SANDIA REPORT

SAND2016-10426
Unlimited Release
October 2016

Using Machine Learning in Adversarial Environments

Warren L. Davis IV, Daniel M. Dunlavy, Yevgeniy Vorobeychik, Karin Butler, Chris Forsythe, Matt Letter, Nichole Murchison, Kevin Nauer

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: http://www.ntis.gov/search

SAND2016-10426
Unlimited Release
October 2016

Using Machine Learning in Adversarial Environments

Warren L. Davis IV, Daniel M. Dunlavy
Scalable Analysis & Viz
Sandia National Laboratories,
Albuquerque, New Mexico 87185

Kevin Nauer
Cyber Security Technologies
Sandia National Laboratories,
Albuquerque, New Mexico 87185

Matt Letter
Software Systems R&D
Sandia National Laboratories,
Albuquerque, New Mexico 87185

Karin Butler, Nichole Murchison,
Chris Forsythe
Human Factors
Sandia National Laboratories,
Albuquerque, New Mexico 87185

Yevgeniy Vorobeychik
Computer Science and Computer Engineering
Vanderbilt University

Abstract

Cyber defense is an asymmetric battle today. We need to understand better what options are available for providing defenders with possible advantages.
Our project combines machine learning, optimization, and game theory to obscure our defensive posture from the information the adversaries are able to observe.

The main conceptual contribution of this research is to separate the problem of prediction, for which machine learning is used, and the problem of computing optimal operational decisions based on such predictions, coupled with a model of adversarial response. This research includes modeling of the attacker and defender, formulation of useful optimization models for studying adversarial interactions, and user studies to measure the impact of the modeling approaches in realistic settings.

ACKNOWLEDGMENTS

We thank members of the team who contributed to the overall success of this work presented here. From Sandia National Laboratories, this includes Warren Davis (Principal Investigator), Curtis Johnson (Program Manager), Danny Dunlavy, Karin Butler, Chris Forsythe, Matt Letter, Nichole Murchison, Kevin Nauer, Bob Carr, Nathan Fabian, Brenda Medina, Kim Ta, Eric Buedell, and Sid Holman. From Vanderbilt University, this includes Yevgeniy Vorobeychik, Bo Li, Sixie Yu, Jiazhi Zhang, Liyiming Ke, Royce Mou and Brandon Arvanaghi.

CONTENTS

FIGURES

TABLES

1. INTRODUCTION

Intrusion/anomaly detection systems are among the first lines of cyber defense. Commonly, they either use signatures or machine learning (ML) to identify threats, but fail to account for sophisticated attackers trying to circumvent them.

This research addresses three key shortcomings of ML in adversarial settings: 1) resulting classifiers are typically deterministic and, therefore, easy to reverse engineer; 2) ML approaches only address the prediction problem, but do not prescribe how one should operationalize predictions, nor account for operational costs and constraints; and 3) ML approaches do not model attackers' response and can be circumvented by sophisticated adversaries.

This project specifically addressed our cybersecurity mission. It significantly advances the state of the art in anomaly/intrusion detection where intelligent, highly capable adversaries are involved. Through science and formal modeling of adversarial behavior, explicit accounting for operational constraints in deploying machine learning IDS tools, principled approaches to moving target defense, and the development of an adversarial machine learning simulation platform, we developed methods and tools that have the capability to substantially increase the costs and uncertainty for adversaries. This increased defensive capability will benefit the DOE, NNSA, and other national security partners.

2. ADVERSARIAL CLASSIFICATION

In a classical supervised learning setting one starts with a data set of instances generated according to some fixed distribution, and learns a function which (one hopes) effectively evaluates new instances generated from the same distribution. While this assumption is often reasonable, it is clearly violated in adversarial settings. For example, if machine learning is used for network intrusion detection, an intelligent adversary will try to avoid detection by deliberately changing behavior to appear benign. There is another important feature of most of the literature on machine learning techniques aimed at adversarial settings, such as network anomaly and intrusion detection. Almost universally, the predictions produced by the application of learning are conflated with operational decisions based on these predictions, even though these are conceptually distinct. This has important consequences. First, the adversary does not respond to predictions per se, but to operational actions based on these. For example, even if the prediction identified an input as malicious, as long as this prediction is not actualized in operations, the adversary would have no reason to change behavior. Second, learned predictions typically do not account for operational costs and constraints (although the literature on cost-sensitive learning attempts to do so to some degree). For example, even if an input is identified as malicious, it may not be worth the cost to act on it if the damage is minimal (say, if an input is a Windows virus and you have only Linux machines). Third, machine learning is best suited for the task of prediction, and in that sense we should focus on trying to develop the most predictive algorithm given the available data. Operational decisions, on the other hand, are most meaningfully an outcome of an optimization problem under uncertainty, and the adversary's response is naturally captured in this framework.

The main theoretical contribution of this project was to separate the problem of prediction, for which machine learning is used, and the problem of computing optimal operational decisions based on such predictions, coupled with a model of adversarial response. Formally, this can be treated as a bi-level optimization program of the following form:

$$\max_{q,a} U_D(q, a; p)$$
$$\text{s.t. } q \in Q, \;\; a = A(q).$$

where p represents the predictions of the learning algorithm, which is an input into the optimization problem, q is the decision function of the defender, operationally constrained to be in some set Q, $A(q)$ is a model of adversarial response to the operational decision function q, and $U_D(\cdot)$ is the utility function of the defender, which depends on the operational decisions q, adversarial response q, and threat predictions p.

2.1. Example of the Adversarial Machine Learning Framework

It is helpful to illustrate the above framework in a more concrete context. To this end, consider the problem of adversarial binary classification over a finite space $\mathcal{X}$ of inputs, where each input feature vector $\vec{x} \in \mathcal{X}$ can be categorized as benign or malicious. The defender, $\mathcal{D}$, starts with a data set of labeled instances, $\{(\vec{x}_1, y_1),...,(\vec{x}_m, y_m)\}$, which we assume to accurately represent the current distribution of input instances and corresponding categories. $\mathcal{D}$ then uses an algorithm of

choice, such as Naive Bayes, to obtain a probabilistic classifier $p(\vec{x})$ which assigns to an arbitrary input vector a probability that it (or, rather, a producer of it) is malicious. In traditional application of machine learning, adversarial or not, one would then use a threshold, θ , and classify an instance $\vec{x}$ as malicious if $p(\vec{x}) \geq \theta$, and benign otherwise, with adversarial aspects of the problem folded into the algorithm that derives the function $p(\cdot)$, but it is on this point that our approach diverges from current art. Specifically, we introduce a function $q(\vec{x}) \in [0,1]$ which prescribes a possibly randomized operational decision for an instance $\vec{x}$ given a prediction $p(\vec{x})$. We can now incorporate a simple operational constraint, letting c be the maximum fraction of possible inputs that is feasible to act on (e.g., inspect manually). To model adversarial response, suppose that the attacker attempts to circumvent the operational response $q(\vec{x})$ by changing his "ideal" instance $\vec{x}$ to another, $\vec{x}'$, attempting to keep it as close as possible to $\vec{x}$ but increase the probability of bypassing operational detection. We can model this behavior as utility-optimizing, for the utility function $\mu(\vec{x}, \vec{x}') = e^{-\delta\|\vec{x}-\vec{x}'\|}(1 - q(\vec{x}'))$, where the parameter δ can be viewed as the importance of being close to the preferred attack method.

Our model of the attacker assumes that the attacker knows the operational decision function $q(\vec{x})$. In this we model a sophisticated attacker who carefully probes the defender's intrusion detection system prior to engaging in malicious activity. Formally, the interaction between the defender and attacker(s) is a Stackelberg game, where the defender moves first by choosing $q(\vec{x})$, the attackers then observe the defender's move and jointly (and independently) respond by choosing their attacks $\vec{x}'$. We can now formulate the problem of optimizing operational decisions based on ML and with adaptive attackers as the following linear program:

$$\min_{\vec{q}\in[0,1]^{\mathcal{X}}} \sum_{\vec{x}\in\mathcal{X}} G\left[q(\vec{x})\left(1 - p(\vec{x})\right)\right] + V\left[p(\vec{x}) \max_{\vec{x}'} e^{-\delta\|\vec{x}-\vec{x}'\|}\left(1 - q(\vec{x}')\right)\right]$$
$$s.t. \quad 0 \leq q(\vec{x}) \leq 1 \qquad \forall \vec{x} \in \mathcal{X}.$$

G and *V* allow us to explicitly trade off the cost of false positives and false negatives.

2.2. Prior Work Related to the Adversarial Machine Learning Framework

Prior research in adversarial machine learning focuses on tuning the machine learning *algorithm* to make it more robust in the face either of adversarial manipulation of training data [7,9] or test data [1-4,6]. None of this literature performs either an extensive analysis of adversarial behavior, or optimal decision making given machine learning predictions. There are also game theoretic modeling efforts in the context of network security [8,10,12], but these generally too abstract to be practical. Our approach, however, aims to focus on detailed adversarial modeling, and operational decision making, taking machine learning predictions as given, which makes it easy to integrate with existing ML-based intrusion detection system tools, and provides a means to address the operational problems directly in a holistic modeling framework, as argued by Sommer and Paxson [11].

2.3. Advances in Adversarial Machine Learning Made During this Project

The work on this project advanced the state-or-the-art in modeling attacker-defender interactions as described above, theoretical guarantees associated with optimal or near-optimal security strategies associated with those models, and computational methods for solving the complicated optimization problems that were used to solve for these optimal strategies. The majority of the theoretical work focused on email security, network security, intrusion detection systems, and risk in data sharing.

Following the formulation of the attacker-defender Stackelberg games [28], the framework was applied to the problem of defending against an attacker trying to send spam email to users on a network. Feature analysis and evasion modeling led to understanding trades off between overfitting and feature selection in an adversarial setting [27]. The more complicated models were addressed computationally using a compact parity basis representation for the operational decision function, leading to a scalable algorithm for computing near-optimal randomized operational decisions [25]. And finally, studies involving users manipulating spam templates to evade security policies associated with spam blocking were performed to assess the efficacy of the near-optimal strategies found [20]. Those studies suggest that feature reduction, a common approach used in applying machine learning methods to large-scale data sets such as email text, can actually help facilitate attacker evasion. Researching the extent of this facilitation requires more work, and could be considered as follow-on research to the work presented here. The email security work was extended to models of multiple defenders and a single attacker, which is a typical scenario in most email systems with multiple users. Optimal strategies were found for such models [26], and theoretical guarantees for the associated Stackelberg games were [21]. Related to this work, but applicable to other security-related data sets, data sanitization for the purpose of securing sensitive email data, was also modeled using the attacker-defender models above [22].

Another application addressed using the attacker-defender framework included network security. Monitoring of networks to detect the propagation of malicious programs (e.g., malware, viruses, etc.) across interdependent entities (e.g., computers on an internal corporate network) is an important security problem that needs to be addressed by computer specialists today. The email work was adapted to this problem and studied from both the theoretical and practical perspectives. The work included addressing adapting adversaries [23], formulating the problem of attack-resilient sensor placement as the problem of selecting a subset from a set of possible observations, with the goal of minimizing the uncertainty of predictions [24], and the impact of using optimal randomized security policies on real-world problems [16]. New problem formulations were required for these more complicated models, leading to a novel mixed-integer linear programming formulation to compute a defender's best response and approximate the Nash equilibria of the game using this formulation [13].

The work on this project also included application of attacker-defender modeling framework presented above to several other applications as a demonstration of the general applicability of this approach. These applications included intrusion detection optimal strategies [18,19], resiliency assessments of recommender systems based on collaborative filtering techniques [17], and risk

associated with de-identified (i.e., anonymized) data sharing [15]. This general approach could be considered for other security applications as well, potentially leading to much follow-on research.

3. ADVERSARIAL GAME SIMULATION

In addition to the analytical work based in game theory and optimization, we decided to approach the problem from an adversarial game simulation perspective. In this context, there isn't an optimal adversarial attack, *per se*. Instead, there are gaps, defined by decision boundaries, in the defender's model. These gaps are likely caused by incomplete model data; the defender most likely has not been exposed to every possible attack, and can only build a model off of the events that have been seen thus far. However, gaps can also be caused by the model itself. For instance, algorithms like k-nearest neighbor and multi-layer perceptrons may construct small "islands" in feature space that capture some subset of malicious events. These islands may not be constructed in non-kernelized support vector machine implementations, and thus may miss this malicious subset for which there might be significant examples.

However these gaps are formed, it is the goal of the adversary to find an attack that falls into any of these gaps. They can do this by sending attacks and treating the responses as labels to a supervised machine learning problem.

3.1. Preliminary Adversarial Game Machine Learning Instantiations

This adversary-defender interaction can be simulated with real data and machine learning algorithms. The exact nature of the interactions will vary depending on the dataset, the adversary's machine learning approach, and the nature of the gaps (small gaps from a good defender model, gaps that occur with some learnable regularity, etc.). To explore design issues, we decided to develop simulations for a few representative adversarial machine learning games.

We chose to instantiate our view of adversarial game simulation utilizing the Hybrid Toolkit [35]. This toolkit has been successfully used in Sandia's cyber incident response environment. In addition, the information processing pipeline nature of Hybrid lends itself quite naturally to the adversarial game set up. Because of this, we decided to implement our adversarial game experiments using the Hybrid Toolkit.

Preliminary experiments with machine learning adversaries were performed in various domains and with varying datasets.

- NumSig – feature vectors consisting of various floating points numbers were created, with the area of interest forming a gap in the numerical feature space. This is a quite literal representation of the problem.
- Ling-Spam – Using the Ling-Spam dataset, the problem is to identify spam using keywords and LDA feature vectors
- Mastermind – the Mastermind game is played, with the adversary knowing nothing about the game itself and choosing the next response based on the feedback. Obviously, this is much harder than if game knowledge is embedded. Nevertheless, an adversarial machine
- learning algorithm was able to learn the correct code from the defender responses.

In all these experiments, the adversaries were able to learn. However, the importance of this task was mostly in fleshing out the necessary components of an adversarial gaming pipeline utilizing the Hybrid Toolkit.

3.2. Preliminary Human Experimentation

As we were studying the aspects of machine learning adversaries, we felt it might be useful to study human adversaries in this context. Doing so would afford several benefits. For one, it would allow us to understand how humans learn patterns in contrast to machine learning algorithms on identical datasets. In addition, human experimentation would allow us to to test obfuscation strategies and their effect on learning. If the learn and obfuscation strategies were similar, then it would allow defenders to transfer policy decisions between adversarial contexts. If the learning and obfuscation strategies were different, characterizing this can enhance attribution.

To aid in this investigation, we enlisted Chris Forsythe (0431) to begin human experimentation on a pattern recognition task in parallel to the machine learning investigations. He enlisted Brenda Medina and Sid Holman (0431) to assist in user interface design and data capture.

3.2.1. Method

3.2.1.1 Subjects

Subjects consisted of 16 employees of Sandia National Laboratories who responded to a lab-wide announcement in the laboratory Daily News.

3.2.1.2 Materials

Stimulus materials were presented through a java-based software routine. The study began with an instruction screen. The instructions were as follows:

> *"You will be presented with two buttons. On each trial, one of the two buttons is the correct selection. After pressing a button, you will receive feedback indicating if you made a correct selection.*
>
> *The correct selection will be based on a rule and it is your task to discover the rule, and then, apply the rule to produce the correct sequence of button presses. For example, the rule may be that you press the left button on one trial, then the right button on two consecutive trials, and then, repeat the sequence. If you make an incorrect selection, the sequence will continue with the next item in the sequence.*
>
> *Once you have demonstrated the correct sequence by producing it over a series of trials, you will be notified that you have successfully completed the round and allowed to progress to the next round."*

On each trial, subjects were presented two side-by-side buttons. If they chose the correct button, the button turned green and the word "Correct" was displayed. If they chose the incorrect button, the button changed to red and the word "Incorrect" was displayed. To progress to the next round, subjects had to select a third button below and equidistance from the other buttons. The time from between the beginning of a round and the selection of one of the buttons was recorded. Also, the time between the button choice and selecting the button to progress to the next round was recorded.

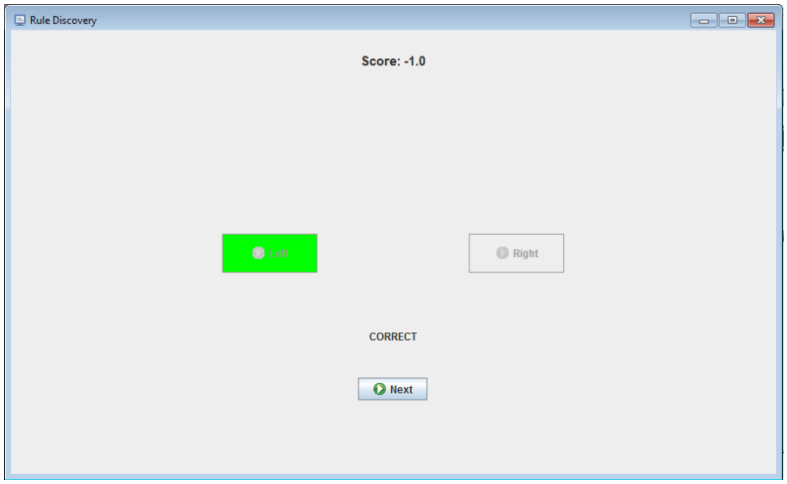

Figure 1. A screen shot of the user interface for the preliminary human trials.

Sequences were constructed in which there were no more than three consecutive button presses on each side. There were two conditions with half of the subjects assigned to each condition. The Simple condition consisted of the following sequences:

Patterns	Sequences
Balanced Alternating	LR, LLRR, LLLRRR, RL, RRLL, RRRLLL
Unbalanced Alternating	LRR, LRRR, LLR, LLRRR, LLLR, LLLRR, RLL, RLLL, RRL, RRLLL, RRRL, RRRLL

In each round of the simple condition, one of the possible sequences was randomly selected.
The second condition combined Simple and Complex sequences. Complex sequences included escalating and descending patterns. Patterns could escalate or descend by either one or two items. The complex sequences included the following:

Patterns	Sequences
Balanced Escalating	LRLLRRLLLRRR, LRLLLRRR, RLRRLLRRRLLL, RLRRRLLL
Balanced Descending	LLLRRRLLRRLR, RRRLLLRRLLRL, LLLRRRLR, RRRLLLRL
Unbalanced Escalating	LRLRRLRRR, LRLLRLLLR, LRLRRR, LRLLLR, RLRLLRLLL, RLRRLRRRL, RLRRRL, RLRLLL
Unbalanced Descending	LLLRLLRLR, LRRRLRRLR, LLLRLR, LLLRLR, RRRLRRLRL, RLLLRLLRL, RRRLRL, RLLLRL

In each round of the Simple and Complex condition, a sequence was randomly selected from either the Simple or Complex set of sequences.
There were twelve rounds. In each round, if subjects made twelve consecutive correct selections, it was concluded that they had correctly determined the sequence. A message was presented stating that they had successfully completed the round and could proceed to the next round. If subjects had failed to produce twelve consecutive correct selections through 100 trials, a message was presented stating that the round was complete and that subjects could continue to the next round.

At the top of the screen, a score appeared that was cumulative across trials and rounds. For each correct selection, subjects received two points and for each incorrect selection 5 points were deducted from their score. In rounds in which subjects produced twelve consecutive correct selections prior to the end of the round, they were awarded the points they would have received for correct selections on each of the remaining trials.

3.2.1.3. Procedure

Subjects were first asked to read the Instructions screen. Then, prior to beginning the first trial, subjects were given additional instructions. These additional instructions consisted of explaining that there was a cumulative score and the objective was to get a high score. Subjects were told that correct selections would add two points to their score and incorrect selections would deduct five points. It was explained that if they produced twelve consecutive correct selections, it would be concluded that they had determined the pattern and they would receive the points for correct selections in the remaining trials resulting in the possibility of receiving large blocks of points. Also, it was explained that the discrepancy served to discourage camping (i.e., repeating the same selection until the pattern was recognized) by imposing a greater cost for incorrect selections. If subjects had no questions, they were then allowed to proceed at their own pace.

3.2.2. Results

The first consideration addresses what facets of the sequences contributed to their relative ease or difficulty. There were three measures that reflect upon the difficulty of a given sequence. First, did the subject determine the sequence prior to the end of the round? If the subject did determine the sequence, how many trials did they take? Finally, the time which elapsed prior to making a selection and the time that elapsed after making a selection before progressing to the next round offer indirect indications of difficulty. The latter two measures assume that subjects take longer when confronted with more difficult sequences.

3.2.2.1. Successful Rule Discovery

Subjects successfully discovered the rule within the 100 trials allotted in 158 of 192 (82%) of the rounds. Figure 1 shows the percentage correctly discovered for each type of pattern. A stepwise regression was calculated in which rule discovery served as the dependent measure with four predictors: (1) round; (2) balanced/unbalanced; (3) escalating-descending/alternating; and (4) sequence length. The resulting equation had a two terms, sequence length (t=-8.09; p<0.001) and balanced/unbalanced (t=-1.70; p<0.10), with R-squared = 27.22 (R-squared (adj) = 26.45). This suggests that the primary factor determining the likelihood of discovering rules was the length of the sequence, with symmetrical sequences being somewhat easier than asymmetrical sequences. This finding is illustrated in Figure 2. It is not surprising that sequence length exerted a significant influence on rule discovery given previous research has shown that rule discovery is highly reliant on working memory capacity.

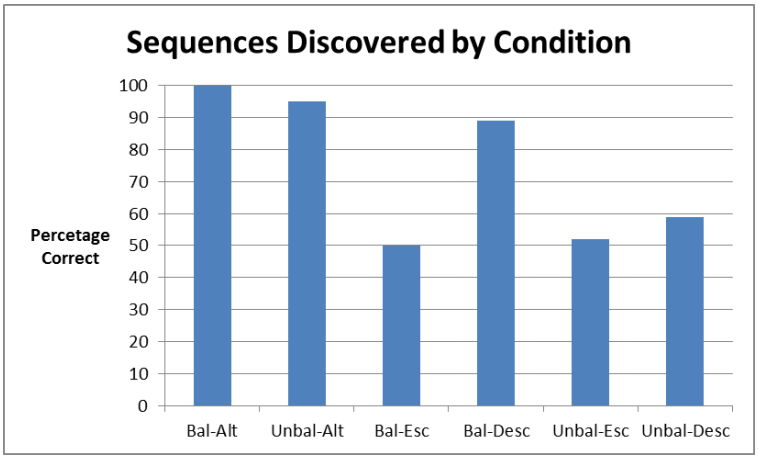

Figure 2. The percentage of sequences discovered for each type of sequence.

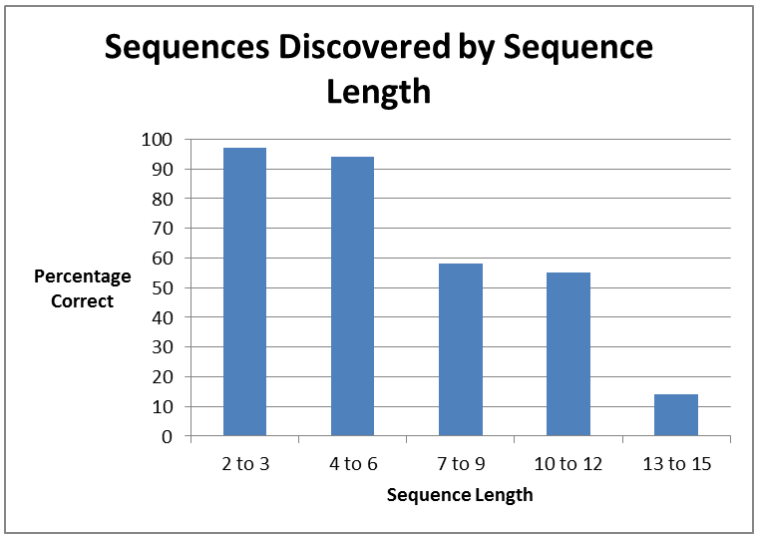

Figure 3. The percentage of sequences discovered for different sequence lengths.

The second measure of difficulty was the number of rounds required to discover a sequence. For this analysis, trials were omitted in which subjects failed to discover the rule within the allotted 100 trials. A stepwise regression revealed four predictors, with R-squared = 38.08 (R-squared (adj) = 36.47). The four predictors were Esc-Desc/Alternating (t=3.52; p<0.001); Round (t=3.71; p<0.001); Sequence Length (t=1.81; p<0.08); and Balanced/Unbalanced (t=1.78; p<0.08). These results indicate that sequences in which the number on each side could increase or decrease over a series of trials (e.g., LRLLRLLLR) were more difficult than sequences in which the number on each side remained the same (e.g., LLRRRLLRRR) (See Figure 3).

Additionally, as subjects progressed over a series of rounds, they required less time to discover sequences. Sequence length and Balanced/Unbalanced were both factors, but had a smaller influence. To assure that the results were not attributable to omitting many of the more difficult sequenced, a follow-on analysis was conducted in which 100 was inserted for the number of trials in each round that the subject failed to discover the pattern. This analysis yielded comparable results with R-squared = 49.31 (R-squared (adj) = 48.23) and the same four predictors: Esc-Desc/Alternating (t=4.26; p<0.001); Round (t=-3.26; p<0.001); Sequence Length (t=3.47; p<0.001); and Balanced/Unbalanced (t=2.20; p<0.03).

A consideration of the time that elapsed prior to making a button selection revealed two predictors (R-squared = 24.61 (R-squared (adj) = 23.81). These predictors were Round (t=-6.25; p<0.001) and Esc-Desc/Alternating (t=4.66; p<0.001). As the experiment progressed, subjects tended to respond more quickly and sequences in which the number on each side could increase or decrease were more difficult than those in which the number of each side was constant. The results were similar when analyzed the time that elapsed following conclusion of one trial and commencing the next (R-squared = 21.62 (R-squared (adj) = 20.79). The same two predictors emerged: Esc-Desc/Alternating (t=6.31; p<0.001) and Round (t=-3.41; p<0.001).

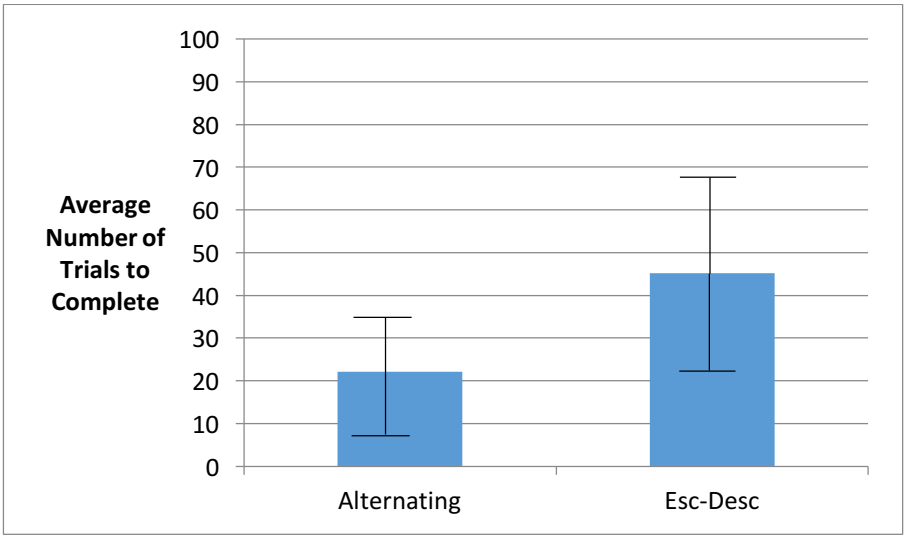

Figure 4. Average number of trials till successful rule discovery for sequences in which the number on each side could change over trials (Esc-Desc), or remained constant (Alternating).

3.2.2.2. Strategies

Post experimentation interviews yielded several strategies that the users believed they employed. A few salient ones are listed below:

- Guessed until had consecutive correct responses, then tried to build sequence one step at a time
- Assumed had seen the whole pattern, then tested assumption, updating hypothesis with each test
- Repeatedly selected one side and counted the number correct, and after incorrect selection, switched and repeatedly selected the other side counting the number correct
- Focused on finding the beginning of the sequence, and when thought had beginning, counted the number on each side
- Focused on determining the length of the sequence, and then determining the actual pattern
- Focused on identifying landmarks (i.e., recurrent sequences), and determining the sequence around these landmarks
- Tried to memorize and reproduce the sequence based on past responses

3.2.3. Discussion

One of the most useful lessons from this preliminary experimentation was that humans learning patterns from responses was clearly parallel in many ways to the previous machine learning experiments. We could study learning based on the difficulty of the embedded patterns, and analyze the increase in incorrect guesses due to increasing difficulty. In addition, we could easily see how to embed obfuscation into these trials, and test the effect using the same techniques. We decided that, for the rest of the research, we should incorporate the idea of human adversaries in a way that would allow for joint study. All ensuing simulation conceptualization and architectural designs are based on the being able to substitute human and machine learning adversaries in a plug and play manner.

3.2. Adversarial Gaming Simulation Concepts

One thing we noticed early was that many of the concepts involved in creating simulations of these games were consistent across games. The concepts are modeled in the event flow diagrams below in Figure 5:

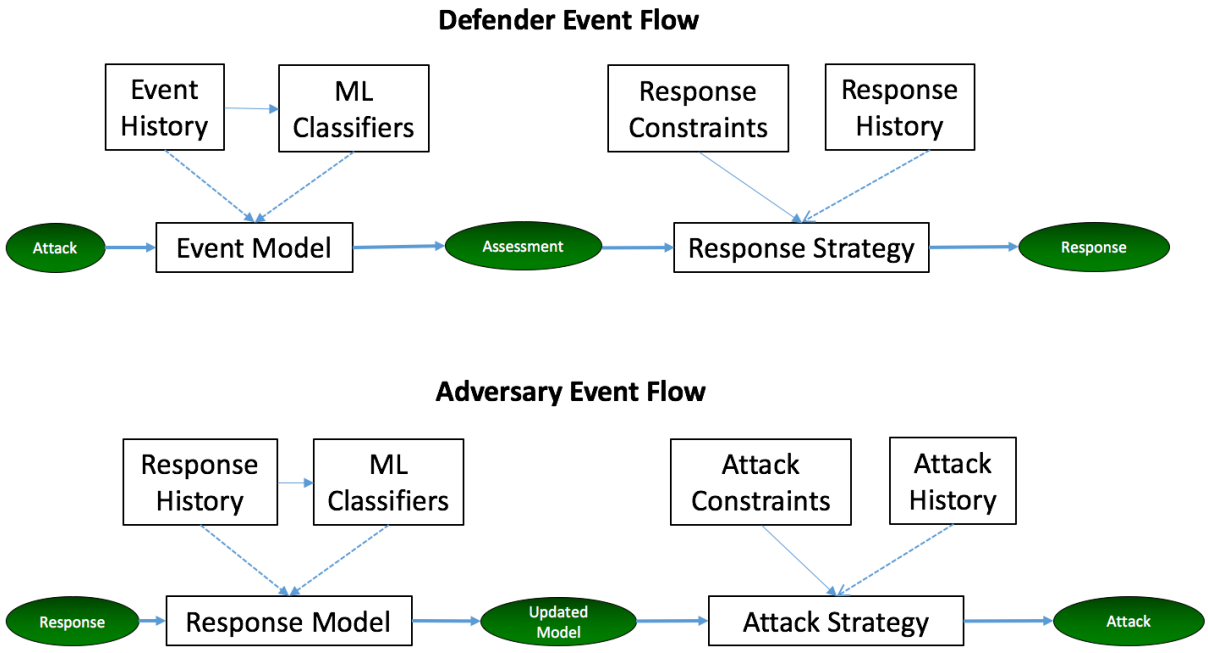

Figure 5. Defender and Adversary Event Flows.

Solid arrows show required information flows, whereas dashed arrows represent optional flows. For instance, a defender's model does not have to incorporate a machine learning classifier. It could simply be a rule based model, informed by previous event history, such as Snort rules.

For the defender, the basic flow is that when an event is received (such as an attack from the adversary), it is processed through an event model which returns an assessment of the classification of the event. This assessment then passes through a response strategy. This is where obfuscation can occur, perhaps informed by previous responses. Responses have constraints that are very specific to the domain, and this must be incorporated into the response strategy as well.

The adversary has a parallel flow, receiving responses, updating the predictive model for new attacks, and generating new attacks, constrained by domain specifics and chosen based on its probability of success, given the updated model.

Both the defender and attacker can make models out of the events they receive. This is not necessarily happening synchronously. In fact, with the event flows above, you can model both static and dynamic defenders, with or without obfuscation. Modeling the static adversary is also possible, however outside of the scope of this research. The four modes of event flow that are of interest to this research are as follows:

- Static
  - Defender has a set of rules used to govern classification
  - Defender applies the rules consistently

    - Examples include Mastermind and virus signatures (at least for some period of time)
- Static - Obfuscated
    - As Static, except that the defender can lie about responses. This represents a separation between the prediction model and the operational decisions discussed in Section 2.
- Dynamic
    - The defender has a set of rules, but these rules can change in response to a detected attack, as the defender updates its model
- Dynamic - Obfuscated
    - The defender can deceive and learn from experience

We found that it was very easy to look at adversarial games and find where they fit in the four modes, then to formulate them in a different mode. For instance, even though Mastermind is a Static game, it could exist as a Dynamic – Obfuscated game, where the *codemaker* is allowed to lie about the responses (with a cost, of course), and can also updated the code (under certain constraints) as they receive pattern guesses.

We also found that, since all of the games we researched could be described in the aforementioned flows, it was conceivable that one could build a platform with reusable components to instantiate all these simulations. The attack and response constraints were the only parts that were not generalizable, as they were dependent directly on the problem domain.

As these commonalties were being fleshed out, the implantation in the Hybrid Toolkit was mirroring these ideas, and we began to develop a generalized architecture for adversarial game simulations.

3.3. Adversarial Gaming Simulation Architecture

The following diagram in Figure 6 shows the components and information flow in the adversarial gaming simulation platform, implemented in the Hybrid Toolkit.

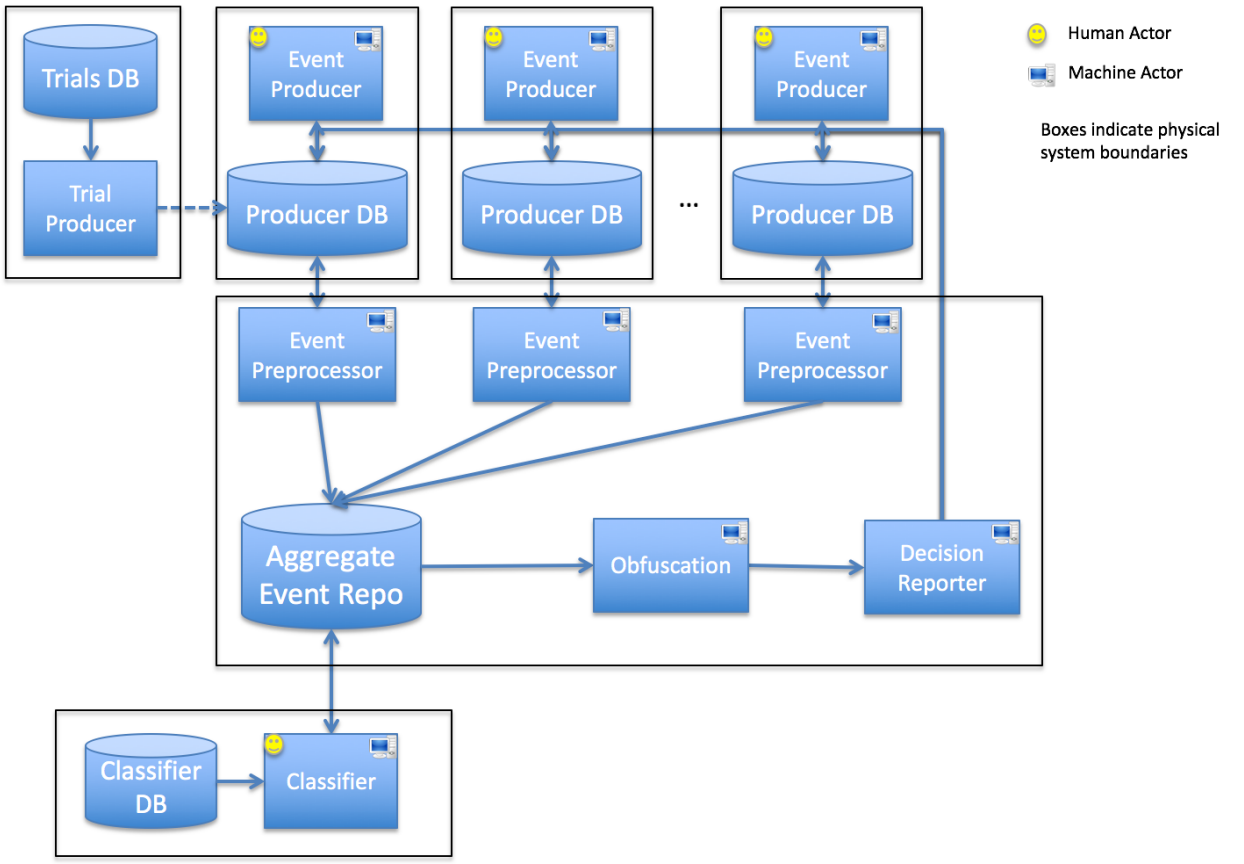

Figure 6. Adversarial Gaming Simulation architecture in the Hybrid Toolkit.

Some components, as shown in the diagram, can be instantiated by human or machine actors. This creates a flexible platform, capable of testing all combinations of human/machine and defender/adversary assignments.

The roles of the components are as follows:

- The Trial Producer, if implemented, utilizes pre-generated attacks in the Trials DB to create trials for the event producer. This allows the adversary to pick from a selection of attacks rather than creating new attacks.
- The Event Producer either creates an attack or selects from a list of valid attacks in the trial, attempting to choose the one with the highest probability of success.
- Producer DB contains the events from the event producer (the choice of query and unique identifier)
- Event Preprocessor adds any additional information necessary for system level processing, such as client URL info.
- Aggregate Event Repo holds the events that will be analyzed by the classifier
- Obfuscation decides whether to obscure the results of the classification (allowing those with a malicious classifier determination to pass)
- Decision Reporter extracts the final event status and event ID and stores that into the Producer DB. No other information is passed back.

- Classifier extracts features from the events and makes a determination as to whether it is malicious, adding that label back into the Aggregate DB
- Although not shown, if the event producer is utilizing machine learning, it will have its own Classifier DB and Classifier, similar to the defender.

3.4. System Specifications

The Adversarial gaming simulation (AGS) platform is currently running on a 16 CPU Xeon 5500/Core i7 Dell power6 edge server with 68 Gigabytes of ram and 4 Terabytes of disk storage.

The AGS front end framework stack is composed of an AngularJS web application using an MVC framework with RESTful web service connections. HTML was used to construct the application view, with CSS used for styling. AngularJS was used for the controller logic and as a web service interface. Grunt was used for deployment.

The simulation application has two state views. The first view is a login state with a username and category HTML form with CSS styling. The second view is an angular repeater reflecting a trial object with another angular repeater reflecting current progressing in the application both styled with CSS.

The AGS back end stack was composed of a CherryPy web server, hybrid pipeline, and Mongo database. The CherryPy web server was used to serve the front end web pages and as a RESTful python web server for interacting with the Mongo database. A Hybrid Toolkit pipeline was leveraged for processing of incoming trial selections where the pipeline determined selection quality and response.

The build dependencies include the following:

- Python 2.7.x
- CherryPy
- PyMongo
- The Hybrid Toolkit v1.0

4. LAB INTEGRATION

One of the desired outcomes for this research was not only to study adversarial machine learning analytically and create a simulation platform, but also to integrate this simulation platform into a Sandia cyber research environment for continued study. To do this, we would need a venue for integration and test case that facilitated both human and machine learning experimentation.

4.1. RECOIL Lab

The Research & Engineering for Cyber Operations & Intelligence Lab (RECOIL) at Sandia was created as a multidisciplinary collaboration focal point, bringing together academia, industry, and government. This lab, run by Kevin Nauer (9312), engages in cyber exercises involving cyber security practitioners, psychologists, sociologists, and computer scientists. The primary emphasis of RECOIL is cybersecurity experiments that utilize 1) case study analysis of adversary techniques and exploit methods; 2) big data analytics and machine learning; and 3) cognitive psychology and cognitive neuroscience.

RECOIL also sponsors the Consortium Enabling Cybersecurity Opportunities & Research (CECOR). Through this, RECOIL interacts directly with academia to promote cybersecurity awareness and training. Several universities engaged in the program have expressed interest in utilizing the Hybrid Toolkit, which serves as the foundation for the adversarial gaming simulations. Collaboration in this space could be a great benefit to enhance the core capabilities of the Hybrid Toolkit, Sandia's own cyber incident response, RECOIL exercises, and benefit to participating institutions.

Because of the focus of RECOIL exercises, the collaboration opportunities with CECOR, and the lab's unique facilities and resources, we feel that RECOIL aligns perfectly with the adversarial machine learning research. We are interested in studying both human and machine learning as applied to a cybersecurity context, both with and without the presence of obfuscation.

To our benefit, Kevin Nauer joined the research team and began helping us brainstorm about simulations that would be relevant in a cyber context. Nauer organized various team members to help us generate experiments and data. This was useful as we were able to gain a greater understanding the types of exercises performed in RECOIL.

Many RECOIL exercises involve both cyber-spelunking and information synthesis components. Participants are given a problem which has clues which can be ascertained forensically, but only make sense in the context of other external information. To this end, much of the cyber data generated for this tasks can be automated through Tracer, the RECOIL environment for producing these events.

We proposed a system where the adversarial gaming simulations can amend these activities, and provide a framework for studying scenarios in which the adversaries or machine learning defense tools can adapt over time, whilst also incorporating deception from the cyber defenders (obfuscation). This can significantly expand the types of activities performed in RECOIL, allowing for a more diverse set of participant experiences. In addition, RECOIL can become a center for adversarial gaming research for SANDIA, accentuated by its multidisciplinary nature and external collaborations.

4.2. Experimentation Test Data

In order to instantiate an adversarial gaming platform, we needed to have a data set to build defender classifiers.

The data requirements are as follows:

- Human/Machine generable – Both humans and machines need to be able to generate events or select from a group of pre-generated events. Because of this, the study of phishing emails we found to be somewhat limited. Having machines auto-generate phishing emails is a separate research path by itself, and collating pre-generated phishing emails that would have an embedded detectable signal and could be discussed openly seemed to be beyond our resources as well.
- Human comprehensible – Although various malware repositories exist, it is extremely difficult, if not impossible for a human to look at the bytes, hashes, etc., and understand anything about the nature of the software in any reasonable time frame. The test observations need to be reasonably understandable by humans within a short period of time (e.g., a minute or two).
- Machine parsable – In order to create tests for a machine learning adversary, the data has to be parsable by automated means, with relevant features extracted for presentation to the machine classifier.
- Labeled events – Unfortunately many datasets have data but do not clearly label malicious events. Other times, there many events, and it is more so something about a combination of events that is malicious rather than one particular event, itself. This is fine, however it adds to this research the additional component of defining events from large temporal datasets (such as packet captures), which is a problem better suited to domain experts for a particular problem, and can conflate this study even more than standard feature extraction on pre-defined events.

The process to create data for testing an adversary's ability to find holes in the cyber defenses are as follows:

1. Collect both normal and malicious events feature vectors. This is defined as collection *A*
2. Train a classifier on *A* which exhibits some specified behavior (e.g., high precision and recall, low precision, high recall, etc.).
3. Collect all the True Positives (malicious events that were detected as malicious). This is now collection *B*.

4. Build another classifier with the normal events from *A* and collection *B*. This classifier should be different or impaired in some way as to produce both true positives and false negatives. The true positives form the set of things that the defender detects and blocks, and the false negatives are the adversary's target, those things which the defender's classifier will not recognize as malicious.

Steps 1 and 2 simulate building a predictive model based on real data. The accuracy to which this classifier is trained is not overly important for adversarial testing, but different behaviors can be useful in certain specific instances. If one is trying to simulate a very conservative system, one should train a classifier which has high true positives but also has high false positives. Choosing a classifier which gives high true positives and low false positives creates a very narrow gap for the adversary, and is intuitively the desired case for cyber defenses. Step 3 is necessary so that it guarantees that there is a signal in the observation which can actually be detected. In other words, it ensures that there is at least one classifier (the one built in Step 2) that can detect the malicious event. Step 4 builds the target defenses that contain the gap that the adversary will target.

For defender classifiers, we chose the SciKit-Learn Decision Tree Regressor. We felt that the decision axes would, by their nature, allow for somewhat definable and explainable classification gaps. However, we feel that any classifier would have been sufficient for this task, as we are more interested in testing the adversary's learning ability than the cleverness of the defender classifier. The combinatorial effects of different styles of defender classifiers vs. different human and machine learning adversaries we shall leave for future research.

We were not able to find any publicly available datasets that met these criteria. One of the more famous cyber datasets is KDD Cup 1999 Data. It consists of 4,000,000 network intrusion attack vectors, with 42 categorical and integer features [30]. Though this dataset received a lot of early attention, its use has waned in ensuing years as problems with the dataset have surfaced. A detailed list of some problems is included in [36].

Nevertheless, we attempted to use this dataset as it was one of the few that met the aforementioned data requirements . We followed the dataset creation steps and produced the data necessary for testing.

Unfortunately, the problems in the dataset came to bear on this task. Most importantly, in order to create sufficient numbers of false negatives that contained a discriminatory signal, we had to hinder the machine learning algorithms in ways that made the decision boundary gaps painfully obvious to the casual observer. For particular classes within the KDD Cup 1999 dataset, creating these gaps was not even possible.

The other cyber-oriented datasets failed to meet the other data requirements as well. We decided to explore generating our own data for the experimentation.

4.3. SQL Injection Detection

We discussed plans to create synthetic cyber data, and engaged RECOIL members to explore what options were available. Because of recent research in the area, we decided that the detection of SQL Injections would not only give data that met the requirements, but could be interesting and relevant cybersecurity research in its own right.

Kim Ta (9312) and Eric Buedel were tasked with creating a data set for experimentation. Ta used SQLMap (http://sqlmap.org), and open source penetration tool, to generate queries against a VM sample Microsoft SQL Server database stack. Buedel was tasked with researching the conversion of the SQL queries to feature vectors. A lot of progress has been made in this areas, such as in research by Kar *et. al* [29].

During this research, we sampled some of the preliminary results from Ta's processing pipeline. Some queries were reasonably humanly comprehensible. Below is an example of the SQL Map output of such a query:

```
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: bill_month=8 AND 8260=8260&sec=HSPO14
```

However, we found that the nature of the SQL injections that were being produced in general were much more complicated:

```
Type: error-based
Title: Microsoft SQL Server/Sybase AND error-based - WHERE or
HAVING clause
Payload: bill_month=8 AND 1339=CONVERT(INT,(SELECT
CHAR(113)+CHAR(106)+CHAR(118)+CHAR(112)+CHAR(113)+(SELECT (CASE
WHEN (1339=1339) THEN CHAR(49) ELSE CHAR(48)
END))+CHAR(113)+CHAR(120)+CHAR(118)+CHAR(122)+CHAR(113)))&sec=HS
PO14
```

Some were much worse:

```
Type: UNION query
Title: Generic UNION query (NULL) - 6 columns
Payload: bill_month=8 UNION ALL SELECT
NULL,NULL,NULL,CHAR(113)+CHAR(106)+CHAR(118)+CHAR(112)+CHAR(113)
+CHAR(122)+CHAR(79)+CHAR(114)+CHAR(119)+CHAR(77)+CHAR(79)+CHAR(8
7)+CHAR(110)+CHAR(98)+CHAR(103)+CHAR(66)+CHAR(68)+CHAR(85)+CHAR(
113)+CHAR(87)+CHAR(68)+CHAR(65)+CHAR(72)+CHAR(113)+CHAR(110)+CHA
R(76)+CHAR(71)+CHAR(119)+CHAR(97)+CHAR(73)+CHAR(78)+CHAR(73)+CHA
R(82)+CHAR(119)+CHAR(117)+CHAR(103)+CHAR(70)+CHAR(74)+CHAR(100)+
CHAR(98)+CHAR(83)+CHAR(105)+CHAR(105)+CHAR(106)+CHAR(80)+CHAR(11
3)+CHAR(120)+CHAR(118)+CHAR(122)+CHAR(113),NULL,NULL--
fwYj&sec=HSPO14
```

After discussion between Davis, Nauer, and Butler, we decided to terminate the SQL injection path. The majority of the queries were not going to be parsable by humans in a reasonable enough time to facilitate the numbers we would need. In addition, the decision boundary for these queries we thought might be non-learnable without domain expertise, which would further limit our participant pool.

We believe that, if feature vector extraction techniques are employed, this would still be a valid test set for machine learning experiments or reasonable numbers of appropriately trained human adversaries.

4.4. Simple Function Data

In the end, we decided to create a simple synthetic dataset that was amenable to the aforementioned data requirements. Using three variables, a, b, and c, we produced random data, where each variable was drawn from a uniform distribution in $[0,1)$, multiplied by 100, and transformed into an integer by truncating towards 0. We then used a function between the variables to create labels. The function defining maliciousness in this data was specifically the following:

$$\text{malicious } iff\ a < b * 2 + 1$$

Notice that the function does not actually use variable c. This was intentional, to establish a nuisance variable and to allow for spurious correlations. This classification function applies to the labeled training set for the initial defender classifier, not the actual rules which the adversary is trying to learn. The adversary is, instead, learning the function of data that the defender classifier *misses*, which is a much more complex function, and will be discussed in Section 5.1.

Unfortunately, this synthetic data separates us from the realism we were hoping to invoke in this game. We will be working with RECOIL in the future to create new games based on more realistic data. This is also another example that argues for more publicly available, processed, and labeled cyber datasets for machine learning research.

Nevertheless, the synthetic function data exhibited the properties we needed, and were able to create a dataset that had false negatives for a particular classifier, in which those false negatives were successfully detected as true positives by a different classifier. The resulting dataset, after examination by Butler, exhibited no obvious regularities that would be instantly exploited, and was thus deemed usable for further human and machine learning adversary experimentation. We created the necessary modules for our Hybrid Toolkit instantiation of our simulation, and incorporated the curated dataset into our simulation for experimentation.

5. HUMAN EXPERIMENTATION

One important dimension of the problem of adversarial learning is studying learning by non-machine learning adversaries. It is important to know whether the techniques developed to study obfuscation and machine learning transfer to human learners as well. Such information can be useful in adjusting the parameters of the obfuscation optimization. In addition, by understanding how humans react to obfuscation vs. machine learning algorithms, one can gain a useful tool for attack attribution.

The human experiment we performed tested whether false feedback would influence human learning of the characteristics of vulnerabilities in systems using an online learning portal. In this simulated hacker scenario, participants with little to no cybersecurity background were challenged to learn the weaknesses in the computer system by launching multiple attacks against the system and using the feedback received about the success and failure of each attack. Half the participants were given false feedback; on attacks that failed there was a 5% chance that the feedback was obfuscated, i.e., indicated success.

5.1. Method

5.1.1. Participants

Participants were 36 employees or contractors of Sandia National Laboratories recruited through ads in the Sandia Daily News, emails sent to student interns, and word-of-mouth. Eighteen were randomly assigned to the obfuscation group and 18 were randomly assigned to the no-obfuscation group. All participants were asked to indicate if they had experience with SQL injections, cybersecurity activities such as Red teaming and capture the flag, or machine learning. Only 9 people reported these types of experiences and the distribution did not obviously differ between the obfuscation group (6 individuals) and the no-obfuscation group (3 individuals).

5.1.2. Materials

Attacks were created that consisted of three whole numbers, labeled A, B, and C feature. Each number could range between 0 and 99. Using a classification algorithm, these attacks were separated by the process described in Section 4.2 into those that would be detected as malicious (failing to penetrate the defenses) and those that would not be detected as malicious (successful attacks). Successful attack vectors are plotted in Figure 1 and were defined by the system of equations in Table 1. As can be seen, the value of the number in the C-column did not have an effect on whether an attack was successful or not.

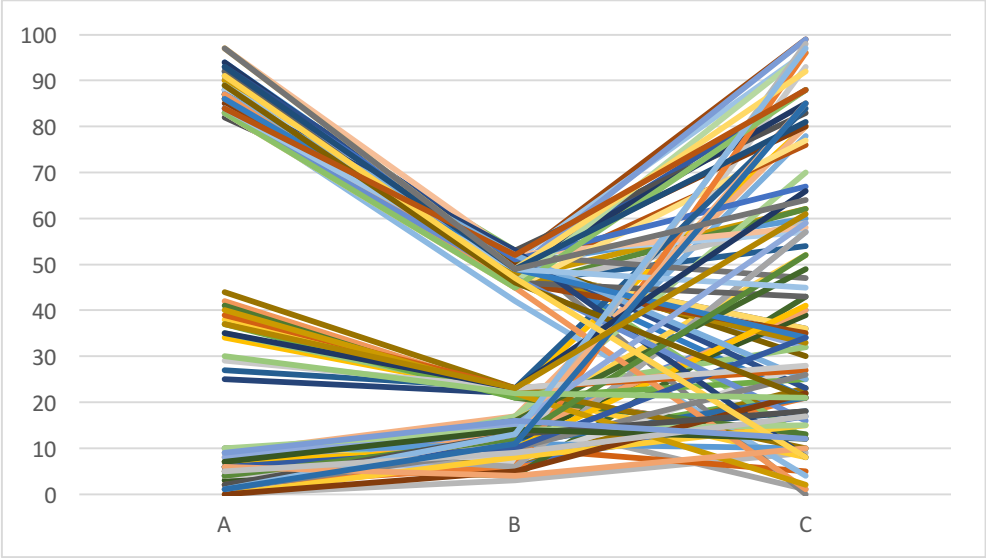

Figure 7. Successful Attacks. The vertical axis is the numerical value of the attribute listed on the horizontal axis.

Table 1. System of Equations Defining Successful Attacks

1. If 0<=A=2 and 2<=B<=17
2. If 3<=A=7 and 4<=B<=17
3. If 8<=A=10 and 6<=B<=17
4. If 25<=A<=42 and 21<=B<=23
5. If 25<=A<=44 and 21<=B<=23
6. If 25<=A<=46 and 21<=B<=23
7. If A=82 and 41<=B<=53
8. If (A=83 or A=84) and 42<=B<=53
9. If (A=85 or A=86) and 43<=B<=53
10. If (A=87 or A=88) and 44<=B<=53
11. If (A=89 or A=90) and 45<=B<=53
12. If (A=91 or A=92) and 46<=B<=53
13. If (A=93 or A=94) and 47<=B<=53
14. If (A=95 or A=96) and 48<=B<=53
15. If (A=97 or A=98) and 49<=B<=53
16. If (A=99) and 50<=B<=53

On each trial of the experiment, participants were shown a set of 5 possible attacks; four were randomly chosen from the set detected by the classifier and one was from the set of successful attacks. Within any set of 5 attacks, no attacks could share more than one value within a feature.

No attack was repeated across any of the trials in the experiment. The attacks within a set were presented in a random order. The same 5 attacks were presented for each trial for all participants. All participants had the opportunity to submit 100 different attacks against the computer system.

Post-experimental awareness and strategy use questionnaire. Following the completion of the learning session participants were asked to answer questions about the strategies they used to learn the relationship that would determine the classifier weaknesses. In addition, participants were asked whether they were aware that obfuscation was being used in the feedback being given.

5.1.3. Procedure

Human Subjects Board approval was obtained for this experiment. Informed consent was obtained from all participants. Participants were invited to an online meeting in which they could talk directly with the experimenter and share their computer monitor with the experimenter to allow for monitoring of performance.

During the learning task, participants were asked to use a web browser to access the online learning-task website, enter their secret codename and category – provided by the experimenter – and begin the experiment. Participants were instructed that their job was to try to figure out the vulnerabilities in a computer system by launching attacks against the system and looking for patterns in the trials that were successful. They were told that at least one trial in each set of 5 attacks would be successful and that the success of an attack was determined by a mathematical function describing a relationship between the features of each attack. Participants were told that they would be able to launch 100 attacks against the computer system and that they should try to guess the successful attack on each trial. Finally, participants were instructed to try to achieve 100% success rate by the end of the experiment.

Immediately following each attack, the system provided feedback that the attack was either successful or failed. Then the attack was added to a table of all prior submissions and the feedback that was provided and a new set of 5 possible attacks was presented. Participants were allowed to review the history at any time.

Obfuscation only occurs when the subject would have gotten the answer wrong, so the obfuscation rate refers to the percentage of time an unsuccessful attack is operationally allowed to be successful.

5.2. Results

Participants varied greatly in how quickly they selected attacks. Some participants took only 20 minutes to select all 100 attacks while some participants did not launch all 100 attacks in the approximately 45 minutes allocated for the learning task. In the obfuscated condition, 14 of 18 participants selected 100 attacks with the remaining participants launching between 43 and 90 attacks. In the non-obfuscated condition, 13 of 18 participants selected 100 attacks with the remaining participants launching between 40 and 98 attacks.

Because participants completed a variable number of trials, two approaches to evaluating performance were used. The first approach calculated the probability of choosing the successful attack across 1) the first 10 trials and 2) trial 11 and later. The second approach captured more of the time course of learning by categorizing trials into one of three groups: 1) early - first 10 trials; 2) intermediate - a variable number of trials in the middle of learning; and 3) late - last 10 trials.

To evaluate if learning occurred, the proportion of successful attacks for trials 11 and greater (i.e., the remaining trials) for all participants (M = .316) was compared to the probability that the successful attack would be chosen randomly (probability (random) = .20). The 95% confidence interval for the observed mean probability of .316 was .270 to .361 which did not include the probability if the choice was random. Therefore, we concluded that learning of the vulnerabilities did occur in this experiment.

In the obfuscated condition when an attack that could be detected by the classifier and would not pass the system defenses was selected, there was a 5% probability that the feedback would be obfuscated. This resulted in obfuscated trials for 1 to 9 trials for participants in the obfuscated condition. The first obfuscated trial occurred as early as the first trial of the experiment and as late as the 44^{th} trial.

To evaluate whether obfuscation influenced learning we compared the probability of choosing the successful attack from trial 11 until the end of testing across the obfuscation conditions. A test of the variances of these two samples suggested that the variances in the samples were not equal, $F(1, 17) = 2.25$, $p = .051$, therefore we compared the means using a two-sample t-test assuming unequal variances. As can be seen in Figure 2, contrary to our hypothesis, the probability of success in the obfuscated condition, M = .364, was greater than in the no-obfuscation condition, M = .267, $t(30) = 2.28$, $p = .0297$.

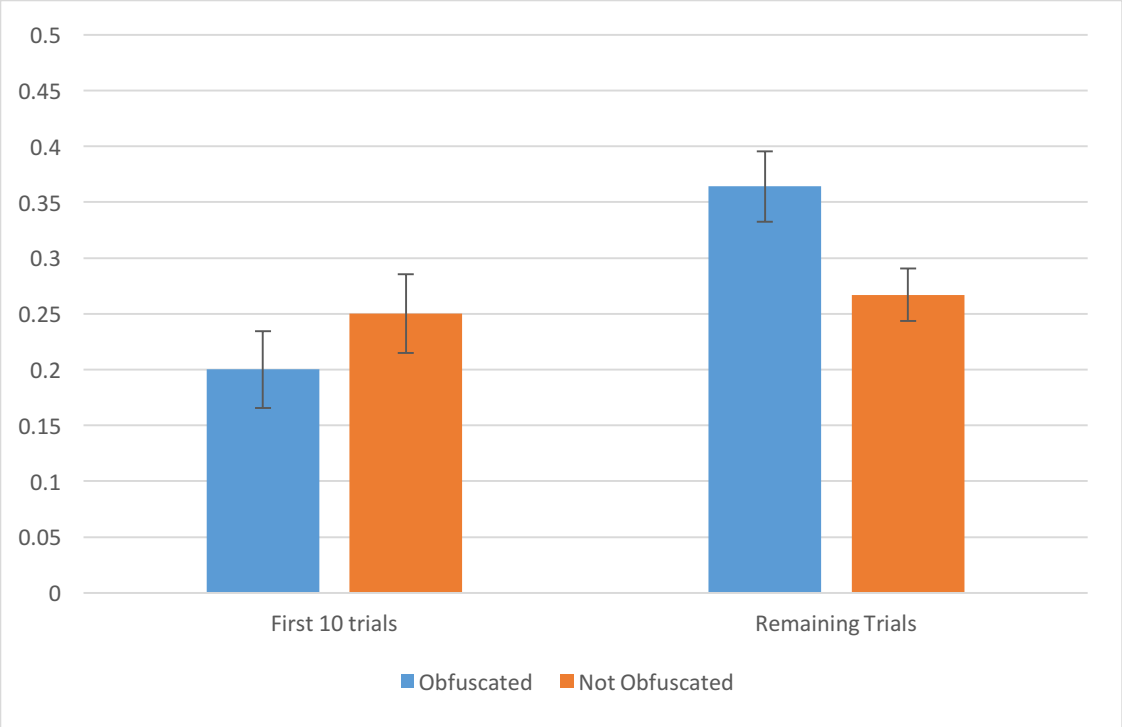

Figure 8. Probability of Choosing the Successful Attack (error Bars are Standard Error of the Mean).

To evaluate whether obfuscation influenced the time course of learning across the testing session, we compared early (first 10 trials), intermediate (the middle trials) and late (last 10 trials) success probabilities using a two-factor Analysis of Variance (ANOVA) with replication on the time-course variable. The means are presented in Figure 3. The ANOVA revealed that success varied across the learning trials, F (2, 102) = 4.5, p = .0136, but the effect of obfuscation condition, F (1, 103) = 2.7, p =0.102, and interaction with obfuscation condition, F (2, 102) = 2.8, p = 0.067, were not more than would be expected due to chance.

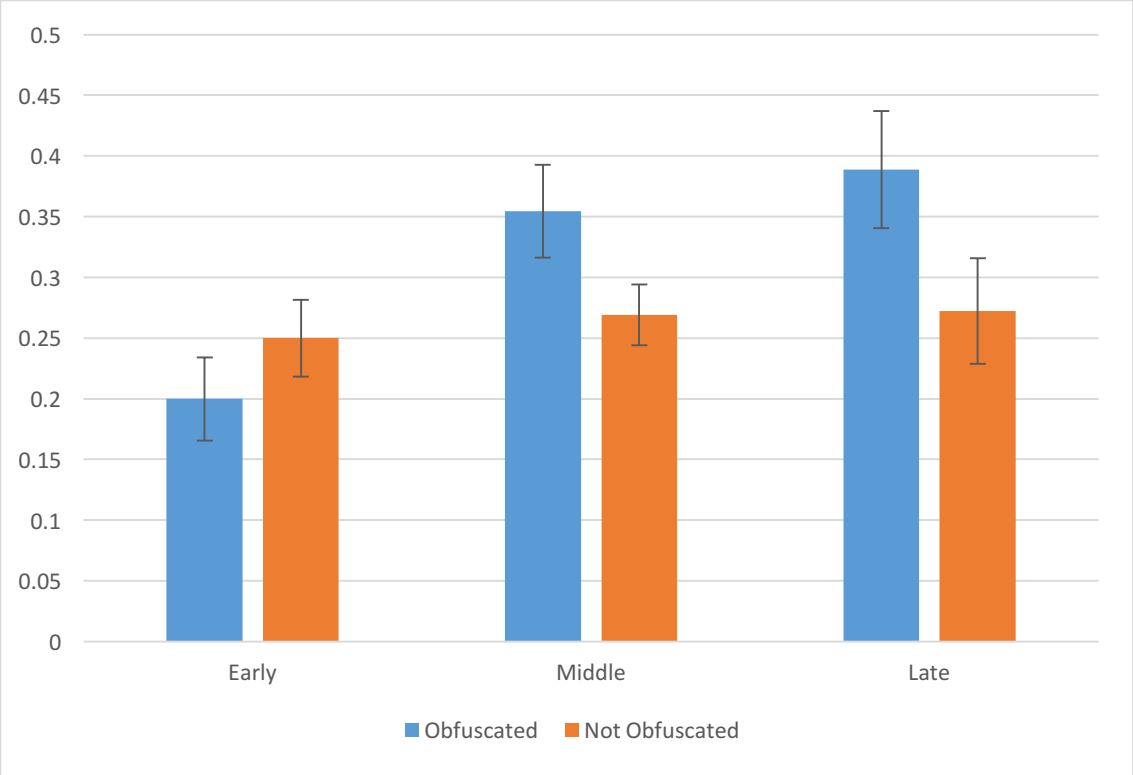

Figure 9. Proportion of Successful Attacks by When Trial Occurred in Learning (Error Bars are Standard Errors of the Mean)

5.3. Discussion

In this online learning environment, participants demonstrated low levels of learning about the vulnerabilities in the computer system. Surprisingly, we observed that in the condition where obfuscation was used (a 5% probability of obfuscation on trials that would have failed to pass the defenses of the system), performance was better than when no obfuscation was used.

The effect of obfuscation in this experiment was unexpected. How could providing individuals with false feedback about their performance improve their learning? The most likely explanation is that obfuscation was not truly the cause of this effect. Rather, in this small-sample experiment, random assignment failed and more individuals with learning strategies that were not well-matched to this task were assigned to the no-obfuscation condition as opposed to the obfuscation condition. For some suggestions on how to reduce the variability in learning across participants, refer to the section on improving the experimental design to assess disruptions in learning. Although failure of random selection is the preferred explanation for our findings regarding obfuscation, the introduction of uncertainty into a learning environment has been shown to improve learning in some conditions for both humans and machines. For humans, desirable difficulties in a learning task are thought to be optimal for motivating learning and for establishing strong long term learning [31]. Likewise, in artificial neural networks back propagation learning algorithms introduce noise into the process of updating weights between units to avoid settling into local maxima or minima.

5.3.1. Improving Experimental Design to Assess Disruptions in Learning

Assessment of the time course of learning would have benefitted from more control over what was being learned. As can be seen in Figure 1 and Table 1, the categories to be learned in this experiment were quite complex. A couple of the more successful participants described what they learned as three categories of vulnerabilities, as illustrated by the following quote from the strategy use questionnaire:

> *"The first sequences that seem to work were low numbers with a spread of about 10. But there weren't always those combinations, so I started to look at other distributions of numbers and found that a v-shape or valley type magnitude distribution usually did well (The first and the third number were close in magnitude and the middle was less). It seemed that the larger numbers were usually around the 80s and the middle one around 50ish. But at times neither of these two options were available, so there had to be at least one more pattern. I hit on one towards the end that was a descending pattern that seemed to emphasize 2's, but I didn't really get to flush that one out very well…"*

On any single trial, only one of these different elements of vulnerability was being tested; The element being tested was chosen randomly. Future work focused on understanding vulnerability learning on simple tasks should limit the vulnerabilities being learned.

Another approach to constraining the learning task, in order to better assess the effects of manipulations like obfuscation, would be to limit the attack selection space to the trials that define the border between successful and failed attacks. In this experiment, on each trial the range of values for success was much smaller than range of values for failure. A real world hacker might limit the problem space and launch attacks that are closer to the vulnerability space. In order to simulate these conditions within this experimental approach, each successful attack could be matched with a set of unsuccessful attacks that only had values within some range from the successful attack.

5.3.2. Improving the Obfuscation Manipulation

The timing of obfuscation was randomly selected to occur with 5% probability on trials where an attack that would fail was selected. By chance, some individuals received obfuscation frequently (as many as 9 trials) while other received it infrequently (as few as 1 trial). Similarly, this false feedback occurred as early as the first trial and as late as the 44^{th} trial. A more systematic manipulation of when and how frequently obfuscation is presented could better identify the effects of obfuscation on human and machine learning.

5.3.3. Statistical Analysis Considerations

This study could be extended in the future to include analyses of information accumulation on a trial-to-trial basis. Comparisons between an obfuscated group (in which obfuscation is more strictly controlled than in this study) and their non-obfuscated counterparts across a time-series could be modeled to show the greater impact of obfuscated versus non-obfuscated feedback over time. Computational theories employ such methods, by comparing predicted outcomes to raw-data time-series [33]. Such a method compares the hypothesized goodness-of-fit to the collected data, on average.

More specifically, one technique is the maximum likelihood estimation for rule learning. Such a technique would model how feedback impacts a learning setting. In this experiment, on each trial, *t*, participants select an attack. According to the *Q-learning model* [34], participants make this response according to their expected value that it is will be a successful attack. The modeling of such an experiment is a "simplified case of the Kalman filter: the Bayesian model uses the same learning rule but has additional machinery that determines the learning rate parameter on a trial-by-trial basis" [32]. It is this learning parameter compared across groups that could give an indication of the learning occurring, and the impact of false feedback. This would be a powerful data analytic tool to utilize in the future in order to better understand how feedback is utilized by participants in such a rule-learning task.

5.3.4. Creating an Automated On-line Platform for Human Data Collection

One goal of this project was to develop a web-based interface that could support faster and more automated human data collection. Although the methods employed for this study were time intensive, the web-based interface could be easily modified for automated administration. Elements that could be included to support automated data collection include 1) a textbox for providing instructions; 2) a mechanism to enhance motivation on every trial, such as a points system that rewards correct guess and, perhaps, penalizes incorrect guesses; 3) an algorithm to assign unique subject numbers to each participant and to select the "condition" of the participant, for example, obfuscated or non-obfuscated; 4) incorporate a timer to limit the amount of outside resources that participants might use; and 5) a mechanism to assess awareness of the experimental manipulation, such as the automated administration of awareness questions following testing.

If a more automated approach is implemented, considerations for the informed consent procedure should be considered. By removing a human experimenter from the process, greater confidentiality and possibly anonymity for participants may be attained. Under these conditions, consideration of whether the data being stored on the server (e.g., with IP addresses) and the demographic questions individuals answer could lead to identification.

6. MACHINE LEARNING EXPERIMENTATION

Because of the modularized architecture, we were able to use the same components used for human experimentation to do the machine learning trials. We substituted the human adversary with a machine learning adversary using a Hybrid Toolkit worker that utilizes logistic regression from SciKit-Learn.

We wrote a separate script to synchronize defender-adversary I/O because we wanted to ensure that the machine learning adversary could learn after every attack, without unnecessary database polling. However, this is not a necessary step, and a more natural simulation could ignore this constraint or even have the machine learning adversary submit events on an entire different schedule than the one-to-one rate enforced in our experiments.

We performed 300 trials with the logistic regression adversary with no obfuscation, and another 300 with obfuscation. All other parameters of the testing, such as number of trials per adversary and the obfuscation rate were kept the same as in the human experiments. The results of the experimentation are summarized below:

Table 2. Machine Learning Adversary Statistics

	Non-Obfuscated	*Obfuscated*	*Obfuscated-Adjusted*
Min Successes	31	26	31
Max Successes	60	57	61
Mean Successes	46.0	43.8	46.7

It appears that obfuscation could have had a small effect on the learning rate of the logistic regression worker. Performing a two-sample t-test with unequal variances gives $t(598)=5.842$, $p=8.599 \times 10^{-9}$, so the results are statistically significant given a threshold of $p<.05$.

However, when one adjusts for the successful attacks generated by the obfuscation itself, counting a operational deception as an actual win for the adversary, the results mirror the non-obfuscated case so much so that they do not quite meet the $p<.05$ statistical significance criteria, $t(598)=-1.86$, $p=.06354$.

Figure 10 below shows a more detailed breakdown, charting the average number of successes per the number of obfuscations:

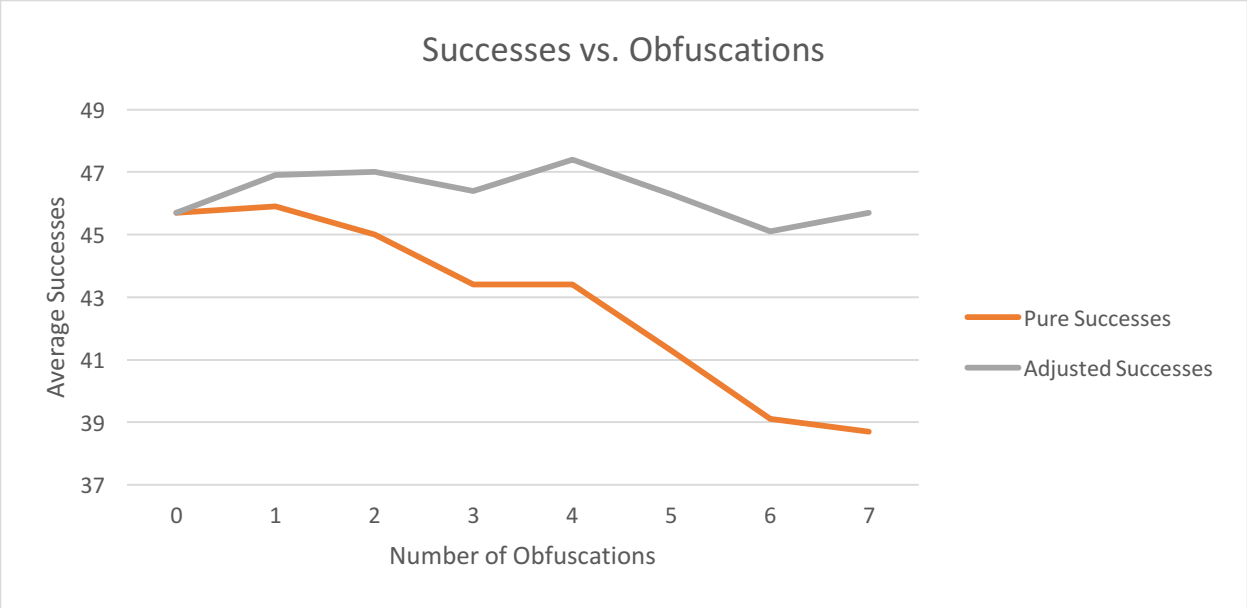

Figure 10. Average successes (pure and adjusted) vs. the number of obfuscations for machine learning adversaries.

Here, tail end obfuscation results with less than 5 adversary rounds per number of obfuscations were omitted. It is clear that the number of Pure successes, in orange, decreases with the number of obfuscations, monotonically so when the number of obfuscations < 1. However, if you adjust the number of successes, counting obfuscations, the average number of successes is fairly stable. Just from the data, above, it appears that the optimal obfuscation rate would be 6 per 100, with not obfuscating at all coming in a close second. It should be noted, however, that over 300 different adversaries that faced potentially obfuscated trials, only 10 received exactly 6 obfuscations, and only 14 received 0 obfuscations. The majority of adversaries received around 2-3 obfuscations, which is consistent with what would be expected given the probability of obfuscation and the average number of incorrect trials per adversary.

Just for comparison, the average number of successful attacks across the humans and the machine learning adversaries are shown below:

Table 3. Human vs. Machine Learning Average Successes.

	Non-Obfuscated	*Obfuscated*	*Obfuscated-Adjusted*
Humans	26.5	35.0	39.3
Machine Learning	46.0	43.8	46.7

It is not surprising that the logistic regression performed better overall. However, it should be taken into consideration that only 13 of the 18 human participants completed all 100 trials, whereas the machine learning adversaries always completed 100% of the trials.

7. DELIVERABLES AND OUTCOMES

Over the course of this project, the team has produced several publications, presentations, and software systems. We list these items here for reference; more details about the work associated with these deliverables and outcomes are provided in the sections above.

7.1 Refereed Publications

1. Andrew Smith, Jian Lou, and Yevgeniy Vorobeychik. Multidefender Security Games. In *IEEE Intelligent Systems*, 2016, to appear.
2. Muqun Li, David Carrell, John Aberdeen, Lynette Hirschman, Jacqueline Kirby, Bo Li, Yevgeniy Vorobeychik, Bradley A Malin. Optimizing annotation resources for natural language de-identification via a game theoretic framework. In *Journal of Biomedical Informatics*, 2016, to appear.
3. Zhiyu Wan, Yevgeniy Vorobeychik, Weiyi Xia, Ellen Clayton, Murat Kantarcioglu, Ranjit Ganta, Raymond Heatherly, and Bradley Malin. A game theoretic framework for analyzing re-identification risk. In *PLoS One*, 10(3):e0120592, 2015.
4. Yevgeniy Vorobeychik and Joshua Letchford. Securing interdependent assets. In *Journal of Autonomous Agents and Multiagent Systems*, 29(2):305-333, 2015.
5. Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Neural Information Processing Systems*, 2016 (NIPS 2016), to appear.
6. Amin Ghafouri, Waseem Abbas, Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Kout- soukos. Optimal thresholds for anomaly-based intrusion detection in dynamical environments. In *Conference on Decision and Game Theory for Security*, 2016 (GameSec 2016), to appear.
7. Aron Laszka, Waseem Abbas, Shankar Sastry, Yevgeniy Vorobeychik, and Xenofon Kout- soukos. Optimal thresholds for intrusion detection systems. In *Symposium and Bootcamp on Science of Security*, 2016 (HotSoS 2016).
8. Liyiming Ke, Bo Li, and Yevgeniy Vorobeychik. Behavioral experiments in email filter evasion. In *AAAI Conference on Artificial Intelligence*, 2016 (AAAI 2016).
9. Aron Laszka, Jian Lou, and Yevgeniy Vorobeychik. Multi-defender strategic filtering against spear-phishing attacks. In *AAAI Conference on Artificial Intelligence*, 2016 (AAAI 2016).
10. Bo Li, Yevgeniy Vorobeychik, Rachel Li, and Bradley Malin. Iterative classification for sanitizing large-scale datasets. In *IEEE International Conference on Data Mining*, 2015 (ICDM 2015).
11. Nika Haghtalab, Aron Laszka, Ariel Procaccia, Yevgeniy Vorobeychik, and Xenofon Kout- soukos. Monitoring stealthy diffusion. In *IEEE International Conference on Data Mining*, 2015 (ICDM 2015).

12. Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Resilient observation selection in adversarial settings. In *IEEE Conference on Decision and Control*, 2015 (CDC 2015).
13. Bo Li and Yevgeniy Vorobeychik. Scalable optimization of randomized operational decisions in adversarial classification settings. In *International Conference on Artificial Intelligence and Statistics*, 599-607, 2015 (AISTATS 2015).
14. Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Optimal personalized filtering against spear-phishing attacks. In *AAAI Conference on Artificial Intelligence*, 958-964, 2015 (AAAI 2015).
15. Bo Li and Yevgeniy Vorobeychik. Feature cross-substitution in adversarial classification. In *Neural Information Processing Systems*, 2087-2095, 2014 (NIPS 2014).
16. Yevgeniy Vorobeychik and Bo Li. Optimal randomized classification in adversarial settings. In *Thirteenth International Conference on Autonomous Agents and Multiagent Systems*, 485- 492, 2014 (AAMAS 2014).

7.2 Presentations

1. Georgia Institute of Technology, September, 2016
2. International Joint Conference on Artificial Intelligence, July, 2016
3. University of California, Berkeley, June, 2016
4. Bar Ilan Symposium on Foundations of Artificial Intelligence (Keynote), May, 2015
5. Conference on Data Analysis, March 2016
6. University of California, Davis, March, 2015
7. University of California, Berkeley, March, 2015
8. University of New Mexico, February 2015
9. University of Southern California, March, 2014

7.3 Software

- Hybrid Toolkit (open-sourced Fall 2016)
- Adversarial Gaming Simulation (AGS) Framework

7.4 Students

All students supported under this project attended Vanderbilt University at the time the work was performed. They all worked directly with Yevgeniy Vorobeychik as their advisor.

- Bo Li: PhD, Computer Science 2016
    - Symantec Labs Graduate Fellowship recipient
    - Currently a post doc at the University of Michigan
- Sixie Yu: M.S. student in Computer Science
- Jiazhi Zhang: B.S. student in Computer Science
- Liyiming Ke: B.S. student in Computer Science
- Royce Mou: B.S. student in Computer Science
- Brandon Arvanaghi: B.S. student in Computer Science

8. CONCLUSIONS

The research we have performed and the tools we developed facilitate research into a vast new area of machine learning and cybersecurity. The first obvious area is research into relevant cybersecurity problems, such as the application of deception to specific network intrusion, malware, and phishing contexts.

The theoretical work on adversarial machine learning on this project focused on Stackelberg models of attacker-defender interactions, but other models could be explored in the future. Moreover, work on this project mainly focused on scenarios involving email attacks, with some initial exploration in other security applications. Extending the Stackelberg model (and other related models) to new applications would help characterize the extent to which these models are generally applicable.

Other fundamental research opportunities exist as well. One very interesting research path that should be studied is the effect of obfuscation on various machine learning algorithms. Algorithms with more stable models, such as support vector machines, may be less affected by obfuscation than algorithms which tend to allow for more flexible decision boundaries, such as decision trees. Understanding the behavior of these algorithms in the presence of deception can also aid in attribution.

There is also a great deal to study concerning obfuscation strategies. The strategies we implemented were very simple, but the architecture allows for easy substitution, including obfuscation strategies that may dynamically choose an obfuscation rate based on the performance of a single adversary or an aggregation of such. In addition, it is quite possible that the placement of deception can play a role in adversarial learning. Badly labeled samples early on could potentially be more destructive than bad samples at later times. However, the reverse might be true for human adversaries, who may forget what they learned earlier, especially if the context in which they are learning does not allow for a recorded history and depends, instead, on memory.

The adversarial gaming simulation platform developed at Sandia is now a tool for studying the effects of defender-initiated deception in a cybersecurity context. The tools allow for controlled and extensible experimentation involving both human and machine learning adversaries and defenders. We will continue to work with the RECOIL lab to develop simulations which instantiate relevant and contemporary cyber exercises.

9. REFERENCES

[1] B. Biggio, G. Fumera, et al. Adversarial pattern classification using multiple classifiers and randomisation. In *Lecture Notes in Computer Science*, pp. 500–509. 2008.

[2] M. Brückner and T. Scheffer. Nash equilibria of static prediction games. In *Advances in Neural Information Processing Systems*, pp. 171–179. 2009.

[3] M. Brückner and T. Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pp. 547–555. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0813-7.

[4] N. Dalvi, P. Domingos, *et al.* Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pp. 99–108. ACM, New York, NY, USA, 2004. ISBN 1-58113-888-1.

[5] J.S. Downs, M. Holbrook, et al. Behavioral response to phishing risk. In *Second Annual eCrime Researchers Summit*, pp. 37–44. 2007.

[6] A. Globerson and S. Roweis. Nightmare at test time: robust learning by feature deletion. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pp. 353–360. ACM Press, New York, NY, USA, 2006. ISBN 1595933832.

[7] M. Kantarcioglu, B. Xi, et al. Classifier evaluation and attribute selection against active adversaries. *Data Min. Knowl. Discov.*, 22(1-2):291–335, Jan. 2011. ISSN 1384-5810.

[8] M. Kodialam and T. Lakshman. Detecting network intrusions via sampling: A game theoretic approach. In *Twenty-Second IEEE Annual Computer and Communications Conference*, pp. 1880–1889. 2003.

[9] W. Liu and S. Chawla. A game theoretical model for adversarial learning. In *Data Mining Workshops, 2009. ICDMW '09. IEEE International Conference on*, pp. 25 –30. dec. 2009.

[10] H. Otrok, M. Mehrandish, et al. Game theoretic models for detecting network intrusions. *Computer Communications*, 31(10):1934–1944, 2008.

[11] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symposium on Security and Privacy*, pp. 305–316. 2010.

[12] O.Vanek,Z.Yin,etal.Game-theoretic resource allocation for malicious packet detection in computer networks. In *Eleventh International Conference on Autonomous Agents and Multiagent Systems*. 2012.

[13] Andrew Smith, Jian Lou, and Yevgeniy Vorobeychik. Multidefender Security Games. In *IEEE Intelligent Systems*, 2016, to appear.

[14] Muqun Li, David Carrell, John Aberdeen, Lynette Hirschman, Jacqueline Kirby, Bo Li, Yevgeniy Vorobeychik, Bradley A Malin. Optimizing annotation resources for natural language de-identification via a game theoretic framework. *Journal of Biomedical Informatics*, 2016, to appear.

[15] Zhiyu Wan, Yevgeniy Vorobeychik, Weiyi Xia, Ellen Clayton, Murat Kantarcioglu, Ranjit Ganta, Raymond Heatherly, and Bradley Malin. A game theoretic framework for analyzing re-identification risk. In *PLoS One*, 10(3):e0120592, 2015.

[16] Yevgeniy Vorobeychik and Joshua Letchford. Securing interdependent assets. *Journal of Autonomous Agents and Multiagent Systems*, 29(2):305-333, 2015.

[17] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Neural Information Processing Systems*, 2016 (NIPS 2016), to appear.

[18] Amin Ghafouri, Waseem Abbas, Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Optimal thresholds for anomaly-based intrusion detection in dynamical environments. In *Conference on Decision and Game Theory for Security*, 2016 (GameSec 2016), to appear.

[19] Aron Laszka, Waseem Abbas, Shankar Sastry, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Optimal thresholds for intrusion detection systems. In *Symposium and Bootcamp on Science of Security*, 2016 (HotSoS 2016).

[20] Liyiming Ke, Bo Li, and Yevgeniy Vorobeychik. Behavioral experiments in email filter evasion. In *AAAI Conference on Artificial Intelligence*, 2016 (AAAI 2016).

[21] Aron Laszka, Jian Lou, and Yevgeniy Vorobeychik. Multi-defender strategic filtering against spear-phishing attacks. In *AAAI Conference on Artificial Intelligence*, 2016 (AAAI 2016).

[22] Bo Li, Yevgeniy Vorobeychik, Rachel Li, and Bradley Malin. Iterative classification for sanitizing large-scale datasets. In *IEEE International Conference on Data Mining*, 2015 (ICDM 2015).

[23] Nika Haghtalab, Aron Laszka, Ariel Procaccia, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Monitoring stealthy diffusion. In *IEEE International Conference on Data Mining*, 2015 (ICDM 2015).

[24] Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Resilient observation selection in adversarial settings. In *IEEE Conference on Decision and Control*, 2015 (CDC 2015).

[25] Bo Li and Yevgeniy Vorobeychik. Scalable optimization of randomized operational decisions in adversarial classification settings. In *International Conference on Artificial Intelligence and Statistics*, 599-607, 2015 (AISTATS 2015).

[26] Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Optimal personalized filtering against spear-phishing attacks. In *AAAI Conference on Artificial Intelligence*, 958-964, 2015 (AAAI 2015).

[27] Bo Li and Yevgeniy Vorobeychik. Feature cross-substitution in adversarial classification. In *Neural Information Processing Systems*, 2087-2095, 2014 (NIPS 2014).

[28] Yevgeniy Vorobeychik and Bo Li. Optimal randomized classification in adversarial settings. In *Thirteenth International Conference on Autonomous Agents and Multiagent Systems*, 485- 492, 2014 (AAMAS 2014).

[29] Debabrata Kar, Suvasini Panigrahi and Srikanth Sundararajan. SQLiDDS: SQL Injection Detection Using Query Transformation and Document Similarity. In *Distributed Computing and Internet Technology: 11th International Conference, ICDCIT*, 2015.

[30] KDD Cup 1999 Data Data Set, University of California, Irvine Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data. Retrieved September 10, 2016.

[31] E.L. Bjork and R.A. Bjork. Making things hard on yourself, but in a good way: Creating desirable difficulties to enhance learning. In M. A. Gernsbacher and R. W. Pew (Eds.) Psychology and the Real World: Essays Illustrating Fundamental Contributions to Society, pp. 56-64. Worth, New York. 2011.

[32] N.D. Daw. Trial-by-trial data analysis using computational models. *Decision making, affect, and learning: Attention and performance XXIII,* 23, 3-38, 2009.

[33] J.P. O'Doherty, P. Dayan, K. Friston, H. Critchley, and R.J. Dolan. Temporal difference models and reward-related learning in the human brain. *Neuron, 38*, 329-337, 2003.

[34] C. J. C. H Watkins. Learning from delayed rewards. Doctoral dissertation, University of Cambridge, 1989.

[35] Warren L. Davis IV and Daniel M. Dunlavy. Hybrid Methods for Cybersecurity Analysis LDRD Final Report, SAND2014-0446. January 2014.

[36] M.V. Mahoney and P. K. Chan. An analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for network anomaly detection. In *Proc. 6th Intl. Symp. on Recent Advances in Intrusion Detection (RAID 2003)*, pp. 220-237, 2003.

DISTRIBUTION

1	MS1326	Warren Davis	1461
1	MS1327	Danny Dunlavy	1461
1	MS1327	Kevin Nauer	9312
1	MS0899	Technical Library	9536 (electronic copy)
1	MS0359	D. Chavez, LDRD Office	1911


Sandia National Laboratories