confounding. part 2: how to identify it and special situations. *Acta Obstetricia et Gynecologica Scandinavica*, 97(4):400–406, 2018.

[13] Neil Dhir, Henrique Hoeltgebaum, Niall Adams, Mark Briers, Anthony Burke, and Paul Jones. Prospective artificial intelligence approaches for active cyber defence. *arXiv preprint arXiv:2104.09981*, 2021.

[14] Arun Bhatt. Evolution of clinical research: a history before and beyond james lind. *Perspectives in clinical research*, 1(1):6, 2010.

[15] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

[16] Sherilyn Houle. An introduction to the fundamentals of randomized controlled trials in pharmacy research. *The Canadian journal of hospital pharmacy*, 68(1):28, 2015.

[17] CH van Werkhoven, S Harbarth, and MJM Bonten. Adaptive designs in clinical trials in critically ill patients: principles, advantages and pitfalls. *Intensive Care Medicine*, 45(5):678–682, 2019.

[18] Nigel Stallard, Lisa Hampson, Norbert Benda, Werner Brannath, Thomas Burnett, Tim Friede, Peter K Kimani, Franz Koenig, Johannes Krisam, Pavel Mozgunov, et al. Efficient adaptive designs for clinical trials of interventions for covid-19. *Statistics in Biopharmaceutical Research*, 12(4):483–497, 2020.

[19] Philip Pallmann, Alun W Bedding, Babak Choodari-Oskooei, Munyaradzi Dimairo, Laura Flight, Lisa V Hampson, Jane Holmes, Adrian P Mander, Lang'o Odondi, Matthew R Sydes, et al. Adaptive designs in clinical trials: why use them, and how to run and report them. *BMC medicine*, 16(1):1–15, 2018.

[20] Niels Provos and Thorsten Holz. *Virtual honeypots: from botnet tracking to intrusion detection*. Pearson Education, 2007.

[21] Vincent Nicomette, Mohamed Kaâniche, Eric Alata, and Matthieu Herrb. Set-up and deployment of a high-interaction honeypot: experiment and lessons learned. *Journal in computer virology*, 7:143–157, 2011.

[22] Eric Alata, Vincent Nicomette, Mohamed Kaâniche, Marc Dacier, and Matthieu Herrb. Lessons learned from the deployment of a high-interaction honeypot. In *2006 Sixth European Dependable Computing Conference*, pages 39–46. IEEE, 2006.

[23] Samuel Kelly Brew and Emmanuel Ahene. Threat landscape across multiple cloud service providers using honeypots as an attack source. In *Frontiers in Cyber Security: 5th*

International Conference, FCS 2022, Kumasi, Ghana, December 13–15, 2022, Proceedings, pages 163–179. Springer, 2022.

[24] Stefan Machmeier. Honeypot implementation in a cloud environment. *arXiv preprint arXiv:2301.00710*, 2023.

[25] Christopher Kelly, Nikolaos Pitropakis, Alexios Mylonas, Sean McKeown, and William J Buchanan. A comparative analysis of honeypots on different cloud platforms. *Sensors*, 21(7):2433, 2021.

[26] Sean Peisert and Matt Bishop. How to design computer security experiments. In Lynn Futcher and Ronald Dodge, editors, *Fifth World Conference on Information Security Education*, pages 141–148, New York, NY, 2007. Springer US.

[27] Office for Health Improvement and Disparities. Randomised controlled trial: comparative studies. *GOV.UK*, 10 2021.

[28] Eduardo Hariton and Joseph J Locascio. Randomised controlled trials—the gold standard for effectiveness research. *BJOG: an international journal of obstetrics and gynaecology*, 125(13):1716, 2018.

[29] KK Gupta, JP Attri, A Singh, H Kaur, G Kaur, et al. Basic concepts for sample size calculation: critical step for any clinical trials! *Saudi journal of anaesthesia*, 10(3):328, 2016.

[30] Gail A. Van Norman. Drugs, devices, and the fda: Part 1: An overview of approval processes for drugs. *JACC: Basic to Translational Science*, 1(3):170–179, 2016.

[31] James Wason. Improving the efficiency of clinical trials with adaptive designs. *Research Design Service Blog*, 10 2019.

[32] Blair Franklin. The digital forecast: 40-plus cloud computing stats and trends to know in 2023. *Google Cloud*, 2023.

[33] Linda Martin, Melissa Hutchens, Conrad Hawkins, and Alaina Radnov. How much do clinical trails cost? *Nature Reviews Drug Discovery*, 16, 2017.

[34] Camelia Simoiu, Christopher Gates, Joseph Bonneau, and Sharad Goel. "i was told to buy a software or lose my computer. i ignored it": A study of ransomware. In *Proceedings of the Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, pages 155–174, 2019.

[35] Tian Lin, Daniel E Capecci, Donovan M Ellis, Harold A Rocha, Sandeep Dommaraju, Daniela S Oliveira, and Natalie C Ebner. Susceptibility to spear-phishing emails: Effects of internet user demographics and email content. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 26(5):1–28, 2019.

[36] Anil Somayaji, Yiru Li, Hajime Inoue, José M Fernandez, and Richard Ford. Evaluating security products with clinical trials. In *CSET*, 2009.

[37] Fanny Lalonde Lévesque, Sonia Chiasson, Anil Somayaji, and José M Fernandez. Technological and human factors of malware attacks: A computer security clinical trial approach. *ACM Transactions on Privacy and Security (TOPS)*, 21(4):1–30, 2018.

[38] Fanny Lalonde Levesque, Jude Nsiempba, José M. Fernandez, Sonia Chiasson, and Anil Somayaji. A clinical study of risk factors related to malware infections. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 97–108, New York, NY, USA, 2013. Association for Computing Machinery.

[39] Ting-Fang Yen, Victor Heorhiadi, Alina Oprea, Michael K Reiter, and Ari Juels. An epidemiological study of malware encounters in a large enterprise. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1117–1130, 2014.

[40] Leon Bošnjak and Boštjan Brumen. Shoulder surfing experiments: A systematic literature review. *Computers & Security*, 99:102023, 2020.

[41] Santiago Gil, Alexander Kott, and Albert-László Barabási. A genetic epidemiology approach to cyber-security. *Scientific reports*, 4(1):1–7, 2014.

[42] Vance W Berger, Louis Joseph Bour, Kerstine Carter, Jonathan J Chipman, Colin C Everett, Nicole Heussen, Catherine Hewitt, Ralf-Dieter Hilgers, Yuqun Abigail Luo, Jone Renteria, et al. A roadmap to using randomization in clinical trials. *BMC Medical Research Methodology*, 21:1–24, 2021.

[43] Jean-Pierre Pignon and Rodrigo Arriagada. Early stopping rules and long-term follow-up in phase iii trials. *Lung Cancer*, 10:S151–S159, 1994.

[44] Amal Kumar and Bhaswat S Chakraborty. Interim analysis: a rational approach of decision making in clinical trial. *Journal of advanced pharmaceutical technology & research*, 7(4):118, 2016.

[45] Derek C Angus, Scott Berry, Roger J Lewis, Farah Al-Beidh, Yaseen Arabi, Wilma van Bentum-Puijk, Zahra Bhimani, Marc Bonten, Kristine Broglio, Frank Brunkhorst, et al. The remap-cap (randomized embedded multifactorial adaptive platform for community-acquired pneumonia) study. rationale and design. *Annals of the American Thoracic Society*, 17(7):879–891, 2020.

[46] Matthew R Sydes, Mahesh KB Parmar, Malcolm D Mason, Noel W Clarke, Claire Amos, John Anderson, Johann de Bono, David P Dearnaley, John Dwyer, Charlene Green, et al. Flexible trial design in practice-stopping arms for lack-of-benefit and adding research arms mid-trial in stampede: a multi-arm multi-stage randomized controlled trial. *Trials*, 13:1–14, 2012.

[47] Jing Ning and Xuelin Huang. Response-adaptive randomization for clinical trials with adjustment for covariate imbalance. *Statistics in medicine*, 29(17):1761–1768, 2010.

[48] Kristian Thorlund, Shirin Golchi, Jonas Haggstrom, and Edward Mills. Highly efficient clinical trials simulator (hect): Software application for planning and simulating platform adaptive trials. *Gates Open Research*, 3, 2019.

[49] Lisa Sullivan. Time to event variables. *Survival Analysis*, Date Accessed: 13 January 2023.

[50] Derek Robinson, Keanelek Enns, Neha Koulecar, and Manish Sihag. Two approaches to survival analysis of open source python projects. *arXiv preprint arXiv:2203.08320*, 2022.

[51] National Institutes of Health (NIH). NIH and FDA Release Protocol Template for Phase 2 and 3 IND/IDE Clinical Trials, Notice Number: NOT-OD-17-064, May 2017.

[52] Gérard Wagener, Radu State, Thomas Engel, and Alexandre Dulaunoy. Adaptive and self-configurable honeypots. In *12th IFIP/IEEE international symposium on integrated network management (IM 2011) and workshops*, pages 345–352. IEEE, 2011.

[53] Yunusa Simpa Abdulsalam and Mustapha Hedabou. Security and privacy in cloud computing: Technical review. *Future Internet*, 14(1):11, Dec 2021.

A Function Definitions

Pseudo code function definitions used in Methods 1, 2, and 3.

Shared functions:

`Deploy(control=`n_1`, corrupted=`n_2`)`: Given the number of devices to observe in each group of the study (n_1 control devices and n_2 corrupted devices), deploy them and record logs from all devices in a centralized location.

`Wait(`t`)`: Wait the length of time specified by t.

`L = SaveLogs()`: Fetch logs and return them to be stored in variable L

`CleanUp()`: Shut down devices and clean up any infrastructure not kept at the end of the trial or stage.

`GetNumToDeploy(`b`, `t`)`: Given budget b and the maximum trial duration t, it returns the maximum number of devices N that can be observed within this study.

`isEarlyStop(L)`: Given the logs collected from the last stage, determine if the pre-specified early stopping conditions have been met and the trial must immediately terminate.

`PowerAnalysis(`p_1, p_2, α, β`)`: See Section A.1

`SurvivalAnalysis(L)`: See Section A.2

A.1 Power Analysis for Sample Size

We denote the trial as robust when it follows a strict calculation of the sample size to be deployed accounts for type I and type II error in the data. For this paper, we follow the equation from HECT [48] which follows **power analysis** to calculate the sample size:

$$N_{\text{total}} = 2 * \left\lceil \frac{((p_1 q_1 + p_2 q_2)(Z_{1-\alpha/2} + Z_{1-\beta})^2}{(p_1 - p_2)^2} \right\rceil \tag{2}$$

$N_{\textbf{total}}$ = Total sample size for the study group, which is later split across the different treatments and regions. We alter the equation to calculate two arms of studies as the same since the split will depend on the ratio found during the trial in interim analysis.

Z = critical Z value for a given α and β-based subscript

p_1 = Control Incidence: The assumed rate of the outcome occurring in the control group is initially $p_1 = 1\%$.

p_2 = Treatment Incidence: The assumed rate of an outcome occurring in the corrupted group based on our pilot study,[6] and conservative rounding is $p_2 = 40\%$.

q_1, q_2 = $1 - p_1$, $1 - p_2$ (respectively)

α = Type I Error: The probability of claiming an infection rate when it is not true. Our setting and the generally accepted probability in clinical studies is $\alpha = 5\%$.

β = Type II Error: The probability of not detecting an accurate infection rate when it is correct. The inverse $(1 - \beta)$ is known as the **power** of the study. In this study,[7] we assume a power of 90% ($\beta = 10\%$) because we know there is a large difference between the control and the corrupted.

In the medical adaptive design, the KM-provided likelihoods would directly form the new p_1 and p_2. This would encourage the model to decrease the number of death events seen within the trial, which would make sense in a healthcare setting. However, we wish to increase the number of death events seen so we invert the likelihoods from the KM, which we call the **risk rates** (RRs). The RRs are marginalised and passed to Equation 2 to get our new N_{total}. Unlike the RCT allocation of honeypots (equally splitting N_{total} between control and corrupted), AD uses the RRs to weight the allocation so regions and corruption assignment with higher RR are given more honeypots in the following stage. This is repeated at every interim analysis for the duration of the trial.

[6] Citation removed for anonymity.

[7] Generally accepted in clinical studies as *power* = 80% which means $\beta = 20\%$.

A.2 Survival Analysis for Updating Risk Rates

We assume in our AD study that each participant will have at most one event of interest before terminating the system, protecting it from becoming a launchpad or providing free resources to attackers. The events of interest are parsed out and passed to a survival function which calculates the likelihood of surviving, i.e., for a system to not see the event of interest. We chose to use a **Kaplan-Meier (KM) Function**, a popular approach for survival analysis [49, 50].

The KM function is calculated by updating the likelihood S upon every event of interest recorded. At $t = 0$, all participants are at risk and $S(t) = 1$. The remaining time steps update following this:

$$S_{t+1} = S_t \times \frac{N_t - D_{t+1}}{N_t} \tag{3}$$

where

N_{t+1} is the current number of participants at risk

D_{t+1} is the number of participants that have seen events of interest since t

If we have full information on when all participants within the study see an event of interest, the KM function is calculated when an infection is recorded. In clinical trials without full information on their patients, a time interval is set to check in with the patient (e.g. every month or year) to collect data and see if they are alive.