Securing the Software Defined Perimeter with Evolutionary Co-Optimization

Michal Shlapentokh-Rothman
Massachusetts Institute of Technology, CSAIL
mshlapen@mit.edu

Erik Hemberg
Massachusetts Institute of Technology, CSAIL
hembergerik@csail.mit.edu

Una-May O'Reilly
Massachusetts Institute of Technology, CSAIL
unamay@csail.mit.edu

ABSTRACT

We investigate the robustness of a network defense isolation technique called Software Defined Perimeter (SDP). SDP tries to limit security risk by isolating users access based on user accounts, rules and resources. We show how a competitive co-evolutionary framework can be used to evaluate different SDP configurations. It employs a simulation to test the strength of different SDP configurations against different attackers. It uses a co-evolutionary algorithm to search through the action spaces of SDP configurations and possible attackers. These results enable the comparison of different SDP configurations based on the objective of minimizing the number of potentially compromised high-value resources .

CCS CONCEPTS

• **Security and privacy → Access control**; *Vulnerability management*; • **Theory of computation → *Evolutionary algorithms***;

KEYWORDS

evolutionary algorithms, cybersecurity, network

ACM Reference Format:
Michal Shlapentokh-Rothman, Erik Hemberg, and Una-May O'Reilly. 2020. Securing the Software Defined Perimeter with Evolutionary Co-Optimization. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377929.3398085

1 INTRODUCTION

It is estimated that in 2017 around 10 billion data records, e.g. passwords and credit card numbers, were lost or stolen since 2013 [1]. A common method of conducting network attacks that target resources such as data records is to introduce a piece of malware into the system, i.e. malware intrusion. The malware compromises the device through which it enters the system and then stealthily moves to other devices in the network. This malicious proliferation can have significant deleterious consequences e.g. in 2012, a malware attack on the national oil company of Saudi Arabia compromised 30,000 devices. [5]

GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7127-8/20/07…$15.00
https://doi.org/10.1145/3377929.3398085

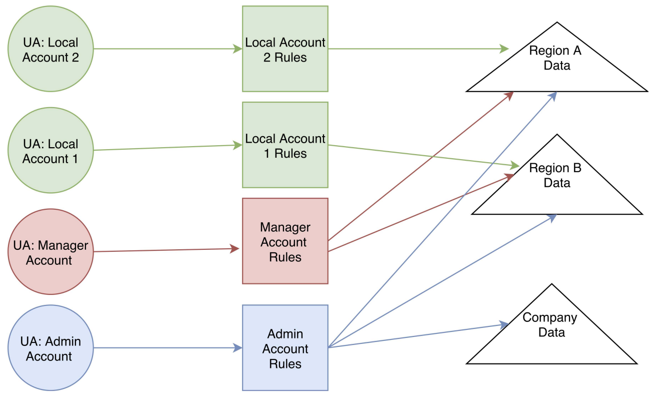


Figure 1: An example of a SDP. Circles represent user accounts(UA), each colored by type, square represent rules and triangles represent resources. Green arrows are for local accounts, red arrows are for manager accounts and blue arrows are for the administrator account.

In this paper we focus on improving security against malware intrusion attacks. We assume the following scenario. An attacker gains access to an enterprise network that has many devices. One such enterprise network could be a 'Bring Your Own Device Network', which is a special wireless network supporting employees use of personal devices. [8]. The attacker compromises one device and is then able to spread to other devices and resources. Our work analyzes an existing defense known as a *Software Defined Perimeter* or SDP [2] for this scenario.

SDP is a respected and often adopted network defense that fits within the broad portfolio of defensive measures network administrators deploy. It creates dynamic configurations of one-to-one relationships between 'users and the data they need to access' [6] within a network. It consists of a set of *rules* that determine whether a specific *user account*, according to its type, should have access to a particular *resource*. For example, consider an organization that has three types of user accounts: administrator, manager or local, plus 3 database resources. Each database holds client information from a different geographic region. The rules for this organization's SDP might be as follows: An administrator account is allowed to access to all of the available resources. Local accounts are only allowed access to the database in their region. Manager accounts have access to the databases that the employees they supervise can access. This SDP set up is diagrammed in Figure 1. While the example organization above has a simple set of rules, if resources and employee types expand and change, the relationships can become quite complicated. For example, consider the case where resource distribution changes and some users accounts now require access to more than one database depending on who their clients are at the moment. Now there are several different combinations of rules that could be used in the SDP. One possible set of rules allows local accounts to access all databases. This introduces a security risk because users without need are being given access. On the flip side, another set of rules requires a local account to obtain permission before it is granted access to a resource and that permission has to be updated with each change of clients. This is costly in time and user work efficiency.

Since eliminating shared resources is seldom practical, SDP puts the onus on the network administrator, when determining which set of rules to use, to make a tradeoff between access loss (or inconvenience) to resources versus increased security risk. Determining security risk entails considering likely attacks and their potential impact. This determination is challenging because of network complexity, the volume of existing attack types, and the possibility of unforeseen attacks that arise when attackers adapt their attacks toward defeating defensive counter-measures. In fact, the network administrator must be concerned with the security "arms race" that could escalate from a SDP rule set configuration.

Currently, despite these challenges, there is no systematic method to guide the process of creating a high access and low risk SDP for a particular network [16]. The number of different combinations of rules is large and grows quickly when new user accounts and resources are added to an SDP. Additionally, a network administrator lacks a quantifiable way to determine if one version of an SDP is better than another because testing options live is infeasible.

Our research goal is to determine the relative accessibility and security risk of different versions of SDPs which will provide decision support for network operators. Our work provides a methodology that evaluates the robustness and strength of different SDP versions against adaptive attackers.

Our solution has two components: a SDP simulation and evolutionary algorithms. The simulation evaluates how effective a specific version of an SDP is at preventing resource compromise. Given specific parameters for both a defender and attacker, and a set of SDP rules, the simulation evaluates the probability of breaches in the defender's network. We run Monte Carlo simulations to aggregate and estimate these probabilities. This provides a fitness score to our evolutionary algorithms. We use evolutionary algorithms in three ways. We first evolve attackers against a manually designed SDP. This provides information about the security risk under attack adaptation. We repeat this step for different domain expert designed SDPs so that a network administrator can compare the choices they design. Next, we manually configure an attack and evolve an optimal set of SDP rules that minimizes the network's security risk when facing it. We repeat this step for different manually configured attacks. This provides a network administrator with support when dealing with different attacks becoming more or less prominent. Lastly, setting aside manual configurations, we competitively co-evolve an attacker and an SDP together, enabling a two-sided arms race. This introduces a unique aspect of our methodology, *concurrently considering the evolution of both the attacker and the defender* rather than just the attacker, something which is rarely done when formulating computer security recommendations [14]. The information provides the network administrator with anticipatory information, allowing some foresight in choosing a configuration.

The contributions of this work are: (1) We model attacks on networks that use SDPs as their defense using a simulation. (2) We use evolutionary algorithms to evaluate different SDPs against adaptive attackers. (3) We use a competitive co-evolutionary algorithm to evaluate different SDPs under adaptive attackers and adaptive defenders. (4) We analyze the performance of best evolved SDPs.

In Section 2, we discuss related work. In Section 3, is the methodology and in Section 4 experiments and results. Finally, we present the conclusion and future work in Section 5.

2 RELATED WORKS

We describe Software Defined Perimeters, the role of simulation in security research and work related to evolutionary algorithms for cyber security in the follow sections.

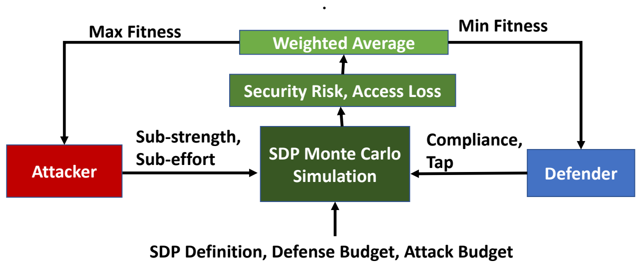
Software Defined Perimeters. SDPs are used for security purposes in various environments including IoT systems [3] and the technology industry [4]. The concept was developed by the Cloud Security Alliance (CSA) [2] with the goal of defending against attacks on 'application infrastructure.' Generally, a SDP consists of hosts and controllers. Controllers determine which hosts can communicate with each other [2].

Organizations interested in using SDPs can either purchase one through a vendor [6] or implement their own. Since we aim for a general methodology, we use a simple abstraction of an SDP to assess our methodology, which is described in Section 3.2. We base our abstraction on Google's Beyond Corp, which is an implementation of an SDP [4]. In their design, Google assigned users and devices to trust tiers and then matched trust tiers to groups of resources. Our SDP abstraction, refers to users and their devices as *user accounts* and trust tiers as *rules.*

SDP Simulation. A major part of our methodology is the use of simulation for determining the risk of various SDP configurations. Simulations address the expense of more realistic candidate solution evaluation and support scenarios that otherwise cannot be tested [10, 13, 18]. There is precedence for using simulations in computer security research. A simulation was developed in [7] to test SDSecurity, which is a security procedure that employs similar concepts as SDP. Waverly Labs [12] also developed an open-source SDP simulator that allows users to create an SDP with different parameters. Instead of using existing simulations, which require many low-level implementation, we developed a Monte-Carlo simulation with domain experts that determines the risk of a particular SDP. Our simulation is lightweight and allows us to abstract many implementation details compared to other simulations. We also are able to sample many different types of scenarios between attackers and defenders which would otherwise be difficult to do. The Monte-Carlo simulation's accuracy was verified by the domain experts using an equation-based model the experts designed. We ended up using the Monte-Carlo simulation since it provided more flexibility and complex model formulations.

Evolutionary Algorithms for Network Security. Evolutionary algorithms have been used in cyber security research for many

Figure 2: An overview of the SDP methodology. Note that tap and compliance are defined in Section 4.2


Max Fitness
Weighted Average
Min Fitness
Security Risk, Access Loss
Sub-strength, Sub-effort
SDP Monte Carlo Simulation
Compliance, Tap
Attacker
Defender
SDP Definition, Defense Budget, Attack Budget

years [11, 17]. Using them to evolve attacks against a static defense or to evolve defenses against a static attack has precedences, e.g. [10] and using co-evolutionary algorithms has been used in recent work, e.g. [9, 10, 15]. We employ *grammatical evolution*, using grammars to encode the different behavioral parameters of the attacks and defenses, following [9, 10]. Our grammars allow fixed budgets of attack strength and effort, plus defensive compliance to be predefined, while their allocation across attacks or defensive rules can be evolved, see Figure 3. The other central differences between the work of [9, 10] and this contribution are that this contribution a) focuses on SDPs– implying risk minimization as an objective, and SDP-specific grammars, b) uses SDP Monte Carlo simulation to obtain robust estimates of risk.

3 METHODS

In Section 3.1, we present the threat model. In Section 3.2, we present the threat scenario. It is parameterized and one set of parameters are fixed when a simulation is set up, while the other set allows attack and/or defense evolution. In Section 3.3, we describe the simulation for the project that is used to emulate attacks on networks. In Section 3.4, we discuss the competitive evolutionary and co-evolutionary search used in the experiments. An overview of the entire system and method can be seen in Figure 2.

3.1 Threat Model

The threat model can be defined as a minmax optimization problem. The defender aims to *minimize* the access loss by configuring an SDP as a defensive strategy to prevent the attack and reduce the security risk. The adversary tries to *maximize* the number and value of compromised resources on a network.

Defense Model. For this we assume:

- The defender's objective is to minimize the average of the access loss and risk,
- The defender's operational decision space is the configuration of an SDP,
- There is a non-compliant user account for non-compliant users,
- If a user in a user account is compromised, then the user is removed from the user account and is placed in the non-compliant user account
- The defender has a budget for network compliance monitoring that limits the extent to which monitoring can occur.

Attack Model. For this we assume:

- The attacker's objective is to maximize the average of the access loss and risk,
- The attacker compromises users through their accounts
- An attack compromises a resource if it compromises even a single user account that has access to that resource,
- The attacker has a limited budget for exploits.

3.2 Threat scenario

We assume that an enterprise uses an SDP for their network. The enterprise wants to determine which of a set of candidate SDPs will be the most robust against current and future attackers in terms of risk and access loss using a weighted average. We defer how these two values are quantified to Section 3.4. The enterprise has several different types of user accounts and different resources. The value of each resource is defined numerically and the administrator also sets up a non-compliance account. This account has limited access to resources (compared to other user account types in the SDP). Entire user account types are downgraded to the non-compliance account if a user account type is compromised. We frame this threat scenario by first initializing an SDP with parameter values that will NOT change and then optimizing, via an evolutionary algorithm, other parameter values that adapt. Table 1 summarizes the parameters, their notation and domains. The non-evolving parameters are:

- *demographic*: a subset that describes the network demographics, $(u_t, n_u, n_\rho, n_r, g_1, \ldots, g_n, 1, \ldots, t)$ where u_t is the total number of users, n_u is the number of different user account types, n_ρ is the number of different rules, and n_r is the number of different resources, $g_1, \ldots, g_n$ is the number of users in each user account and $g_n, 1, \ldots, t$ is the value of resources,
- *user-rule-resource mappings*: These coupled bi-mapping from user accounts to rules to resources dictate if and how a user account type is allowed access to a resource. We denote the user account types mapping to rules as g_r and rules mapping to resources as r_r.
- *type*: each SDP is classified as strong or weak. Strong SDPs have compliance budgets that represent expected 80 % compliance in each user attack and weak SDPs have compliance budgets that represent expected 50 % compliance in each user attack
- *probability of access loss*: the likelihood that a user account will be compromised
- *attack assumptions*:

– *attack strength*: an integer, a_s that represents how damaging an attacker can be to an SDP
– *attack effort*: an integer, a_e denoting total attack strength
– *type*: each attacker is classified as either strong or weak. Strong attackers have higher attack strengths compared to weak attackers.

- *defender assumptions*:

– *compliance budget*: The total compliance of the user account's compliance values must be less than or equal to the compliance budget,
– *type*: each defender is classified as either strong or weak. Strong defenders have higher compliance compared to weak defenders.

The following parameters can be selected for evolution:

An attacker's evolved behavior:

- *sub-strength*: a tuple, α_s that contains n_u values which describes what percentage of the attack budget is spent on that user

Table 1: Parameters. The top rows are assumed to be given for a scenario, e.g. by the SDP network administrator. Parameters in upper case (middle rows) are integrated into the grammar, see Figure 3. The bottom set are evolved by the evolutionary algorithms.

Parameter	Notation	Domain
Total number of users	u_t	250
Number of types of user accounts	n_u	$[0,\ldots,5]$
Number of rules	n_ρ	$[0,\ldots,5]$
Number of resources	n_r	$[0,\ldots,5]$
Number of users in each user account	$g_1,\ldots,g_n$	$[0,1,\ldots,250]$
Mapping of user accounts to rules	g_r	$\{0:(0,\ldots,\rho),\ldots,g_n:(0,1,\ldots,\rho)\}$
Mapping of rules to resources	r_r	$\{0:(0,\ldots,r),\ldots,\rho_n:(0,1,\ldots,r)\}$
Value of resources $1,\ldots,t$	$v_1,\ldots,v_r$	$\{40,50,60,70,80\}$
Probability of Access Loss	p_l	$\{0.0,\ldots 1.0\}$
COMPLIANCE_BUDGET	c_b	$\{2,3,4,5\}$
ATTACK_EFFORT	a_e	$\{0,\ldots,250\}$
ATTACK_STRENGTH	a_s	$\{2,3,4,5\}$
Compliance values per account type $1,\ldots,n$	$c_1,\ldots,c_n$	$[0,1]$
Attack Sub-Strength	α_s	$\{0.0,\ldots 1.0\}$
Attack Sub-Effort	α_e	$\{0,\ldots,250\}$

account. E.g., if there are three user accounts, $n_u = 3$, then a possible sub-strength could be $\alpha_s = (0.5, 0.5, 0)$. These can be interpreted as the probability of a malicious user existing in the account.

- *sub-effort*: a tuple, α_e that contains n_u values which describes what percentage of the attack effort is spent on that user account. For example, if there are three user accounts, $n_u = 3$, then a possible sub effort could be $\alpha_e = (0.5, 0.5, 0)$.

A defender's evolved behavior:

- *compliance values*: a tuple with n_u values representing the likelihood that a user in that account will become compromised.

3.3 SDP Simulation

We utilize Monte Carlo simulation, that we run 10 times, described in Algorithm 1, to estimate the performance of different attackers and defenders. There are two outputs from the simulation: the expected overall risk to resources and the expected number of users having their access downgraded because of non-compliance.

Step 1 Iterate through *all* of the user accounts and test if a user is compliant based on compliance values. If a random number is larger than a compliance value, then the user is compliant. Non-compliant users n_c^i in the account i are then moved to the non-compliant user account n_0 while the rest remain. The number of users in the non-compliant account n_0 becomes $n_0 = \sum_{i=1}^{n_u-1} n_c^i$.

Step 2 Determine the security risk, R_s level in each resource. For each resource r, each user, u, in each user account ua that has access to the resource is selected and compromised users are determined probabilistically, using the attacker sub_strength. A user becomes compromised if the attack sub_strength is larger than a random value. If a user account, ua has at least one compromised user, all users of its account type are considered compromised. If at least one user account ua is compromised, then the entire resource r is compromised. If a resource r is compromised, then the risk of that resource r being exploited maliciously is the resource value v_r divided by the sum of all resource values for the possible resources $\sum_{i=0}^{n_r} v_i : R_s = v_r/(\sum_{i=0}^{n_r} v_i)$. Otherwise, the risk is 0.

Step 3 Determine the access loss, L_a for each resource. This is determined by the pre-specified probability of access loss, p_l (which is the same for all resources) multiplied by the resource value v_r divided by the sum of all resource values possible resources $\sum_{i=0}^{n_r} v_i : L_a = p_l * v_r/(\sum_{i=0}^{n_r} v_i)$

```
begin
    for each user account type do
        Move subset of users in user account to non-compliant
          account using compliance value;
    end
    for each resource do
        for each user account connected to resource in user_resource
          _rule mapping do
            Determine if user account is compromised based on
              attack sub-strength and attack sub-effort;
        end
        if one user account is compromised then
            Entire resource is compromised;
        end
    end
    return security risk and access loss
end
```

Algorithm 1: Monte Carlo Simulation for SDP. The simulation computes the resource risk of an SDP when an attacker attacks the SDP. The input to the simulation is the attacker sub_strength, sub_effort, the SDP user_resource_rule mapping and compliance values.

3.4 Evolutionary and Co-Evolutionary Computation

To explore the results of different parameters, we use an evolutionary search that is developed in [10]. This consists of grammars and evolutionary and co-evolutionary algorithms.

Grammars are used to represent the different parameters in the attack and defense settings. Using grammars allows for many different variants of attacker and defenders to be explored. The attack parameters are different attack sub-strengths and attack sub-effort values, which are assigned to each user account. The sum of the attack sub-strengths across all the user accounts has to be equal to the attack strength. For example, if there are three user accounts, and the attack strength is 3, then the three attack sub-strengths assigned to user accounts 1,2,3 could be 2, .5, .5. Similarly, the sum of the attack sub-effort values has to be equal to the total attack effort. For each user account, the attack sub-strength and attack sub-effort are multiplied to determine the total attack on that individual user account. In addition to having different user-rule-resource and topologies, the 5 SDPs that we evaluate have internal variables that vary within a given SDP. We represent these variables in individual defense grammars for each SDP. The variables are resource values, compliance values for each user account and a compliance budget. The sum of the compliance values have to sum to the compliance budget. An example of the attack and defense grammars for two user accounts can be seen in Figure 3.

The attacker's objective is to maximize the weighted average of access and risk while the defender's objective is to minimize the weighted average of access and risk. The fitness function for the attacker is $f_a = \frac{R_s + L_a}{2}$, defender fitness is $f_d = -f_a$.

Figure 3: Attacker and Defender grammars.

```
### ATTACKER GRAMMAR ###
<attack_sub_strength>  ::= <strength_account_1>,<strength_account_2>
<attack_effort> ::= <effort_user_account_1>, <effort_user_account_2>
<strength_account_1>  ::= <value>
<strength_account_2> ::= ATTACK_STRENGTH - <value>
<effort_user_account_1>  ::= <value>
<effort_user_account_2> ::= ATTACK_EFFORT - <value>
<value>     ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10

### DEFENDER GRAMMAR ###
<compliance_values>  ::= <value_account_type_1>,<value_account_type_2>
<value_account_type_1>  ::= <value>
<value_account_type_2> ::= COMPLIANCE_BUDGET - <value_account_type_1>
<value>     ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10
```

Table 2: Spread (the number of user accounts divided by the number of connections between the rules and the resources) for each SDP and the rank from lowest to highest

SDP	Spread	Rank
Min-Isolated	2/6	1
Min-Total	2/7	2
Medium-Total	1/2	3
Medium-Isolated	3/7	4
Max	5/8	5

4 EXPERIMENTS AND RESULTS

The goals of our experiments are to investigate the performance of using evolutionary and co-evolutionary algorithms on SDPs and to explore the effects of having a weak attacker vs a strong attacker and a weak defender vs a strong defender. More concretely, we are interested in the effect of topology, user-rule-resource, compliance and attacker and defender type (strong or weak) on the overall performance of an SDP against an evolving attacker. Note, the attacker's objective is to maximize the measure of access and risk while the defender's objective is to minimize the measure of access and risk.

4.1 SDP Descriptions

In our experiments, we investigated five different SDP topologies and user-rule-resources that were determined to be the best by domain experts. Differences in the topologies include different amounts of user accounts and different user-rule-resource which include how many resources are being shared between accounts and the non-compliant account. In general, the different layouts can be described by two parameters 'spread', how the different resources divided are between groups. Mathematically, we can define spread of an SDP as the number of user accounts divided by the number of connections between the rules and the resources. The spread for each of the 5 SDPs can be seen in Table 2.

Two of the topologies we look at are *Min-Isolated* and *Min-Total*. Diagrams of these SDPs can be seen in Figures 4a and Figure 4b. The blue groups in both of the diagrams represent the non-compliant accounts. Both of these topologies have a minimum amount of spread since all of the resources are split up between two user accounts. We would expect these two topologies to be easily exploited by attackers due to their low spread values. The difference between *Min-Isolated* and *Min-Total* is that the non-compliant group in *Min-Isolated* has access to one resource while in *Min-Total*, the non-compliant account has access to 2 resources.

The remaining three topologies are Medium-Total, *Medium-Isolated* and Max. They can be seen in Figure 4c, Figure 4d, and Figure 4e. The non-compliant account is in blue. Overall, these three topologies are stronger than the first two because resource access is more divided across user accounts. *Medium-Total* divides its resources amongst 4 user accounts while *Medium-Isolated* has only 3 user accounts. This is what contributes to *Medium-Total* having a higher spread than *Medium-Isolated*. *Max* is the strongest topology because it has the most number of user account so resource access is spread out and it has an isolated user account and resource as well. We expect that *Max* would have least number of resources compromised from an attack.

Table 3: Experiment parameters

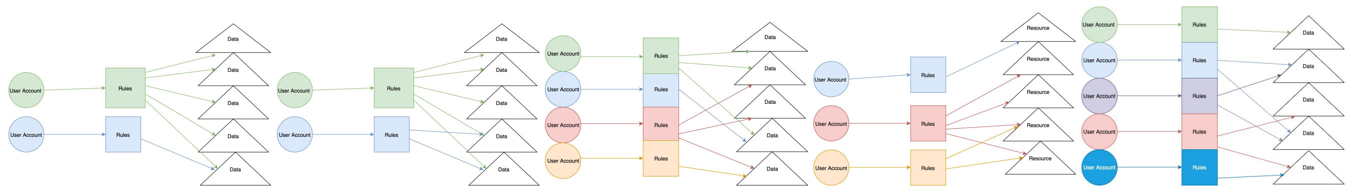
Parameter	Value
Population size	30
Generations	30
Crossover Probability	0.8
Mutation Probability	0.1
Max Length	100
Tournament Size	2
Elite size	1
Attack trials	30
Defender trials	30
Attacker objective	maximize risk and access loss
Defender objective	minimize risk and access loss

4.2 Setup

We follow the same methodology as described in [10]. We perform a defense evolution with a fixed attacker, an attack evolution with a fixed defender and then we co-evolve the two together. During the evolution we follow the same tournament selection and crossover as in [10] as well. A summary of the experiment parameters we use is described in Table 3. We are running the simulation for 30 trials.

The SDP determines the search space size. The smallest SDP has a small attack search space: there are only two user accounts for an attack budget of two so the approximate search space size is 10^4. The largest SDP has 5 user accounts with an attack budget of 5 so the search space size is approximately 10^{16}. The smallest SDP has a defense search space size of 1,200 and the largest SDP has a defense search space size of 10^5.

We define a strong attacker as one where the minimum attack sub-strength is 0.8 and a weak attacker as one where the minimum attack sub-strength is 0.5. Similarly, a strong defender has minimum compliance values of 0.8 and a weak defender has minimum compliances values of 0.5. Note that we use strong and weak attackers and defenders because the types of 'real world' attackers and defenders vary greatly. An example of a strong attacker is a nation state while a weak attacker is a teen hacker. A strong defender could be a large organization that has multiple roles like Amazon Web Services while a weak defender could be a small company that only has a few resources. We consider four pairings of attackers



(a) ***Min-Isolated*** (b) ***Min-Total*** (c) ***Medium-Total*** (d) ***Medium-Isolated*** (e) ***Max***

Figure 4: SDP configurations

and defenders: (1) strong attacker vs strong SDP (2) strong attacker vs weak SDP (3) weak SDP vs strong SDP (4) weak attacker vs weak SDP.

For each strength pairing, we evolve the SDP against a fixed attacker. Then we evolve an attacker against the strongest fixed defender. Finally, we perform co-evolution with both the SDP and an attacker. To evaluate the results overall, we compare the best solutions from the evolutionary phase experiments and the co-evolutionary experiments. We evaluate the performance of the best solutions against an out-of-sample attacker, which will be an attacker where each user account has a sub-strength value of 0.75. We also evaluate the performance of the attack performance of the out-of-sample attacker against the different SDPs.

We have three sets of SDP configurations:

No Taps baseline, we do not make any changes to the SDPs.

Taps we add *taps*, which are network monitoring devices. The job of the tap is to perform extra monitoring of a particular user account. We add taps in because SDPs alone are not enough to stop strong attackers. When taps are incorporated, one of the user accounts is designated as the tap user account. The tap user account has 'extra' monitoring on it: After the simulation begins, all user accounts have non-compliant users removed from the user accounts. The probability of being non-compliant is a parameter that is part of the search. After, the removal of non-compliant users from all the user accounts, the tap user account has extra monitoring on it. The tap user account has non-compliant users removed again, except the parameter to determine if a user account is compliant or not is only known by the tap user account. For example, consider user account 0 which has a compliance value of 0.8 and the tap has a compliance value of 0.5. If user account 0 is the tap account, then during the initial non-compliance user removal, a user would be considered non-compliant with probability 0.8. Then during the tap user account removal, a user would be considered non-compliant with probability 0.5.

Phases & Taps we incorporate *phases* into the simulation. During the fitness evaluation of an individual, instead of evaluating the individual once using the simulation, we will run the simulation multiple times, where each run of the simulation is considered a phase. We expand the search space when phases are used by giving both the attacker and defender a phase budget. The sum of all the attack strengths from each phase must sum to the attacker phase budget. Similarly, the defender's compliance budgets have to sum to the defender phase budget.

The next section describes the results from the experiments.

Table 4: Average best evolved attacker and evolved defender fitness values

		Attack		Defense	
Experiment Set	**SDP Name**	**Strong**	**Weak**	**Strong**	**Weak**
	Min-Isolated	16.5 ± 0.0	16.5 ± 0.0	−16.5 ± 0.0	−16.5 ± 0.0
	Min-Total	16.5 ± 0.0	16.5 ± 0.0	−16.5 ± 0.0	−16.5 ± 0.0
Taps	*Medium-Total*	14.93 ± 0.0	13.9 ± 0.25	−15.6 ± 0.01	−15.55 ± 0.10
	Medium-Isolated	16.5 ± 0.0	16.5 ± 0.0	−16.49 ± 0.001	−16.49 ± 0.01
	Max	**13.75 ± 0.11**	**13.41 ± 0.21**	**−14.38** ± 0.11	**−13.84 ± 0.13**
	Min-Isolated	16.5 ± 0.0	16.5 ± 0.0	−16.5 ± 0.0	−16.5 ± 0.0
	Min-Total	16.5 ± 0.0	16.5 ± 0.0	−16.5 ± 0.0	−16.5 ± 0.0
No Taps	*Medium-Total*	15.04 ± 0.15	13.42 ± 00.15	−15.3 ± 0.09	−15.5 ± 0.11
	Medium-Isolated	16.5 ± 0.0	16.5 ± 0.0	−16.49 ± 0.01	−16.47 ± 0.01
	Max	**13.71 ± 0.36**	**13.41 ± 0.21**	**−14.37** ± 0.07	**−14.29** ± 0.05
	Min-Isolated	18.0 ± 0.0	18.0 ± 0.0	−18.0 ± 0.0	−18 ± 0.0
	Min-Total	18.0 ± 0.0	18 ± 0.0	−18.0 ± 0.0	−18 ± 0.0
Phases & Taps	*Medium-Total*	16.81 ± 0.57	17.76 ± 0.19	−17.01 ± 0.17	−16.9 ± 0.17
	Medium-Isolated	18.0 ± 0.0	18.0 ± 0.0	−17.95 ± 0.01	−17.92 ± 0.03
	Max	**14.08 ± 0.28**	**13.52 ± 0.15**	**−15.87 ± 0.08**	**−15.85 ± 0.07**

Table 5: Average best co-evolution fitness values when attack and defense has same strength setting.

		Attack Strong, Defense Strong		Attack Weak, Defense Weak	
Experiment Set	**SDP Name**	**Attack**	**Defense**	**Attack**	**Defense**
	Min-Isolated	16.49 ± 0.00	−14.41 ± 0.08	16.44 ± 0.06	−15.11 ± 0.21
	Min-Total	16.5 ± 0.0	−16.12 ± 0.03	16.49 ± 0.00	−16.26 ± 0.06
Taps	*Medium-Total*	11.62 ± 0.67	−7.64 ± 0.24	11.91 ± 0.47	−8.11 ± 0.06
	Medium-Isolated	12.01 ± 2.05	−8.02 ± 0.26	12.2 ± 1.97	−8.01 ± 0.24
	Max	10.80 ± 0.54	−7.18 ± 0.33	10.44 ± 0.67	−7.24 ± 0.32
	Min-Isolated	15.65 ± 0.94	−7.34 ± 0.42	16.46 ± 0.05	−15.70 ± 0.02
	Min-Total	16.5 ± 0.0	−16.19 ± 0.04	16.5 ± 0.0	−16.44 ± 0.01
No Taps	*Medium-Total*	11.36 ± 0.75	−7.53 ± 0.26	11.38 ± 0.60	−7.89 ± 0.17
	Medium-Isolated	11.99 ± 2.05	−8.01 ± 0.25	12.52 ± 1.89	−8.01 ± 0.22
	Max	10.37 ± 0.72	−6.59 ± 0.38	8.41 ± 0.16	−6.30 ± 0.02
	Min-Isolated	14.55 ± 0.59	−7.85 ± 0.25	14.12 ± 0.59	−6.0 ± 0.0
	Min-Total	12.49 ± 1.77	−8.42 ± 0.15	10.91 ± 2.31	−6.0 ± 0.0
Phases & Taps	*Medium-Total*	11.54 ± 1.38	−6.96 ± 0.22	10.35 ± 1.43	6.0 ± 0.0
	Medium-Isolated	8.89 ± 1.03	−6.85 ± 0.04	11.54 ± 1.33	−6.0 ± 0.0
	Max	9.53 ± 0.49	−6.48 ± 0.12	9.49 ± 0.46	−6.54 ± 0.14

4.3 Results and Discussion

The aim of the experiments is to investigate how evolutionary search performs on the SDP threat scenario. A summary of the evolutionary experiments can be seen Table 4. Summaries of the co-evolution experiments can be seen in Table 5 and Table 6. The following subsections will investigate the results in further detail.

4.3.1 Defense Experiments. The goal of the defense experiments was to investigate the relative performance of the SDPs against a fixed attacker as well as how the SDPs optimize the placement of the compliance values. The defense experiments are visualized over generations for the Phases & Taps Strong Defense Experiments in Figure 5. The full defense results can be seen in Table 4.

The relative performance of the SDPs is consistent with spread rankings. With only two user accounts, the SDPs with the lowest spread were unable to isolate the attacker and all the resources

Table 6: Average best co-evolution fitness values when attack and defense has different strength (strong or weak) setting. Bolded values indicate strongest SDP.

		Attack Strong, Defense Weak		Attack Weak, Defense Strong	
Experiment Set	**SDP Name**	**Attack**	**Defense**	**Attack**	**Defense**
	Min-Isolated	16.47 ± 0.05	−14.00 ± 0.17	16.45 ± 0.05	−15.60 ± 0.06
	Min-Total	16.49 ± 0.004	−15.86 ± 0.08	16.49 ± 0.00	−16.42 ± 0.01
Taps	*Medium-Total*	11.56 ± 0.69	−7.80 ± 0.10	11.93 ± 0.48	−8.19 ± 0.06
	Medium-Isolated	12.71 ± 2.23	−8.00 ± 0.27	12.41 ± 2.08	−8.01 ± 0.24
	Max	10.54 ± 0.67	−7.05 ± 0.34	9.70 ± 0.83	−7.78 ± 0.14
	Min-Isolated	16.48 ± 0.03	−14.55 ± 0.12	16.47 ± 0.06	−15.68 ± 0.02
	Min-Total	16.5 ± 0.0	−15.47 ± 0.36	16.5 ± 0.0	−16.44 ± 0.004
No Taps	*Medium-Total*	11.32 ± 0.77	−7.41 ± 0.28	11.48 ± 0.62	−7.84 ± 0.15
	Medium-Isolated	12.18 ± 2.01	−7.91 ± 0.35	11.84 ± 2.01	−8.03 ± 0.22
	Max	8.60 ± 0.12	−6.19 ± 0.08	9.81 ± 0.74	−7.36 ± 0.23
	Min-Isolated	14.76 ± 0.62	−7.93 ± 0.31	13.98 ± 0.71	−6.0 ± 0.0
	Min-Total	12.36 ± 1.91	−8.17 ± 0.12	10.84 ± 2.39	−6.0 ± 0.0
Phases & Taps	*Medium-Total*	11.75 ± 1.15	−6.83 ± 0.15	10.72 ± 1.56	−6.0 ± 0.0
	Medium-Isolated	9.05 ± 0.89	−6.85 ± 0.04	11.85 ± 1.28	−6.0 ± 0.0
	Max	9.45 ± 0.50	−6.48 ± 0.12	7.98 ± 0.41	−6.48 ± 0.18

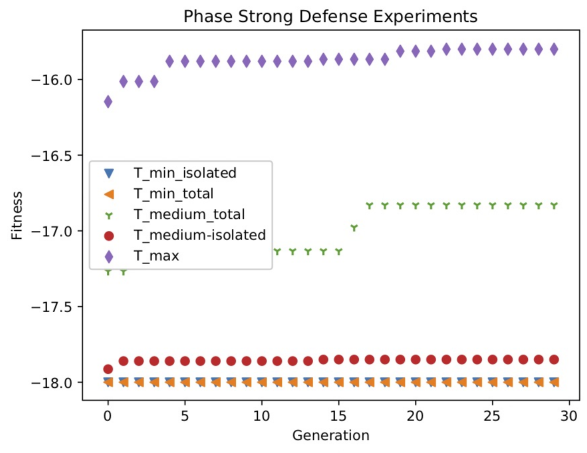


Figure 5: Fitness values Phase & Taps Strong Defense Experiments

became compromised. This indicates, that when an attacker is fixed, spread is correlated with risk to the network.

We did not see many differences between the strong and weak defender. The differences were relatively small and some fitness values increased while others decreased. For example, the fitness value for the weak *Max* was higher then the strong *Max*. This result was puzzling, so we examined the actual compliance values in the solution. We found that the general strategy of the defender is to place a large percentage $\approx 85 - 95\%$ of the compliance on one user account. The rest of the user accounts split the remaining compliance. Generally, the user account that had large compliance values was one of the accounts with the most access. However, the defense strategy was also to arrange the values of the resources such that the user account with the highest compliance values does not have access to the most highly valued resources. In general, the defender prefers to give the user accounts compliance values closer to 0. This is advantageous to the defender because if a user account has a compliance value close to 0, then all of the users will be moved to the non-compliant account and thus, resulting in that user account having 0 users. If there are 0 users, then the

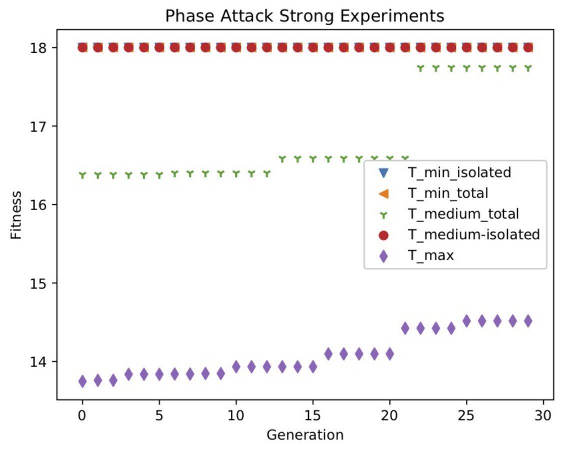


Figure 6: Fitness values for the Attack Strong Phase & Taps Experiments

attacker cannot compromise that user account and therefore, has fewer chances at compromising the resources connected to that attacker. Given this strategy, having a *lower* compliance budget can improve fitness for the defender in the simulation. This is likely why the weak defender *Max* fitness is higher than the strong defender *Max* fitness.

All of the SDPs had the worse performance during the phases part of the experiments. This implies that the SDPs were unable to find sets of compliance values in the larger search space that had a higher fitnesses then in the smaller search spaces. The SDPs that saw the largest decrease in performance were the ones with the lowest spread. We did not see much of a difference in performance when taps were removed, except for *Max* and *Medium-Total*. The performance actually improved when the taps were removed from *Medium-Total* and decreased with *Max*.

4.3.2 Attack Experiments. For the attack experiments, we evolved an attacker against a fixed version of each SDP. The goal was to see which SDPs were able to minimize the risk, represented by an attack's fitness. The attack experiments are visualized over the generations for the Phases & Taps. Strong Attack experiments can be seen in Figure 6 and full results are in Table 4.

The results were mostly consistent with the spread measurement. The highest spread, *Max* had the weakest attack while the lower risk SDPs, had the strongest attackers. However, the *Medium-Isolated* SDP, which had the third highest spread, had the same risk level as the two SDPs with the lowest spreads, which also occurred during the defense experiments. This could indicate that spread does not approximate relative risk when spread values are close together (the spread difference between *Medium-Isolated* and the second smallest is less than 0.1). In addition, the differences in strong and weak attacks were consistent with our expectations. The changes were fairly small but the attacker fitness did decrease for *Medium-Total* and *Max* when a weak attacker was used.

The SDPs had similar performance with and without taps. The attacker on strong *Medium-Total* was slightly weaker when taps

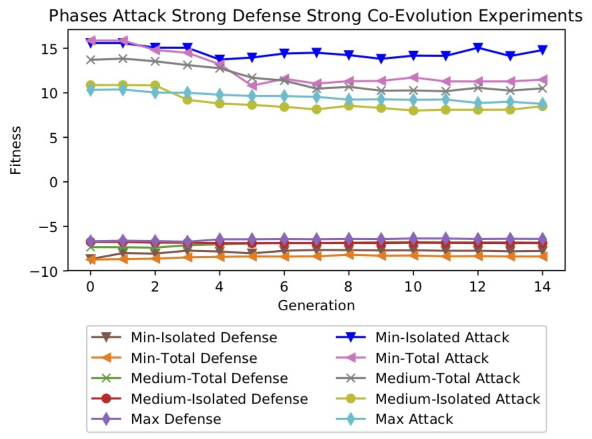


Figure 7: Fitness values for co-evolution of attackers and defenders. Attackers and defenders from the same topology have the same shape but different colors.

were used then when taps were not used. However, the weak attacker had a higher fitness when taps were not used. In both the static attacker and defender experiments, the lack of fitness improvement when taps were used, indicates that either the values in the larger search space have the same fitness as the smaller ones or that the evolutionary search space was not able to find the max values. When phases are used, the risk level for all SDPs was increased. This is consistent with the larger search space and is indicative of what would happen in a real network if multiple attacks occur in sequence. Ultimately, the attacker would be more damaging.

4.3.3 Co-Evolutionary Experiments. For each type of experiments, we looked at 4 different co-evolutions. The full results of the co-evolution for strong attacker and strong defender and weak attacker and weak defender can be seen in Table 5 and the remaining co-evolution results can be seen in Table 6.

Figure 7 show the coevolution of Phase & Taps strong coevolution experiments. The results of the co-evolution were somewhat surprising. We expected that for each SDP, the performance for both the attacker and the defender would decrease in co-evolution. This did occur in several of the SDPs with the attacker fitness but there were also some experiments that showed an increase in performance of the attacker. The most surprising result was an increase in defender performance in the co-evolution setting.

The end goal of the co-evolution is to anticipate how an SDP would perform against an unknown attacker. In order to make an objective comparison between the subjectively co-evolved SDPs, we tested the co-evolved SDPs from the phases trials against an unseen attacker. The results were generally consistent (see Table 7) with earlier results. The best performing SDPs, from both the static and co-evolved SDPs, were the ones with the highest spread, like *Max* and *Medium-Total*. Even after co-evolution, the lower spread SDPs, were unable to improve the fitness from the starting point from −16.5. In other experiments, *Medium-Isolated*, did have better performance. However, in the test cases, *Medium-Isolated*, was unable to do better than *Min-Isolated* or *Min-Total* after coevolution. This indicates that spread as an approximation of risk is likely continuous and more discrete. The spread of the SDP has to reach a certain value before the risk is actually lowered.

Table 7: Out of Sample Results from Phase & Taps Experiments

Experiment	SDP	Defender Fitness	Attacker Fitness
Static	*Min-Isolated*	−16.5	16.5
	Min-Total	−16.5	16.5
	Medium-Total	−15.35	15.92
	Medium-Isolated	−16.48	16.5
	Max	−15.59	16.06
Coevolution	*Min-Isolated*	−16.5	16.5
	Min-Total	−16.5	16.5
	Medium-Total	−14.45	13.8
	Medium-Isolated	−16.5	16.5
	Max	−13.49	13.76

For the stronger SDPs, we see a large contrast between the evolutionary SDP performance and the co-evolved SDP performance. The two strongest SDPs, *Medium-Total* and *Max* only performed slightly better than the other SDPs. We also see that evolutionary *Medium-Total* actually performs better than the evolutionary *Max*. When using the co-evolutionary SDPs, we see that *Max* has better performance than *Medium-Total*. The difference in performance rankings shows the advantage of using co-evolution. The relative performance ranking of the different SDPs after evolutionary search could be different than the actual performance ranking of the SDPs.

The co-evolution results demonstrate how the search handled the tradeoff of access loss and resource risk. Given the types of solutions, it seems that the search favored heavily protecting a small group of users and restricting access to many of the other users. Most solutions usually gave extremely high compliance values to one user account and the remaining user accounts had extremely small compliance values, which resulted in many users losing access. Given our parameters and the simulation used, it seems that the search favored restricting risk over preventing access loss.

5 CONCLUSION & FUTURE WORK

A key aspect for our method was the capability of coevolution to model adaptive adversaries, thus permitting an anticipatory approach for cyber defenses. For this approach evolutionary computation seems to be an appropriate method since it allows the specification of adversaries decisions and objectives and then a search based on these objectives.

We showed that the network defense methodology, Software Defined Perimeters (SDP) can be modeled using a Monte Carlo simulation, evolutionary and co-evolutionary algorithms. Our experiments demonstrate a technique that can enable network administrators to test the strength of their SDP without a large time or computation investment. We also showed that generally, the more spread out resources are, the stronger the SDP will be.

To verify the results, future work will include long term simulation and testing. It will also include more extensive SDP configurations that are larger and have more complex relations. Future work would also include an attack and/or defense model that incorporates new variables such as time.

ACKNOWLEDGEMENTS

We would like to thank MIT Lincoln Labs for all of their help and expertise.

REFERENCES

[1] [n.d.]. *The Need for a Global Response Against Cybercrime: Quatar as a Case Study.*

[2] Cloud Security Alliance. 2013. *Software defined perimeter. White paper.* Technical Report.

[3] R. E. Balfour. 2015. Building the "Internet of Everything" (IoE) for first responders. In *2015 Long Island Systems, Applications and Technology*. 1–6. https://doi.org/10.1109/LISAT.2015.7160172

[4] Betsey Beyer Barclay Osborn, Justin McWilliams. 2016. Beyond Corp: Design to Development at Google. *;login* 41, 1 (2016).

[5] Christopher Bronk and Eneken Tikk-Ringas. 2013. The Cyber Attack on Saudi Aramco. *Survival* 55, 2 (2013), 81–96. https://doi.org/10.1080/00396338.2013.784468 arXiv:https://doi.org/10.1080/00396338.2013.784468

[6] Cyxtera. 2018. *Definitive Guide to Software-Defined Perimeter.* Technical Report.

[7] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos. 2015. SDSecurity: A Software Defined Security experimental framework. In *2015 IEEE International Conference on Communication Workshop (ICCW)*. 1871–1876. https://doi.org/10.1109/ICCW.2015.7247453

[8] Kathleen Downer and Maumita Bhattacharya. 2016. BYOD Security: A New Business Challenge. *CoRR* abs/1601.01230 (2016). arXiv:1601.01230 http://arxiv.org/abs/1601.01230

[9] Dennis Garcia, Anthony Erb Lugo, Erik Hemberg, and Una-May O'Reilly. 2017. Investigating Coevolutionary Archive Based Genetic Algorithms on Cyber Defense Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17)*. ACM, New York, NY, USA, 1455–1462. https://doi.org/10.1145/3067695.3076081

[10] Erik Hemberg, Joseph R. Zipkin, Richard W. Skowyra, Neal Wagner, and Una-May O'Reilly. 2018. Adversarial Co-evolution of Attack and Defense in a Segmented Computer Network Environment. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '18)*. ACM, New York, NY, USA, 1648–1655. https://doi.org/10.1145/3205651.3208287

[11] Sara Khanchi, Ali Vahdat, Malcolm I Heywood, and A Nur Zincir-Heywood. 2018. On botnet detection with genetic programming under streaming data label budgets and class imbalance. *Swarm and evolutionary computation* 39 (2018), 123–140.

[12] Juanita Koilpillai. 2016. *Open Source Software Defined Perimeter.* Technical Report.

[13] Mona Lange, Alexander Kott, Noam Ben-Asher, Wim Mees, Nazife Baykal, Cristian-Mihai Vidu, Matteo Merialdo, Marek Malowidzki, and Bhopinder Madahar. 2017. Recommendations for Model-Driven Paradigms for Integrated Approaches to Cyber Defense. *CoRR* abs/1703.03306 (2017). arXiv:1703.03306 http://arxiv.org/abs/1703.03306

[14] Antonio Roque. 2017. Validating Computer Security Methods: Meta-methodology for an Adversarial Science. *CoRR* abs/1710.01367 (2017). arXiv:1710.01367 http://arxiv.org/abs/1710.01367

[15] George Rush, Daniel R. Tauritz, and Alexander D. Kent. 2015. Coevolutionary Agent-based Network Defense Lightweight Event System (CANDLES). In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO Companion '15)*. ACM, New York, NY, USA, 859–866. https://doi.org/10.1145/2739482.2768429

[16] Igor Saenko and Igor Kotenko. 2018. Genetic algorithms for role mining in critical infrastructure data spaces. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 1688–1695.

[17] Guillermo Suarez-Tangil, Esther Palomar, José María de Fuentes, J Blasco, and Arturo Ribagorda. 2009. Automatic rule generation based on genetic programming for event correlation. In *Computational Intelligence in Security for Information Systems*. Springer, 127–134.

[18] Allan Wollaber, Jaime Peñna, Benjamin Blease, Leslie Shing, Kenneth Alperin, Serge Vilvovsky, Pierre Trepagnier, Neal Wagner, and Leslie Leonard. 2019. Proactive Cyber Situation Awareness via High Performance Computing. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 1–7.