An Artificial Coevolutionary Framework for Adversarial AI

Una-May O'Reilly and **Erik Hemberg**
Massachusetts Institute of Technology

Abstract

Cyber adversaries are engaged in a perpetual arms race. They are continuously maneuvering to outwit the opposing posture. Replicating and studying the dynamics of these engagements provides a route to proactive, adversarially-hardened cyber defenses. The constant struggle can be computationally formulated as a competitive coevolutionary system which generates many arms races that can be harvested for robust solutions. We present a paradigm, techniques and tools that recreate the coevolutionary process in the context of network cyber security scenarios. We describe its current use cases and how we harvest defensive solutions from it.

Introduction

The greatest concern a prepared cyber defender might raise is: "What if my assumptions are wrong?" It is common knowledge that the only certainty is that an intelligent adversary will always keep trying to gain an advantage. Moreover, once forced to react, a defender is too late. So, how can a defender use Artificial Intelligence (AI) to gain an edge in an environment that is stacked to the attacker's advantage, where the defender seems doomed to always be one step behind?

One approach, *adversarial AI*, is to deploy defensive configurations, that consider multiple possible *anticipated* adversarial behaviors and already take into account their expected impact, goal, strategies or tactics. Note that the precise metrics in this accounting can vary. For example, impact can be any combination of financial cost, disruption level or outcome risk. Or, a defender could prioritize a worst case, average case or a trade-off configuration.

One way that such defensive configurations can be found is by using stochastic search methods that first explore the simulated competitive behavior of adversaries and then generate ranked configurations according to a variety of different objectives so a decision maker can choose among them. In particular, the field of coevolutionary algorithms (Popovici et al. 2012) provides search heuristics that specifically direct competitive engagements. The engagements are between members of adversarial populations with opposing objectives that each undergo selection on the basis of performance and variation to adapt. Coevolutionary logic results in population-wide adversarial dynamics. Such dynamics can expose possible adversarial behaviors that a defense would like to anticipate. A competitive coevolutionary algorithm can be a component of a larger system, see for example Figure 1, in which a complementary component sets up the environment where pairs of adversaries engage and measures the outcome for each adversary. These measures can be used by the coevolutionary algorithm to judge an adversary's fitness.

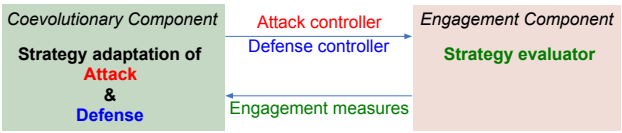

Figure 1: Component overview of our coevolutionary adversarial AI framework. The coevolutionary component performs search over the adversary controllers. The engagement component evaluates the strategies of the adversaries and returns the measurements of the engagement.

Herein we summarize a framework that we have used to generate robust defensive configurations (Prado Sanchez 2018; Pertierra 2018). It is composed of different coevolutionary algorithms to help it generate diverse behavior. The algorithms, for further diversity, use different "solution concepts", i.e. measures of adversarial success. Because engagements are frequently computationally expensive and have to be pairwise sampled from two populations each generation, the framework has a number of enhancements that enable efficient use of a fixed budget of computation or time.

The framework supports a number of use-cases using simulation and emulation of varying model granular-

 In: Joseph Collins, Prithviraj Dasgupta, Ranjeev Mittu (eds.): Proceedings of the AAAI Fall 2018 Symposium on Adversary-Aware Learning Techniques and Trends in Cybersecurity, Arlington, VA, USA, 18-19 October, 2018, published at http://ceur-ws.org

ity. These include: *A)* Defending a peer-2-peer network against Distributed Denial of Service (DDOS) attacks (Garcia et al. 2017) *B)* Defenses against spreading device compromise in a segmented enterprise network (Hemberg et al. 2018), and *C)* Deceptive defense against the internal reconnaissance of an adversary within a software defined network (Pertierra 2018)

The framework is linked up to a decision support module named `ESTABLO` (Sanchez et al. 2018; Prado Sanchez 2018). The engagements of every run of any of the coevolutionary algorithms are cached and, later, `ESTABLO` gathers adversaries resulting from different algorithms for its *compendium*. It then competes the adversaries of each side against those of the other side and ranks each side's members according to multiple criteria. It also provides visualizations and comparisons of adversarial behaviors. This information informs the decision process of a defensive manager.

The adversarial AI framework's specific contributions are:

- The use of coevolutionary algorithms to adaptively generate adversarial dynamics supporting preemptively investigating adversarial arms races that could occur.
- A suite of different coevolutionary algorithms that diversify the behavior of the adversaries to broaden the potential dynamics.
- Use cases that model a variety of adversarial threat and defensive models.
- A decision support module that supports selection of a superior anticipatory defensive configuration.

Background provides context on modeling and simulation and coevolutionary search algorithm. **Framework** describes our coevolutionary method, engagement component and decision support module. **Use Cases** provides examples applying to cyber security and network attacks. **Conclusions** summarizes and addresses future work.

Background

The strategy of testing the security of a system by trying to successfully attack it is somewhat analogous to software fuzzing (Miller, Fredriksen, and So 1990). Fuzzing tests software adaptively to search for bugs while adaptive attacks test defenses. In contrast to software where a bugs is fixed by humans, our approach automatically adapts a defense. This forms a novel *counter attack*. Fuzzing is driven by genetic algorithms (GA) whereas, to drive cyber arms races in which both adversaries adapt, our approach uses coupled GAs called competitive coevolutionary algorithms.

Coevolutionary Search Algorithms

Coevolutionary algorithms, related to evolutionary algorithms (Bäck 1996), explore domains in which the quality of a candidate *solution* is determined by its ability to successfully pass some set of *tests*. Reciprocally, a *test*'s quality is determined by its ability to force errors from some set of *solution*s. In competitive coevolution, similar to game theory, the search can lead to an arms race between *test* and *solution*, both evolving while pursuing opposite objectives (Popovici et al. 2012). An example of learning in a coevolutionary algorithm is shown in Algorithm 1. A basic coevolutionary algorithm evolves two populations with e.g. tournament selection and for variation uses crossover and mutation. One population comprises attacks and the other defenses. In each generation, competitions are formed by pairing attack and defense. The populations are evolved in alternating steps: first the attacker population is selected, varied, updated and evaluated against the defenders, and then the defender population is selected, varied, updated and evaluated against the defenders. Each attacker–defender pair is dispatched to the engagement component to compete and the result is used as a component of fitness for each of them. Fitness is calculated over all an adversary's engagements.

The representation of *test*s (and *solution*s) is customizable in any coevolutionary algorithm (Rothlauf 2011) under the design constraint that it be amenable to stochastic variation, e.g. "genetic crossover" or mutation. It may directly express the test or it may do so indirectly, e.g. with a grammar. In the latter case, an intermediate interpreter works with a rule-based grammar to map from a "genome" that undergoes variation to a "phenome" that expresses an executable behavior. Grammars (and GA representations, in general) offer design flexibility: changing out a grammar and the environment of behavioral execution does not require any changes to the rest of the algorithm.

Coevolutionary algorithms can encounter problematic dynamics where *tests* are unable improve *solutions*, or drive toward a solution that is the *a priori* intended goal. There are accepted *remedies* to specific coevolutionary pathologies (Bongard and Lipson 2005; Ficici 2004; Popovici et al. 2012). They generally include maintaining population diversity so that a search gradient is always present and using more explicit memory, e.g. a *Hall of Fame* or an archive, to prevent regress (Miconi 2009). The pathologies of coevolutionary algorithms are similar to those encountered by generative adversarial networks (GANs) (Goodfellow et al. 2014; Arora et al. 2017)

Modeling and Simulation

A coevolutionary algorithm includes an environment that supports executing the tests and solutions to compete against each other in each engagement. We use modeling and simulation for this purpose. Mod-sim systems range in complexity, level of abstraction and resolution. Modeling and simulation comprise a powerful approach, "*mod-sim*", for investigating general security scenarios (Tambe 2012), computer security (Thompson, Morris-King, and Cam 2016; Lange et al. 2017; Winterrose and Carter 2014) and network dynamics in particular, e.g., in CANDLES – the Coevolutionary, Agent-based, Network Defense Lightweight Event System of (Rush, Tauritz, and Kent 2015), attacker and defender strategies are coevolved in the context of a sin-

Algorithm 1 Example Coevolutionary Algorithm

Input:

T: number of iterations $\quad \mathcal{L}$: Fitness function

μ: mutation probability, $\quad \lambda$: population size

$\mathbf{A}_0 \leftarrow [\mathbf{a}_{1,0}, \ldots, \mathbf{a}_{\lambda,0}] \sim \mathcal{U}(\mathcal{A})$ ▷ Initialize minimizer population

$\mathbf{D}_0 \leftarrow [\mathbf{d}_{1,0}, \ldots, \mathbf{d}_{\lambda,0}] \sim \mathcal{U}(\mathcal{D})$ ▷ Initialize maximizer population

$t \leftarrow 0$ ▷ Initialize iteration counter

repeat

  $t \leftarrow t + 1$ ▷ Increase counter

  $\mathbf{A}_t \leftarrow select(\mathbf{A}_{t-1}))$ ▷ Selection

  $\mathbf{A}_t \leftarrow perturb(\mathbf{A}_t, \mu))$ ▷ Mutation

  ▷ Best minimizer

  $\mathbf{a}'_*, \mathbf{d}'_* \leftarrow \arg\min_{\mathbf{a} \in \mathbf{A}_t} \arg\max_{\mathbf{d} \in \mathbf{D}_{t-1}} \mathcal{L}(\mathbf{a}, \mathbf{d})$

  ▷ Replace worst minimizer

  if $\mathcal{L}(\mathbf{a}'_*, \mathbf{d}'_*) < \mathcal{L}(\mathbf{a}_{\lambda,t-1}, \mathbf{d}_{\lambda,t-1})$ **then**

    $\mathbf{a}_{\lambda,t-1} \leftarrow \mathbf{a}'_*$ ▷ Update population

  end if

  $\mathbf{A}_t \leftarrow \mathbf{A}_{t-1}$ ▷ Copy population

  $t \leftarrow t + 1$ ▷ Increase counter before alternating to maximizer

  $\mathbf{D}_t \leftarrow select(\mathbf{D}_{t-1}))$ ▷ Selection

  $\mathbf{D}_t \leftarrow perturb(\mathbf{D}_t, \mu))$ ▷ Mutation

  ▷ Best maximizer

  $\mathbf{a}'_0, \mathbf{d}'_0 \leftarrow \arg\min_{\mathbf{a} \in \mathbf{A}_t} \arg\max_{\mathbf{d} \in \mathbf{D}_t} \mathcal{L}(\mathbf{a}, \mathbf{d})$

  ▷ Replace worst maximizer

  if $\mathcal{L}(\mathbf{a}'_0, \mathbf{d}'_0) > \mathcal{L}(\mathbf{a}_{\lambda,t}, \mathbf{d}_{\lambda,t-1})$ **then**

    $\mathbf{d}_{\lambda,t-1} \leftarrow \mathbf{d}'_0$ ▷ Update population

  end if

  $\mathbf{D}_t \leftarrow \mathbf{D}_{t-1}$ ▷ Copy population

until $t \geq T$

$\mathbf{a}_*, \mathbf{d}_* \leftarrow \arg\min_{\mathbf{a} \in \mathbf{A}_T} \arg\max_{\mathbf{d} \in \mathbf{D}_T} \mathcal{L}(\mathbf{a}, \mathbf{d})$ ▷ Best minimizer

return $\mathbf{a}_*, \mathbf{d}_*$

gle, custom, abstract computer network defense simulation.

Framework Components

Coevolutionary Algorithms

The framework supports diverse behavior by executing algorithms that vary in synchronization of the two populations and solution concepts. (Prado Sanchez 2018; Pertierra 2018). Working within a fixed time or fitness evaluation budget, the framework also

1. Caches engagements to avoid repeating them;
2. Uses Gaussian process estimation to identify and evaluate the most uncertain engagement (Pertierra 2018);
3. Uses a recommender technique to approximate some adversary's fitnesses (Pertierra 2018); and
4. Uses a *spatial grid* to reduce complete pairwise engagements to a Moore neighborhood quantity (Mitchell 2006; Williams and Mitchell 2005).

Engagement Environment

The engagement component is flexible and can support a problem-specific network testbed, simulator or model. The abstraction level of the use case determines the choice of a simple to more detailed mod-sim or even the actual engagement environment. Mod-sim is appropriate when testbeds incur long experimental cycle times or do not abstract away irrelevant detail.

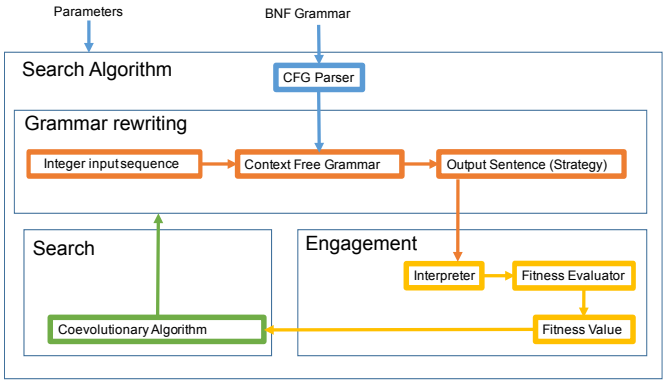

Figure 2: A BNF grammar and search parameters are used as input. The grammar rewrites the integer input to a sentence. Fitness is calculated by interpreting the sentence and then evaluate it. The search component, a coevolutionary algorithm, modifies the solutions using two central mechanisms: fitness based selection and random variation.

Adversary Representation

The framework uses grammars to express open ended behavioral action sequences for attack and defense strategies (a.k.a controllers). See Figure 2 and (O'Neill and Ryan 2003) for more details. While the framework's grammars currently are strategic in nature, we foresee incorporating higher level behavior related to plans and goals.

A grammar is introduced in Backus Naur Form (BNF) and describes a language in the problem domain. The BNF description is parsed to a context free grammar representation. Its (rewrite) rules express how a sentence, i.e. test or solution, can be composed by rewriting a start symbol. The adversaries are fixed length integer vectors that are use to control the rewriting. To interpret them, in sequence each of the vector's integers is referenced. This resulting sentence is the strategy that is executed. For solving different problems, it is only necessary to change the BNF grammar, engagement environment and fitness function of the adversaries. This modularity, and reusability of the parser and rewriter are efficient software engineering and problem solving advantages. The grammar additionally helps communicate the framework's functionality to stakeholders by enabling conversations and validation at the domain level. This contributes to stakeholder confidence in solutions and the framework.

Decision Support

Competitive coevolution has the following challenges (Sanchez et al. 2018; Prado Sanchez 2018):

1. Solutions and tests are not on comparable on a "level playing field" because fitness is based solely on the context of engagements.
2. Blind spots, unvisited by the algorithms may exist.
3. From multiple runs, with one or more algorithms, it is unclear how to automatically select a "best" solution.

The framework's decision support module, `ESTABLO`, see Figure 3, addresses these challenges. `ESTABLO`: *A)* runs competitive coevolutionary search algorithms with different solution concepts; *B)* combines the best solutions and tests at the end of each run into a compendium; *C)* competes each solution against different test sets, including the compendium and a set of unseen tests, to measure its performance according to different solution concepts; *D)* selects the "best" solutions from the compendium using a ranking and filtering process; and *E)* visualizes the best solutions to support a transparent and auditable decision.

Use Cases of the Framework

In this section we demonstrate use cases of the Adversarial AI framework. Broadly their goal is to identify defensive configurations that are effective against a range of potential adversaries.

DOS Attacks on Peer-to-Peer Networks

A peer-to-peer (P2P) network is a robust and resilient means of securing mission reliability in the face of extreme distributed denial of service (DDOS) attacks. The project named RIVALS (Garcia et al. 2017) assists in developing P2P network defense strategies against DDOS attacks. It models adversarial DDOS attack and defense dynamics to help identify robust network design and deployment configurations that support mission completion despite an ongoing attack.

RIVALS models DDOS attack strategies using a variety of behavioral languages ranging from simple to complex. A simple language e.g. allows a strategy to select one or more network servers to disable for some duration. Defenders can choose one of three different network routing protocols: shortest path, flooding and a peer-to-peer ring overlay to try to maintain their performance. A more complex one allows a varying number of steps over which the attack is modulated in duration, strength and targets and can even include online adaptation based on observed impact. Defenders can adapt based on local or global network conditions. Attack completion and resource cost minimization serve as attacker objectives. Mission completion and resource cost minimization are the reciprocal defender objectives. RIVALS has a suite of coevolutionary algorithms that use archiving to maintain progressive exploration and that support different solution concepts as fitness metrics.

An example of attackers from `ESTABLO` on a mobile resource allocation defense used in RIVALS (Sanchez et al. 2018) is shown in Figure 4. The mobile asset placement defense challenge is to optimize the strategic placement of assets in the network. While under the threat of node-level DDOS attack, the defense must enable a set of tasks. It does this by fielding feasible paths between the nodes that host the assets which support the tasks. A mobile asset is, for example, mobile personnel or a software application that can be served by any number of nodes. A task is, for example, the connection that allows personnel to use a software application.

Availability Attacks on Segmented Networks

Attackers also often introduce malware into networks. Once an attacker has compromised a device on a network, they can move to connected devices, akin to contagion. This use case considers *network segmentation*, a widely recommended defensive strategy, deployed against the threat of serial network security attacks that delay the mission of the network's operator (Hemberg et al. 2018) in the context of malware spread.

Network segmentation divides the network topologically into *enclaves* that serve as isolation units to deter inter-enclave contagion. How much network segmentation is helpful is a tradeoff. On the one hand, a more segmented network provides less mission efficiency because of increased overhead in inter-enclave communication. On the other hand, smaller enclaves contain compromise by limiting the spread rate, and their cleansing incurs fewer mission delays. Adding complexity, given some segmentation, a network operator can further use threat monitoring and network cleansing policies to detect and dislodge attackers but they come with a tradeoff of cost versus efficacy.

The use case assumes a network supports an enterprise in carrying out its business or *mission*, and that an adversary employs *availability attacks* against the network to disrupt this mission. Specifically, the attacker starts by using an exploit to compromise a vulnerable device on the network. This inflicts a mission delay when a mission critical device is infected. Then, the attacker moves laterally to compromise additional devices and maximally delay the mission. The network and its segments are pre-determined but the placement of critical devices within an enclave and the deployment of defensive threat monitoring device are open to optimization.

The use case employs a simulation model as its engagement environment. Malware contagion of a specific spread rate is assumed. The defender decides placement of mission devices and tap sensitivities in the enclaves. The attacker decides the strength, duration and number of attacks in an attack plan targeting all enclaves. For a network with a set of four enclave topologies, the framework is able to generate strong availability attack patterns that were not identified *a priori*. It also identifies effective configurations that minimize mission delay when facing these attacks.

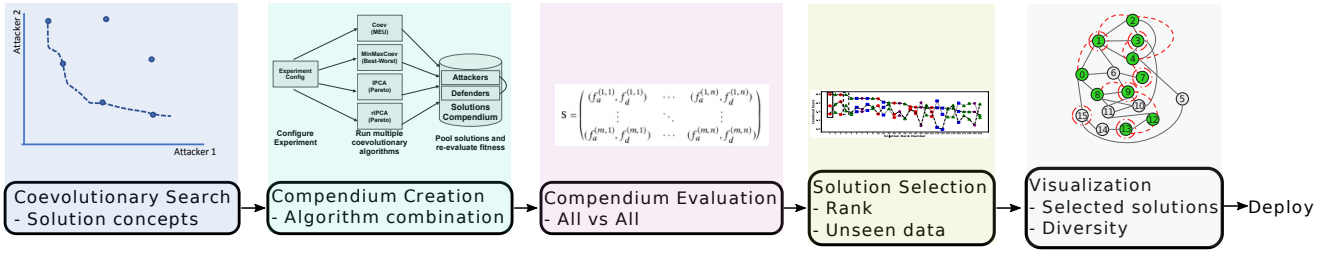

Figure 3: Overview of the ESTABLO framework for decision support through selection and visualization by using a compendium of solutions from coevolutionary algorithms.

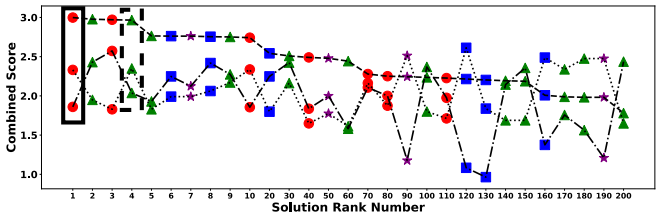

Figure 4: The x axis shows a sorted subsample of attackers (note, the top 10 are shown and then every tenth) and the y axis shows the ranking score. The ranking is done on the scores from the compendium. The values for the same run and unseen test sets are shown on separate lines. The algorithm used to evolve the attacker is shown by the marker and the color. The attacker in the box with the solid line is the top ranked solution from the Combined Score ranking schemes. The solution in the dashed box is the top ranked solution from the Minimum Fitness ranking scheme.

Internal Reconnaissance in Software Defined Networks

Once an adversary has compromised a network endpoint, they can perform network reconnaissance (Sood and Enbody 2013). After reconnaissance provides a view of the network and an understanding of where vulnerable nodes are located, they are able to execute a plan of attack. One way to protect against reconnaissance is by obfuscating the network to delay the attacker. This approach is well suited to software defined networks (SDN) such as those being used in many cloud server settings because it requires programmability that they support (Kirkpatrick 2013). The SDN controller knows which machines are actually on the network and can superficially alter (without function loss) the network view of each node, as well as place decoys (honeypots) on the network to mislead, trap and slow down reconnaissance.

One such multi-component deceptive defense system (Achleitner, Laporta, and McDaniel 2016) foils scanning by generating "camouflaged" versions of the actual network and providing them to hosts when they renew their DHCP leases. We use this deception system and mininet (Team 2018) within the framework as an engagement environment. This allows us to explore the dynamics between attacker and defender on a network where the deception and reconnaissance strategies can be adapted in response to each other (Pertierra 2018). A deception strategy is executed through a modified POX SDN controller. A reconnaissance strategy is executed by a NMAP scan(Lyon 2018). The attacker strategy includes choices of: which IP addresses to scan, how many IP addresses to scan, which subnets to scan, the percent of the subnets to scan, the scanning speed, and the type of scan. The defender strategy includes choices of: the number of subnets to setup, the number of honeypots, the distribution of the real hosts throughout the subnets, and the number of real hosts that exist on the network. Fitness is comprised of four components: how fast the defender detects that there is a scan taking place, the total time it takes to run the scan, the number of times that the defender detects the scanner, and the number of real hosts that the scanner discovers. Through experimentation and analysis, the framework is able to discover certain configurations that the defender can use to significantly increase its ability to detect scans. Similarly, there are specific reconnaissance configurations that have a better chance of being undetected.

Conclusion

We have described an AI framework that recreates, in an abstract way, the adversarial, competitive coevolutionary process that occurs in security scenarios. We presented its current use cases and how we harvest defensive solutions from it. Future work includes extending it to support more cyber security applications, considering other use cases and developing more efficient or true to reality algorithms.

Acknowledgments

This material is based upon work supported by DARPA. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements. Either expressed or implied of Applied Communication Services, or the US Government.

References

Achleitner, S.; Laporta, T.; and McDaniel, P. 2016. Cyber deception: Virtual networks to defend insider reconnaissance. *In Proceedings of the 2016 International Workshop on Managing Insider Security Threats* 57–68.

Arora, S.; Ge, R.; Liang, Y.; Ma, T.; and Zhang, Y. 2017. Generalization and Equilibrium in Generative Adversarial Nets (GANs). *arXiv preprint arXiv:1703.00573*.

Bäck, T. 1996. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press.

Bongard, J. C., and Lipson, H. 2005. Nonlinear system identification using coevolution of models and tests. *IEEE Transactions on Evolutionary Computation* 9(4):361–384.

Ficici, S. G. 2004. *Solution concepts in coevolutionary algorithms*. Ph.D. Dissertation, Citeseer.

Garcia, D.; Lugo, A. E.; Hemberg, E.; and O'Reilly, U.-M. 2017. Investigating coevolutionary archive based genetic algorithms on cyber defense networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, 1455–1462. New York, NY, USA: ACM.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2672–2680.

Hemberg, E.; Zipkin, J. R.; Skowyra, R. W.; Wagner, N.; and O'Reilly, U.-M. 2018. Adversarial co-evolution of attack and defense in a segmented computer network environment. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 1648–1655. ACM.

Kirkpatrick, K. 2013. Software-defined networking. *Communications of the ACM* 56(9).

Lange, M.; Kott, A.; Ben-Asher, N.; Mees, W.; Baykal, N.; Vidu, C.-M.; Merialdo, M.; Malowidzki, M.; and Madahar, B. 2017. Recommendations for model-driven paradigms for integrated approaches to cyber defense. *arXiv preprint arXiv:1703.03306*.

Lyon, G. 2018. Nmap network scanner. `https://nmap.org/`. [Online; accessed 6-July-2018].

Miconi, T. 2009. Why coevolution doesn't "work": superiority and progress in coevolution. In *European Conference on Genetic Programming*, 49–60. Springer Berlin Heidelberg.

Miller, B. P.; Fredriksen, L.; and So, B. 1990. An empirical study of the reliability of unix utilities. *Communications of the ACM* 33(12):32–44.

Mitchell, M. 2006. Coevolutionary learning with spatially distributed populations. *Computational intelligence: principles and practice*.

O'Neill, M., and Ryan, C. 2003. *Grammatical evolution: evolutionary automatic programming in an arbitrary language*, volume 4. Springer.

Pertierra, M. 2018. Investigating coevolutionary algorithms for expensive fitness evaluations in cybersecurity. Master's thesis, Massachusetts Institute of Technology.

Popovici, E.; Bucci, A.; Wiegand, R. P.; and De Jong, E. D. 2012. Coevolutionary principles. In *Handbook of natural computing*. Springer. 987–1033.

Prado Sanchez, D. 2018. Visualizing adversaries - transparent pooling approaches for decision support in cybersecurity. Master's thesis, Massachusetts Institute of Technology.

Rothlauf, F. 2011. *Design of modern heuristics: principles and application*. Springer Science & Business Media.

Rush, G.; Tauritz, D. R.; and Kent, A. D. 2015. Coevolutionary agent-based network defense lightweight event system (candles). In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, 859–866. ACM.

Sanchez, D. P.; Pertierra, M. A.; Hemberg, E.; and O'Reilly, U.-M. 2018. Competitive coevolutionary algorithm decision support. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 300–301. ACM.

Sood, A., and Enbody, R. 2013. Targeted cyberattacks: a superset of advanced persistent threats. *IEEE security & privacy* 11(1):54–61.

Tambe, M., ed. 2012. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

Team, M. 2018. Mininet - realistic virtual sdn network emulator. `http://mininet.org/`. [Online; accessed 6-July-2018].

Thompson, B.; Morris-King, J.; and Cam, H. 2016. Controlling risk of data exfiltration in cyber networks due to stealthy propagating malware. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, 479–484. IEEE.

Williams, N., and Mitchell, M. 2005. Investigating the success of spatial coevolution. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 523–530. ACM.

Winterrose, M. L., and Carter, K. M. 2014. Strategic evolution of adversaries against temporal platform diversity active cyber defenses. In *Proceedings of the 2014 Symposium on Agent Directed Simulation*, 9. Society for Computer Simulation International.