SAND2022-14011

SANDIA REPORT
SAND20XX-XXXX
Unclassified Unlimited Release
Printed Click to enter a date

Sandia National Laboratories

ADROC: An Emulation Experimentation Platform for Advancing Resilience of Control Systems

Jamie E. Thorpe
Ray Fasano
Michael Livesay
Meghan A. Sahakian
Hannah Stroble
Eric Vugrin

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

ABSTRACT

Cyberattacks against industrial control systems have increased over the last decade, making it more critical than ever for system owners to have the tools necessary to understand the cyber resilience of their systems. However, existing tools are often qualitative, subject matter expertise-driven, or highly generic, making thorough, data-driven cyber resilience analysis challenging.

The ADROC project proposed to develop a platform to enable efficient, repeatable, data-driven cyber resilience analysis for cyber-physical systems. The approach consists of two phases of modeling: computationally efficient math modeling and high-fidelity emulations. The first phase allows for scenarios of low concern to be quickly filtered out, conserving resources available for analysis. The second phase supports more detailed scenario analysis, which is more predictive of real-world systems. Data extracted from experiments is used to calculate cyber resilience metrics. ADROC then ranks scenarios based on these metrics, enabling prioritization of system resources to improve cyber resilience.

ACKNOWLEDGEMENTS

The ADROC team would like to acknowledge all the people who have worked on the ADROC project over the last two years: Amanda Gonzales, Josh Morris, Tim Ortiz, Chris Mairs, Andrew Hahn, Chris Lamb, and Trisha Miller. We also thank Craig Lawton, Derek Hart, Jen Depoy, and the Resilient Energy Systems Mission Campaign for their programmatic efforts in support of this project. This work was funded under the Sandia National Laboratories Laboratory Directed Research and Development Program Project Number 222442 and Titled "ADROC: An Emulation Experimentation Platform for Advancing Resilience of Control Systems".

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS AND TERMS

Acronym/Term	Definition
ADROC	Advancing resilience of control systems
CARV	Consensus Algorithm Relay Voting (branch of HARMONIE-SPS LDRD project)
CIR	Critical infrastructure resilience
DOS	Denial of service
DQN	Deep Q-learning with neural network
HMM	Hidden Markov model
ICS	Industrial control system
IT	Information technology
LDRD	Lab Directed Research and Development; internally funded projects at Sandia National Laboratories
LSTM	Long short-term memory
MDP	Markov decision process
POMDP	Partially observable Markov decision processes
PWR	Pressurized water reactor
RL	Reinforcement learning
RTDS	Real time digital simulator, usually in the context of power system modeling
SCADA	Supervisory control and data acquisition
SCRAM	Safety cut rope axe man (historically); in modern day, refers to a rapid emergency shut down for a nuclear reactor
SME	Subject matter expert
STAB	State trait aggregation in Baum-Welch

This page left blank.

1. INTRODUCTION

Over the past decade, industrial control systems (ICS) and supporting information networks have experienced an increasing number of cyberattacks. Recent attacks on the Ukrainian power grid [1]; nuclear power plants in India [2]; and shipping ports [3], a fuel pipeline [4], and a water treatment facility [5] in the United States have demonstrated that cyberattacks can disrupt operations for a wide variety of infrastructures. Further, the tools available to adversaries to perform these attacks continue to grow [6]. As a result, improved cyber resilience for critical infrastructure has become a national and international priority. However, with so many different threats out there, and the fact that each ICS is typically a unique system with different vulnerabilities and risks, it can be extremely difficult for system owners to know how to direct their limited resources to best protect their systems.

One vital tool to ICS owners is the ability to model and study critical systems before an attack occurs. This can help in understanding how resilient or vulnerable the system is to attack. The vast majority of critical infrastructure resilience (CIR) modeling work has focused on physical threats such as natural disasters and physical interdiction [7] [8] [9]. These techniques don't easily extend to cyberattack scenarios that include adaptive, dynamic, and stealthy threats. Established cyber resilience assessment methods [10] [11] provide qualitative analyses, but these rely heavily, if not exclusively, on expert judgment, preventing validation and limiting their usefulness for supporting cyber resilience design efforts. Commercial cyber threat emulation tools such as CALDERA [12]and Cobalt Strike [13] provide high fidelity threat representations, but these approaches have scalability limits that prevent rapid exploration of the large number of possible cyber threats.

Overall, the following technical gaps exist and pose significant challenges to characterization of cyber risks and designing more resilient ICSs:

- Modeling approaches that don't represent adaptive, dynamic cyber threats in a high-fidelity manner that is scalable and that can be validated.
- Reliance on qualitative assessments that don't provide quantitative, cyber-specific measures to inform cyber risk analyses and the design of cyber resilience ICSs.

ADROC (ADvancing Resilience Of Control systems) proposed to address these gaps through the development of a cyber threat modeling and resilience experimentation platform for characterization and quantification of cyber vulnerabilities and risks. We hypothesized that the noted challenges could be overcome through hybrid threat modeling techniques and cyber-physical resilience metrics. Therefore, the ADROC project strived to develop a platform for data-driven, repeatable cyber experimentation and threat prioritization.

The ADROC platform consists of two phases: computationally efficient Math Modeling in Phase 1 (see the top half of Figure 1) and high-fidelity Emulation Modeling in Phase 2 (see the bottom half of Figure 1). In this way, ADROC leverages multiple levels of fidelity for system and threat modeling in order to achieve a more efficient analysis. Low-fidelity math models are used to down-select scenarios to those of higher interest, and high-fidelity emulation is applied to model these high-interest scenarios in a way that can be more deeply understood and quantified. The application of the platform will be discussed later in this report.

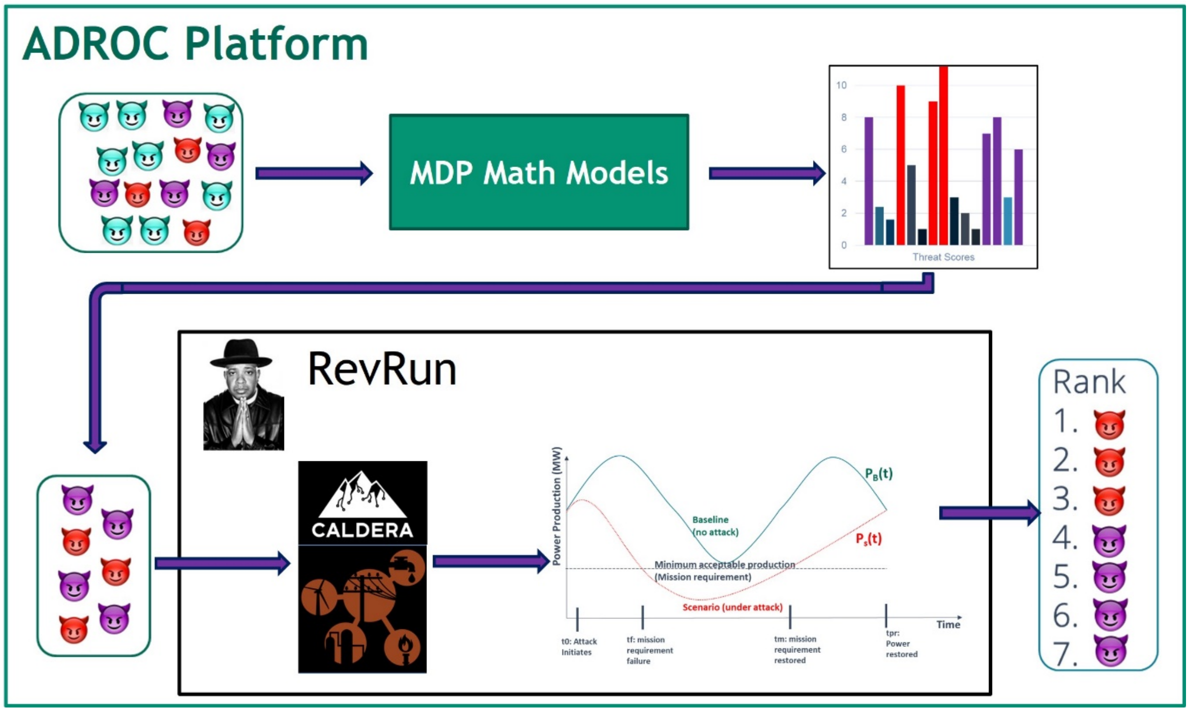


Figure 1. Overview of the workflow of the ADROC platform.

The rest of this report is organized as follows. Sections 2 and 3 discuss the development of the math and emulation models respectively. Section 4 illustrates how these models were combined into a single integrated platform and describes our methods of experiment control and metric calculation. Section 5 overviews how a future user might apply the ADROC platform to perform their own analysis. Sections 6, 7, and 8 discuss the three use cases that were used for platform development over the course of the project. Section 9 lists the papers, presentations, and intellectual property that resulted from the ADROC project. Section 10 concludes the report with a discussion on future work.

2. ADROC PHASE 1: MATH MODELING

This section describes the classes of mathematical models included in the first phase of the ADROC approach, specifically Markov decision processes (MDPs) and partially observable Markov decision processes (POMDPs).

2.1. MDP Models

MDPs are a commonly used modeling technique used to find optimal policies for managing stochastic control processes. An MDP is a quintuple $(\mathcal{S},\mathcal{A},T,R,\gamma)$ can be compactly defined as follows. Where $t = 0,1,\ldots$, denotes a set of discrete timesteps, let

- $\mathcal{S} = 0,1,\ldots,N$ be a set of finite achievable states.
- $\mathcal{A} = 0,1,\ldots,K$ be a set of finite actions that can be taken by an agent.
- T be the matrix consisting of all state transition probabilities. Let $\mathbb{P}(\mathcal{S}_{t+1} = j \mid \mathcal{S}_t = i,\mathcal{A}_t = a)$ denote the probability of transitioning to state j at timestep $t+1$, given the process is in state i at timestep t and action a is taken at timestep t. In MDPs, state transition probabilities are constant with respect to timestep.
- $R(\mathcal{S}_{t+1} = j,\mathcal{S}_t = i,\mathcal{A}_t = a)$ denotes the reward received when the process transitions to state j at timestep $t+1$, given the process is in state i at timestep t and action a is taken at timestep t. Note that a reward may be a positive or negative value.
- $0 \leq \gamma \leq 1$ is a constant discount factor that weights recent rewards more heavily than less recent rewards.

One can define a policy[1] to be a function such that $\pi(s) = a$; that is, given a state s, the policy specifies the action a that the agent will take. For a specified initial state s_0 , solution of the MDP problem finds the optimal policy π^* such that

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(\mathcal{S}_{t+1},\mathcal{S}_t,\mathcal{A}_t = \pi(i)) \middle| \mathcal{S}_0 = s_0\right]$$

That is, one attempts to find the optimal policy that maximizes the expected rewards. Similar optimization problems can be formulated for finite time horizon problems.

MDPs have been studied for decades (i.e., [14]) and have a large body of established theoretical results[2], including existence proofs for optimal policies. Numerical methods for solving MDPs, such as linear programming, policy iteration, and value iteration methods, are well established and have been demonstrated[3]. Additionally, these methods can be used for relatively large problems (consisting of thousands of states). Due to their relatively simple formulation, established numerical solution methods, and scalability, MDPs have been used to study a wide variety of applications, including cybersecurity [15], supply chains [16], satellite networks [17], and emergency response management [18].

[1] Using the policy definition above specifies a deterministic policy, i.e., one action is output per state input. With minor modifications, one can specify stochastic policies and analogous optimization problems.
[2] For example, see M. Littman et al. (1995), "On the complexity of solving Markov decision problems," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 394-402.
[3] Ibid.

Under the ADROC project, we use MDPs as a computationally efficient means to model attacker actions, their progress, and defender response options. We model a cyberattack on an ICS as a multi-stage process.

- The set of states $\mathcal{S}$ represents the various stages of attacker progress.
- The set of actions $\mathcal{A}$ represents that options that the ICS defender can take to thwart attacker progress.
- The matrix of transition probabilities T describes the rate at which the attacker progresses towards its objective, given a set of assumptions about attacker behaviors and strategies.

Using an MDP, one can rank attacks according to the rate at which they achieve their attack objective, the probability of achieving attack goals, probability that the attack fails, etc. Additionally, one can determine the optimal combination of actions that the defender can take to thwart the attack.

The computational efficiency of MDPs make them attractive for modeling cyberattack scenarios, but MDPs include a fundamental assumption that is often inconsistent with cyberattacks. That is, MDPs assume that the defender has full knowledge of the attacker's current state and progress.

Additionally, parameterizing the MDP can be challenging. Determining the set of defender actions for cyberattack scenarios is fairly straightforward; the set of actions is determined by the ICS design, defender investments made prior to the attack, and resourcefulness of the defender. However, some practical challenges make development of attacker stages and transition probabilities more difficult. Though threat intelligence can provide information about attacker tactics, techniques, and procedures, it is unlikely that threat intelligence can provide precise, quantitative parameters that are needed for MDP models.

Thus, despite the attractiveness of MDPs for modeling cyberattack scenarios, MDPs have some significant limitations that can hinder the use of MDPs to analyze such scenarios. The following sections describe the use of POMDPs and reinforcement and model learning to address these limitations.

2.2. POMDP Models

POMDPs (i.e., [19]) relax the assumptions made by MDPs and are able to model the case where the defender does not have full knowledge of attacker progress. Rather than having this knowledge, defender observations inform which action the defender should take. In a cybersecurity-driven context, observations can be informed by intrusion detection sensors and other features one might use to monitor a computer network. Observations inform the defender about the attacker's progress, but they do not provide the defender with 100% clarity on the state of the attack.

This modeling approach is much more consistent with real-world cyberattacks, but uncertainties related to attack implementation and the potential for false positives and negatives from intrusion detection systems can pose significant challenges. Such uncertainties cause the agent in the POMDP to struggle to make relevant predictions.

When the Markov model of the POMDP is known, the POMDP can be converted to an MDP of state distributions called a Belief-MDP [19], which is more easily learned by the agent. However, we are unlikely to have such information about the cyberattack directly, and so we say the model is "hidden". This hidden model needs to be understood before we can apply reinforcement learning techniques to understanding the attacker's strategy.

Under ADROC, we break down the POMDP learning into two parts: hidden Markov model (HMM) learning [20] to convert the POMDP into a Belief-MDP, and reinforcement learning (RL) to learn a good policy. These two parts can be thought of as first learning a model of the attacker's strategy and likely position at a given time, and then learning a defender strategy to slow the attacker down.

2.2.1. *Hidden Markov Model Learning*

This section describes how a cyberattack is modeled by the learning agent using the HMM-learner Baum-Welch algorithm [21], and how Baum-Welch is extended to our new algorithm, State Trait Aggregation in Baum-Welch, or "STAB".

A POMDP is an eight-tuple $(\mathcal{S},\mathcal{O},\mathcal{A},T,O,d,R,\gamma)$ which are the state space, observation space, action space, transition rate matrix, emission rate matrix, initial distribution, reward function, and discount factor respectively. More specifically,

- $\mathcal{S} = 0,1,\ldots,N$ be a set of finite achievable states.
- $\mathcal{O} = 0,1,\ldots,M$ be a set of finite observations.
- $\mathcal{A} = 0,1,\ldots,K$ be a set of finite actions that can be taken by an agent.
- T be the matrix consisting of all state transition probabilities. Let $\mathbb{P}(\mathcal{S}_{t+1} = j \mid \mathcal{S}_t = i, \mathcal{A}_t = a)$ denote the probability of transitioning to state j at timestep $t + 1$, given the process is in state i at timestep t and action a is taken at timestep t. In MDPs, state transition probabilities are constant with respect to timestep.
- O be the emission rate matrix consisting of probabilities $\mathbb{P}(Y_{t+1} = j \mid \mathcal{S}_t = i, \mathcal{A}_t = a)$.
- d be the initial belief distribution and is a vector consisting of N numbers ranging between 0 and 1 (inclusive) and that sum to 1.
- $R(\mathcal{S}_{t+1} = j, \mathcal{S}_t = i, \mathcal{A}_t = a)$ denote the reward received when the process transitions to state j at timestep $t + 1$, given the process is in state i at timestep t and action a is taken at timestep t. Note that a reward may be a positive or negative value.
- $0 \leq \gamma \leq 1$ is a constant discount factor that weights recent rewards more heavily than less recent rewards.

We chose to use HMM-learning to reduce the POMDP into a Belief-MDP instead of the direct approach of using long short-term memory (LSTM) networks on the POMDP [22] because of the high uncertainty and sparseness of observables in cyber problems. Under these conditions, $\mathbb{P}(Y_{t+1} \mid Y_t,\ldots,Y_{s+1})$, in which Y_t represent observations, is a poor approximation of $\mathbb{P}(Y_{t+1} \mid Y_t,\ldots,Y_1)$ for large enough values of "s" so that using LSTM networks on the POMDP is not effective.

Consider the map $r{:}\mathcal{S}\to\mathcal{P}(\mathcal{S})$, where $\mathcal{P}$ is the power set. We say that r is a *state relation*, if for any i_1, $i_2 \in \mathcal{S}$, $j_1 \in r(i_1)$, $j_2 \in r(i_2)$, we have that

$$\mathbb{P}\big(X_{t+1} = j_1 \,\big|\, X_t = i_1\big) = \mathbb{P}\big(X_{t+1} = j_2 \,\big|\, X_t = i_1\big).$$

Furthermore, we call $\mathbb{P}\big(X_{t+1} = j_1 \,\big|\, X_t = i_1\big)$ the *state relation rate* of r.

Note that for any $i,j \in \mathcal{S}$ there is a *single edge state relation* r so that $r(i) = \{j\}$ and $r(\tilde{i}) = \{\}$ for $\tilde{i} \neq i$. If r_1 and r_2 are distinct *state relation*, and if $r_1(i)$ and $r_2(i)$ intersect for any $i \in [N]$, then they are called *conflicting state relations*.

Suppose that $\mathcal{R}$ is a set of non-conflicting state relations, so that for any $i,j \in \mathcal{S}$ there is an $r \in \mathcal{R}$ so that $j \in r(i)$. Let T_r be the *state relation rate* of r. Then by letting $T_{ij} = T_r$ for $i \in \mathcal{S}$ and $j \in r(i)$ we reduce the amount of data needed for learning which is an exponential function the number of edges to the number of relations.

In this way, the vanilla Baum-Welch algorithm naturally extends to our STAB algorithm, where we use all the inferred experience of a *state relation* to maximize our certainty of the transition rates that it expresses. This allows us to limit the amount of data needed for learning the HMM, which is key in use cases such as cyber systems where there are a large number of potential states with potentially a limited amount of data about each individual state.

Let *H* be the time horizon, *N* the state space order, *M* the observation space order, *T* the transition rate matrix, *O* the emission rate matrix, *d* the distribution, $\{y_t\}_{t=0}^{H}$ the sequence of observations in a given episode, and $\mathcal{R}$ the set of non-conflicting state relations so that for every $i,j \in \mathcal{S}$ there is an $r \in \mathcal{R}$ so that $j \in r(i)$. The STAB algorithm which updates the HMM (T,O,d) using the observations $\{y_t\}_{t=0}^{H}$ and the state relations $\mathcal{R}$ is seen here.

```
Algorithm STAB
Given: $H, N, M, T, O, d, \{y_t\}_{t=0}^{H}, \mathcal{R}$
  $\alpha_i(0) \leftarrow \pi(i)O_{iy_0}$
  for t = 0,...,H-1 do
      $\alpha_i(t+1) \leftarrow O_{iy_{t+1}} \sum_{j=1}^{N} T_{ji}\alpha_j(t)$
  end for
  $\beta_i(H) \leftarrow 1$
  for t = H-1,...,0 do
      $\beta_i(t) \leftarrow \sum_{j=1}^{N} \beta_j(t+1)T_{ij}O_{jy_{t+1}}$
  end for
  for t = 0,...,H do
      for i = 1,...,N do
          $\gamma_i(t) \leftarrow \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N}\alpha_j(t)\beta_j(t)}$
      end for
  end for
  for t = 0,...,H-1 do
      for i = 1,...,N do
          for j = 1,...,N do
              $\xi_{ij}(t) \leftarrow \frac{\alpha_i(t)\beta_j(t+1)T_{ij}O_{jy_{t+1}}}{\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n(t)\beta_m(t+1)T_{nm}O_{my_{t+1}}}$
          end for
      end for
  end for
  for $r \in \mathcal{R}$ do
      $\widehat{T}_r \leftarrow \frac{\sum_{t=0}^{H-1}\sum_{i=1}^{N}\sum_{j \in r(i)} \xi_{i,j}(t)}{\sum_{t=0}^{H-1}\sum_{i=1}^{N}\sum_{j \in r(i)} \gamma_i(t)}$
  end for
  for i = 1,...,N do
      $\widetilde{d}_i \leftarrow \gamma_i(0)$
      for o = 1,...,M do
          $\widehat{O}_{io} \leftarrow \frac{\sum_{t=0}^{H} \mathbf{1}_{\{y_t=o\}}\gamma_i(t)}{\sum_{t=0}^{H}\gamma_i(t)}$
      end for
  end for
  for i = 1,...,N do
      for $r \in R$ do
          for $j \in r(i)$ do
              $\widetilde{T}_{ij} \leftarrow \widehat{T}_r$
          end for
      end for
      $\widetilde{T}_{ij} \leftarrow \frac{\widetilde{T}_{ij}}{\sum_{n=1}^{N}\widetilde{T}_{in}}$
  end for
```

Two illustrations where STAB would be effective due to the high portion of edges coming from common state relations are seen in Figures 2 and 3. The state relations utilized by STAB would drop the complexity significantly compared to vanilla Baum-Welch, as the algorithm's complexity depends on the number of relations instead of the number of edges, so significantly less data is required.

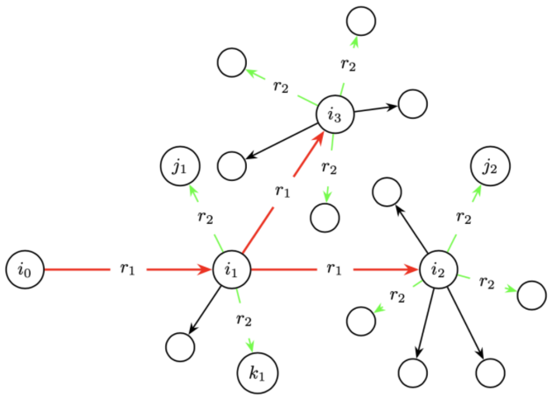


Figure 2. Illustration of supply chain network with State Trait Aggregation.

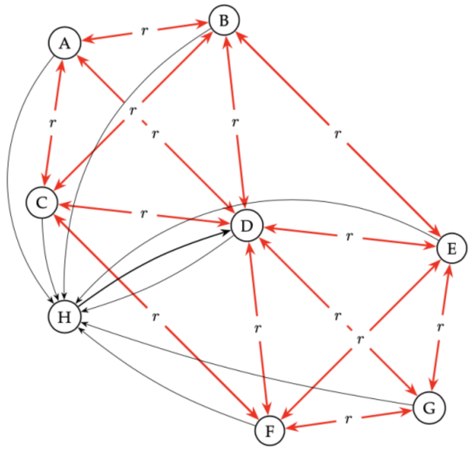


Figure 3. Illustration of peer-to-peer network with State Trait Aggregation.

STAB, a generalization of Baum-Welch from states to trait-based state classes, allows us to take advantage of similar components in networks to limit the amount of data required to learn the model. As an analogy, imagine how a disease spreading in one city tells us much about how that disease may spread in another city.

For the development of the POMDP models, we focused our study on cyberattacks with worm-like behavior (referred to in 3.2.2 simply as "cyberattacks" for brevity), meaning the attack moves from node to node (or machine to machine) in a system. Model learning is very complex in the state and observation spaces, so we restricted ourselves further to worms that only transfer to other nodes, rather than those that duplicate, as this would cause an exponentially large state space. Even with this simplification, we found that large networks required a lot of data to learn, and that the STAB algorithm was an improved approach to the problem over vanilla Baum-Welch.

2.2.2. Reinforcement Learning Algorithms on Belief-MDP

This section describes how RL is used to determine optimal policies against the attack based on the Belief-MDP determined by HMM learning, with our STAB algorithm. The performance of the RL agent determines the severity of the attack – the better the agent/defender performs, the more they confound the attacker, and, in general, the less severe the attack will be.

The Belief-MDP approach might appear daunting because the new state space of the Belief-MDP is the space of distributions of the POMDP's states. Fortunately, the value function of the Belief-MDP is convex and piecewise linear and thus not much more complex for the agent to learn than the completely observable MDP.

Deep Q-Learning with neural networks (DQN) [23] is a commonly used RL method that seeks to find optimal policies to deterministic formulations, but the uncertainty with our problems of interest can cause problems for DQN. In particular, the optimal policies may not be robust to uncertainties and small errors. Under this project, we developed a generalization of the DQN algorithm to be robust to potential uncertainties. We introduce an additional term, σ, to generalize the DQN value update. We call the new algorithm σ-Safe DQN. The σ-Safe DQN value update equation is

$$q(X_t,A_t) \leftarrow q(X_t,A_t) + \alpha[R(X_t,A_t) + \gamma q_\sigma(X_{t+1}) - q(X_t,At)], with$$

$$q_\sigma(X) := (1-\sigma)\max_{a\,\in\,\mathcal{A}} q(X,a) + \frac{\sigma}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} q(X,a).$$

Note that 0-Safe DQN and DQN are equivalent. The performance improvement for σ-Safe DQN was significant in the presence of state transition uncertainty. For example, in use cases where a wrong move can be very costly, vanilla DQN can learn to take those risks anyway. σ-Safe DQN is better at learning to avoid such risky states.

In a completely observable MDP environment we called "observable cyber worm", we demonstrated how the σ-Safe DQN algorithm was able to revert a typical issue of DQN where it unlearns good policy solutions, see Figure 4.

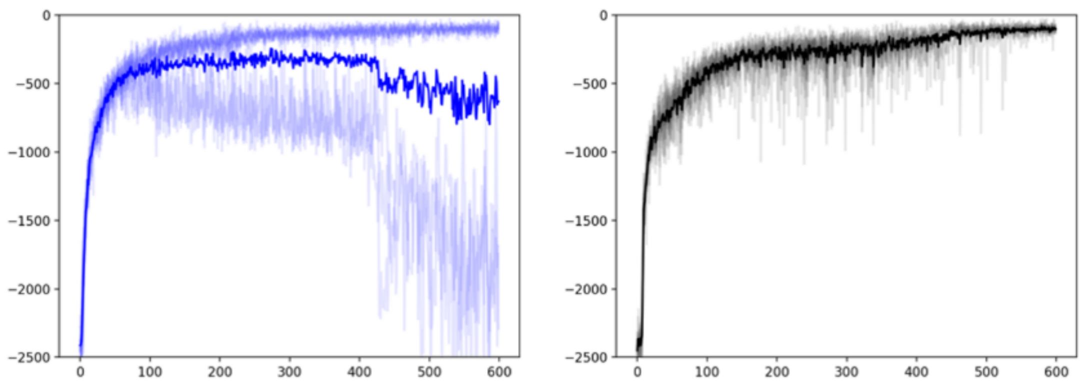


Figure 4. Ten training sessions of DQN (left) and 0.1-Safe DQN (right) in observable cyber worm.

In Section 2.2.1, we discussed the STAB algorithm for learning the HMM of a POMDP for the sake of converting the POMDP into a Belief-MDP, especially on networks with relatable edges, such as in cyber infrastructures. In Section 2.2.2, we explain how the Belief-MDP is similar in complexity to the completely observable MDP for optimal policy learning, and we give our σ-Safe DQN algorithm which thrives on uncertainty and risky states.

This page left blank.

3. ADROC PHASE 2: EMULATION MODELS

The second phase of the ADROC platform involves high-fidelity modeling of the system of concern and of threats against that system. This section will give a brief overview of some of the considerations made for the high-fidelity emulation experiments. More details, including discussion of the Use Case covered in Section 6 of this report, are included in already published work under ADROC [24].

3.1. System Modeling

Often, the term "Industrial Control System" is used to refer to the network that manages, monitors, and helps operate key physical processes in a system. Modeling this type of system might include supervisory control and data acquisition (SCADA) devices, typical operational technology communication protocols (e.g., modbus), and some representation of a physical process such as a power plant or water treatment facility. However, it is also important to consider components that might be connected to the ICS. These components may include firewalls, public-facing networks including web servers or databases, and internal networks including software for typical business functions (e.g., payroll or document processing). When considering system vulnerability to cyber threats, it is important to include all cyber-related aspects of the system in the model.

ADROC uses the SCEPTRE emulation platform [25] to emulate the system under study. In this context, emulation can be thought of as a virtual testbed. The level of fidelity modeled is dependent on the system, the resources available for developing the model, and the level of detail required to answer the key questions being asked by the analysis of the model. For example, if a key capability of an attack being studied is its ability to pivot through a system firewall, then the firewall should be modeled with realistic firewall rules in place.

Emulation models for ICS typically include virtual machines running real system software and communicating using realistic protocols. Virtual machines acting as firewalls implement realistic firewall rules for the given system. Physical processes in the ICS (e.g., pressure and temperature in a power plant) are simulated using mathematical models. Depending on the level of fidelity, models may include other factors such as end-user action simulation or connection to a virtual machine representing the Internet.

3.2. Threat Modeling

Traditional "threat" exploration within SCEPTRE has historically been done via manual interactive red teaming activities. These activities do not provide the level of automation and repeatability needed for the ADROC platform.

For automated threat representation, one of the simplest approaches is to model only the impact of the threat, taking success in early attack stages for granted. However, the ADROC project was interested in modeling multi-stage attacks in a higher fidelity way.

After an extensive review of available tools, ADROC chose to leverage CALDERA [12] to model a multi-stage cyberattack against the system. Attacks modeled in CALDERA may be defined by a variety of attacker capabilities, available "insider" information, and attack goals. The final stage(s) of the attack may include executing payloads, which can be flexible. The deterministic execution of CALDERA was also an attractive feature for ADROC as it allowed for experiment repeatability, which is a key trait of experimental rigor. In addition, the team developed code to fully automate the startup and execution of CALDERA, further increasing experiment repeatability.

One of the drawbacks of CALDERA for ADROC is that it was designed for penetration testing on real information technology (IT) systems. There were two key hurdles for ADROC to overcome in order to use CALDERA effectively:

- CALDERA often attempts to exploit resources that would be available in a lived-in system but not in a from-scratch emulated system, such a SSH configurations and internet access.
- CALDERA has a variety of capabilities to use against IT infrastructure, but very little (if anything) that could be used to impact the processes of an ICS.

Therefore, work under ADROC included efforts to integrate CALDERA into the emulated system such that its built-in attack capabilities could run, and to integrate the use of ICS-specific attack tools with CALDERA in order to impact the ICS components of the modeled system. The latter effort is discussed further in use case Sections 6, 7, and 8, illustrating the push to design the ADROC platform to be flexible to a variety of threats and payloads.

4. INTEGRATED PLATFORM

This section will overview the ADROC platform and discuss how the two previously described phases work together. This section will also briefly discuss some of the surrounding infrastructure, including data collection and cyber resilience metric calculation. More details about reinforcement learning for the math modeling stage can be found in an in-progress paper by Livesay, et al. [26]. More details about the emulation stage and the surrounding infrastructure, including discussion of the Use Case covered in Section 6 of this report, are included in previously published work by Thorpe, et al. [24].

4.1. Overview

An image of the developed ADROC Platform workflow is shown in Figure 5. The workflow is illustrated around a problem aiming to analyze and prioritize a series of threats against a system, although this is not the only purpose for ADROC. The goal of the ADROC platform is to quantify and compare the impact different scenarios have to the system's ability to perform its mission, regardless of the exact nature of those scenarios.

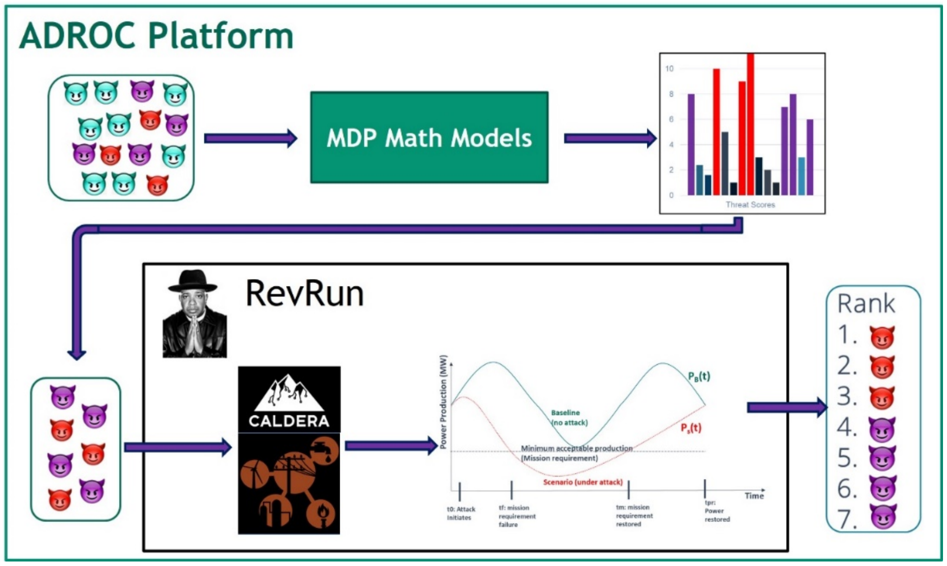


Figure 5. Overview of the workflow of the ADROC platform.

The process starts in the top left, with a potentially large pool of scenarios of interest. In the Figure 5 example, these scenarios are threat-focused, and may represent a set of minor variations on a single threat, or a group of very different threats to the system. In an analysis more focused on defensive strategies, the scenarios could be defined more broadly to encompass differences in system architecture or tools for intrusion detection.

Suppose that the number of scenarios exceeds the user's resources for emulation. To help down-select scenarios efficiently, scenarios enter Phase 1, the math modeling stage (illustrated here as a Markov Decision Process model, but the POMDP approach is an option as well). The goal of this stage is to get a first pass at characterizing the threats without requiring large amounts of computational power or time. This stage essentially acts as a filter, helping to eliminate scenarios that are clearly not of great concern to the system. For a threat-focused analysis, this might mean filtering out threats that have very limited impact to the mission of this specific system, or threats that the current system and tools can already mitigate effectively.

Scenarios that are still deemed to be of high concern are then sent to Phase 2 to be modeled using tools for high-fidelity ICS emulation (SCEPTRE) and threat modeling (CALDERA). Data is collected during emulation experiments and extracted to be processed. Processed data can then be used to compute cyber resilience metrics. These metrics enable data-driven prioritization of the scenarios. This prioritization can highlight which threats should be addressed before others, or which system configurations were more resilient to attack than others.

In summary, the goal of ADROC was to integrate threat models, algorithms, and metrics with SCEPTRE models of ICSs to create a platform for repeatable, quantitative cyber experimentation in support of cyber risk assessment and resilient design. Sections 4.2 and 4.3 discuss the two phases practically in the context of the rest of the platform.

4.2. Phase 1

Phase 1 involves using the lower-fidelity math models (i.e., MDPs and POMDPs) to represent systems and threats in a computationally efficient way. Section 2 discussed the technical details of the models developed for this phase under ADROC. This section will talk about how this phase fits in the context of the rest of the ADROC platform.

Within the ADROC platform, math models are used to describe potential cyberattacks on systems. These models are implemented numerically, and they can be rapidly solved and analyzed to determine which attacks, scenarios, and mitigations should be further studied in the second phase of the ADROC platform. The ADROC platform is agnostic to the type of math model used in this phase. A key step in the development of the models is the selection of appropriate model parameters. SME judgment and analysis of previously observed attacks can be used to select parameters, and information about the system and the cyberattack being described are key inputs to this process. See Section 6 and Thorpe, et al. [24] for an example of the parameterization of MDP's in a specific use case.

Regardless of the model selected, the output should be high-level characteristics about the scenarios studied. The exact characteristics will depend on the design of the math models. Several different characteristics may be studied for each defined scenario, and the set of characteristics must be considered as a whole. Section 6 discusses an example of potential output of the MDP's – characteristics such as the probability that the attack succeeds or the timesteps it took to succeed. The output of the RL agent is slightly different. The agent uses RL to learn a control response for the learned Belief-MDP and collects a score based on accumulated rewards. This score determines the danger of an attack based on its illusive or aggressive traits. In both examples, the results of the math models should help with the determination of which scenarios will be modeled in Phase 2, which we refer to as threats of "highest concern". This decision requires manual intervention and consideration of a variety of factors, including:

- For each characteristic output, how did each scenario compare? Were there any outlier scenarios, or did most scenarios perform approximately the same?
- Which characteristics are deemed most concerning or most important to this system? For example, is there more concern about the attacker's probability of success or the speed with which they can succeed?
- What resources are available for Phase 2? How many of the scenarios investigated in Phase 1 can you afford to run in emulation?

4.3. Phase 2

Phase 2 involves the high-fidelity system emulation and threat modeling for the scenarios of highest concern from the Phase 1 math models. Section 3 and Thorpe, et al. [24] talk about how the system and threat modeling was approached, as well as the use of a tool called RevRun [27]. This section will discuss more about RevRun and how it is used to drive ADROC Phase 2.

The input to Phase 2 is a description of a set of scenarios that are of highest concern to the system. These scenarios, which should have some threat component, can be emulated using CALDERA within the SCEPTRE platform. In addition to tools for representing the threat and the system, we need a tool to manage the experiments and to extract and process data. This is the function RevRun performs in the context of ADROC.

RevRun takes as input several configuration files. The primary configuration file defines the data that will be collected from each experiment, how that data with be processed, and how the processed data will be combined into a single cyber resilience score for each scenario. In addition, each scenario has its own configuration file which specifies the system, threat, and any other components specific to that scenario. At analysis time, RevRun is started with these configuration files. RevRun then runs each emulation experiment in serial in an automated fashion, applying the specified scenario components. RevRun uses a set of data collection tools to collect the specified data from each experiment. At the end of an experiment, data is extracted and stored in Elasticsearch [28]. From there, RevRun applies its library of defined preprocessing functions to process the collected data.

Once data has been collected and processed for each scenario, RevRun calculates cyber resilience metrics. The data of each scenario is compared to data from a baseline "null" scenario, and differences are calculated in a user-defined way. Impacts to physical processes, network traffic, and host health can be captured and combined into a single cyber resilience score for each scenario which quantifies how the scenario impacted the system's ability to perform its mission. The resulting cyber resilience scores given by RevRun are the primary output of the ADROC platform. These metrics can be used to prioritize scenarios by how greatly they impacted the system.

RevRun is the driving mechanism for ADROC Phase 2. The user must manually define the scenarios to be run based on high-concern scenarios identified in Phase 1. In addition, system mission drives the main RevRun data collection and processing configuration. After configuration, RevRun runs autonomously, eventually outputting the metrics which become the output for the entire ADROC platform.

This page left blank.

5. ADROC APPLICATION USER GUIDE

We designed the ADROC platform to be generally applicable across a variety of infrastructures, threats, and analysis goals. This generalizability was demonstrated through application to several different use cases over the course of the project (refer to Sections 6, 7, and 8 as examples of previously performed ADROC analyses). This section will walk through the process of applying the ADROC platform to analyzing a new system.

5.1. Before You Begin

Before you start planning any experiments, consider the following factors to determine whether ADROC is right for your work.

- *System Modeling* – What is the system you are interested in studying? Do you already have a model developed? Recall that ADROC leverages both math models and emulation models. If you do not have models pre-made, you will need to develop these models. Consider the time and resources you have available for this effort – high-fidelity emulation modeling in particular can take a lot of time.
- *Emulation Platform* – If your system of interest is already modeled as an emulation, what emulation platform are you using? The emulation experimentation components of ADROC have been developed to interface with SCEPTRE or minimega [29]. If you are using a different emulation platform, you may need to convert your existing models to one of these two platforms, use a different tool to manage the emulation experiments, or adapt the ADROC tools accordingly.
- *Scenarios of Interest* – Do you have definable scenarios of interest to run? All scenarios run by ADROC must be manually defined, implemented, and configured. ADROC cannot be told to search for or develop new scenarios, including threats, vulnerabilities, or mitigation strategies.
- *Resilience-Focused Goal* – Is your end analysis goal related to system cyber resilience? Although the basic tools of ADROC can be used to achieve a variety of goals which require system modeling, data collection, and/or metric calculation, the existing data processing and metrics functions are focused on calculating the cyber resilience of the system under different scenarios. Utilizing ADROC for other purposes may be possible, but it could take extensive additional work to develop new metrics, processing functions, or even data collection tools.

5.2. Experiment Brainstorm

It is very important to have a clear plan in mind before beginning to implement and run experiments with ADROC. If you take time to plan your experiments, you are much more likely to run an informative analysis. RevRun, which ADROC uses to manage the emulation phase of experimentation, has a set of experiment brainstorming questions available with its documentation. These questions were written with emulation and data processing in mind, but many could be applied to the math modeling phase of ADROC as well. The remainder of Section 5.2 will walk through experiment brainstorm questions which are inspired by those found in the RevRun documentation. The goal of these questions is not necessarily to limit your thinking, but to inspire discussion that will drive experiment design.

1. Set Goals for the Analysis

   a. What is the overarching goal of this analysis? Is this goal based around understanding threats, vulnerabilities, mitigation strategies, system architecture, or something else?

   b. What specific questions do we want to answer with this analysis? What information do we hope to gain?

   c. What information do we need from these experiments to achieve the analysis goal and answer the questions asked?

2. Fully Understand the System Being Modeled
   a. What are the key components of this system? Which MUST be modeled in order for the analysis to be meaningful? Which can be abstracted or simplified? Can any components be grouped into classes (e.g., Windows vs Linux machines) without loss of generality?

   b. What does the typical operation of the system look like? What features/settings need to be or can be configured or modified (e.g., firewall rules set by a config file that is injected into the experiment)? Are there any anticipated timing constraints or startup artifacts, particularly for the emulated model of the system?

   c. What are the key features of the system's network? What kind of communication typically occurs? How are different components networked together?

   d. What is the mission of the system? Typically, for industrial control systems, this is based on the physical processes of the system. What are these and what do they represent? How will they be modeled for analysis?

3. Describe the Scenarios of Interest

   a. What information is needed to address the goals of this analysis? For example, if the goal is threat-focused, are we comparing many different threats or are we comparing different implementations of the same threat?

   b. In general, what kinds of threats are we concerned about on this system? Cybersecurity and resilience do not exist without the presence of threats, and regardless of the goals of the analysis, some kind of threat will need to be considered, either as the focus of the scenarios or as a test for other scenario components.

   c. What aspects of the described scenarios are actually important to model? This depends largely on the questions being asked and what the system models can capture. For instance, if you only care about the final impact of a threat, it may not be necessary to model the full multistage attack. Limiting scenarios to only the necessary components can save you development time and experimentation time, but you should be careful not to limit your options in the future if you want to dig into further details about the experiment.

   d. What is the "baseline" state against which all other scenarios will be compared?

4. Determine How to Proceed

a. Consider the resources you have available to run emulation experiments. Approximately how many emulations can you run? Do you have enough resources to fully emulate all scenarios of interest, or do you need to leverage the math modeling phase of ADROC?

b. If you will be using the math modeling phase, what type of math models will be most informative for your analysis? What information will you gain from the math models that will help to eliminate some of the scenarios of interest? Are there any math models that have already been developed for your system that you could use?

c. Once you are moving toward the emulation phase of ADROC, it is recommended that you review the experiment design questions in the RevRun documentation. While several are related to the questions asked here, certain questions are more emulation-specific and will help guide decisions around RevRun configuration.

Remember that the experimentation process will not often be linear. First, the design of the math models and the design of the emulation models may inform each other in an iterative process which improves both models and increases how predictive the analysis results will be to a real system. Second, you may reach the end of your planned analysis and decide that the results do not answer the questions you were asking. At this point, the next step is to return to the experiment design questions, decide what needs to be changed (e.g., a different scenario to test, new data to collect, modified processing or metrics configurations), and rerun stages of the ADROC analysis.

5.3. Math Model Design

The math modeling phase of ADROC can be fulfilled by many different types of math models. The objective is to use a model which is computationally inexpensive to run many times for evaluating many proposed scenarios. So long as a model fits these criteria and gives informative results that can be used to narrow down the scenarios of interest, then it is a viable option for the ADROC math modeling phase. But the selection and parameterization of this model is up to analyst craft.

Development work on ADROC has involved two different math models: Markov Decision Process models [14] and Reinforcement Learning on Belief-Markov Decision Process models [26]. Details on general approaches to designing these models can be seen here.

5.3.1. Markov Decision Process (MDP)

As with the development of most types of models, development of MDPs is both an art and a science. The "artistic" element refers to the development of the model components. The model developer needs to consider the attack being modeled and how the various stages of the attack can be represented as states in the MDP. This process must also consider how the attacker interacts with the system itself.

As previously noted, specification of the transition probabilities poses a significant challenge. For attacks that have been observed and analyzed, transition probabilities can be estimated using the observations. However, for attacks that have not yet been observed, one may have to rely upon SME judgment or experiments conducted in controlled testbeds. Probabilities constructed from any of these data sources are likely imprecise and fraught with uncertainties.

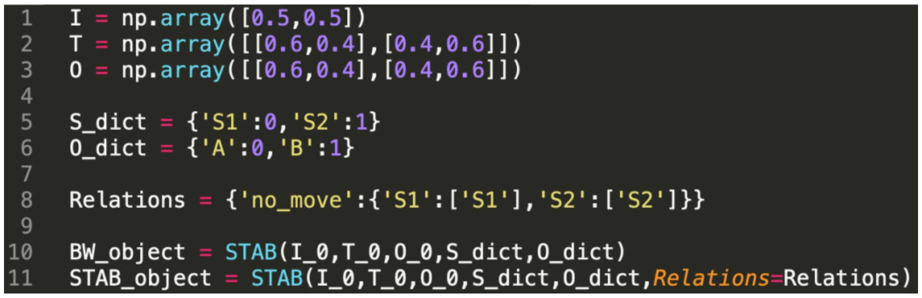
The scientific aspect of model development refers to the numerical solution of MDPs. This process has been well-studied, and a number of numerical methods are readily available (i.e., Chades et al.

[30] [31], Cordwell et al. [32], sachinbiradar [33]). Once the analyst provides the MDP parameters in the appropriate format, the numerical methods provide results effectively and efficiently.

5.3.2. *Model Reinforcement Learning on POMDP*

As discussed in Sections 2.2.1 and 2.2.2, reinforcement model learning on POMDP's can also be an effective solution to the mathematical modeling phase of ADROC. Under ADROC, a novel algorithm, STAB, was developed to make this application more efficient for potentially large cyber systems. This section describes how to use the STAB algorithm implemented in Python, and how to transform a DQN agent into a σ-Safe DQN agent in the stable baselines 3 RL package [34] for Python.

A STAB object is designed to take episodes of observations of a HMM and improve a guess of the HMM, which is the triplet (I,T,O) initial distribution, transition rate matrix, and observation rate matrix. To construct a STAB object, you need the components of the HMM guess as well as Python dictionaries for the state space and the observation space. The keys of these dictionaries are the identifying names of the states and observations, and the values are an ordering of these keys by whole number, see Figure 6 (lines 5 and 6).

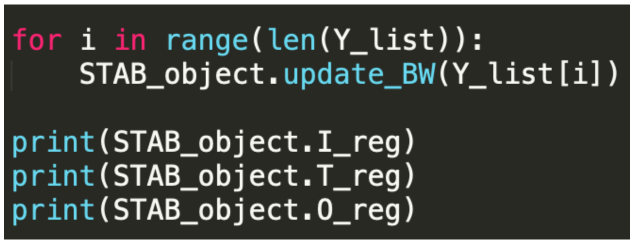


```
1   I = np.array([0.5,0.5])
2   T = np.array([[0.6,0.4],[0.4,0.6]])
3   O = np.array([[0.6,0.4],[0.4,0.6]])
4
5   S_dict = {'S1':0,'S2':1}
6   O_dict = {'A':0,'B':1}
7
8   Relations = {'no_move':{'S1':['S1'],'S2':['S2']}}
9
10  BW_object = STAB(I_0,T_0,O_0,S_dict,O_dict)
11  STAB_object = STAB(I_0,T_0,O_0,S_dict,O_dict,Relations=Relations)
```

Figure 6. Constructing a STAB object from STAB.py

With these parameters alone, the STAB object will only give you the Baum-Welch algorithm, unless you also give it the optional state relations. The state relations are a dictionary from each relation's name to their map, and each map is a dictionary from each node to the subset of nodes that relation maps to (Figure 6 line 8).

After the STAB object has been constructed, it can take a collection ("Y_list" in Figure 7), of sequences of observations in its "update_BW" method. Then the object variables *I_reg* , *T_reg* , *O_reg* are the updated HMM with regularization to protect against states with sparsely inferred visits, see Figure 7. After receiving this updated HMM it is advisable to use it as a next guess on the same data (or Y_list) multiple times. If the data collected represents the HMM well then updating the HMM with the STAB algorithm will converge to a strictly better estimate of the model than one started with.


```python
for i in range(len(Y_list)):
    STAB_object.update_BW(Y_list[i])

print(STAB_object.I_reg)
print(STAB_object.T_reg)
print(STAB_object.O_reg)
```

Figure 7. Run Baum-Welch and Inspect the STAB Objects

The σ-Safe DQN is easy if you first install the stable_baselines3 (Revision 29f6687b) [34] Python package. Go to the file located at "stable_baselines3.dqn.dqn", in the "DQN" class in the "train" method, there is a line "next_q_values, _ = next_q_values.max(dim=1)" which needs to be updated to include the diffusion term and the σ parameter, according to the equation

$$q(X_t,A_t) \leftarrow q(X_t,A_t) + \alpha\left[R(X_t,A_t) + \gamma\left((1-\sigma)\max_{a\in\mathcal{A}} q(X_{t+1},a) + \frac{\sigma}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} q(X_{t+1},a)\right) - q(X_t,A_t)\right].$$

Integrating the HMM learning and the RL on the corresponding Belief-MDP is a matter of generating the gym environment from the learned Markov model, (I,T,O), and creating a wrapper to feed in information about the original POMDP so that the RL agent can continue to learn the HMM and the control response to the Belief-MDP. This integration would require a wrapper that takes a POMDP gym environment and the learned Markov model to create a Belief-MDP. This wrapper would not be difficult to write, however it was not written for this LDRD, partly because the interaction between the reward function and the learned HMM in an environment is somewhat situation dependent. Like the design of the MDP models discussed in Section 5.3.1, the actual parameterization of the POMDP is an art.

5.4. Emulation Design

Emulation design includes three sub-tasks: system design, threat design, and experiment control. System design includes everything that is going to be either simulated or emulated. System requirements determine which features are included and which can be excluded. Threat design includes the type of attacks that are going to be included in the experiment. Included threats may come from dedicated threat modeling or from subject matter experts. Experiment control includes the flow of the experiment and the actions that must occur either conditionally or chronologically. Experimental control takes requirements formulated in the system and threat design steps and executes requirements that are dynamic within the flow of the experiment.

5.4.1. System Design

At a very basic level, a system is a set of components that work together in an interconnecting network. From an ADROC perspective, a system is the set of components relevant to the research objective. Identifying what components are relevant to the objective can be approached using a protected volume method. It is important to iteratively do a component analysis to ensure that all relevant dependencies are captured. Components may include non-physical and physical objects. Examples of non-physical components are software, operation guides/procedures, and penalties. Physical objects relevant to ICSs may include structures, programmable logic controllers, physical states of matter, and sensors. Interactions between physical and non-physical objects are important to capture as small changes in a systems state trajectory can lead to vastly different results.

It is helpful to use a graded approach when determining the fidelity at which a system should be modelled. Components that are highly relevant to the experiment should be high fidelity. Supporting components with dependencies to highly relevant components can be modelled at medium fidelity. Components that need to perform a narrowly defined function can be modelled at low fidelity. For example, if a vulnerability in software is being investigated, the exact implementation of the software is needed. A router used to transport data generated by the software may be able to use a medium fidelity implementation. In this case, the router should be representative of a real system (i.e., include firewall rules, routing protocols), but it is not the primary objective of the experiment. Finally, a low fidelity component could be a small piece of software the communicates with the primary piece of software being investigated. This software *simulates* a specific functionality of, potentially, a more complex piece of software. Its primary objective is to provide dynamic feedback to the software under test but does have any additional functionality beyond its prime objective.

Other considerations for system design important to ADROC include the ability to fully automate the system startup process, the ability to vary the system parameters that you want to study with your analysis, and the ability to access and extra the data that you need to answer your analysis questions.

5.4.2. Threat Design

The threat design can manifest in several different implementations but overall, it is the disturbance to the system which leads to a consequence. Based on its attack surface, a system may have hundreds or thousands of threats. Thus, for ADROC, it is important to design the threats that support the many objectives of the research and leverage the functionality of the high-fidelity components in the system. It can be difficult to determine which single, specific threat to use in a given scenario, as it is hard to know a priori how the threat will impact the system and what implementation of the threat would be most informative. Thus, building a test matrix that includes a spectrum of best guesses is an approach in which RevRun excels. By explicitly defining the threats, they can be included in the experimental configuration/control. RevRun will be able to run through the threat matrix and collect the data, which leads to a more broadly based analysis. Note that cybersecurity does not exist without the presence of threats, so whether the intent of the analysis is to understand the threats themselves or to test efficacy of a mitigation tool, some threat should be implemented in the model.

An excellent framework that describes different types of cyber-ICS threats is the MITRE ICS ATT&CK framework [35].

5.4.3. Experiment Control (RevRun)

RevRun is a tool for emulation experiment control, data collection and processing, and cyber resilience metric calculation. ADROC leverages RevRun to manage Phase 2 of the ADROC workflow.

Different versions of RevRun exist to manage experiments in SCEPTRE and minimega. Consider which platform you are using, and whether you need RevRun to perform the experiment management at all. The data processing and metrics calculation stages of RevRun are typically applied to data collected by RevRun, but this is not required. If you have pre-existing system data, or data collected using a different emulation platform or experiment orchestration tool, then the data processing and metrics functions from RevRun can still be used to quantify the impact of the emulated scenarios and to produce the metrics which help to prioritize threats.

Refer to the RevRun documentation for additional information on configuring and using this tool.

5.4.3.1. RevRun for Model Verification

Although the primary goal of RevRun is not to provide a means to verify your emulation, something should be said here on the subject.

In the ideal setting, all aspects of the emulation model (including the scenario-specific components, such as threat and mitigation implementation) should be tested in an emulation environment similar to the one that will be used for ADROC prior to the experiments being run in RevRun. The goal of RevRun is not to try to detect problems in the scenario tools or system emulation.

That said, RevRun does provide something that can be difficult to obtain from typical system emulations: data. In almost every analysis performed by RevRun thus far, an initial analysis revealed some issue or ill-formed assumption in the emulated model, to include physical process initialization, attack impacts, and automated tool startup. Therefore, it is important to carefully review data with the system owners and/or SME's, particularly during the first few runs of the analysis. This gives people who know the system well a chance to recognize potential improvements or bugs based on the data that RevRun is collecting. Once these concerns are addressed, you can re-run the RevRun analysis.

It is very important to ensure that the results of your analysis can be attributed directly to stimulus intentionally applied in the course of experimentation, and not to bugs or limitations in the experimentation environment or system implementation.

5.5. Running ADROC

The ADROC workflow has been previously described and illustrated in this paper. It is important to note that this workflow is not completely automated, and it requires manual intervention to parameterize and configure models and tools and to run an analysis. Here, we'll give the general steps to perform the analysis, assuming that you have gone through the process of designing your analysis and selecting scenarios.

1. Select and parameterize your math model, based upon the system and scenarios of interest to the analysis.
2. Run your math model. The exact nature of this step will depend on the math model type you have selected and implemented.
3. Interpret the output of your math models. Visualizations can be very helpful here. Use this information to down-select to only the number of scenarios your resources are able to support in emulation.
4. Create your emulation model. You may be able to model the system ahead of time based only on the system definition and the known goals of the analysis. However, the exact scenario configurations will need to be parameterized at this stage with the knowledge of the scenarios selected in Step 3.
5. Configure RevRun to run a series of experiment to execute each of your defined scenarios. Be sure to configure the data processing and metrics calculation stages as desired for

RevRun. Your choice of functions and parameters here determine how the cyber resilience of the system is quantified, although they can be adjusted later if desired.

6. Run RevRun. Ideally, most if not all aspects of the experiments can be automated (including scenario parameterization and data collection), and you will be able to just let RevRun run for a while. Otherwise, you may need to manually interface with the RevRun-managed experiments from time to time.
7. RevRun extracts a lot of data in order to calculate cyber resilience metrics. One great use of this data is the ability to more deeply inspect what happened in the system as the scenario was running. Since the data is stored in Elasticsearch by default, a straightforward approach is to create visualizations in Kibana [36] to help with data interpretation.

6. USE CASE 1: NUCLEAR POWER SYSTEM THREAT

The ADROC toolset was applied to a use case involving a hypothetical cyberattack on a nuclear power plant (NPP) facility. A detailed overview of the experiment and results for the nuclear power plant use case can be found in Thorpe et al. [24]. We briefly summarize the use case and results here.

The use case considers a NPP with a pressurized water reactor (PWR). The system included:

- Corporate Network: This portion of the network is a typical information technology (IT) network within a company and is connected to the external internet.
- ICS: The ICS is comprised of a SCADA network that monitors and controls a pressurized water reactor (PWR), including a reactor coolant pump and steam valve. Field devices such as programmable logic controllers and remote terminal units are included.
- Demilitarized Zone (DMZ) and Firewall: The DMZ and firewall separate the corporate network from the ICS and are designed to limit potentially malicious traffic from reaching the ICS.

The use case further assumes that intelligence reporting indicates adversaries are interested in targeting the NPP and forcing unsafe operating conditions for the PWR. The assumption is that the adversaries would initially gain a foothold on a machine in the corporate network (e.g., through a phishing campaign), attempt to pivot from one machine to another, before landing on the IT administrator's machine, harvesting the necessary credentials, and then passing through the firewall into the ICS. Once in the ICS, the attacker establishes a presence on an engineering workstation and delivers a malicious payload to PLCs controlling the reactor coolant pump and/or the steam generator valve. The goal of analysis with ADROC, therefore, is to analyze several variations of this attack to determine how they impact the system and which variations impact the system most.

Given that capabilities of the attacker are not fully known, we used a set of MDP models to analyze which type of attack (defined as a combination of capabilities) would be of greatest concern. Table lists the attack feature variations that were considered. Our analysis considered all possible combinations, i.e., 24 different attacks.

Table 1. Attack Features

Attack Dimension	Options	Description
Directness	min path	Attacker had insider knowledge, so it pivots to admin machine immediately after initial infection
	full path	No insider knowledge: must perform network discovery & possibly infect multiple machines before getting to admin machine
Malware Memory	Yes	Does not reinfect machines
	No	May reinfect machines

Attack Dimension	Options	Description
Operating System	Designed to operate on Linux & Windows	Has tools for both operating systems
	Designed for Windows Only	Tools only work on Windows OS machines
Target PLC	Reactor Coolant Pump	Directly affects temperature
	Steam Generator	Directly affects pressure
	Both	Combined effects

The associated Markovian models were set up as follows:

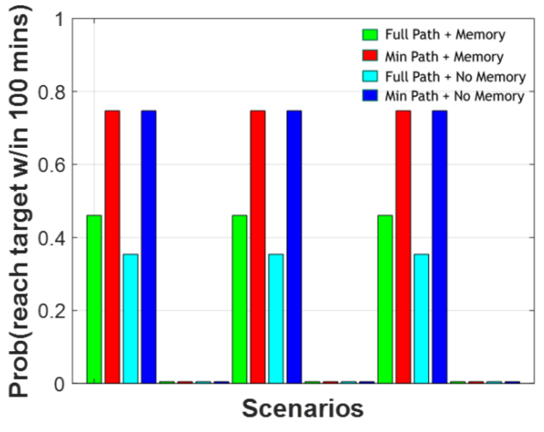
- States are defined to indicate what type of machine the attacker has most recently pivoted to (Windows, Linux, Administrator's, or Engineering Workstation). If the malware is assumed to have memory, the states also encode which machines the attacker previously pivoted to. A "Failure" state was included to represent when the attacker had either run out of options or given up.
- Transition probabilities are determined according to
    - Time: from our emulation testbed, we empirically observed that the CALDERA threat emulator spent approximately 5 minutes analyzing Linux machines before pivoting and approximately 6 minutes analyzing Windows machines before pivoting.
    - Current machine: we assumed that if the attacker was currently on a Windows machine, it was more likely to pivot to a different Windows machine next, and similarly if the attacker was currently on a Linux machine, it would most likely pivot to another Linux machine.
    - Operating System: if the malware was not designed to operate on Linux platforms, then the attacker was assumed to fail if it pivoted to a Linux machine.
- Actions (and rewards) were not defined at this stage of the project, so we actually only used Markov chain models instead of MDPs.

We evaluated the attacks according to:

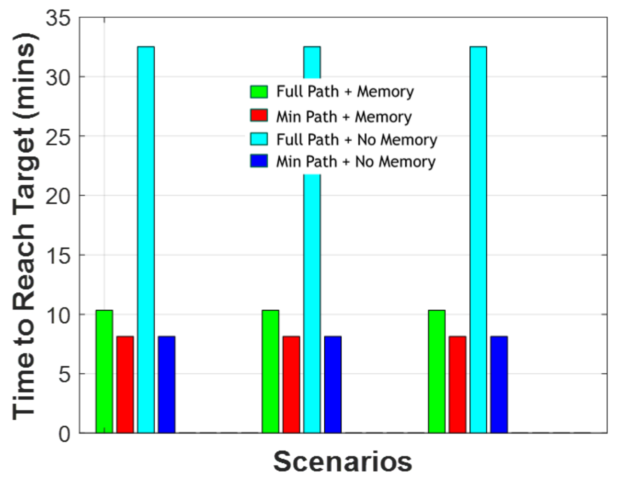
- Success: the probability that the attacker reached the engineering workstation.
- Time: the time required for an attacker to reach the engineering workstation, if the attacker reached it.

Figure 8 provides summary results from the MDP runs. The scenarios with no attacker success correspond to the attacks with which the attacker only had capabilities for Windows machines and could not successfully attack Linux-based machines. Overall, the attacks that had the highest probability of reaching the engineering workstation target were the attacks that had insider information and took the minimum path by pivoting directly to the Administrator's machine. For these two scenarios, probability and speed of success were not significantly affected by the presence or lack of "attacker memory".

The MDP models provide a means for ranking the attacks considered in this scenario. Furthermore, they were computationally efficient (less than 1 second to run) and fairly easy to set up (about 2 days total were spent building the models). However, these models cannot be used to estimate the potential consequences of an attack, so the ICS and threat emulation platforms were used to provide that additional information.



(a)



(b)

Figure 8. Results from MDP models for nuclear power plant use case: probability of attacker success (a) and average time to target (b)

Once the scenarios of highest concern were identified, they were modeled using CALDERA for threat emulation and an emulated model built around the Asherah model [37] of a PWR facility. Asherah is a physics model of a generation II PWR. The control systems that are responsible for steam generation and a coolant flow rate were emulated. These control systems are found as part of the steam generator and reactor coolant pump respectively. Maintaining temperature and pressure in the system is critical for safe operation of the reactor. Loss of flow or heat rejection to main portions of the system may lead to nuclear core damage and possibly radiological release if left unmitigated.

Nuclear reactors have numerous safety systems to maintain safe operating conditions. In the Asherah model, a reactor SCRAM (rapid emergency shutdown) would be triggered if pressure in the PWR exceeded 8.97 MPa or the reactor core temperature exceeded 580 Kelvin. A departure from nucleate boiling ratio[4] below 1.3 would also have negative effects on the PWR. Cyberattacks that could lead to reactor SCRAM are attacks of very high consequence. If executed on a production system, the reactor may stay offline for months or never come back online due to the nuclear industries current cyber security posture. The economic and safety implications could be severe.

Using the tools developed in the ADROC project several attack scenarios that lead to SCRAM were successfully identified. The Asherah physics model is not a high-fidelity representation of an existing reactor, so these findings do not put currently operating reactors a risk. However, whether a version of these attacks could be executed on production systems is still an open question. It is possible that additional safety controls and physics not represented in Asherah would invalidate these results.

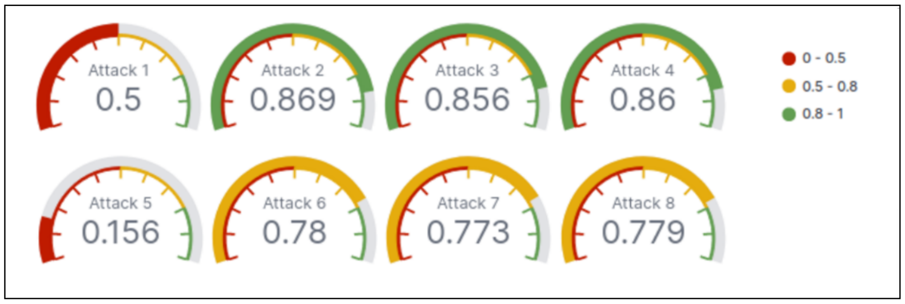


Figure 9. Cyber resilience scores for Use Case 1.

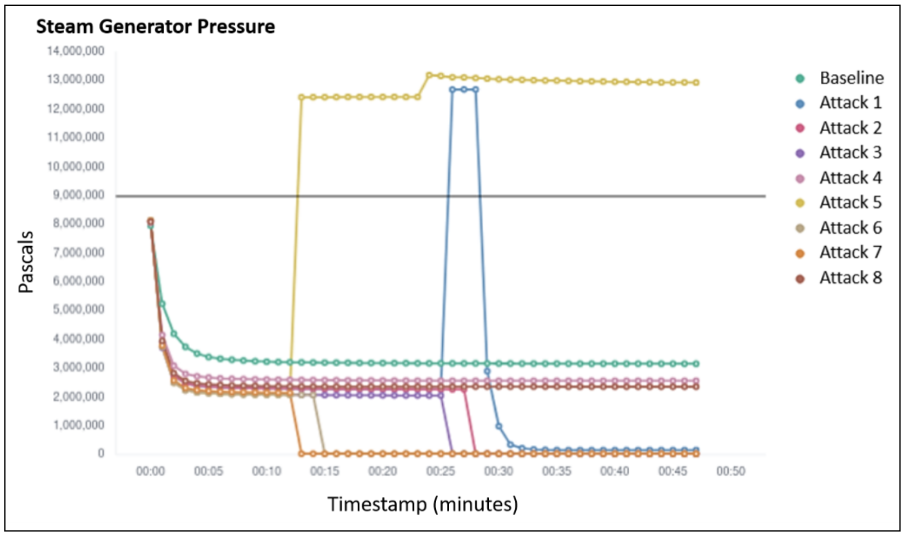


Figure 10. System pressure data for Use Case 1. Note Attacks 1 and 5 cross the safety threshold.

More details about the emulation experiments and the results of this analysis are found in Thorpe et al. [24].

[4] "The ratio of the heat flux needed to cause departure from nucleate boiling to the actual local heat flux of a fuel rod" [45]

7. USE CASE 2: ELECTRIC POWER GRID THREAT

One of the major goals of ADROC is to ensure that the tool is relatively flexible to a variety of systems, threats, and analysis goals. After Use Case 1, we theorized that the emulation phase of ADROC was fairly modular, such that the ICS system model and CALDERA payload script could be "swapped out" to perform an analysis similar to Use Case 1 for for a different system.

For this effort, we selected the Cyber Flag 19 (CF19) topology, which is a model of a power system and fuel distribution station. The threat of concern was the Crash Override [1] malware, which is used to send malicious DNP3 packets to disrupt the power system, and thereby also impact the fueling station. Besides these two points, the emulation experiments were designed very similarly to those described in Thorpe, et al. [24].

We validated several important technical aspects on this topology. We were able to

1) Migrate the ADROC corporate network from SCEPTRE's Phenix V5 to Phenix V6
2) Fully automate the startup and execution of CALDERA
3) Execute a different attack on a completely different physical model using a completely different ICS protocol.

Some of these results are implementation-specific, but they allowed us to demonstrate the modularity of the ADROC approach.

This page left blank.

8. USE CASE 3: ELECTRIC POWER GRID MITIGATION

Thus far, the motivating problem, in-text examples, and discussed use cases for ADROC have all been focused on analyzing and prioritizing threats. However, we also wanted to illustrate how ADROC could be used for other types of analyses. Therefore, ADROC partnered with another LDRD project, HARMONIE [38] [39] to apply the emulation phase of the ADROC platform, primarily RevRun, to analyzing the efficacy of a mitigation tool that HARMONIE was developing called CARV [40] [41].

At a high level, the goal of HARMONIE was to address unpredictable disruptions (e.g., cyberattacks, extreme weather) in the power grid. While protection schemes currently exist to help the grid withstand predictable, known disruptions, traditional schemes cannot defend the grid against more unpredictable events. Therefore, the HARMONIE project aimed to develop a mitigation tool which would sit on relays in the grid and help defend against both predictable and unpredictable disturbances. A major thrust of this work was an algorithm called Consensus Algorithm Relay voting, or CARV. CARV is intended to enable consensus voting among a connected group of relays in the power grid. This voting allows the relays to work together to determine system protection actions and to reach consensus on the values of key variables in the system. Ideally, CARV would improve the resilience of the power grid to unpredictable disruptions, particularly cyberattacks. Attacks which compromise one relay in a system would have less chance of impacting the grid because the other relays could still communicate and come to consensus about actions to take.

The goal of ADROC's analysis with HARMONIE was to evaluate the efficacy of CARV within a 9-bus electric power system against a Denial of Service (DoS) attack plus some other disruption to system power. When CARV is running on several relays within a system, they can chat amongst themselves to vote and come to consensus on the state of the system and on whether to trigger a protection scheme. Although this communication could be very beneficial to the system's resilience to spoofing and man-in-the-middle attacks, or even physical failure on a relay, it does incur additional burden to the system during nominal operating conditions. The HARMONIE team wanted to use ADROC as a mechanism to quantify 1) the increase in system resilience due to CARV, and 2) the burden CARV places on the network, effectively performing an initial cost-benefit analysis of the tool.

Unfortunately, we were not able to complete the analysis with HARMONIE due to time and budget constraints. However, ADROC was able to go through the full exercise of designing the analysis, configuring the ADROC tools, and implementing the planned scenarios (details in sections below). Accomplishments and lessons learned from the exercise are captured here.

8.1. Analysis Approach

This section describes the main thrusts of our approach when designing the analysis of HARMONIE's CARV tool.

8.1.1. Design

The proposed CARV analysis was broken down into two parts, Analysis A and Analysis B. The Analysis A matrix is seen in Table 9. For this analysis, the goal was to study the effectiveness of different configurations of CARV against a notional Denial of Service (DOS) attack which was developed by the HARMONIE team, and to better understand the impact CARV had on the system when it was running nominally. Scenario 1 was the baseline against which all other scenarios in the analysis would have been compared, where neither CARV nor the DOS attack are run. The different configurations of CARV represented different sets of timing settings, including the time given for the CARV-enabled relays to vote and to reach consensus.

Table 2. Analysis A scenario matrix. Investigating the effectiveness of different configurations of CARV.

	No CARV	CARV Config 1 (Default)	CARV Config 2	CARV Config 3
No Attack	1	2	3	4
DOS Config 1	5	6	7	8

The scenario matrix for Analysis B can be seen in Table 10. Here, the goal was to verify the robustness of one configuration of CARV against several variations of the DOS attack. When we only use one implementation of a specific attack to verify, test, and tune a mitigation tool, we run the risk of configuring the tool too narrowly such that it does not work well in general against a wider variety of attacks. Therefore, it is beneficial to verify that a configuration is robust to variations of the same attack or to different attacks. Again, for Analysis B, the baseline was a scenario where neither CARV nor the attack are run. The scenarios where CARV was running would use the best configuration of CARV identified in Analysis A, keeping in mind that these configurations are not necessarily optimal, but were the best in a finite group of configurations tested.

Table 3. Analysis B scenario matrix. Investigating the robustness of the "best" CARV config from Analysis A.

	No CARV	Best CARV Config
No Attack	1	2
DOS Config 1	3	4
DOS Config 2	5	6
DOS Config 3	7	8

8.1.2. Data Collection

The analysis goals for HARMONIE centered around physical process data from their system simulation and on quantity of network traffic, particularly traffic generated as part of the CARV tool.

For physical process data, the RTDS output three-phase voltage and current for each of nine buses as well as statuses for a variety of circuit breakers in the system. The HARMONIE team's main concern was with current in the system, although certain situations could also cause low-voltage conditions. Thus, we decided to start by casting a wide net, visualizing the data we got, and then circling back with the HARMONIE team to determine what was important.

For network data, because the crux of the analysis was the impact of CARV, the network traffic of interest was traffic quantity coming from relays involved in the CARV voting scheme. In theory, CARV would create a lot of chatter in the network as relays would vote and come to consensus.

In addition, to help track the progress of the attack, traffic sent by the designated attacker machine was captured.

8.1.3. Metrics and Visualizations

Most of the metrics were calculating the deviation in the scenarios from the baseline for each of the collected datastreams. The only other major metric was concerning the time of the first DOS packet from the attacker, where an earlier time would constitute a lower resilience score.

For the initial runs of the analysis, the metrics would have been slightly more weighted toward changes in the physical process data, since minor changes in network data do not necessarily indicate major problems for the system, or even sub-optimal conditions. However, the plan was to meet with the HARMONIE team again after initial data collection to get better direction on metric parameters.

Several visualizations would have been designed to:

- Compare network traffic with and without CARV in no-attack scenarios. This would have helped to address one of the analysis goals of understanding how CARV impacts the system under nominal conditions
- Compare network traffic in scenarios with and without attack. This would have illustrated the impact of the DOS attack on network traffic compared to the impact of CARV alone on network traffic
- Compare key physical process data during attack scenarios with and without CARV. This would have shown the efficacy of CARV at mitigating the impacts of the attack, which was another analysis goal

These metrics and visualizations were intended to help verify the efficacy of CARV and to perform a cost-benefit-like analysis of the tool.

8.2. Accomplishments

Although the ADROC team was not able to fully complete the planned HARMONIE analysis, there were still several key accomplishments made through the process of preparing for this use case.

1. The ADROC team was able to go through the exercise of designing a mitigation-focused analysis for RevRun. We worked with the HARMONIE team to understand the system, identify analysis goals, and select important data streams. This exercise helped verify that ADROC and RevRun can be used for analyses that are not purely threat focused.
2. We went through the process of integrating the ADROC emulation topology and toolset with another team's emulation topology. Through all three use cases, we showed that ADROC is flexible enough to integrate with a variety of system models.
3. We were able to update RevRun to integrate with the latest version of SCEPTRE's phenix.
4. This was the first use case for RevRun involving hardware in the loop (HITL). Although we've theorized that RevRun would work on a system with integrated HITL, the HARMONIE use case allowed us to verify this for at least one example.

5. Although RevRun was designed to run in an automated fashion, these mechanisms don't always work out of the box. In addition, system models may be designed to require manual intervention, especially if the model was not developed with repeatable experimentation in mind. The latter certainly applied to the HARMONIE model, but we were able to modify RevRun to maximize the ability for automated experimentation.
6. Although RevRun is not intended to be used for model verification, we were able to illustrate once again the utility of data collection via RevRun as a mechanism for verifying system models are working as expected. Through initial integration testing, data collected by RevRun showed that HARMONIE tools were not running as expected within the emulated environment.

8.3. Lessons Learned

The ADROC team has recorded several lessons learned from the experience of preparing for the HARMONIE analysis.

1. Accomplishment 1 (Section 8.2) noted that the ADROC team was able to design a mitigation-focused analysis plan for the first time. Instead of comparing threat variations, we have here an opportunity to perform a cost-benefit analysis and to check its robustness to attack variants. This can be challenging with a limited number of experiments that are manually configured. This process required a different way of thinking about RevRun experiment design and considering a different set of questions. For example:
    a. What data do we need in order to show that the mitigation tools worked as intended?
    b. What data do we need to show side-effects to the system due to these mitigation tools (e.g., quantity of network traffic)?
    c. What is the baseline scenario? Do we need multiple baselines? If so, how do we reconcile resulting scores from RevRun?
    d. Because there are potential variations in both the mitigation and test attack, the number of experiments to run can blow up quite quickly. The most important vectors for variation must be chosen carefully.
    e. Since the threat is no longer the focus of the analysis, how do you choose a good example "test" attack which will demonstrate the value of the mitigation tool?
2. RevRun was initially developed with the assumption that the user has root privileges by default on the server where the emulation will be running. If instead the user would need to use `sudo` in order to escalate privileges, this significantly disrupts RevRun's ability to manage emulation experiments in an automated way. The process of adapting RevRun to run on the HARMONIE team's dedicated server highlighted this inflexibility.
3. Accomplishment 4 (Section 8.2) noted that RevRun was able to run using a topology with HITL. Although this process did not affect RevRun's ability to collect data (for this use case), it did affect RevRun's ability to run in an automated way. Furthermore, this exercise highlighted the fact that even if RevRun were extended to support parallel experimentation in the future, not all system topologies, particularly with HITL, would support parallelism.
    a. The HARMONIE network relied on a Real Time Digital Simulator (RTDS) connected to the emulation via HITL to simulate the physical processes of the

system. The RTDS could only support a single outbound connection to an experiment, so parallel experiments would not have been possible.

   b. Restarting the simulation on the RTDS required manual intervention via remote desktop connection. This needed to be done between each experiment in order to maintain system state consistency.

   c. If the RTDS got into an unexpected state, it was necessary to reboot the simulator. This required manually hitting a power button on the machine itself.

   d. To connect the RTDS simulation into the emulated environment, a network tap needed to be created. For the purposes of RevRun analysis, this step could be scripted, but it still constituted an additional step for RevRun.

4. The rate of data flow in the system could be an issue for RevRun. RevRun is intended to be fairly passive in the emulation experiment, and to be able to connect into the emulation without significantly affecting the model. One of the outcomes of this goal is that the RevRun data collection tools, by default, listen to and record every update broadcast by the physical process provider for the configured fields. In the case of HARMONIE, this was 30 updates per second for up to approximately 65 fields. This could create a situation where RevRun needs to manage an unwieldy amount of data which it was not necessarily designed to handle.

5. Just as we needed to think carefully about how to model and implement threats in previous use cases, we need to put the same level of thought into how to configure and start mitigation tools (or other significant scenario components) in a scriptable or automated way.

This page left blank.

9. ADROC PAPERS, PRESENTATIONS, AND COLLABORATIONS

ADROC has shared work through a variety of papers, talks, and collaborations, captured below.

9.1. Papers/Presentations

- Presentation at Medical Device Innovation Consortium (MDIC) working group. MDIC is a working group with representatives from various areas in the medical technology community focused on the science required to improve patient access and safety of medical devices [42].
    - (Invited) E. D. Vugrin, M. A. Sahakian, J. Thorpe, "Threat Modeling and Medical Device Cybersecurity, a National Security Perspective," Center for Medical Device Cybersecurity, August 2021.
- Presentation at Sandia's Infrastructure Resilience Community of Practice (IRCOP)
    - (Invited) E. D. Vugrin, M. Livesay, and J. Thorpe, "ADROC: ADvancing Resilience Of Contol Systems", Sandia National Laboratories Infrastructure Resilience Community of Practice, September 2021.
- Presentation at Systems Engineering Research Center (SERC). SERC is a university affiliated research center which acts as a national resource for systems engineering research and expertise [43]. Presentation led to conversation with project team from FFRDC (see Section 9.2).
    - (Invited) E. D. Vugrin, "How Can We Model Cyberattacks and Systems to Characterize Resilience of Critical Infrastructure Systems?", Systems Engineering Research Center, February 2022.
- Several presentations to various project groups at Sandia to spread awareness of the ADROC capability, including:
    - The Sandia group who helped us develop Use Case 1, focused on resilient SCADA architectures for nuclear power plants.
    - A large multi-lab group focused on situational awareness for industrial control systems.
    - A Sandia project group developing a space-cyber emulation platform.
- Paper and Presentation at ACM's Workshop on Secure and Trustworthy Cyber-Pysical Systems [24].
- Paper to be submitted on Reinforcement Learning research under ADROC [26].

9.2. Technical Collaborations

- Collaboration with HARMONIE (see Section 8 above) during FY22. HARMONIE was an LDRD under the Resilient Energy Systems Mission Campaign FY21-FY22, led by Shamina Hossain-McKenzie.
- Sandia cyber forensics training team are considering using tools, data, and Use Case 1 model from ADROC in a future training course.

- ADROC-developed corporate network emulation and CALDERA integration into SCEPTRE currently being leveraged by another Sandia project modeling a bio-manufacturing facility.
- Project group at an external Federally Funded Research and Development Center (FFRDC) reached out about using ADROC components for inclusion in a critical infrastructure resilience project.
- The Designing for Resilience through Emulation project (DRE) has been funded for FY23-FY25 through Sandia's LDRD program (Resilient Energy Systems Mission Campaign), and it will expand on the work that ADROC has done to create a usable cyber resilience modeling platform.

9.3. Documented Intellectual Property

- RevRun TA – SD#16138 (Submitted 4/18/2022)
- ADROC TA – SD#16127 (Submitted 4/27/2022)
- RevRun government use copyright in-progress

10. FUTURE WORK AND CONCLUSIONS

There are several areas for future work left after development of the ADROC platform.

First, the ADROC workflow itself could be further developed. Future work could study the ADROC workflow as a whole to look for additional opportunities for automation or helper tool development to ease the process. In addition, we did not have the time and budget to fully integrate and demonstrate the reinforcement learning modeling approach in the ADROC workflow. Some effort in the future would be required to see how well they integrate, but the biggest challenge is likely in the configuration of the POMDP to adequately capture the key characteristics of the cyber system.

There are a few areas of future work on the STAB algorithm. First, STAB currently only applies aggregation on the transition rates, but the same concept applies for the observations as well. Such an extension could be valuable when dealing with high rates of false positives and negatives. Second, STAB should be updated to regularize the α, β terms so that they don't vanish exponentially over time (see Eq. 92b in Rabiner [20]).

There are two other generalizations of Baum-Welch that would be non-trivial but valuable. One is to handle a meta-state which represents many components' states at once. This type of state would cause an exponential explosion of state space if done directly. Extending Baum-Welch in this way would allow for modeling a multiplying virus rather than one that simply transfers from component to component.

The other generalization for future work is to generalize Baum-Welch to multigraphs, which would allow for using multiple degrees of fidelity of the model in superposition during learning. This would enable much faster improvements of model learning during early training.

On the emulation side, the ADROC and HARMONIE teams are hoping for opportunities for future collaboration so that we can complete the planned analysis. Some of the lessons learned discussed in 8.3 also highlight areas of current challenges or inflexibilities that could be addressed by future work.

One of the biggest challenges still for this platform is the fact that scenarios to be analyzed must be manually configured. Not only does this take a great deal of time (especially in emulation), but it means that configuration efficacy is limited by human time and imagination. Future work planned under a new LDRD in FY23-FY25 will address this and other challenges by integrating uncertainty quantification and optimization into the emulation experimentation pipeline.

In conclusion, the ADROC project made a great deal of progress toward making efficient, quantitative cyber resilience experimentation a realistic approach for threat prioritization on ICSs. More than that, we were able to show through several demonstrations how the ADROC toolset is flexible to addressing a variety of questions for a variety of systems. We hope this toolset will be useful to others and will continue to grow in the coming years.

REFERENCES

[1] Dragos, Inc., "CRASHOVERRIDE Analysis of the Threat to Electric Grid Operations," Dragos, Inc., Hanover, MD, 2017.

[2] M. Robbins, *Cyberattack Hits Indian Nuclear Plant,* Arms Control Association, 2019.

[3] C. Cimpanu, *Port of San Diego suffers cyber-attack, second port in a week after Barcelona,* ZDNET, 2018.

[4] M. Hoffman and T. Winston, *Recommendations Following the Colonial Pipeline Cyber Attack,* Hanover, MD: Dragos, Inc., 2021.

[5] S. Kardon, *Florida Water Treatment Plant Hit with Cyber Attack,* Industrial Defender, 2021.

[6] Dragos, Inc., *CHERNOVITE's PIPEDREAM Malware Targeting Industrial Control Systems (ICS),* Hanover, MD: Dragos, Inc., 2022.

[7] J.-P. Watson, R. Guttromson, C. Silva-Monroy, R. Jeffers, K. Jones, J. Ellison, C. Rath, J. Gearhart, D. Jones, T. Corbet, C. Hanley and L. T. Walker, "Conceptual Framework for Developing Resilience Metrics for the Electricity, Oil, and Gas Sectors in the United States," Sandia National Laboratories, Albuquerque, NM, 2014.

[8] R. Jeffers, A. Wachtel, A. Zhivov, C. Thompson, A. Srivastava and P. Daniels, "Integration of Resilience Goals into Energy Master Planning Framework for Communities," *ASHRAE,* vol. 126, no. 1, 2020.

[9] N. J. K. Brown, J. L. Gearhart, D. A. Jones, L. K. Nozick, N. Romero and N. Xu, "Multi-objective optimization for bridge retrofit to address earthquake hazards," in *Winter Simulation Conference (WSC 2013)*, 2013.

[10] R. Ross, V. Pillitteri, R. Graubart, D. Bodeau and R. McQuaid, "Developing Cyber Resilient Systems: A Systems Security Engineering Approach," *Cyber Resilience Framework: NIST SP 800-160,* vol. 2, no. 1, 2021.

[11] P. M. Mell, K. A. Scarfone and S. Romanosky, "A complete guide to the common vulnerability scoring system, version 2.0," FIRST - Forum of Incident Response and Security Teams, 2007.

[12] The MITRE Corporation, *CALDERA,* Bedford, MA: The MITRE Corporation, 2021.

[13] HelpSystems, *Cobalt Strike,* Minneapolis/St. Paul MN: HelpSystems, 2022.

[14] R. Bellman, Dynamic Programming, Princeton, NJ: Princeton University Press, 1957.

[15] J. Zheng and A. Siami Namin, "Enforcing Optimal Moving Target Defense Policies," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, Milwaukee, WI, 2019.

[16] I. Giannoccaro and P. Pontrandolfo, "Inventory management in supply chains: a reinforcement learning approach," *International Journal of Production Economics,* vol. 78, no. 2, pp. 153-161, 2002.

[17] W. Usaha and J. Barria, "Reinforcement Learning for Resource Allocation in LEA Satellite Networks," *IEEE Transations on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 37, no. 3, pp. 515-527, 2007.

[18] S. K. Keneally, "A Markov Decision Process Model for the Optimal Dispatch of Military Medical Evacuation Assets," in *Theses and Dissertations*, 2014.

[19] L. P. Kaelbling, M. L. Littman and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence,* vol. 101, pp. 99-134, 1998.

[20] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *IEEE,* vol. 77, no. 2, 1989.

[21] S. Z. Li and A. Jain, Eds., "Baum-Welch Algorithm," in *Encyclopedia of Biometrics*, Boston, MA, Springer, 2009.

[22] D. Wierstra, A. Foerster, J. Peters and J. Schmidhuber, "Solving Deep Memory POMDPs with Recurrent Policy Gradients," in *International Conference on Artificial Neural Networks*, 2007.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature,* pp. 529-533, 2015.

[24] J. Thorpe, R. Fasano, A. Gonzales, A. Hahn, J. Morris, T. Ortiz, H. Reinbolt, M. Galiardi Sahakian and E. Vugrin, "A Cyber-Physical Experimentation Platform for Resilience Analysis," in *Proceedings of ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (SaT-CPS 2022)*, Virtual, 2022.

[25] Sandia National Laboratories, *SCEPTRE,* Albuquerque, NM: Sandia National Laboratories, 2016.

[26] M. Livesay, E. Vugrin and J. Thorpe, *Diffusion in POMDP Methods for Uncertainty in Infrastructural Problems,* In progress, 2022.

[27] M. Galiardi, A. Gonzales, J. Thorpe, E. Vugrin, R. Fasano and C. Lamb, "Cyber Resilience Analysis of SCADA Systems in Nuclear Power Plants," in *Proceedings of 28th Conference on Nuclear Engineering and Joint with the ASME 2020 Power Conference (ICONE 2020)*, 2020.

[28] Elasticsearch B.V., *Elasticsearch,* Elasticsearch B.V., 2022.

[29] D. Fritz and J. Floren, *minimega,* Albuquerque, NM: Sandia National Laboratories, 2013.

[30] I. Chades, G. Chapron, M.-J. Cros, F. Garcia and R. Sabbadin, "MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems," *Ecography,* vol. 37, no. 9, pp. 916-920, 2012.

[31] M.-J. Cros, *Markov Decision Processes (MDP) Toolbox,* MATLAB Central File Exchange.

[32] S. Cordwell, Y. Gonzalez and Theja, *Markov Decision Process (MDP) Toolbox for Python,* Github, 2013.

[33] sachinbiradar9, *Markov Decision Processes (python),* Github, 2007.

[34] German Aerospace Center - Institute of Robotics and Mechatronics, *Stable Baselines3,* Github, 2022.

[35] The MITRE Corporation, *ATT&CK for Industrial Control Systems,* Bedford, MA: The MITRE Corporation, 2021.

[36] Elasticsearch B.V., *Kibana,* Elasticsearch B.V., 2022.

[37] R. A. Busquim E Silva, D. A. Correa, F. R. Antunes, F. C. S. Souza, J. R. C. Piqueira and R. P. Marques, "The Asherah Nuclear Power Plant Simulator (ANS) as a Training Tool at the Brazilian Cyber Guardian Exercise," in *Proceedings of International Conference on Nuclear Security 2020*, 2020.

[38] S. Hossain-McKenzie, D. Calzada, J. Nicholas, C. Goes, A. Summers, K. Davis, H. Li, Z. Mao, T. Overbye and K. Shetye, "Adaptive, Cyber-Physical Special Protection Schemes to Defend the Electric Grid Against Predictable and Unpredictable Disturbances," in *2021 Resilience Week*, 2021.

[39] A. Summers, C. Goes, D. Calzada, N. Jacobs, S. Hossain-McKenzie and Z. Mao, "Towards Cyber-Physical Special Protection Schemes: Design and Development of a Co-Simulation Testbed Leveraging SCEPTRE," in *2022 IEEE Power and Energy Conference at Illinois (PECI)*, 2022.

[40] N. Jacobs, A. Summers, S. Hossain-McKenzie, D. Calzada, H. Li, Z. Mao, C. Goes, K. Davis and K. Shetye, "Next-Generation Relay Voting Scheme Design Leveraging Consensus Algorithms," in *2021 IEEE Power and Energy Conference at Illinois (PECI)*, 2021.

[41] N. Jacobs, S. Hossain-McKenzie and A. Summers, "Consensus Algorithm Relay Voting (CARV) for Improving Resilience in Grid Cybersecurity," *[To Be Submitted],* 2022.

[42] Medical Device Innovation Consortium, "Mission & Purpose," Medical Device Innovation Consortium, Arlington, VA, 2022.

[43] Systems Engineering Research Center, "SERC," Hoboken, NJ, 2022.

[44] Dragos, Inc., "TRISIS Malware Analysis of Safety System Targeted Malware," Dragos, Inc., Hanover, MD, 2017.

[45] United States Nuclear Regulatory Commission, "Departure from nucleate boiling ratio (DNBR)," U.S.NRC, 2021.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Technical Library	1911	sanddocs@sandia.gov

Email—External (encrypt for OUO)

Name	Company Email Address	Company Name

Hardcopy—Internal

Number of Copies	Name	Org.	Mailstop

Hardcopy—External

Number of Copies	Name	Company Name and Company Mailing Address

This page left blank.

This page left blank.