Coevolutionary Agent-based Network Defense Lightweight Event System (CANDLES)

George Rush, Daniel R. Tauritz
Natural Computation Laboratory
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri 65409-0350, USA
gdr34b@mst.edu, dtauritz@acm.org

Alexander D. Kent
Cyber Futures Laboratory
Los Alamos National Laboratory
Los Alamos, NM 87545, USA
alex@lanl.gov

ABSTRACT

Predicting an adversary's capabilities, intentions, and probable vectors of attack is in general a complex and arduous task. Cyber space is particularly vulnerable to unforeseen attacks, as most computer networks have a large, complex, opaque attack surface area and are therefore extremely difficult to analyze. Abstract adversarial models which capture the pertinent features needed for analysis, can reduce the complexity sufficiently to make analysis feasible. Game theory allows for mathematical analysis of adversarial models; however, its scalability limitations restrict its use to simple, abstract models. Computational game theory is focused on scaling classical game theory to large, complex systems capable of modeling real-world environments; one promising approach is coevolution where each player's fitness is dependent on its adversaries. In this paper, we propose the Coevolutionary Agent-based Network Defense Lightweight Event System (CANDLES), a framework designed to coevolve attacker and defender agent strategies and evaluate potential solutions with a custom, abstract computer network defense simulation. By performing a qualitative analysis of the result data, we provide a proof of concept for the applicability of coevolution in planning for, and defending against, novel attacker strategies in computer network security.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.6 [**Simulation and Modeling**]: Miscellaneous; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

cyber security, coevolution, simulation

GECCO '15, July 11 - 15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739482.2768429

1. INTRODUCTION

Great strides are needed in the defensive tools and technologies available to cyber security practitioners, as the asymmetric nature of cyber warfare [8] puts defending practitioners at a distinct disadvantage; i.e., cyber attackers get to decide when and where to attack, without the need for physical presence providing advance notice to the cyber defenders who must scramble to quickly determine that an attack is occurring, select an appropriate defense, and execute it. In cyber security, as in many security-related fields, it can be prudent to evaluate worst-case attack scenarios. Whether facing insider threats or sophisticated adversaries outside one's networks, it is important to develop threat and attack models such that one can predict an adversary's capabilities, intentions, and probably vectors of attack. Cyber space is particularly vulnerable to unforseen attacks due to most computer networks having a large, complex, opaque attack surface area and therefore being extremely difficult to analyze. Invariably, the large number of variables involved in both offensive and defensive strategies makes an exhaustive search of all strategies infeasible. Abstract adversarial models which capture the pertinent features needed to analyze such strategies, can reduce the complexity sufficiently to make analysis feasible.

Game theory allows for mathematical analysis of adversarial models; however, its scalability limitations restrict its use to simple, abstract models. Computational game theory is focused on scaling classical game theory to large, complex systems capable of modeling real-world environments; one promising approach is coevolution where each player's fitness is dependent on its adversaries. Coevolution can be employed to explore the associated search spaces, examining strategic capabilities and estimating how each side will adapt in response to moves made by an opponent.

It makes sense to focus resources on securing systems such that attacks become less effective. There are several well-known ways to harden systems against attack, from limiting access permissions to patching software and reducing exposed network services. These are useful measures to reduce the attack surface and make it easier to defend vital assets. Yet for all the efforts taken to secure systems, intrusions still occur in many organizations, often without their knowledge [8]. Once it is acknowledged that intrusions cannot be stopped entirely, awareness, damage mitigation, and disaster recovery become important goals. In particular, knowing how an adversary could break into a network and devising possible counter-strategies is vital. In this paper,

we propose the Coevolutionary Agent-based Network Defense Lightweight Event System (CANDLES), a framework designed to coevolve attacker and defender agent strategies and evaluate potential solutions with a custom, abstract computer network defense simulation. Our approach coevolves two populations containing attacker and defender strategies in a network defense scenario, and the fitness of strategies is determined through simulations. In this way, unique, near-optimal strategies for both sides can be discovered. By performing a qualitative analysis of the result data, we provide a proof of concept for the applicability of coevolution in planning for, and defending against, novel worst-case attacker strategies in computer network security, allowing an end user to develop strategies and set up test scenarios. This offers new possibilities for analyzing and reinforcing defensive capabilities against unknown threats.

2. RELATED WORK

Coevolution has been used to find better placements for Flexible AC Transmission System (FACTS) devices [10], which are used to prevent cascading blackouts in electric transmission systems [7], and also to evolve attackers and defenders for graph-based network theory models [1]. Defenses have been designed that can predict the behavior of adaptive adversaries using a combination of game theory and machine learning [5], and machine learning has likewise been used to predict the nature of relationships in adversarial social networks [4]. Strategies have been developed for defending against Distributed Denial-of-Service (DDoS) attacks using a Bayesian game theoretic framework [12], and hidden Markov models have been developed to detect cyber attacks in network traffic [6].

Current approaches to developing computer network defense strategies largely focus on either pure game theoretic models [11] or real world implementations and emulated systems [2, 9]. Emulated systems more accurately reflect real world conditions, but require a fair amount of resources and configuration knowledge. Game theoretic models are mathematically elegant, but they do not scale well to larger solution spaces. Our approach falls somewhere in the middle, as our experiments employ a custom network security simulation. Coevolution allows us to approximate game theory solutions for large search spaces, thus providing scalability beyond the limits of classic game theory, and the simulation provides a more realistic strategy evaluation without the configuration difficulty of most emulated systems.

3. METHODOLOGY

There are two main parts to the CANDLES framework: the coevolutionary algorithm (CoEA) and the network security simulation. A CoEA was chosen over other stochastic methods since it most closely represents the natural dynamic between attackers and defenders in cyber space. Namely, they evolve their capabilities over time in response to actions taken by opponents. The network security simulation was developed to evaluate potential solutions in the CoEA, and it is used to simulate cyber defense scenarios given an abstract set of capabilities for both attackers and defenders.

In our model, attacker capabilities include exploits and reconnaissance, or recon. Recon techniques identify features of defender machines or networks and increase the chance of exploit success. Exploits are used to compromise a vulnerability and deliver a payload used to exfiltrate data. In our simulation, a vulnerability may be specific to a service or operating system, and there are no payloads designed to damage enemy systems. This is meant to model a stealthy attacker with advanced intrusion capabilities and a desire to gather as much information as possible. Attacker profit is defined in the simulation as the total amount of information exfiltrated from the defender's machines.

Defender capabilities include detection systems, mitigation techniques, and shutting down machines. Detection systems attempt to detect both exploits and reconnaissance and inform the defender. Upon detection, dynamic mitigation techniques can be used to attempt to stop the attack. Should the attack not be detected or if dynamic mitigation fails, then static mitigation techniques may still prevent the attack from working. Dynamic mitigation is meant to represent an Intrusion Prevention System (IPS), whereas static mitigation represents passive defenses like closed network ports, software patches, or limited user privileges. Paranoia is also used to represent how alert a defender is after exploits have been detected, and high paranoia will eventually cause the defender to shut down targeted machines.

3.1 Classes

This section describes classes that provide the structural basis for the rest of the framework.

3.1.1 Action

The Action class represents a possible action by any entity in the simulation, and it specifies both a target and a technique. Since currently only Attackers use Actions, techniques can be either Recon Techniques or Exploits.

3.1.2 Attacker

The Attacker class is initialized with an Attacker Solution and executes Actions during the simulation. Which Actions to use are determined using capabilities in the Attacker Solution, and state information is used to track reconnaissance information and event history. The structure for this class is visualized in Fig. 1.

3.1.3 Attacker Solution

An Attacker Solution specifies attack capabilities by defining available Recon Techniques and Exploits, and it specifies strategy through a target list of Defender Machines. In general, an attacker should not attack too many targets since that raises the Defender's paranoia level, which makes it harder to extract information once machines are shut down. On the other hand, attacking too few targets or targets of low value will not provide enough profit to be worthwhile. Attacker Solutions compose the first of two populations in the CoEA, and they are initialized using population seeds as described in Subsection 4.4.

3.1.4 Defender

The Defender class is initialized with a Defender Solution and responds to Actions by an attacker using detection systems and mitigation techniques. State information is used here to track the paranoia level. Paranoia is a mechanism that increases with detected exploits and allows the Defender to shut down machines upon reaching certain thresholds. Note that shutting down machines incurs a passive

Table 1: Experiment Configuration Parameters

Experiment ID	Attacker Pop. Seed	Defender Pop. Seed	Attacker Evolution	Defender Evolution
X1	Weak	Weak	Static	Static
X2	Weak	Weak	Static	Dynamic
X3	Weak	Weak	Dynamic	Static
X4	Weak	Weak	Dynamic	Dynamic
X5	Weak	Strong	Static	Static
X6	Weak	Strong	Static	Dynamic
X7	Weak	Strong	Dynamic	Static
X8	Weak	Strong	Dynamic	Dynamic
X9	Strong	Weak	Static	Static
X10	Strong	Weak	Static	Dynamic
X11	Strong	Weak	Dynamic	Static
X12	Strong	Weak	Dynamic	Dynamic
X13	Strong	Strong	Static	Static
X14	Strong	Strong	Static	Dynamic
X15	Strong	Strong	Dynamic	Static
X16	Strong	Strong	Dynamic	Dynamic

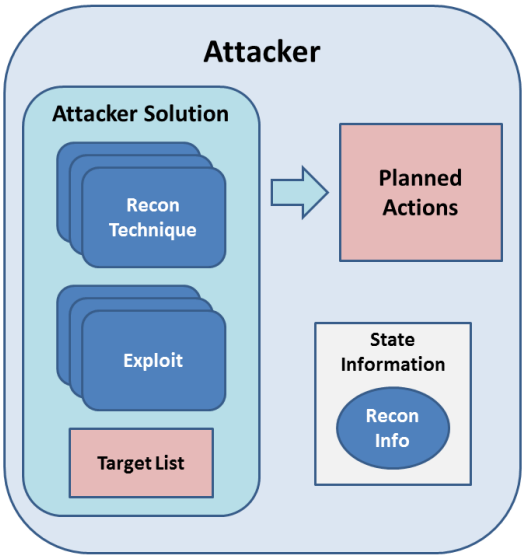


Figure 1: Attacker Diagram

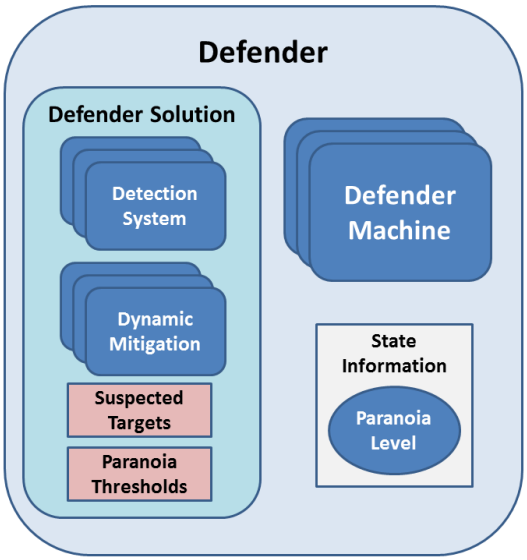


Figure 2: Defender Diagram

cost for productivity loss, but it prevents further data exfiltration. The structure for this class is visualized in Fig. 2.

3.1.5 *Defender Machine*

Each Defender Machine has a unique ID, intrinsic value, operating system, and list of available services. State information tracks whether the machine is active and any status effects incurred as a result of Exploits or Recon Techniques.

3.1.6 *Defender Solution*

A Defender Solution specifies a Defender's capabilities by defining available Detection Systems and Dynamic Mitigations, and it specifies strategy through suspected targets and paranoia thresholds. Suspected targets are used to perform extra defensive measures with a one-time cost, and paranoia thresholds are used to determine when to shut down machines. Defender Solutions compose the second population in the CoEA, and they are initialized using population seeds as described in Subsection 4.4.

3.1.7 *Detection System*

Each Detection System has a one-time installation cost and four separate detection probabilities for the following events against any target: successful recon, failed recon, successful exploit, and failed exploit.

3.1.8 *Dynamic Mitigation*

Each Dynamic Mitigation has a local execution cost, a user cost (to represent spent effort or time), and a probability of success. Note that Dynamic Mitigations can also be used as a static defensive measure on suspected targets. This means the costs are only incurred once per suspected target, but they are incurred regardless of whether or not that target is attacked.

3.1.9 *Exploit*

Each Exploit has a value multiplier, a probability of success, and one or more constraints. The value multiplier indicates the utility of the payload since a higher multiplier

will extract more value, representing information, from a given Defender Machine. The constraints specify the operating system or services required for the Exploit to function correctly.

3.1.10 Recon Technique

Each Recon Technique has type information and a probability of success. Type information indicates which effect the Recon Technique will have if successful, and the three possible effects are to increase the probability of exploit success, identify the OS, or identify running services.

3.2 Coevolutionary Algorithm

The CoEA evolves a population of Attacker Solutions against a population of Defender Solutions. Attacker Solutions use targets, Recon Techniques, and Exploits while Defender Solutions use paranoia thresholds, a budget, suspected targets, Detection Systems, and Dynamic Mitigations. Note that in both solution classes, targets are references to Defender Machines. Both Attacker and Defender Solutions are initialized from population seeds as covered in Subsection 4.4.

Parent selection is random, and survivor selection uses truncation. For recombination, a random subset of values are chosen for each variable from each parent. For mutation, a target or technique is randomly added or removed from the given solution. Defender Solutions can also mutate paranoia thresholds to any value with a step size of 0.1 in the range [0, 1]. Note that paranoia thresholds specify the necessary paranoia level for a Defender to shut down machines, whereas the paranoia level itself just specifies how paranoid the Defender is at any time during a given simulation.

Fitness evaluation for any individual requires measuring its performance multiple times against several opponent solutions in the network security simulation and averaging the results. The exact number of opponents and number of simulations per opponent are determined by the CoEA configuration (listed in Subsection 4.3). Pseudocode for the fitness function is provided in Algorithm 1.

One unique aspect of our CoEA is that the fitness function is asymmetric. The attacker only attempts to maximize its profit (representative of information gained), but the defender wants to both minimize the attacker's profit and minimize its own costs. To do this, the fitness value *per simulation* is calculated as shown in (1) and (2).

$$attacker\ fitness = -(attacker\ profit) \tag{1}$$

$$defender\ fitness = -(attacker\ profit) - (defender\ costs) \tag{2}$$

It is important to point out that fitness is determined on a scale of $(-\infty, 0]$, and attackers always want to minimize their fitness value while defenders want to maximize theirs. This mirrors reality in that the best case for defenders in cyber security is having zero losses from attacks and zero resources spent on defense, resulting in a maximum fitness value of zero. Attackers attempt to minimize their fitness value so that the attacker profit can be represented the same way in both equations.

3.3 Network Security Simulation

The network security simulation follows this process:

1. Attacker and Defender are initialized from solutions.
2. Defender prepares initial defenses.
3. Event loop begins:
   (a) Attacker performs an action (or does nothing).
   (b) Defender responds as necessary.
   (c) Passive costs are calculated.
   (d) Exit the loop if Attacker does not act.

During initialization, the Attacker will build attack sequences based on all possible combinations of available targets and techniques. It should be emphasized that the Attacker Solution only specifies the targets and techniques available for composing Actions in the attack sequence, and the Attacker's strategy for building attack sequences is static. The Defender then prepares its initial defense capabilities, which involves calculating installation costs for detection systems and static defensive measures. At this point, the main event loop begins. During each iteration of the event loop, the Attacker will start by looking up its next planned Action. If there are recon Actions remaining, those will be given priority. Should an exploit Action be selected, it will be executed only if the exploit constraints match any known recon information for the target. The Defender will respond to either recon or exploits by attempting to detect and mitigate them. If the event is successfully detected, dynamic mitigations are applied. If dynamic mitigations fail or if the event is not detected, then static mitigations may still take effect for suspected targets. The Defender also has a paranoia level which increases when exploits are detected. Should paranoia reach certain thresholds, then targeted machines will be shut down upon further detection of exploits or data exfiltration. Passive costs are also calculated for each iteration of the event loop. The only passive costs are currently those for productivity loss by inactive Defender Machines, which are meant to balance out the tendency of a Defender Solution to simply shut down machines in order to block all attacks. Finally, should the Attacker have zero planned actions remaining, the event loop terminates.

4. EXPERIMENT PROCEDURE

The objective of our experiments is to demonstrate that coevolution can be used to explore and evaluate strategies in a cyber defense simulation. To this end, we have designed a number of experiments to explore various offensive and defensive scenarios.

4.1 Experiment Variables

The variables for individual experiments are the attacker population seed, defender population seed, and whether or not each population evolves (listed in Table 1). Population seeds are individual solutions used as a template for all population members during initialization, and they are listed as weak or strong depending on their capabilities. We also examine both static and dynamic populations in order to examine evolution in different contexts. It is easier, for example, to determine how well a population is evolving against a static opponent since it is not considered a moving target.

4.2 Result Data Format

There are 30 CoEA runs per experiment, and several types of result data are provided for each run. During a single run of the CoEA, the best Attacker Solution and Defender

Algorithm 1 Fitness Function

```
function CALCULATEFITNESS(individual, enemyPopulation)
    randomlyChosenOpponents ← CHOOSEOPPONENTS(enemyPopulation, NUM_OF_OPPONENTS)
    allOpponentAverages ← None
    for each opponent in randomlyChosenOpponents do
        simulationResults ← None
        for i in RANGE(NUM_OF_SIMULATIONS) do
            defenderCosts, attackerProfit ← RUNSIMULATION(individual, opponent)
            if individual.class = Attacker then
                result ← −attackerProfit
            else if individual.class = Defender then
                result ← −attackerProfit − defenderCosts
            end if
            simulationResults.APPEND(result)
        end for
        opponentAverage ← AVERAGE(simulationResults)
        allOpponentAverages.APPEND(opponentAverage)
    end for
    return AVERAGE(allOpponentAverages)
end function
```

Solution are stored from each generation. After the run is complete, all the best solutions are output to file and used to generate a CIAO plot (Current Individual vs. Ancestral Opponents), which is used to visually convey the progress of two populations during coevolution [3]. The Defender Machine configuration is also output to provide contextual information if needed during analysis.

To briefly explain CIAO plots, an example is provided in Fig. 3. This particular plot is from the attacker's perspective in experiment X8. For this perspective, darker regions indicate more success for the attacker, and the attacker's generation is plotted along the increasing y-axis while the defender's generation is plotted along the increasing x-axis. If this were the defender's perspective, darker regions would indicate more success for the defender, and the defender's generation would be plotted along the increasing y-axis while the attacker's generation would be along the increasing x-axis. In this CIAO plot, the attacker is most effective towards the last of its generations against the earliest generations of defenders as indicated by the nearly black pixels.

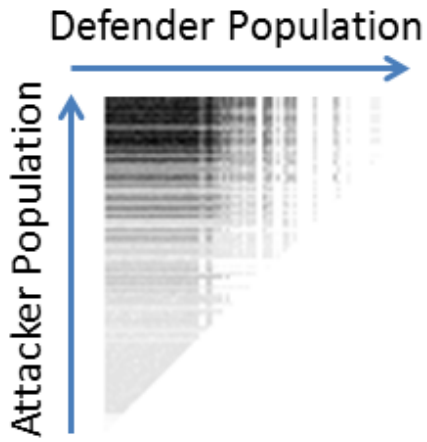


Figure 3: Example CIAO Plot

4.3 Important Configuration Parameters

All experiments ran with certain fixed values defined in the configuration, and this section will examine those values.

- 30 CoEA Runs
  This specifies the number of times a full CoEA is run per experiment. 30 CoEAs will result in 60 different CIAO plots (one each for attacker and defender perspective). This means that outliers can be identified, but when discussing results, we chose a representative pair of plots for each experiment.
- 10 Defender Machines
  This specifies how many machines the Defender has running on their network.
- 10 Fitness Opponents
  This is the number of opponents used to evaluate a given population member for fitness.
- 5 Simulation Runs
  This is the number of simulation runs between any two opponents during fitness evaluation. With 10 fitness opponents, this means that 50 simulation runs are needed to evaluate any population member.
- 100 Generations per CoEA
- Attacker Population Size of 100
- Defender Population Size of 100
- 30 Parents
  This is the number of parents selected from each population per generation, and each pair of parents produces a single offspring (i.e., 15 offspring for each population).
- Random Parent Selection
- Truncation Survivor Selection

- No Pre-Defined Defender Machines
  This is used to specify a Defender Machine configuration for the network security simulation. Since one is not given, the program will generate a random set of Defender Machines for the duration of each CoEA.

Many of these constants were chosen to meet time constraints in the experiments or to simplify the model, though there are a few exceptions. Random parent selection was chosen to add genetic variety to the offspring, and using truncation for survivor selection ensured that the population would attempt to improve over time, or at least not throw away the best solutions. There were never any predefined settings for the Defender Machines aside from the total number, as this forced solutions to be robust enough to succeed against multiple network configurations.

4.4 Population Seeds

Both of the initial populations were generated using either strong or weak solutions for each experiment. The design of each population seed is presented here.

4.4.1 Weak Attacker Seed

The weak attacker solution has only three out of ten possible targets and zero recon techniques or exploits. All useful capabilities must be developed through mutation.

4.4.2 Strong Attacker Seed

The strong attacker solution has all ten defender machines as targets, one recon technique for each possible effect, and an exploit with the highest possible multiplier for the current configuration. All techniques have a success rate of 100 percent. While not a perfect solution, this provides the attacker with a powerful set of starting capabilities.

4.4.3 Weak Defender Seed

The weak defender solution starts both shutdown thresholds at 1.0 (the highest level), which means that the defender will wait the longest possible time before shutting down machines despite detecting exploits and data exfiltration many times. Only three out of ten machines are listed as suspected targets, and there are no detection systems or dynamic mitigation techniques. Like the weak attacker seed, all useful capabilities must be developed through mutation.

4.4.4 Strong Defender Seed

The strong defender solution starts both shutdown thresholds at 0.5, which is meant to strike a balance between increasing security and minimizing productivity losses from having to shut down machines. It also has a cheap detection system with perfect detection rates against all recon techniques and exploits, and it has an even cheaper mitigation technique with a success rate of 100 percent. All ten machines are listed as suspected targets. This provides a nearly perfect defense.

5. RESULTS

Typical CIAO plots for each experiment are shown in tables 2 and 3. The reason for having CIAO plots with both the attacker and defender perspectives is because each population uses a slightly different fitness function (as explained in Subsection 3.2).

5.1 Entirely White or Black CIAO Plots

CIAO plots are entirely white or black for ten experiments, indicating that little or no evolution occurred. The following is an examination of these cases.

5.1.1 X1, X5, X9, and X13

These are experiments where both populations were static (no evolution was allowed to occur), so this behavior is entirely expected.

5.1.2 X2, X6, and X14

These are experiments with a static attacker and dynamic defender. In X2, both sides start with weak population seeds. Since the weak attacker essentially starts out with zero attack capabilities, there is no motivation for the defender to evolve. In X6, the defender is strong while the attacker is still weak, so the same thing happens. In X14, both sides start out strong, and this leads to a different problem: If both sides are currently near their peak capabilities, there is little room to improve via evolution.

5.1.3 X7 and X15

These are experiments with a dynamic attacker and static defender. In X7, the attacker starts out weak while the defender starts out strong. While this provides plenty of room for the attacker to evolve, the defender was simply too powerful. The attacker never managed to evolve stronger capabilities since it could never gain a foothold in the defender's network. In X15, both sides start out strong, so neither has room to evolve past their peak capabilities.

5.1.4 X16

In this experiment, both populations were dynamic and started out strong. However, since both started near their peak, there was no room to evolve for their best members.

5.2 Gradient CIAO Plots

CIAO plots have gradients for six experiments, indicating that evolution occurred for one or both sides. The following is an examination of these cases.

5.2.1 X3 and X11

These are experiments with a dynamic attacker and static defender. In X3, both sides start out weak, which allows the attacker to evolve many new capabilities and produce a smooth gradient in the corresponding CIAO plots. In X11, attackers started out strong while defenders started out weak. As a result, the attacker's best members only evolved a few times since they were already strong. This created CIAO plots with a few large bands in the gradient. In both cases, these CIAO plots show clear evolution by the attacker.

5.2.2 X10

In this experiment, the defender was dynamic and started out weak while the attacker was static and started out strong. Because of the attacker's strength, the defender had an incentive to evolve better capabilities, and since the defender started out weak, it had a lot of room to evolve. Also, the attacker being static made it easier for the defender to improve since it was not chasing a moving target. This led to a smooth gradient in the corresponding CIAO plots representing the clearest evolution by the defender in all experiments.

Table 2: Typical Resulting CIAO Plots (Attacker Perspective)

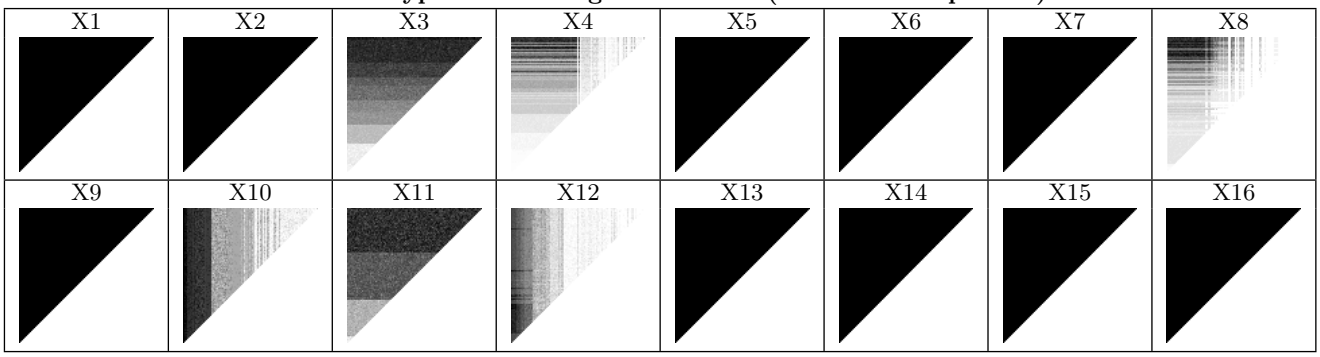
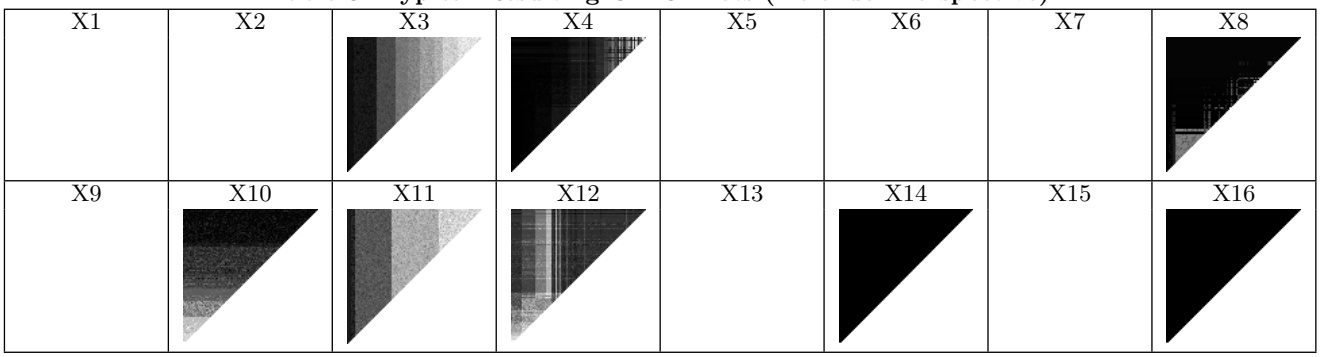


Table 3: Typical Resulting CIAO Plots (Defender Perspective)



5.2.3 *X4, X8, and X12*

These are experiments where both populations were dynamic. In X4, both populations started out weak, which leads to a lot of evolution on both sides. This experiment's CIAO plots show the attacker doing better against earlier defenders, whereas the later defenders eventually became strong enough that the attackers had trouble keeping up.

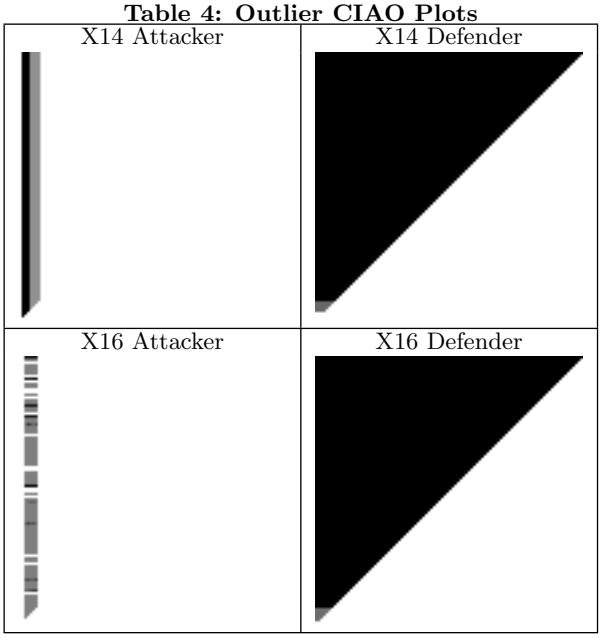
In X8, the attacker started out weak while the defender started out strong. The attacker managed to evolve enough to do better against earlier opponents, though against the defenders compensated for this by the end of the CoEA. Interestingly, the defender's CIAO plot in X8 is mostly black since the defender was doing well nearly the entire time according to its own fitness estimates.

In X12, the attacker started out strong while the defender started out weak. In the attacker's CIAO plot, it is clear that the attacker was getting weaker over time according to its own fitness estimates. This is because it started out strong while the defender was weak, meaning that the attacker did not have much room to evolve and was also chasing a moving target. The defender's CIAO plot for X12 has mixed gradients, which means that the attacker was at least attempting to counter the defender's evolution. However, it still shows that the defender eventually evolved to do well against almost all the attackers.

5.3 Outlier CIAO Plots

There were a few outlier CIAO plots for experiments X14 and X16, and examples of them are displayed in Table 4. In both experiments, each side starts out with a strong population seed, though the attacker is static for X14 while the defender is dynamic in both experiments. The CoEA starts out with the attackers doing decently well early on, but then they are clearly dominated by all later defender solutions. This is consistent with other CIAO plots for X14 and X16 in the sense that little evolution seems to occur, at least after a certain point. With little room for improvement, it appears that the defender will always dominate due to a nearly perfect defense.

Table 4: Outlier CIAO Plots



6. DISCUSSION

The results show that coevolution can be used to explore strategies by both attackers and defenders in network security, but there are still several opportunities for improvement. First, we chose to develop our own network security simulation as a compromise between a purely mathematical model and real world testing. Unfortunately, there seem to be few realistic simulations with the capabilities to test different types of cyber attacks. This is an important point since both the scenarios and strategies need to map accurately to real world entities. With our current simulation, it is possible to connect individual components in solutions to offensive and defensive capabilities in modern networks, but it will likely be an inaccurate or inconsistent mapping.

Second, the attacker and defender solutions mostly contained capabilities and a few important thresholds. The fundamental strategy for both sides does not change according to the solutions. Attackers always attempt to maximize their own profit by attacking a specific list of targets with all reconnaissance techniques and exploits available to them. Exploits will only be ignored for a given target if reconnaissance on that target has shown it to violate the exploit's own constraints (e.g. cannot use a SQL exploit if that service does not exist on the target). Defenders only respond to attacks rather than taking independent actions, and beyond detection and mitigation techniques, their only available option for stopping attacks is to shut down machines entirely.

Third, there are a number of parameter combinations that can affect the simulation or CoEA in interesting ways. One can change the number of defender machines, the number of generations, parent and survivor selection methods, the fitness function for either side, population sizes, etc. So far only limited hand tuning has been performed, due to the high computational cost of running experiments.

7. CONCLUSION AND FUTURE WORK

The goal of this work is to demonstrate the viability of developing cyber defense strategies by applying coevolution to network security simulations. We created our own simulation to test attacker and defender capabilities since existing frameworks did not meet our needs, and the results have shown that coevolution is a capable model in this solution space. In particular, experiments X3, X7, X11, and X15 are significant since they reflect the current situation in cyber security. This is because they model a dynamic attacker and static defender, an accurate situation as a defender will typically deploy defenses once and rarely change them while attackers constantly update their capabilities.

Note that our project is designed as a proof of concept. The network simulation in CANDLES is purposely somewhat abstract for the sake of simplicity. By utilizing coevolution with a more accurate simulation, it is likely that solutions with a stronger mapping to real world systems would emerge. Therefore, in future work, the first priority is to increase the fidelity of the simulation, such as introducing the notion of time (currently the simulation only recognizes order). Another important goal would be to develop more of the general strategy and less of the capabilities for attackers and defenders, as this could lead to more dynamic behavior on both sides. Finally, an extensive sensitivity study needs to be performed for identifying high-quality CoEA and simulation parameters.

8. ACKNOWLEDGMENTS

This work was supported in part by Los Alamos National Laboratory via the Cyber Security Sciences Institute under subcontract 259565 and in part by the Missouri S&T Intelligent Systems Center.

9. REFERENCES

[1] H. Arnold, D. Masad, G. A. Pagani, J. Schmidt, and E. Stepanova. NetAttack: Co-Evolution of Network and Attacker. In *Proceedings of the Santa Fe Institute Complex Systems Summer School 2013*.

[2] T. Benzel, B. Braden, T. Faber, J. Mirkovic, S. Schwab, K. Sollins, and J. Wroclawski. Current Developments in DETER Cybersecurity Testbed Technology. In *Proceedings of the Cybersecurity Applications & Technology Conference For Homeland Security (CATCH)*, pages 57–70. IEEE, 2009.

[3] D. Cliff and G. F. Miller. Tracking the Red Queen: Measurements of Adaptive Progress in Co-Evolutionary Simulations. In *Advances In Artificial Life*, pages 200–218. Springer, 1995.

[4] R. Colbaugh and K. Glass. Leveraging Sociological Models for Prediction I: Inferring Adversarial Relationships. In *2012 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 66–71. IEEE, 2012.

[5] R. Colbaugh and K. Glass. Predictability-Oriented Defense Against Adaptive Adversaries. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2721–2727, 2012.

[6] J. Grana, D. Wolpert, J. Neil, D. Xie, T. Bhattacharya, and R. Bent. HMMs for Optimal Detection of Cybernet Attacks. Technical Report SFI-2014-06-022, Santa Fe Institute, June 2014.

[7] N. G. Hingorani, L. Gyugyi, and M. El-Hawary. *Understanding FACTS: Concepts and Technology of Flexible AC Transmission Systems*. Wiley-IEEE Press, Dec. 1999.

[8] A. T. Phillips. Now Hear This–The Asymmetric Nature of Cyber Warfare. In *US Naval Institute*, volume 138/10/1,316, Oct. 2012.

[9] L. Pridmore, P. Lardieri, and R. Hollister. National Cyber Range (NCR) Automated Test Tools: Implications and Application to Network-centric Support Tools. In *Proceedings of the 2010 IEEE Systems Readiness Technology Conference (AUTOTESTCON)*, pages 1–4, Sept. 2010.

[10] T. Service and D. Tauritz. Increasing Infrastructure Resilience Through Competitive Coevolution. *New Mathematics and Natural Computation*, 5(2):441–457, July 2009.

[11] M. Van Dijk, A. Juels, A. Oprea, and R. L. Rivest. FlipIt: The Game of "Stealthy Takeover". *Journal of Cryptology*, 26(4):655–713, 2013.

[12] G. Yan, R. Lee, A. Kent, and D. Wolpert. Towards a Bayesian Network Game Framework for Evaluating DDoS Attacks and Defense. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*, pages 553–566, 2012.