DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

SANDIA REPORT
SAND2025-12402

Printed September 2025

Sandia National Laboratories

DRE: Designing for Resilience through Emulation

Jamie E. Thorpe
Todd S. Coffey
Nicholas Jacobs
J. Adam Stephens
Bethanie Williams

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

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

ABSTRACT

Threats to cyber- and cyber-physical systems have continued to increase over the past decade. Now more than ever, it is vital that cyber-physical system stakeholders have the tools to deeply understand their systems and the threats they face. High-fidelity modeling capabilities are powerful tools to support system understanding and decision-making, but they currently lack scientifically rigorous experimentation practices and infrastructures. The purpose of the project Designing for Resilience through Emulation (DRE) was to bring together past research and existing tools into a new pipeline to improve our ability to quantitatively evaluate future cyber scenarios. DRE offers new integrated capabilities for large dataset collection, noise studies, sensitivity analysis, uncertainty quantification, and surrogate modeling using a cyber-physical system emulation environment. These new capabilities significantly improve our ability to establish the credibility of the answers derived from emulation environments, ultimately leading to better critical decision support.

ACKNOWLEDGEMENTS

The authors would like to thank the following people for their significant technical contributions over the course of the project: Hannah Stroble, Nathaniel Krakauer, Zachary Kirkeby, and Eric Binnendyk.

The authors would also like to thank Brian Adams and Michael Baca for consulting with the team during the first year of the project, and Chris Goes for his modeling support in the later years of the project.

Finally, the authors would like to acknowledge the Resilient Energy Systems Mission Campaign for funding this work. This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS AND TERMS

Acronym/Term	Definition
ANN	Artificial Neural Networks
C2	Command and control, a network communication pattern associated with cyber attackers sending commands and receiving feedback from infected hosts
DMZ	Demilitarized Zone, a common computer networking term
DOS	Denial of Service
DRE	Designing for Resilience through Emulation (the name of this project)
Emulytics™	A portmanteau of “Emulation” and “Analytics”, coined by Sandia practitioners to refer to a holistic approach to cyber emulation and analytics.
HARMONIE	LDRD and follow on capability that includes a specific Power System ICS
ICS	Industrial Control System
IDW	Inverse Distance Weighting
ILP	Integer-Linear Program
Jupyter	A Python notebook tool to emulate a scientific notebook with Python capabilities
KNN	K Nearest Neighbors
KPEC	Kansas Power and Energy Conference
KPLS	Kernel Principal Component Regression
KPLSK	KPLS with Kernel
LP	Linear Program
LS	Linear Surrogate Model
MILP	Mixed Integer-Linear Program
ML	Machine Learning
QP	Quadratic Surrogate Model
RBF	Radial Basis Function
RevRun	Resilience Verification Unit
RMTB and RMTC	Response Surface Methodology B and C
SCADA	Supervisory Control And Data Acquisition
SCEPTRE	Sandia’s cyber-physical system emulation platform
SMT	Surrogate Model Toolbox
SVM	Support Vector Machines
SVR	Support Vector Regression
UQ	Uncertainty Quantification

This page left blank

1. INTRODUCTION

Critical infrastructure today faces a relatively new risk: threat of cyberattack. Over the past decade in particular, attacks to critical infrastructure systems have been increasing [1-5]. Now more than ever, it is vital that system designers, owners, and operators have the tools they need to deeply understand their systems and the threats they face.

Emulytics™, a holistic approach that combines system emulation with analytics [6], can be a powerful tool for cyber-physical system understanding. Sandia-developed tools and methodologies such as minimega [7], SCEPTRE [8][9], and RevRun [10] have strengthened our capabilities in high fidelity, repeatable, quantifiable cyber experimentation. A recent overview of SCEPTRE [8] summarizes existing work in this area. However, work remains to develop methodologies that support truly rigorous scientific experimentation in this context. The SECURE Grand Challenge [11] made substantial progress in this area but did not result in a general tool for performing statistics-driven studies in SCEPTRE or minimega environments.

"Emulation" can mean different things to different communities of practice. For the Sandia Emulytics™ community, "emulation" refers to a method of modeling complex systems at a high level of fidelity, where system hardware is virtualized but runs real system software and communication protocols. SCEPTRE, for example, is a platform designed to emulate cyber-physical systems (typically industrial control systems) in this way. Compared to experimentation on the real system or on a physical twin, emulation is typically less costly in terms of time, money, and safety measures. In addition, that level of realism often allows analysts to study cyber factors that could not otherwise be studied using lower-fidelity modeling approaches.

Historically, emulated environments have been used for training exercises and one-off experiments. More recently, emulation environments have been used to run as many as dozens of experiments in an analysis. However, rigorous statistics-based analysis, which requires far more experiment iterations, has not typically been part of an emulation-driven workflow. These analyses may include:

- Noise Studies, where identical iterations of a model are run multiple times to help identify natural areas of variation in the model;
- Uncertainty Quantification (UQ), which considers how uncertainty in model inputs results in uncertainty in predictions;
- Optimization, where inputs to the model are sought that result in extreme (maxima or minima) values of the output;
- Machine learning applications, where many sample iterations of the model may need to be run in order to train a machine learning model for some purpose, such as producing a surrogate to predict emulation model outputs

Sandia National Laboratories has a wealth of existing experience and tooling in these areas, but it is often applied to physics simulations, which are lower fidelity but more computationally efficient than the high-fidelity control systems emulations of SCEPTRE. The primary reason for this is because emulations of this kind run in real time, making collecting large numbers of samples for the purpose of statistics-driven analysis cumbersome at best and infeasible at worst.

DRE (Designing for Resilience through Emulation) was proposed to integrate cyber emulation, resilience metrics, UQ, and optimization to support better informed resilient system design and threat mitigation. This combination of capabilities could be used to deeply investigate proposed

threats and mitigations to critical systems. Bringing these capabilities together is a significant advancement to Sandia's Emulytics™ toolset. Analysts can now use DRE to more rigorously study system models, better understand model sensitivity to various inputs, test proposed model changes, quantify confidence in experiment metrics, and more.

This report will describe the research of the DRE project spanning three years and two demonstration use cases. Section 2 describes the team's approach to the problem of statistics-driven emulation experimentation. Section 3 summarizes research from FY23 into uncertainty quantification. Section 4 describes research from FY24 and FY25 into surrogate modeling. Section 5 overviews some early work into optimization which could be integrated into the DRE pipeline in the future. Section 6 discusses other technical advancements which were contributed to existing Emulytics™ tools over the course of this project. Section 7 overviews practical application of the DRE pipeline. Section 8 looks ahead to future work in this area and indicates ongoing work with the DRE pipeline.

2. APPROACH

DRE was proposed to integrate Sandia's existing capabilities for cyber-physical system emulation and cyber resilience with capabilities for UQ, optimization, and other statistics-driven studies. In order to motivate this work, the team developed an example application of DRE as depicted in Figure 1. This example applies UQ to a threat of concern to better understand how variations in the configuration of the threat could change how it impacts the system of study. Then, supposing that a mitigation for this threat is being considered, the example applies optimization to the configuration of the mitigation. The optimization process is further supported and made more efficient by use of a surrogate model of the system. The result of this whole analysis would be an optimized mitigation configuration which is robust to the studied variations in the threat.

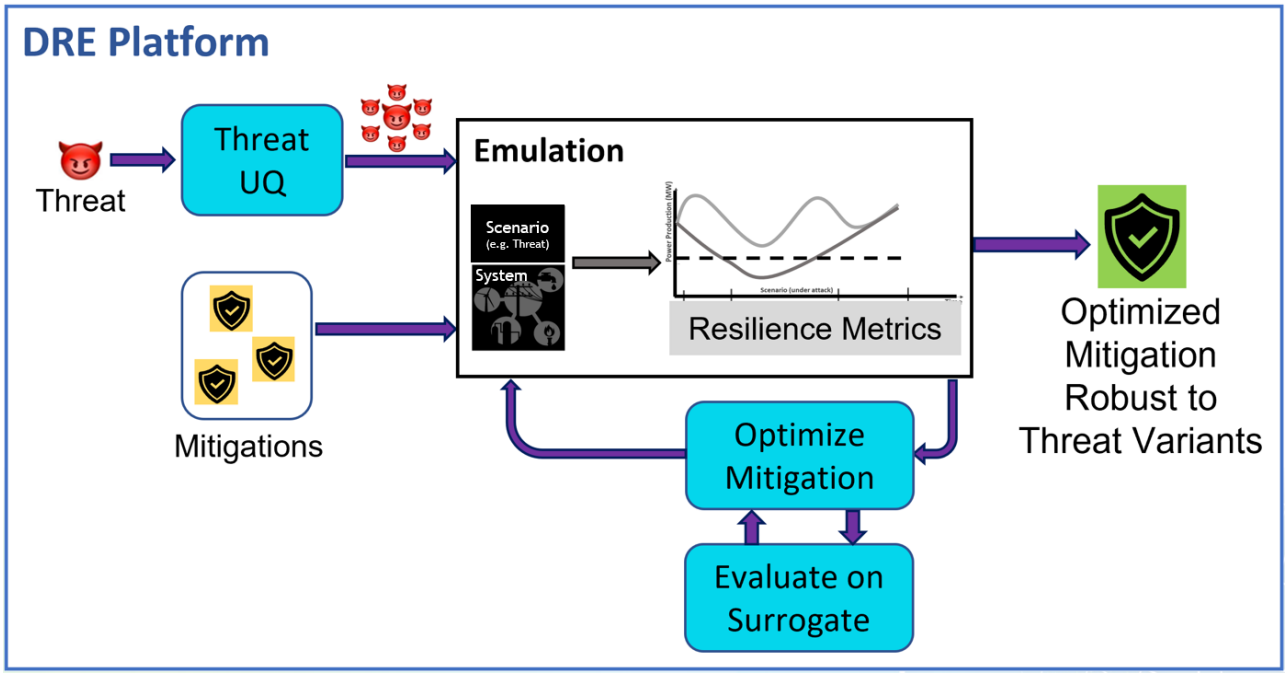

Figure 1. One Potential Configuration of the DRE Platform.

In Figure 1, the black box labeled Emulation represents existing Sandia capabilities in Emulytics™, such as SCEPTRE and RevRun. These capabilities provide a platform to model cyber-physical systems to a high level of fidelity, manage experiments with these models, collect data, and calculate cyber resilience metrics.

The blue boxes in Figure 1 represent existing Sandia work and capabilities in UQ, Optimization, and Surrogate Modeling. These tools are frequently applied to computationally efficient simulations of physical phenomena. Existing tools such as Dakota [12] provide the necessary capabilities for parameter sampling, iterative sampling and experimentation, mathematical modeling, and more.

Much of the research left to the DRE project is represented by the purple arrows in Figure 1. This involved designing and developing interfaces between Dakota and RevRun, extending existing code to accommodate a new experimentation workflow, and researching a new methodology to support efficient experimentation in this workflow. Historically, statistics-driven studies in a cyber emulation environment have been very limited. DRE needed to consider how to integrate scientifically rigorous statistics-driven concepts into an emulation-based experimentation process, and to do so in

a way that would encourage adoption amongst other Emulytics™ efforts. Thus, the three driving research thrusts for DRE corresponded to integrating each of the blue boxes from Figure 1 (Uncertainty Quantification, Surrogate Modeling, and Optimization) into the Emulytics™ workflow – the organization of this report will reflect these thrusts.

It is also important to note that one of the main goals for the project was to keep these capabilities modular and flexible. Figure 1 only shows one potential configuration of the DRE platform. UQ may be applied to other aspects of the system or experimentation rather than just to threat configuration. Likewise, optimization does not have to solely be focused on mitigation tools. In addition, although the Surrogate Modeling effort is depicted supporting optimization, a surrogate model which is more computationally efficient than the emulation environment can be used to reduce the amount of time required to complete any study of the system which requires a large number of samples.

The following sections will discuss work on UQ, Surrogate Modeling, and Optimization respectively. The team proceeded through each of these topic areas over the course of the project, focusing the bulk of our efforts on one topic area at a time. Each area had a defined "proof of concept" problem to motivate technical development, inform experimentation methodology, and spark discussion around a concrete and realistic problem that DRE might be able to help solve.

3. UNCERTAINTY QUANTIFICATION

The concept of "Threat UQ", as seen in Figure 1, was the motivating research problem for FY23 (Year 1) of the DRE project. The goal was to integrate Dakota with RevRun to enable an uncertainty quantification study. This section will briefly discuss the technical integration of the two tools and then overview the demonstration of the capability.

In order to better grasp the description of the tool integration below, the authors supply here an excerpt from the RevRun documentation containing some key jargon. Additional information can be found in RevRun and Dakota tool documentation.

The following bullet points have been copied from the RevRun User Guide [13].

- RevRun – The full Resilience Verification Unit toolset.
- Master Controller – The primary orchestrating script in the RevRun toolset.
- Project Folder – Contains the configuration files and scripts used to customize your analysis.
- Scenario – The high-level concept applied to an emulated system. This could be a threat, a mitigation, a system configuration, or some combination thereof… This is not to be confused with a SCEPTRE Scenario, which is one of two types of configuration files (the other being Topology) that is used to define a SCEPTRE emulation experiment.
- Experiment – An instance of an emulated system with a specific scenario applied. The resulting data from multiple experiments is used to compare the resilience of the system to the associated scenarios.
- Iteration – One "run" of a particular configured experiment. Comes into play when we talk about running a particular experiment configuration multiple times. [Note: DRE research also referred to these as "replicates"].
- RunID – Used to identify the configuration for a single iteration of a single experiment. Depending on the analysis being run, each experiment may only have one iteration, in which case RunID is synonymous with Experiment ID.

3.1. Tool Integration

Technical integration between Dakota and RevRun was relatively straightforward. Dakota is designed to run an arbitrary modeling tool. From the Dakota standpoint, DRE needed to be able to tell Dakota how to run RevRun. On the other side, RevRun is designed to manage SCEPTRE experiments, data collection, and metrics based upon a series of configuration files. From a RevRun standpoint, DRE needed a way for the RevRun configuration to be modified by parameters selected by Dakota. To fulfill both needs, a series of DRE-specific interface code was developed. This code has been more extensively documented as the DRE pipeline, and it is available upon request to the authors. In addition, steps for practical application are outlined in Section 7 of this document. As an overview of the pipeline components:

- DRE Driver: Unpacks input values generated by Dakota. Makes calls to the Configuration Manager (see below) to finish configuring the RevRun project. Makes calls to execute each phase of RevRun (including experiment running, data processing, and metrics calculation). Packages the results from RevRun and writes them out for Dakota.

- Configuration Manager Base Class: Tasked with performing the basic functions required to transform a template RevRun project into a usable project folder, including setting the number of samples requested by Dakota.

- Configuration Manager User Class: This is the only piece of the interface code which must be written by the user. This code will take the inputs from Dakota, convert them to parameters required by the RevRun project, and write them to the relevant RevRun configuration file. This component is left to the DRE user to allow for maximum flexibility of what the Dakota input can correlate to in a RevRun/SCEPTRE experiment.

3.2. Demonstration Exercise

To test and prove out the UQ component of DRE, as well as the basis for the DRE pipeline itself, the team applied the DRE pipeline to a proof-of-concept problem on a use case system. This work has been documented in [14]. For completeness, this report will briefly overview contents that are already captured in the paper and will further expand upon work that happened after the paper was published.

[14] *Toward Rigorous Experimentation and Analysis in Cyber-Physical System Emulation.* SAND2024-04810C, https://doi.org/10.1109/KPEC61529.2024.10676298, published Sept. 2024.

3.2.1. Use Case System

The system in this case represented a shipping port refueling station with connection to the local power grid (Figure 2). There were four primary components of this network, connected by a firewall: the corporate network, the DMZ, the open internet, and the SCADA network. The SCADA network contained several subnetworks, including fueling control, primary and backup power control, and field devices.

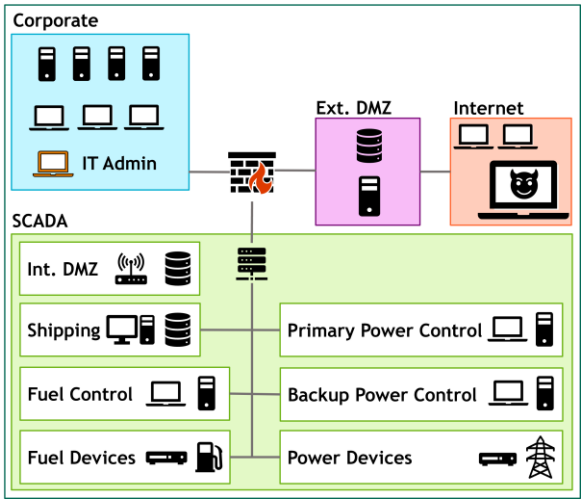

Figure 2. Use Case System for UQ Proof of Concept Problem

The threat studied in this system (Figure 3) was a cyber worm [15] which started in the corporate network, pivoted through the firewall to a workstation in the SCADA network, and executed a payload to disrupt physical connection to the power system. MITRE's CALDERA [16] was used to implement and execute the worm. The attack is managed by the CALDERA server from the open internet using command and control (C2) traffic.

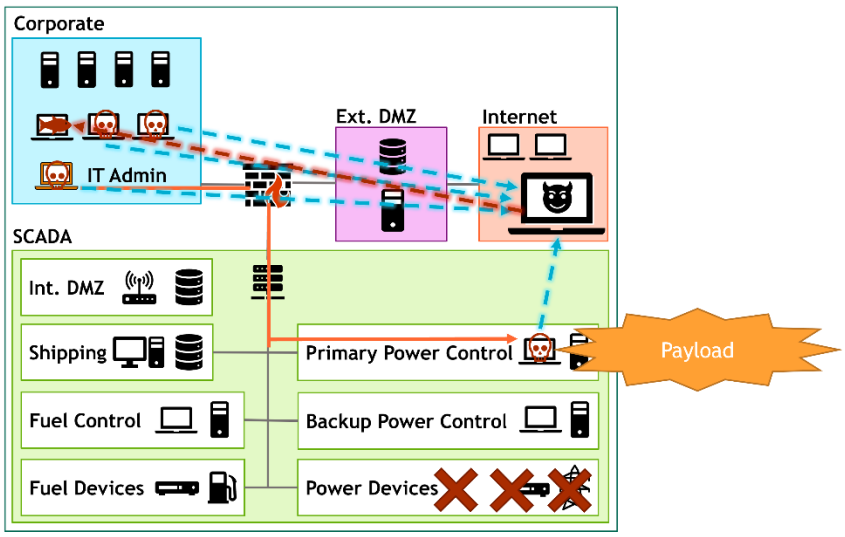

Figure 3. Use Case Threat for UQ Proof of Concept Problem

This system and threat were chosen largely because they had been used recently in related work, and the team believed the behavior of the system was well-understood.

3.2.2. Proof-of-Concept Problem Statement

The goal of the UQ study was to determine how variations in the threat (i.e., how "aggressively" the attacker behaves) affected the resilience of the system to the attack. Input parameters were two settings available in CALDERA to dictate C2 behavior: 1) Beacon rate, determining how frequently infected machines beacon to the CALDERA server, and 2) Jitter, inserting additional time between steps of the attack. As output, the team defined resilience metrics using RevRun. Resilience was measured using a series of metrics which were designed to capture the system's ability to perform its mission in the presence of the attack.

3.2.3. Results

The team expected results similar to Figure 4, showing a linear relationship between the independent variables and the final RevRun-calculated resilience metric. However, the real results are depicted in Figure 5, where there is much more noise than initially expected. In fact, there appears to be very little if any correlation in the data.

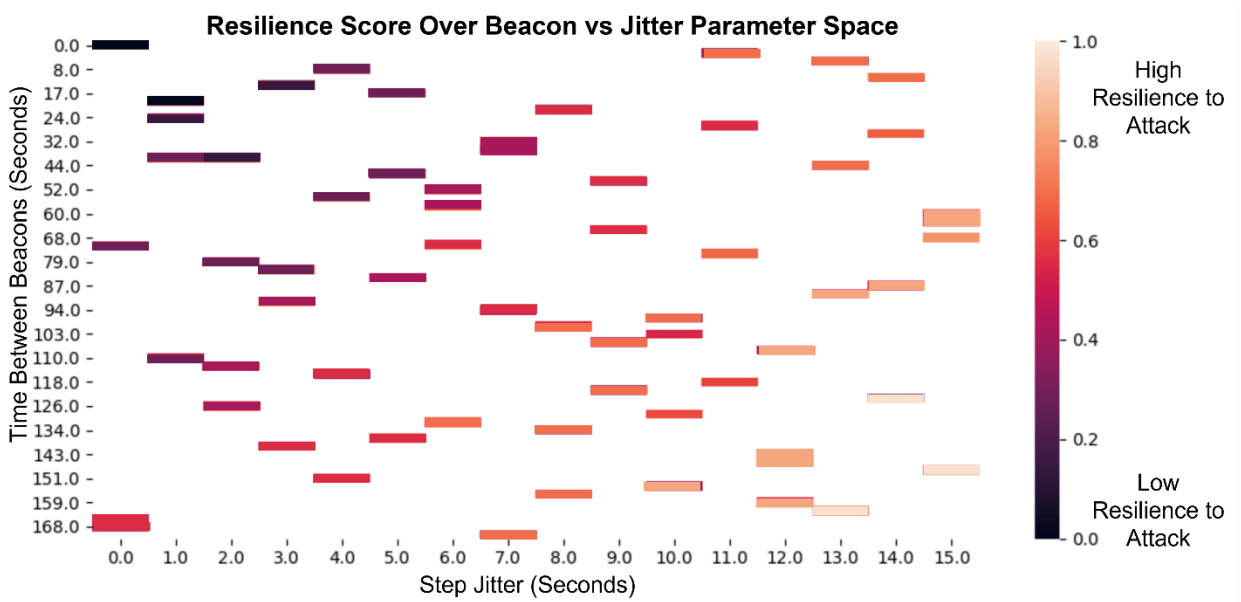

Figure 4. Heatmap depicting expected results of UQ study. "Heat" values come from RevRun metrics, where "0" is least resilient (i.e., attack had the most impact) and "1" is most resilient.

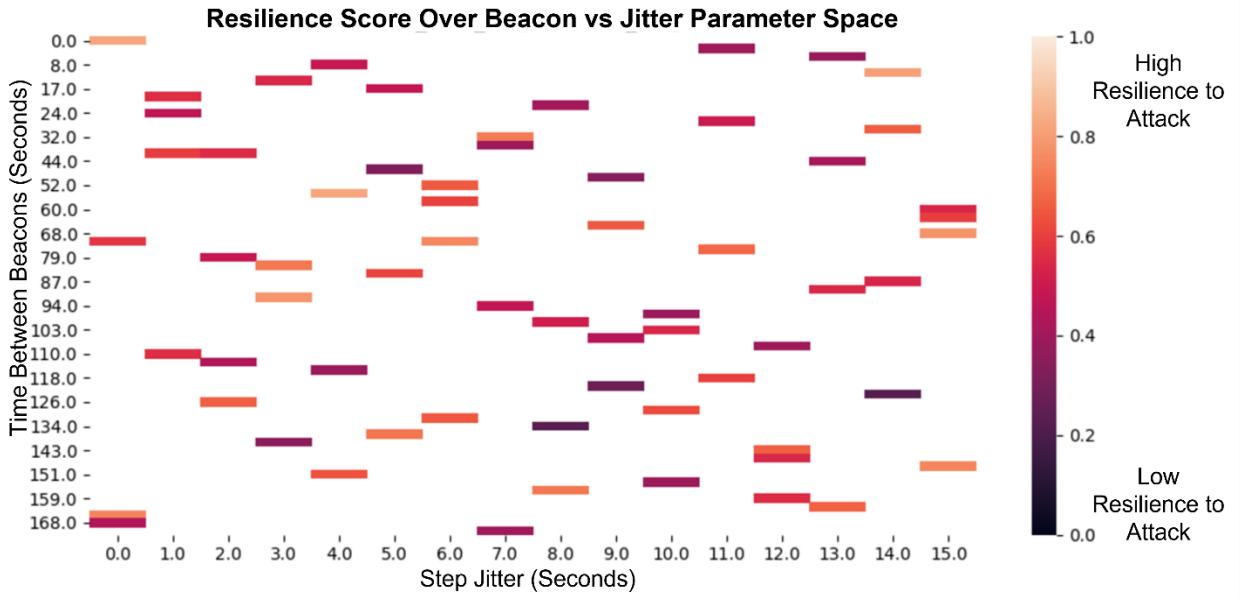

Figure 5. Heatmap depicting real results of UQ study.

After finding these results, the team performed several small follow-up activities to identify (and potentially neutralize) the cause of the unexpected noise levels. These activities included reviewing the configured metrics and data collection techniques for possible unintended errors and checking aspects of the model for inherent variability. Some more details of these studies are presented in the KPEC paper [14], but the conclusion was that the noise came from different aspects of the experiment environment itself, and not from the variation of the chosen independent variables. And without any initial studies to better understand the behavior and variability of the system on its own, it was not possible to attribute all the noise to specific sources. Overall, the team believed there was

a much stronger noise signal than any possible signal from the beacon and jitter variables, so no conclusions could be drawn about the impact of beacon and jitter on the effectiveness of the attack against the system.

Later, after publication of the KPEC paper, one final follow-on study was performed to investigate sources of noise in the attack implementation in CALDERA. The team had identified at this time that most of the noise in the data seemed to come from the attack scenario (as opposed to the baseline system). While a small amount of variability came from the payload, most of the variation in attack timing could be isolated to the time before the attacker reached the target machine. Therefore, the team suspected that CALDERA's implementation of the stages of the attack was causing most of the variability in attack timing.

Investigation revealed that the timing and order with which the cyber worm propagates through the network is unpredictable. In addition, the worm can re-infect machines that it has already infected, which can lead to a sort of denial of service if the host does not have sufficient memory to support a large number of CALDERA agent processes. The combined impact of these two factors on the worm's behavior, in particular the time at which the attacker reached the target machine, was much greater than the influence of the beacon and jitter.

3.3. Lessons Learned

Although the results of the study itself were inconclusive, the team was still able to demonstrate the utility of DRE as an enabler of larger, multi-sample studies. The outcomes of the initial study and follow-on investigations resulted in a much deeper understanding of the system and cyberthreat modeled. Overall, this was a successful first demonstration of the DRE platform. The following subsections dive deeper into specific areas for improvement that were highlighted by this initial study.

3.3.1. Noise Studies and Sources of Variability

The results in Figure 5 were surprising due to a lack of understanding of the system model. The DRE team incorrectly assumed the model was deterministic. Initial exploratory studies would have revealed the run-to-run variability in the system model and threat implementation sooner. For this reason, one of the biggest lessons from this demonstration was the importance of initial noise studies to ensuring full understanding of the modeled system and scenario(s) behaviors. The authors recommend this as a de facto first step in any analysis which similarly tests scenarios on a system model.

Consider that models are always abstractions of real-world systems to some extent. Even if a model is designed to be a one-to-one representation of the real-world system, it is highly unlikely that the model is completely identical. In fact, some level of deviation is expected, owing to the common desires to represent the system at a different level of fidelity or to focus on certain areas of the system and abstract others. Thus, the behavior of the model may have subtle (or not-so-subtle) differences to the behavior of the real system. Depending upon the purpose of the experimentation, these differences may or may not be important to understand and characterize. Noise studies help to identify these variabilities in the first place so that analysts can make informed decisions about how to handle them.

It is important to conduct noise studies in such a way that any noise/variability identified can be correctly attributed to its source at a relevant level of abstraction. For this reason, design of noise study experiments should be very methodical. One approach is to break the system down into

discrete, smaller subcomponents and run small studies on each subcomponent separately. This isolates any noise to those specific components. Another approach (which the DRE team has leveraged in later analyses) involves identifying the logical "layers" of complexity in the model. An initial noise study should be conducted on the simplest version of the model. Then, each layer can be systematically added and re-tested. As new variability is identified, it can then be attributed to the most recently added layer.

The desired input space should also be considered at this time. Typically, a noise study will hold all input values constant, so that any variability in the outputs can be attributed to the model itself rather than changing input values. However, it may be useful to select several points in the input parameter space at which to conduct noise studies. Variability at one point in the input space may be different from another.

Finally, once sources of noise are identified, the analyst must decide how to handle them.

- Certain variability in components or behavior may indicate an error in the implementation of the model or the scenario. These errors are not reflective of the real system. In this case, the ideal approach is to fix the error which is the root cause of the variability. Although the primary purpose of DRE is not to catch these kinds of errors, we have found in the past that this is a helpful byproduct of this kind of extensive experimentation.
- Other kinds of variability may not indicate errors per se, but analysts may still wish to eliminate them for a number of reasons. The variability may be a product of the fact that experiments are being run on a model rather than on a real system (e.g., byproducts of the infrastructure of the emulation environment). Or the variability may be caused by factors the analyst would rather ignore – for example, removing variability caused by the effect of weather patterns on the system by assuming that the weather is always ideal. Or it could be that the model is determined to be too complex, and the desire is to simplify the model or scenario in order to avoid the variability entirely. The technical implementation of removing sources of variability will depend on the model and what the source is, but it typically involves removing or otherwise neutralizing certain capability in the model.
- Some sources of noise may be worth characterizing rather than eliminating. Noise studies can reveal inherent variability in the behavior of a certain aspect of the model, such as threat behavior. However, if this behavior is inherent to the scenario, and is believed to be representative of real-world variability, then eliminating the behavior may cause the model to deviate further from the real world and become less informative. In these cases, it may be worth only characterizing the noise. This way, a statement can be made at the end of the analysis about the impact this particular source of noise may have had on the results.
- Certain sources could simply be ignored without taking any additional action. If the noise signal is significantly weaker than the changes caused by varying input parameters under test, then it may be enough to be aware of the noise signal and not do anything about it.

Regardless of how sources of variability are handled, the important first step is to ensure that they are properly identified and effectively attributed to their relevant sources. This identification step is where DRE can be incredibly useful, enabling large, multi-experiment analyses which can help weed out sources of variability.

3.3.2. Need for Parallelization

One of the major bottlenecks for DRE was the fact that emulation experiments were being run in serial – one experiment at a time, one after the other. Indeed, this has been a limitation of RevRun since its initial development. Now, with DRE pushing RevRun to manage larger numbers of experiments, it behooves us to consider how to parallelize RevRun.

The most time-consuming aspect of the RevRun process is the actual running of the emulation experiment. This is because emulation runs in real time, and some experiments may need to be run for longer periods of time in order to capture data about desired events. As an example, each experiment in the UQ study described in this section ran for one hour. Although the data processing and resilience metrics phases of RevRun may take time (depending on volume of data, compute power, etc.), it's nowhere near as intensive as the experimentation itself.

Initially, the reason for serialization was because RevRun was designed to only run or communicate with a single host. RevRun cannot currently manage multiple experiments on a single host, as it disrupts the data extraction process. However, there is no reason why experiments can't run in parallel across **multiple** hosts. By design, each experiment is completely independent of any other experiment. So, once configured, experiments may be run across any number of hosts.

The DRE team took one approach to RevRun parallelization, discussed in more detail in Section 6.1, which enabled less time-intensive experimentation during surrogate modeling development and exercise (Section 4). However, the approach chosen had some drawbacks. Future work will need to investigate or develop other options to support parallelization in DRE and/or RevRun.

This page left blank

4. SURROGATE MODELING

The second research focus for DRE was surrogate modeling. Surrogate modeling refers to using a more computationally efficient model in place of a complex system representation. Since emulations run in real time, a surrogate model of our emulation environment would be a lower fidelity model that can run faster than real time but is still predictive of answers we would get from the emulation environment. The initial purpose for including this as part of the DRE research plan was to help make the optimization research and implementation more efficient. That said, surrogate models can help make most statistics-driven analyses more efficient by reducing the time and compute resources required to run the large number of sample experiments required to calculate the desired statistics.

4.1. Early Research

4.1.1. Survey of Surrogate Model Approaches

Surrogate modeling represents a specialized application of supervised machine learning (ML) techniques. Early in the DRE project, the team performed some general literature review around previous work with surrogate models in a cyber system context. Although these resources were not necessarily directly used by the research that followed, they still bear listing here as a collection of interesting resources on the subject (organized here by publish date) [17-26]:

- M. Liljenstam, D.M. Nicol, V.H. Berk, R.S. Gray. Simulating Realistic Network Worm Traffic for Worm Warning System Design and Testing. *WORM'03: Proceedings of the 2003 ACM workshop on Rapid Malcode*. ACM. 27 October 2003.
- D.M. Nicol, W.H. Sanders, K.S. Trivedi. Model-Based Evaluation: From Dependability to Security. In *IEEE Transactions on Dependable and Secure Computing*. Vol 1, No 1. IEEE. January-March 2004.
- D. Jin, D.M. Nicol, G. Yan. An Event Buffer Flooding Attack in DNP3 Controlled SCADA Systems. In *Proceedings of the 2011 Winter Simulation Conference*. IEEE. 2011.
- S. Jauhar, B. Chen, W.G. Temple, X. Dong, Z. Kalbarczyk, W.H. Sanders. D.M. Nicol. Model-Based Cybersecurity Assessment with NESCOR Smart Grid Failure Scenarios. In *IEEE 21st Pacific Rim International Symposium on Dependable Computing*. IEEE. 2015.
- L. Jia, R. Alizadeh, J. Hao, G. Wang, J.K. Allen, F. Mistree. A rule-based method for automated surrogate model selection. *Advanced Engineering Informatics*. Vol 45. 2020.
- C. Menghi, S. Nejati, L. Briand, Y.I. Parache. Approximation-Refinement Testing of Compute-Intensive Cyber-Physical Models: An Approach Based on System Identification. *IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE/ACM. 2020.
- S. Asgari, H. Moazamigoodarzi, R.J. Tsai, S. Pal, R. Zheng, G. Badawy, I.K. Puri. Hybrid surrogate model for online temperature and pressure predictions in data centers. *Future Generation Computer Systems*. Vol 114. 2021.
- H.H. Nguyen. Modeling and Analysis of Trustworthy Systems using Extensions of Network Reliability. *Doctoral dissertation*. University of Illinois Urbana-Champaign. 2022.
- X. Qin, Y. Xian, A. Zutshi, C. Fan, J.V. Deshmukh. Statistical Verification of Cyber-Physical Systems using Surrogate Models and Conformal Inference. *ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. ACM/IEEE. 2022.
- M. Rahman, A. Khan, S. Anowar, M. Al-Imran, R. Verma, D. Kumar, K. Kobayashi, S. Alam. Leveraging Industry 4.0 – Deep Learning, Surrogate Model and Transfer Learning with Uncertainty Quantification Incorporated into Digital Twin for Nuclear System. *Preprint*. 30 September 2022. https://doi.org/10.48550/arXiv.2210.00074

Various popular ML methodologies have been adopted as surrogate models in the context of cyber-security research at Sandia National Laboratories. These include:

- Polynomial Regression
- Support Vector Machines (SVM)
- Gaussian Processes
- K-Nearest Neighbors (KNN)
- Artificial Neural Networks (ANN)

The following approaches have been evaluated based on their applicability to the current dataset and research objectives within the DRE. It is important to note that these rankings reflect the opinions of the researcher, Bethanie Williams, and are organized from best to worst.

1. **Gaussian Processes**
    - Gaussian Processes are particularly effective for small to moderate-sized datasets, providing accurate predictions along with uncertainty estimates. They excel in modeling non-linear relationships and offer a flexible and interpretable framework.
    - **Rationale**: Given that DRE currently has a limited number of data samples (potentially no more than 50), Gaussian Processes are well-suited for this scenario. The existing dataset does not exhibit linear relationships, particularly with respect to jitter and beacon range versus resilience score. Gaussian Processes can effectively capture these non-linear dynamics while providing probabilistic predictions.
2. **Support Vector Regression (SVR)**
    - SVR is beneficial for small to moderate datasets, provided there is a sufficient number of samples to learn from. It is straightforward to implement, utilizes kernel functions, and has fewer hyperparameters to tune. SVR is robust against noise and outliers, aiding in the prevention of overfitting through its margin-based approach.
    - **Rationale**: While DRE's dataset is currently small, an increase to around 50 samples could make SVR a viable option. Its margin-based approach and kernel function allow for the development of an accurate surrogate model, although careful selection of hyperparameters is essential.
3. **Polynomial Regression**
    - This method is effective with limited datasets and is straightforward and interpretable, making it suitable for problems of low to moderate complexity. Polynomial regression fits a polynomial equation to the data, allowing for the modeling of non-linear relationships.
    - **Rationale**: Given the limited dataset, polynomial regression presents a simple approach to fitting the data. However, there is a risk of overfitting if the polynomial degree is not chosen appropriately, especially since the current results show significant non-linearity.
4. **K-Nearest Neighbors (KNN)**
    - KNN is easy to implement and can be useful for small datasets with local patterns. It approximates outputs by averaging the nearest data points but struggles in high-dimensional spaces or regions with sparse data.
    - **Rationale**: Although DRE has a limited number of samples, the presence of sparse regions in the data may hinder KNN's effectiveness. The observed localized

behavior in resilience scores may change as more data is collected, potentially diminishing KNN's applicability.

5. **Artificial Neural Networks (ANN)**
    - ANN requires large datasets and can be computationally intensive, making it less suitable for the current research context. While capable of handling complex and high-dimensional problems, the resource demands are prohibitive.
    - **Rationale**: Given the current focus on only three parameters (jitter range, beacon range, and resilience score), the time and resources required to train an ANN are not feasible for DRE at this stage.

The top three approaches identified for DRE's surrogate modeling efforts are:

1. **Gaussian Processes**: Best suited for uncertainty quantification, which is essential for DRE. The challenge is that these models require hyperparameter tuning and may be more computationally intensive than SVR and Polynomial Regression.
2. **Support Vector Regression (SVR)**: Note that SVRs involves selecting optimal hyperparameters, kernel functions, and data preprocessing methods, particularly in noisy datasets.
3. **Polynomial Regression**: This model type requires careful selection of polynomial degree to avoid overfitting.

The following approaches were deemed unrealistic for DRE**:**

- **KNN**: May struggle with sparse regions and lack of local patterns.
- **ANN**: Impractical due to excessive time and data requirements.

4.1.2. Early Experiments Conducted with Surrogate Models

Following this initial survey of surrogate model methods and tools, the team adopted the Python Surrogate Model Toolbox (SMT) and leveraged a training Jupyter notebook with several surrogate model examples and modified the code to work for the DRE models.

In the context of DRE's cyber-security research efforts, various surrogate models were evaluated to assess their performance in predicting outcomes based on limited datasets. The following sections describe the surrogate models implemented in the Jupyter notebook, along with a brief overview of each model's characteristics and how they align with the previously discussed surrogate modeling approaches.

4.1.2.1. Linear Surrogate Model (LS)

The Linear Surrogate Model is a fundamental approach that fits a linear equation to the training data. It is particularly useful for problems where relationships between variables are expected to be linear.

- **Implementation**: The model was initialized, trained on the dataset, and predictions were made on validation points.
- **Performance**: The relative error was computed to evaluate the model's accuracy.

4.1.2.2. Quadratic Model (QP)

The Quadratic Model extends the linear approach by fitting a quadratic equation to the data, allowing for the modeling of non-linear relationships while remaining interpretable.

- **Implementation**: Similar to the LS model, the QP model was trained on the dataset and validated against test points.
- **Performance**: The model's effectiveness was assessed through relative error calculations.

4.1.2.3. Kriging Model

Kriging is a powerful surrogate modeling technique that provides not only predictions but also estimates of uncertainty. It is particularly well-suited for small to moderate datasets and can capture complex, non-linear relationships.

- **Implementation**: The model was initialized with specified hyperparameters, trained on the data, and predictions were made for validation points. Additionally, the model provided variance estimates for confidence intervals.
- **Performance**: The Kriging model's predictions were validated, and the estimated variance was visualized to illustrate the model's uncertainty quantification capabilities.

4.1.2.4. Kernel Principal Component Regression (KPLS)

The KPLS model combines principal component analysis with regression, allowing for dimensionality reduction while modeling relationships in the data. This approach is beneficial when dealing with multicollinearity among input variables.

- **Implementation**: The model was initialized with a specified number of components, trained on the dataset, and predictions were made for validation points.
- **Performance**: The model's accuracy was evaluated through relative error metrics.

4.1.2.5. Kernel Principal Component Regression with Kernel (KPLSK)

The KPLSK model is an extension of KPLS that incorporates kernel methods, enhancing its ability to model complex relationships in the data.

- **Implementation**: Similar to KPLS, the KPLSK model was trained on the dataset and validated against test points.
- **Performance**: The model's predictions were assessed, and confidence intervals were generated to evaluate uncertainty.

4.1.2.6. Inverse Distance Weighting (IDW)

The IDW model is a non-parametric method that estimates values based on the distance from known data points, making it suitable for spatial data interpolation.

- **Implementation**: The model was trained on the dataset and predictions were made for validation points.
- **Performance**: The accuracy of the IDW model was evaluated through relative error calculations.

4.1.2.7. Radial Basis Function (RBF)

The RBF model utilizes radial basis functions to interpolate data points, making it effective for capturing non-linear relationships in multi-dimensional spaces.

- **Implementation**: The model was trained on the dataset and predictions were made for validation points.
- **Performance**: The model's effectiveness was assessed through relative error metrics.

4.1.2.8. Response Surface Methodology (RMTB and RMTC)

The RMTB and RMTC models are advanced surrogate modeling techniques that optimize responses based on a defined objective. They are particularly useful for problems requiring optimization under constraints.

- **Implementation**: Both models were trained on the dataset, and predictions were made for validation points.
- **Performance**: The accuracy of both models was evaluated through relative error calculations.

4.1.3. Summary of Surrogate Models

The experiments conducted with these surrogate models demonstrate a range of methodologies suitable for different types of data and research objectives. The models align with the previously discussed surrogate modeling approaches, with Gaussian Processes and Support Vector Regression being particularly relevant for DRE's needs due to their ability to handle uncertainty and non-linear relationships effectively. The results from these experiments informed later modeling efforts of the DRE team.

4.2. Demonstration Exercise

The purpose of this demonstration exercise was threefold: 1) motivate any code development required to train a surrogate model from DRE-generated data, 2) inform requirements and recommendations for methodology future users should employ when using DRE, and 3) evaluate the viability of our approach as a precursor to an optimization study. If this exercise did not result in a successful surrogate model, there would be no point in trying to use it as part of an optimization study.

The demonstration exercise was divided into four phases:

1. Noise Study of the use case system. Our earlier exercise (Section 3.2) demonstrated the importance of initial noise studies for understanding sources of variability within the model and desired test scenarios.
2. Data collection from the use case system. Once we understood our model, we could run large numbers of experiments, varying parameters and collecting resulting data and resilience metrics. The quality of this data would determine the quality of the resulting surrogate model.
3. Surrogate model development. Several surrogate model structures had been proposed as described in Section 4.1. The goal of this phase would be to use the data collected in the previous phase and attempt to train these models.

4. Comparative study for validation. In order to validate the performance of the surrogate model, we would compare its results on a specified proof of concept problem to the results of the full emulation environment.

The outcomes of each of these phases will be described under the Results section (4.2.3) below.

4.2.1. Use Case System

Based upon the results of the UQ exercise described in Section 3.2, the team was not confident that the UQ use case system could produce data that was likely to be able to train a useful surrogate model. Therefore, the team moved to using the HARMONIE model [27] as the use case system and disruption implementation.

As a summary, the model builds upon the Western System Coordinating Council 9-bus model [28], adding a SCADA network, a notional and flexible attacker presence, and configurable disruption(s) to the system. The HARMONIE model consists of three spatially diverse regions each with its own generator and loads along with transmission lines between the regions. In one of the disruption scenarios, a physical disruption is conducted that removes one of the generators from the network and cuts the transmission lines between two of the regions. This results in a condition where there is insufficient generation to satisfy the loads, and the expected mitigation is to issue a load shed command to two of the regions. In one of the other disruption scenarios, the physical disruption is also coordinated with a Denial-of-Service attack (DOS). The goal of the DOS is to keep the load shed commands from making it to the destination regions. If the DOS is successful, the whole network will go down and power will go out to all regions.

The authors recommend reviewing work published on the HARMONIE model for further details.

4.2.2. Proof-of-Concept Problem Statement

The DOS has two primary parameters that are evaluated in this exercise. The first parameter is start_delay, or the amount of time (in seconds) between the start of the experiment and the start of the DOS attack. The second parameter is the attack_duration, which indicates the amount of time (in seconds) the DOS will run for. We would expect that if the DOS attack starts before the physical disruption and continues for some time afterward, then the physical mitigation will not be effective and the system will be less resilient. On the other hand, if the DOS starts too late or is too short in length, we expect the load shed packet to make it through the network and successfully mitigate the physical disruption.

4.2.3. Results

There were many challenges in the process of getting results for this demonstration. Each of the following sections will cover relevant challenges and how they were overcome. The final section discusses the results of the last simulation study we completed.

4.2.3.1. Noise Study Outcomes

The first phase of the surrogate modeling exercise involved performing a noise study to investigate the HARMONIE model under various disruption scenarios. For the DOS disruption cases in particular, six additional sets of input values were intentionally selected to look at different cases of expected behavior from the DOS in the environment.

Table 1: Noise Study Experiments

Disruption Type	DOS Start Time (s)	DOS Duration (s)	# Experiment Replicates Run
No Disruption	N/A	N/A	16
Physical Disruption	N/A	N/A	16
Cyber Disruption	60	120	16
Cyber + Physical Disruption	60	120	16
Cyber + Physical Disruption	30	180	16
Cyber + Physical Disruption	30	360	16
Cyber + Physical Disruption	60	330	16
Cyber + Physical Disruption	120	270	16
Cyber + Physical Disruption	0	60	16
Cyber + Physical Disruption	360	60	16

Experiments that used the exact same experimental setup (i.e., all replicates from the same row in Table 1) could be compared to identify potential sources of inherent variability. Experiments that all ran the DOS under different sets of inputs (i.e., Table 1 rows three onward) could be compared to characterize DOS behavior in more detail.

The following set of box-and-whisker plots helps to depict the level of variability in a select set of data streams across samples collected using the first four settings from **Error! Reference source n ot found.**

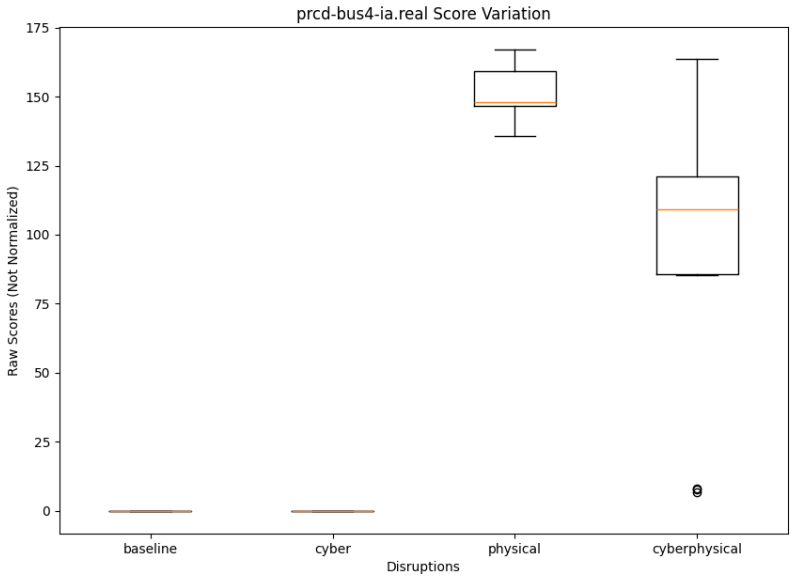

Figure 6: Current in Phase A on Bus 4, one of the busses impacted by physical disruption.

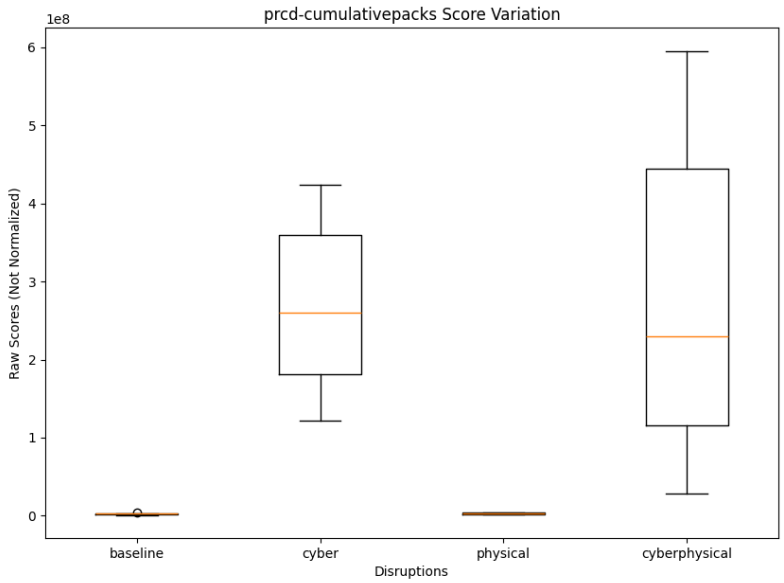

Figure 7: Cumulative volume of network traffic.

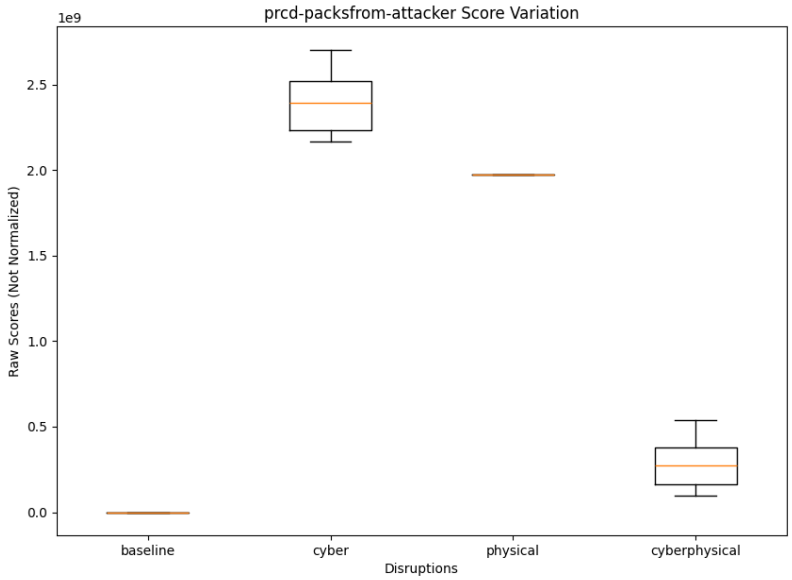

Figure 8: Cumulative traffic from the DOS attacker IP address.

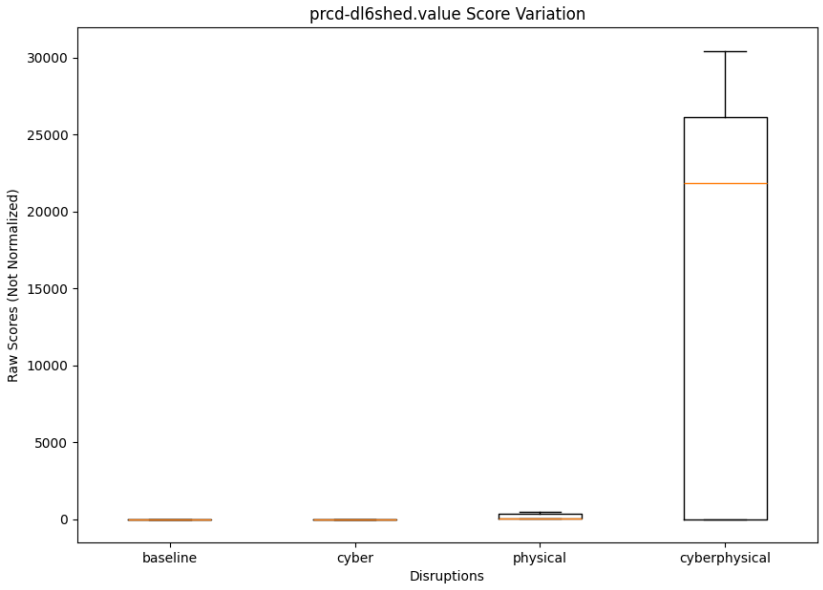

Figure 9: Timing of Load Shed at Load 6.

This study resulted in several key findings for the environment.

- Sampling highlighted the dependency between selected DOS input values, leading to discussion on what variables the team wanted to study and how to implement those in the pipeline so that we could most effectively sample the parameter space we cared about.

- Looking across experiments where DOS parameters were invariant, the DRE team identified inconsistent performance of the DOS and whether it had its intended effect on the load shedding mitigation or not. This variability is depicted in Figure 10. After consulting with the HARMONIE team, it was determined that this variability should be characterized but not eliminated for future experimentation.

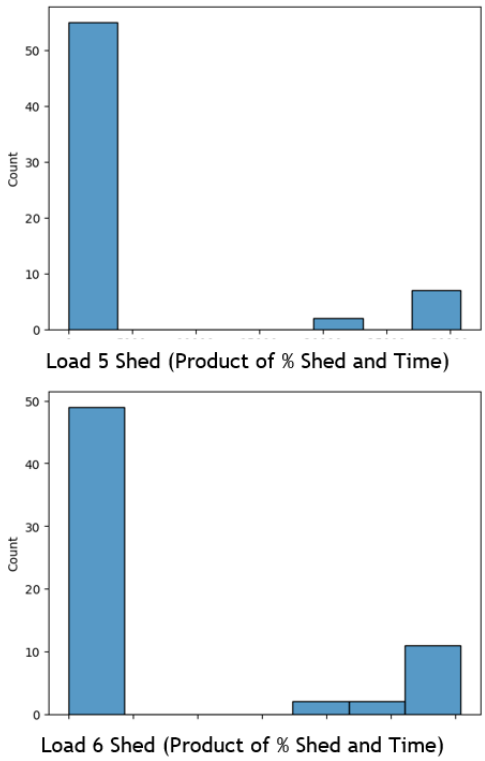

Figure 10: Histogram of Load Shed Metric.

- By varying the input parameters for the DOS beyond how it was originally used by the HARMONIE team, the DRE team was able to identify some fragility in the original implementation of the disruption code and recommend a more generalized solution.
- All of the collected data from this study was used to help investigate modifications to the RevRun metric configuration for this exercise. Samples highlighted which data streams were significantly impacted by the various disruptions and which remained more stable. Data streams which experienced greater impact would be more effective components of the overall experiment metrics.

4.2.3.2. Observations from the Data Collection Process

The second phase of the surrogate modeling exercise involved data collection.

Ideally, this research would produce a general set of guidelines for sampling data and developing effective metrics and surrogate models of emulated systems which would not rely on a priori knowledge of scenarios under test and how scenarios impact the system. However, the DRE team started this phase with a simpler goal – could we generate data that could be used to train a surrogate model specifically for HARMONIE, even if it required using a priori knowledge of the attack scenarios under study? Additional research would be needed to generalize any findings.

This is the phase where the team spent the most time iterating on how to collect the "right" data in the "right" way. There were several challenges and interesting questions raised about the data collection process – some technical and some methodological.

Technical challenges with the DRE pipeline included:

- A couple of common failure modes in the SCEPTRE model and tool stack meant that several experiments needed to be rerun after each batch. As time went on, the team developed methods to catch these failures and start rerunning experiments more automatically.
- RevRun relies on Elasticsearch [29] as its underlying database. Elasticsearch settings on the host had to be modified in order to accommodate our use case and allocate enough space to the Elasticsearch database. An important consideration here is that RevRun creates a large number of relatively small indices, which is not the typical use case for Elasticsearch.
- Several capability gaps were identified around data collection in RevRun, including
    - Growing complexity of the RevRun configuration file,
    - Overreliance on packetbeat [30] to collect network data,
    - Inefficiencies in how RevRun handles raw data files,
    - Inefficiencies in how RevRun processes raw data.

The following methodological topics were also considered:

- RevRun metrics rely on a "comparative baseline" – an additional experiment which is meant to represent the system as-is, or to at least be a consistent view of the system against which any scenarios of interest can be compared. Good selection of this baseline was an interesting question during the surrogate modeling exercise. Initially, the baseline was the HARMONIE model with no disruption running. This baseline was compared against samples of the cyber-physical disruption scenarios, meaning all data and metrics incorporated the impacts of both the physical disruption and the DOS together. The team later determined this wasn't ideal – in order to study the impact of varying DOS timing parameters, what we really wanted was to look at the impact of the DOS to the cyber network and physical system operation. Thus, we shifted to using a physical-only disruption scenario as the baseline.
- Prior to this work, common practice did not necessarily involve characterizing the input space for the models. But with a downstream surrogate modeling use case in mind, the team began to ask some questions about the input space. For instance, what is the shape of the input space (the answer to which will depend on how dependent/independent our inputs variables are)? How do we make sure sampling covers the input space we care about? How space-filling is our sampling process? Are there any other considerations to be made based on information collected during the initial noise study?
- In the same vein as the previous comment, this work also motivated more characterization of the output space (or "response surface") created by our experimentation. We asked, what are our metrics actually calculating? Do these metrics create a smooth surface? Are there outliers, and if so, what do we do with them? How varied is the output surface – do the inputs we tested result in noticeable differences in the response surface?

- Like with the Uncertainty Quantification study, we are left asking – how do we handle inherent stochasticity in a system or model? This question deserves continued study as we seek a variety of options to better support characterizing variability in models.

During this phase of the study, the team (acting as analysts) made certain choices about the model and metrics configuration in order to maximize our chances of being able to train a usable surrogate model. Our modifications were largely intended to make the data conform to some of our ideals about shape, fillingness, smoothness, etc. In a way, using a priori knowledge of the experiment environment can feel unrealistic, since in a real-world experiment we wouldn't necessarily have such knowledge. In a perfect world, we wouldn't *need* to make all of these modifications based upon the data. Instead, we'd be able to set up the model, inputs, and outputs so as to reflect the system and our understanding of it to the best of our ability, and simply use the data as-is. However, real life is unlikely to EVER reach this ideal state. Any users of this approach will likely have to make a judgement call about their boundaries here: when do you design the analysis to fit the data, and when do you make the data work for the analysis?

For the purpose of this study under the DRE project, the team reasoned that if we can't demonstrate the proposed workflow when we ARE manipulating the data/configs in this way, then it's highly unlikely we'd be able to make it work at all WITHOUT these manipulations. Therefore, we chose to investigate the problem with added constraints derived from a priori knowledge. Future work or future users may need to investigate the problem without such constraints.

4.2.3.3. Surrogate Model Outcomes

The next phase of the surrogate modeling study involved using the collected data to train surrogate models. After addressing the challenges with the data collection process, we conducted a replicate study with 40 replicates for 36 different samples of the DOS start_delay and attack_duration (note that the motivation for collecting replicates of samples is discussed further in Section 6.2). This results in a triangular sample space. By using a beta distribution, we were able to overcome a skew issue in the initial sampling of the domain.

There were many challenges in processing the huge amount of data that we generated, and the results provided here consist of 3 replicates from each of the 36 samples.

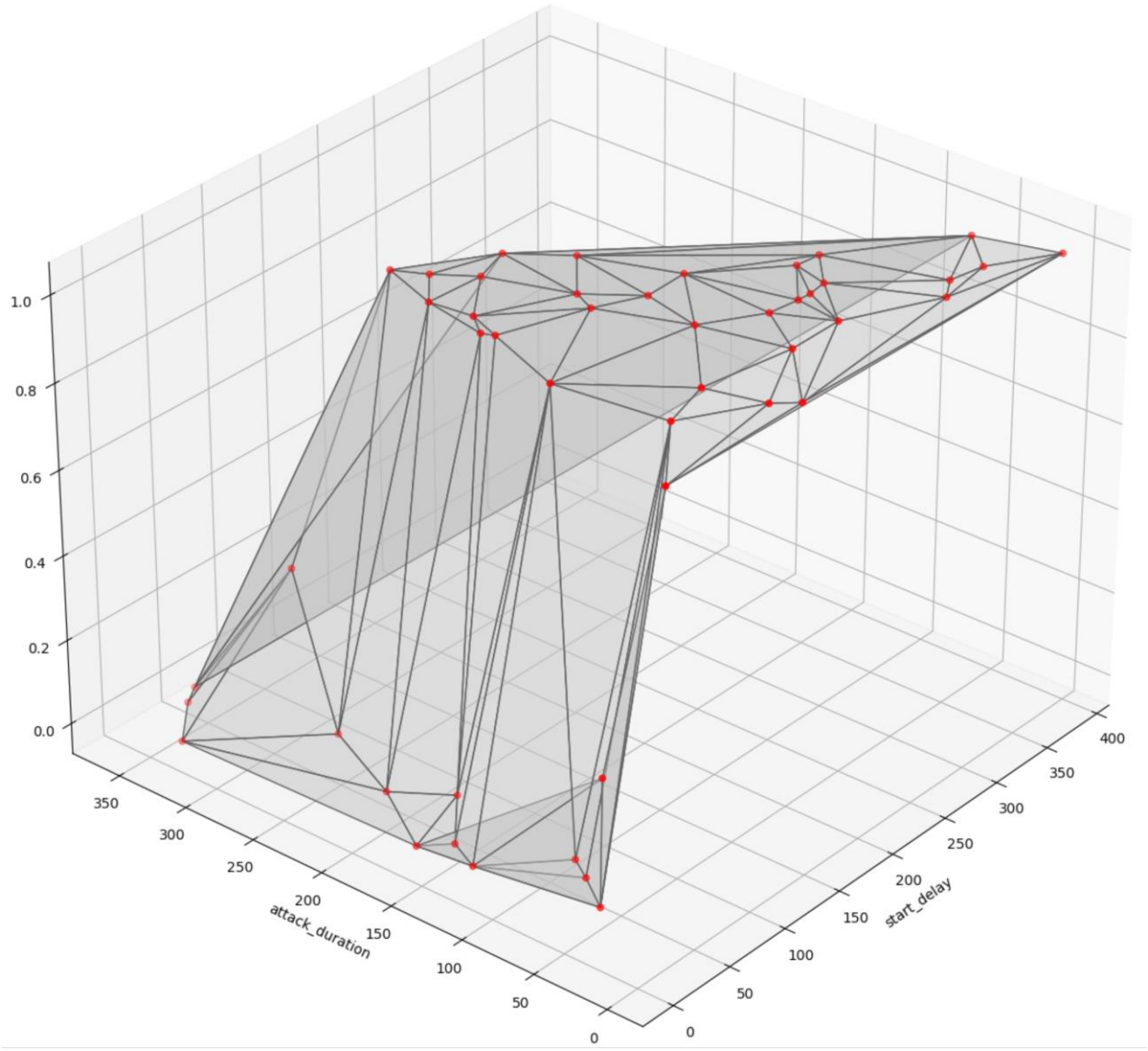

Figure 11: Resilience scores for samples where the load shed occurred within a given time threshold.

Figure 11 shows resilience score data with 3 replicates for each sample. The data has a very strong cliff in it and the surrogate models in the Python SMT library had a hard time producing a good model. It is also worth noting that the resilience score can only be one of the following four values: zero, 0.33…, 0.66…, or one.

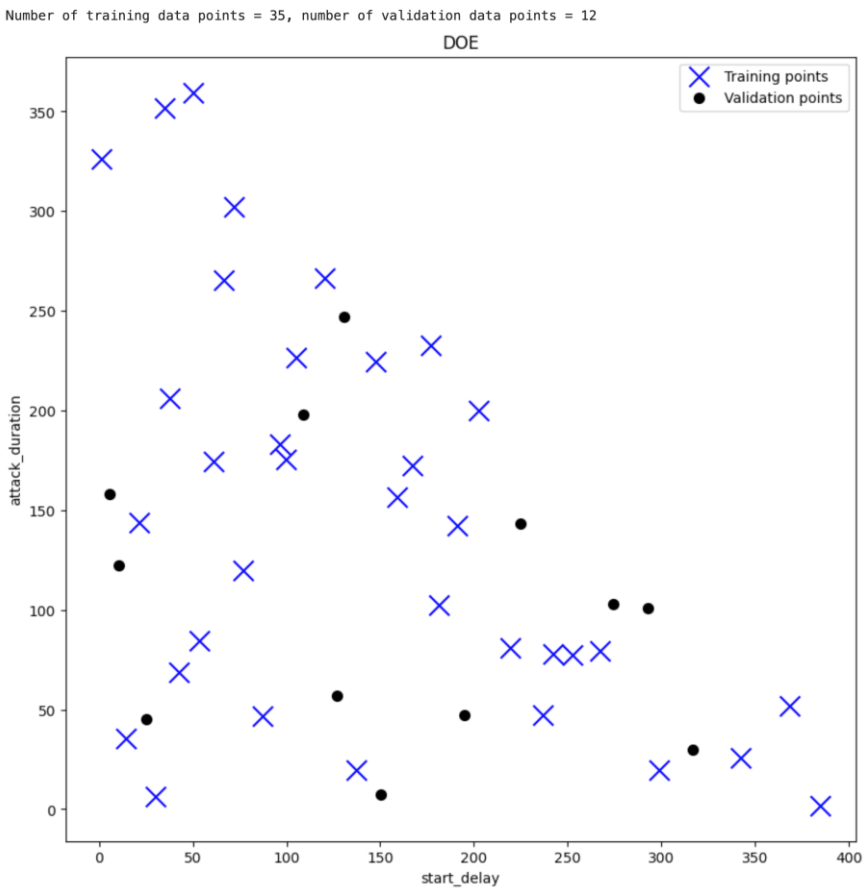

Figure 12: Samples and Training data for Surrogate model on samples with 3 replicates per sample.

In all of the results, 75% of the data is used to train the surrogate model and 25% of the data is used to test the surrogate model. Figure 12 shows the sample points and the training points for the 36-sample experimental run. Table 2 shows all of the samples along with their numeric input values and final score.

Table 2: Parameters for Initial Experimental Run

	RunID	current_disruption	start_delay	attack_duration	final_score
0	tes40	cyber_physical	105.20	226.52	1.000000
1	tes80	cyber_physical	14.47	35.42	0.333333
2	tes120	cyber_physical	267.46	79.35	1.000000
3	tes160	cyber_physical	241.95	78.00	1.000000
4	tes200	cyber_physical	42.82	68.85	0.000000
5	tes240	cyber_physical	176.92	232.60	1.000000
6	tes320	cyber_physical	30.33	6.47	1.000000
7	tes360	cyber_physical	72.03	302.31	0.333333
8	tes400	cyber_physical	219.46	80.76	1.000000
9	tes440	cyber_physical	147.64	224.26	1.000000
10	tes480	cyber_physical	342.02	25.53	1.000000
11	tes520	cyber_physical	96.21	182.87	1.000000
12	tes560	cyber_physical	120.14	266.21	1.000000
13	tes600	cyber_physical	236.99	47.21	1.000000
14	tes640	cyber_physical	137.44	19.87	1.000000
15	tes680	cyber_physical	21.56	143.88	0.000000
16	tes720	cyber_physical	298.78	19.65	1.000000
17	tes760	cyber_physical	1.32	326.22	0.000000
18	tes800	cyber_physical	166.90	172.15	1.000000
19	tes840	cyber_physical	368.14	51.82	1.000000
20	tes880	cyber_physical	61.19	174.43	0.000000
21	tes920	cyber_physical	53.49	84.58	0.000000
22	tes960	cyber_physical	191.25	142.12	1.000000
23	tes1000	cyber_physical	384.40	1.95	1.000000
24	tes1040	cyber_physical	99.55	175.39	1.000000
25	tes1080	cyber_physical	37.69	206.19	0.000000
26	tes1120	cyber_physical	77.10	119.84	1.000000
27	tes1160	cyber_physical	252.34	77.45	1.000000
28	tes1200	cyber_physical	66.31	265.45	0.000000
29	tes1240	cyber_physical	181.15	102.41	1.000000
30	tes1280	cyber_physical	35.07	351.52	0.000000
31	tes1320	cyber_physical	158.88	156.37	1.000000
32	tes1360	cyber_physical	50.22	359.57	0.000000
33	tes1400	cyber_physical	87.12	46.63	1.000000
34	tes1440	cyber_physical	202.35	200.18	1.000000
35	tes1480	cyber_physical	25.01	44.93	0.000000
36	tes1520	cyber_physical	274.26	102.93	1.000000
37	tes1560	cyber_physical	292.59	101.08	1.000000
38	tes1600	cyber_physical	194.84	46.95	1.000000
39	tes1640	cyber_physical	316.47	29.88	1.000000
40	tes1680	cyber_physical	150.06	7.16	1.000000
41	tes1720	cyber_physical	130.69	246.85	1.000000
42	tes1760	cyber_physical	10.64	122.28	0.000000
43	tes1800	cyber_physical	127.12	56.92	1.000000
44	tes1840	cyber_physical	108.85	198.12	1.000000
45	tes1880	cyber_physical	224.86	143.10	1.000000
46	tes1920	cyber_physical	5.66	158.27	0.000000

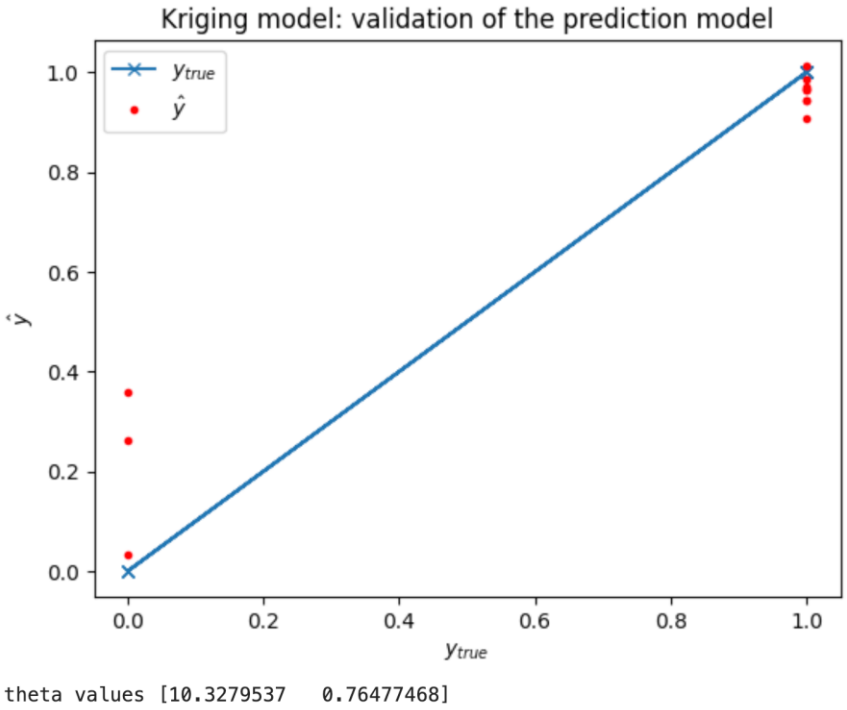

Figure 13: Kriging model error for the 3-replicate sample data.

In Figure 13, the Kriging model produced the best surrogate model for this data compared to other models of the set evaluated in Python SMT. There are only two values in the resilience score space for these test samples, 0 and 1. The data appears to be so discontinuous that all of the models performed poorly on this data.

4.2.3.4. Model Testing and Validation

The final phase of the surrogate modeling exercise was to test and validate the developed model. A more thorough validation process would compare the results of the surrogate model back to the results of the original emulation environment when used to answer some other question (e.g., an uncertainty quantification study). However, due to time constraints, the DRE team was only able to do some early testing of some of the trained models.

Table 3: Final Sample Data for Surrogate Model Training and Testing

	RunID	current_disruption	start_delay	attack_duration	final_score	num_values
0	tes40	cyber_physical	105.20	226.52	1.000000	7
1	tes80	cyber_physical	14.47	35.42	0.000000	6
2	tes120	cyber_physical	267.46	79.35	1.000000	5
3	tes160	cyber_physical	241.95	78.00	1.000000	2
4	tes200	cyber_physical	42.82	68.85	0.000000	2
5	tes240	cyber_physical	176.92	232.60	1.000000	3
6	tes280	cyber_physical	79.29	225.93	0.833333	6
7	tes320	cyber_physical	30.33	6.47	0.875000	8
8	tes360	cyber_physical	72.03	302.31	0.142857	7
9	tes400	cyber_physical	219.46	80.76	0.750000	4
10	tes440	cyber_physical	147.64	224.26	0.250000	4
11	tes480	cyber_physical	342.02	25.53	1.000000	6
12	tes520	cyber_physical	96.21	182.87	1.000000	2
13	tes560	cyber_physical	120.14	266.21	1.000000	3
14	tes600	cyber_physical	236.99	47.21	1.000000	5
15	tes640	cyber_physical	137.44	19.87	0.666667	9
16	tes680	cyber_physical	21.56	143.88	0.000000	5
17	tes720	cyber_physical	298.78	19.65	0.888889	9
18	tes760	cyber_physical	1.32	326.22	0.000000	8
19	tes800	cyber_physical	166.90	172.15	1.000000	7
20	tes840	cyber_physical	368.14	51.82	0.666667	3
21	tes880	cyber_physical	61.19	174.43	0.000000	7
22	tes920	cyber_physical	53.49	84.58	0.000000	3
23	tes960	cyber_physical	191.25	142.12	0.750000	4
24	tes1000	cyber_physical	384.40	1.95	1.000000	8
25	tes1040	cyber_physical	99.55	175.39	0.857143	7
26	tes1080	cyber_physical	37.69	206.19	0.000000	12
27	tes1120	cyber_physical	77.10	119.84	0.909091	11
28	tes1160	cyber_physical	252.34	77.45	1.000000	8
29	tes1200	cyber_physical	66.31	265.45	0.000000	10
30	tes1240	cyber_physical	181.15	102.41	1.000000	4
31	tes1280	cyber_physical	35.07	351.52	0.000000	8
32	tes1320	cyber_physical	158.88	156.37	1.000000	1
33	tes1360	cyber_physical	50.22	359.57	0.000000	5
34	tes1400	cyber_physical	87.12	46.63	0.800000	5
35	tes1440	cyber_physical	202.35	200.18	1.000000	1
36	tes1480	cyber_physical	25.01	44.93	0.000000	3
37	tes1520	cyber_physical	274.26	102.93	1.000000	4
38	tes1560	cyber_physical	292.59	101.08	1.000000	5
39	tes1600	cyber_physical	194.84	46.95	0.750000	4
40	tes1640	cyber_physical	316.47	29.88	0.750000	4
41	tes1680	cyber_physical	150.06	7.16	1.000000	2
42	tes1720	cyber_physical	130.69	246.85	1.000000	4
43	tes1760	cyber_physical	10.64	122.28	0.000000	4
44	tes1800	cyber_physical	127.12	56.92	1.000000	3
45	tes1840	cyber_physical	108.85	198.12	1.000000	3
46	tes1880	cyber_physical	224.86	143.10	1.000000	3
47	tes1920	cyber_physical	5.66	158.27	0.000000	1

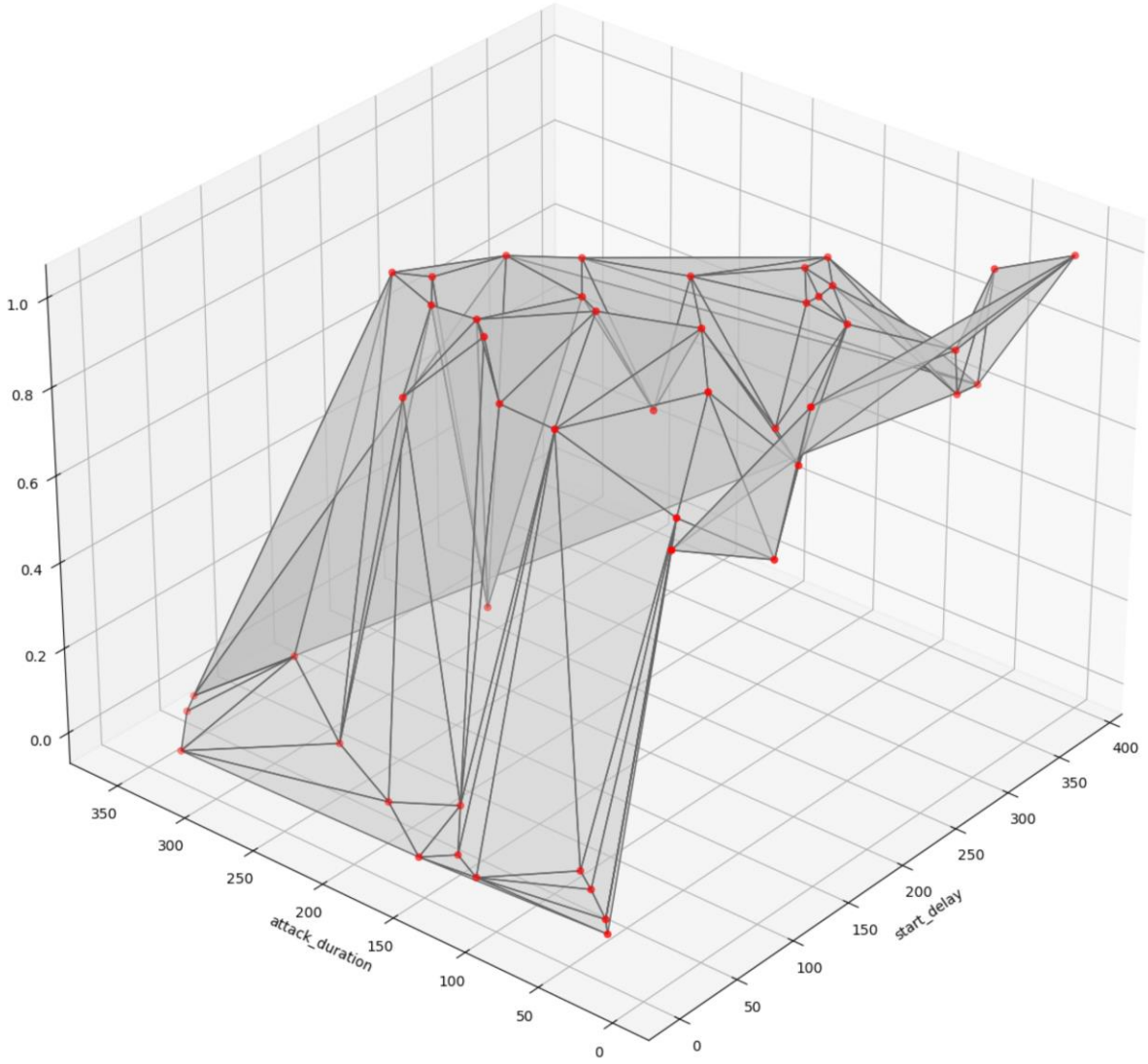

Figure 14: Resilience score for replicate study as a function of start_delay and attack_duration.

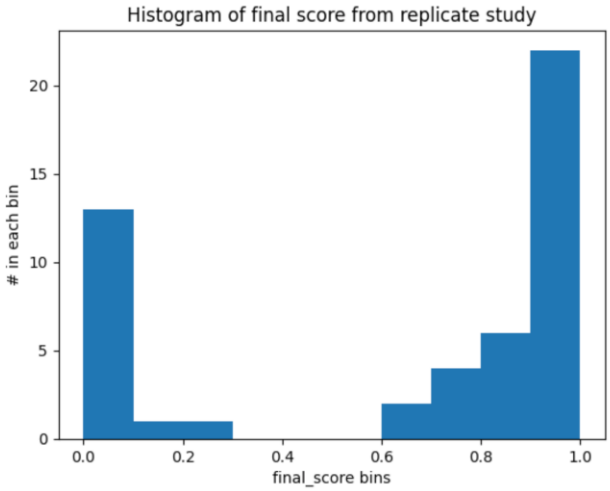

Figure 15: Broader range of final resilience scores from larger set of replicates.

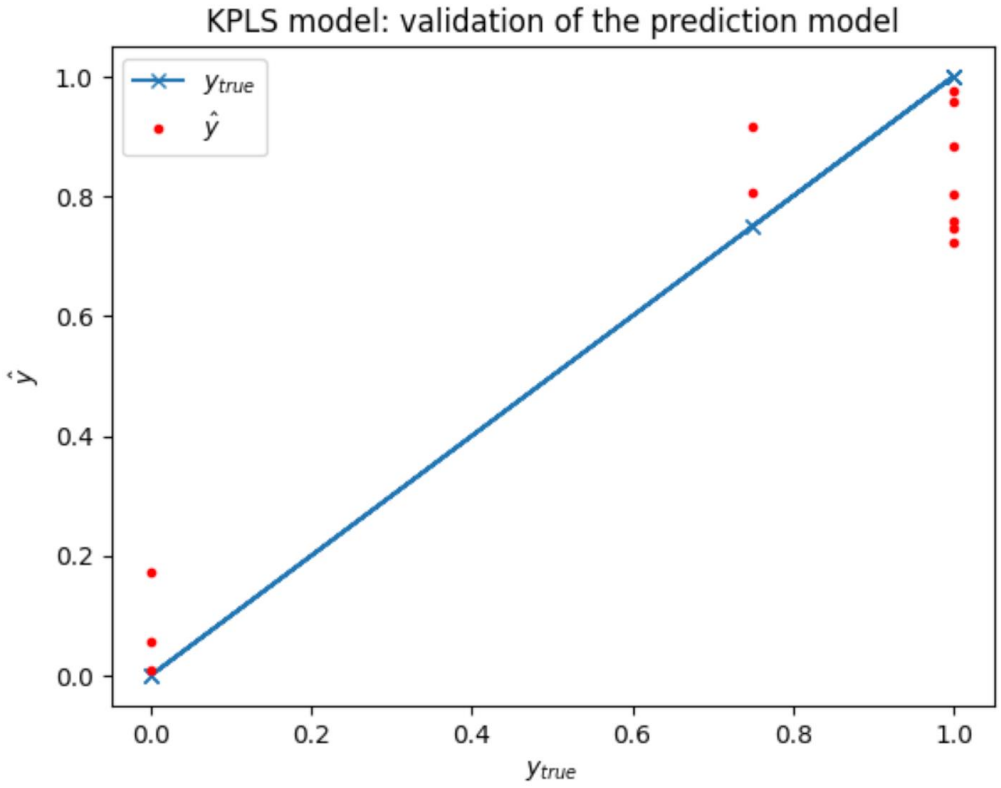

Figure 16: Error plot for KPLS surrogate model on larger number of test replicates test samples.

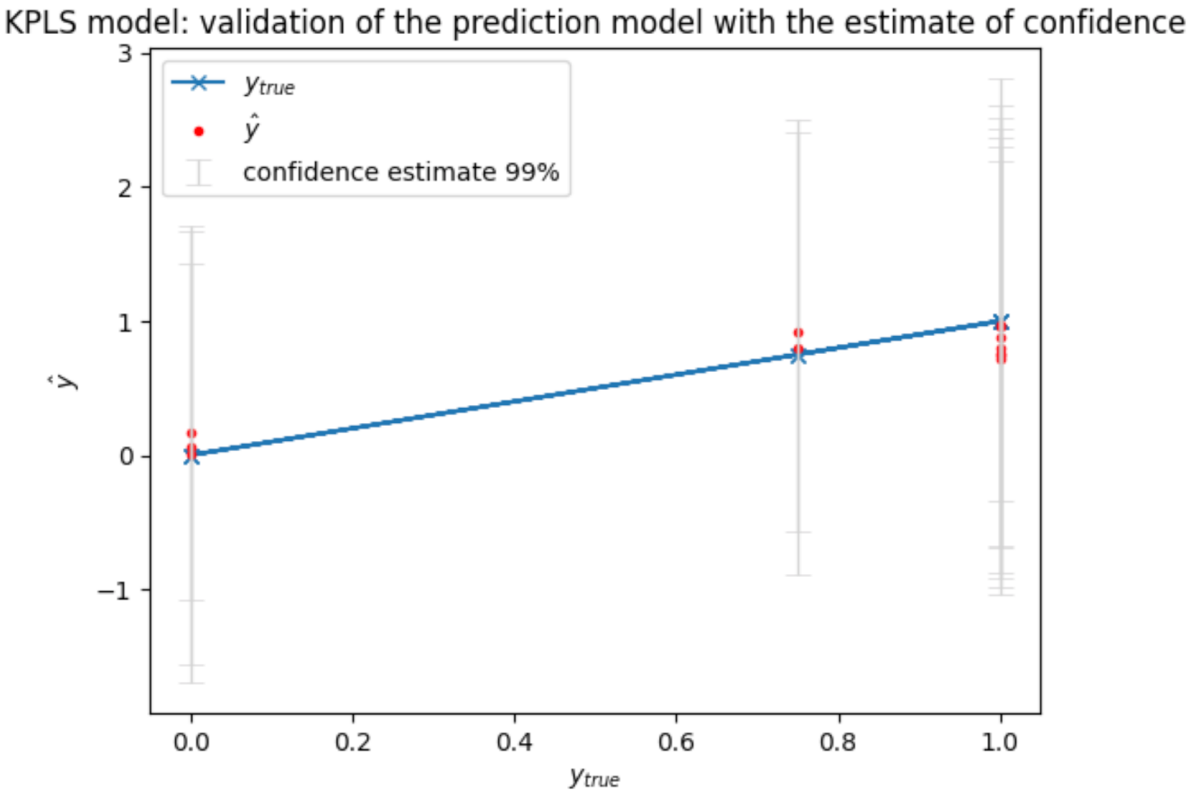

Figure 17: 99% confidence intervals for the KPLS Surrogate model as trained on the given data

We had several limitations in gathering the data for these final plots. Resource constraints and inefficient architecture slowed our ability to process and move the data, making it difficult to get into the Jupyter notebook. Of the data that was collected, while there are more sampled points than from the previous results, there are fewer replicates, and therefore fewer datapoints overall. Several (~7) of the samples have only one or two replicates, compared to three in the previous charts. And while the results in Figure 16 appear to show a reasonably good model for the data, the error bars are quite large as seen in Figure 17. We include a histogram of the resilience score in Figure 15 to demonstrate that it does take on more than four values with this larger data set. The full data set is included in Table 3, and the 3D graph of that data is in Figure 14. The distribution of sample points is the same as in Table 2.

4.3. Lessons Learned

4.3.1. Spot Checks

Section 4.2.3.1 discussed the outcomes of the noise study during this leg of the DRE project. A particularly useful variation on the typical noise study is to perform smaller "spot checks" earlier in the model testing process. For this exercise, several points in the input space were specifically selected for spot checking. Then, a small number of experiments (five to ten) were run at each point. This process helped provide some mini noise studies that could be used to test whether the model was responding to variations in the input space in a logical way.

For the HARMONIE use case, the spot check approach helped to identify several bugs (including fragility in the disruption implementation). In addition, the data was used to hone the configuration for cyber resilience metrics from the environment. Sample data gave early indication of how movement in the input space would translate to changes in the output data, and thus to useful metrics for the system.

4.3.2. Future Surrogate Modeling Work

The resilience score plots demonstrate how sharp the modeling surface is and all of the surrogate models we tested produced very poor results on this data. Future research could include exploring surrogate models that are capable of representing these kinds of discontinuities. It is also possible that cyber security systems will inherently produce these kinds of results because of the nature of the underlying phenomena – cyber systems are, by nature, more often discontinuous than natural phenomena.

Another observation is that the stochasticity of the DOS's effectiveness at blocking the load shed mitigation caused some problems training the surrogate models. Future work would need to better understand how to represent these uncertainties.

The team also explored changing the resilience score as calculated by RevRun in an attempt to improve the relation between the inputs and the outputs, but these changes did little to ameliorate the above issues.

This page left blank

5. OPTIMIZATION

Finally, DRE investigated developing an approach and methodology to help determine optimal choices for system mitigations, designs, and other choices that make the system more secure and resilient. By formulating and incorporating optimization into the DRE pipeline, we can enable the formulation of important system questions to make a variety of more optimized choices. This section will go over the general formulation for this problem as well as some initial proof-of-concept results to demonstrate application.

5.1. Formulation

Optimization is a rather broad and very deep field of research, so it is important to build an appropriate formulation and methodology for DRE that connects to the core capability needs of the work. Going back to the discussion in Section 2, this includes testing mitigations in either emulation or surrogate models, rigorous experimentation that can assess system design or configuration choices, and the ability to determine better options for more resilient systems.

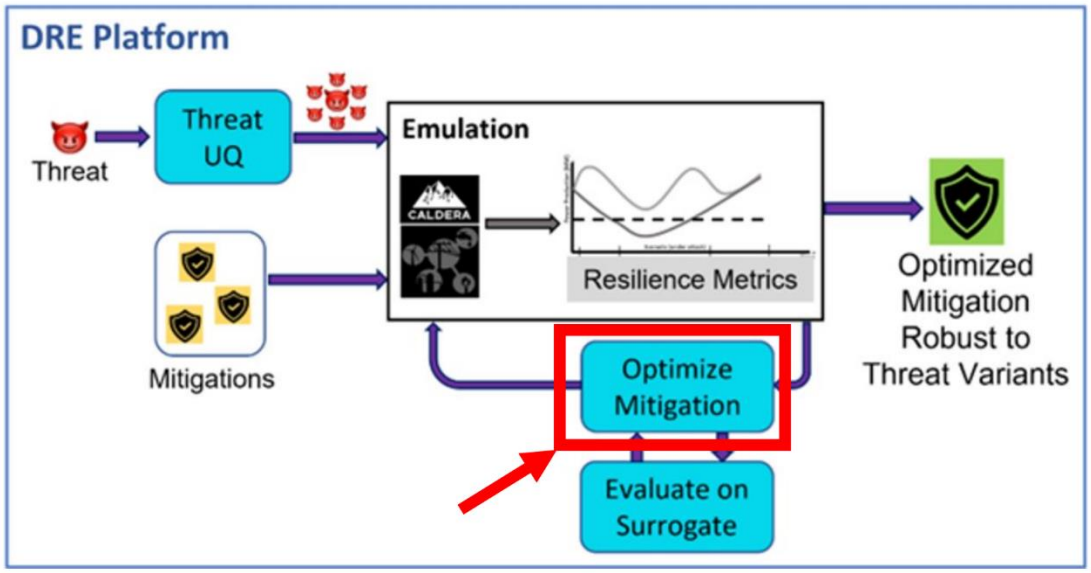

Figure 18: Optimization Placement in DRE Platform, Assisting Choice of Mitigation and Mitigation Settings to Make System More Resilient

When formulating on optimization problem it is important to be able to connect it both to the problem that a system user or stakeholder needs to answer, and to be able to write that problem appropriately into a mathematical statement. For this process, there are several basic components that are needed for the optimization problem:

1. Objective Function (Cost / Utility)
    a. This is where the core question is being asked of the optimization problem. This objective function specifies the connection between values of variables and performance. Generally, we are either minimizing or maximizing the result of this objective function.
2. Decision Variables

a. Decision variables are inputs where choices can be made to optimize the system. These variables are adjusted by the optimization solution to find the best value for each of them. Depending on the type of optimization problem, these may be integer-valued ($x \in \mathbb{Z}$) or real-valued ($x \in \mathbb{R}$).

3. Input Variables
    a. There may be set values that are incorporated as inputs to an objective function that cannot be modified as part of the optimization process. These are input variables, which are separate from decision variables (which can be modified).
4. Constraints
    a. Constraints are additional requirements placed on the optimization problem that specify the acceptable solution space. This is generally in the form of an equality or inequality specification that needs to be met. For instance, one could require that for an optimization problem where there exists a set of solutions, we only pick one solution.

Having different structures for these components may result in different variations of optimization problems. For this formulation for DRE, we are allowing for several different options based on user need, and this could be extended in the future. An important element of this is that in DRE we are providing the framework and foundation, but this needs to be highly applicable to system stakeholders, analysts, and users to be able to formulate their problem into this platform. For this reason, we setup this initial framework for the DRE platform to be mainly in the form of standard optimization problems so that it is easy to specify and incorporate with well-known optimization libraries.

There are two optimization problems that we have setup for an initial foundation for the DRE platform:

1. Choose the optimal system configuration from a set of options (best experiment). This can be structured as an **Integer-Linear Program (ILP)**
2. Find the best parameter choices for configuration options. This can be structured as a **Linear Program (LP)**

An ILP utilizes integer decision variables but may have some real-valued input variables or constraints. This can be expanded into a Mixed Integer-Linear Program (MILP) if the decision variables also include real-valued variables. The standard form for an ILP is as follows:

$$\text{Maximize } \boldsymbol{c^T x}$$

$$\text{Subject to } \boldsymbol{Ax + s = b}$$

$$\boldsymbol{s \geq 0}$$

$$\boldsymbol{x \geq 0, \forall x_i \in x, x_i \in \mathbb{Z}}$$

Where $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$, and $\boldsymbol{A} \in \mathbb{R}^{n \times m}$. In these equations, $\boldsymbol{c}$ is a vector with n separate real-valued numbers that weight the relative cost of each of the variables in $\boldsymbol{x}$. Note, $\boldsymbol{x}$ is the vector of the decision variables for the ILP, so the equation $\boldsymbol{c^T x}$ is just a vector multiplication of weights for each decision variable, which are then summed together (since this is a vector multiplication). The

last three equations in the ILP are for constraints and can be used to specify what the solution space allows as feasible. This may include cost or complexity constraints (if cost is included in the problem formulation), allowable location of solutions, and basically just narrow down which solutions are allowed.

A LP is different from an ILP in that it utilizes real-valued decision variables. This is the type of problem that appears often in system modeling contexts and in parameter optimization. These problems are often solved through linear regression [31], gradient descent [32], the simplex algorithm [33], and similar algorithms to find the best parameter values for the objective function. A LP is structured as follows:

$$\text{Maximize } \boldsymbol{c^T x}$$

$$\text{Subject to } \boldsymbol{Ax \leq b}$$

$$\boldsymbol{x \geq 0}, \forall x_i \in \boldsymbol{x}, x_i \in \mathbb{R}$$

Where $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^n$, and $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. Just like for the ILP case, $\boldsymbol{c}$ is a vector of weights for the relative cost of each variable in $\boldsymbol{x}$, and $\boldsymbol{Ax \leq b}$ and $\boldsymbol{x \geq 0}$ are constraints that specify the allowable solutions. Since an LP specifies a combination of parameters, the type of question being asked for this part of the optimization is centered less around choosing a "best" mitigation and more around choosing appropriate settings for a certain mitigation or system choice.

Note that this entire discussion connects to the previous discussions in this report on emulation and experimentation, uncertainty quantification, and surrogate modeling. The decision and input variables for these optimization problems are the outputs of tools such as RevRun (resilience scores), and the solutions determined by this optimization procedure can feed back into further experimentation. So, when we refer to maximizing objective in these optimization problems, that tends to mean maximizing resilience of the system as scored by RevRun or by surrogate models.

The following section will show what this looks like in practice and discuss how this optimization framework has been implemented in DRE, as well as some possible future extensions.

5.2. Optimization Demonstration

To demonstrate how the optimization framework described in this work can be implemented, a simple implementation and demonstration was developed. This code is meant to be a foundation for future extension and a demonstration of how this toolset can be leveraged to help in system experimentation.

This initial set of demonstration code was developed in Python and uses the SciPy MILP solver [34], which wraps around the HiGHS linear optimization solver [35], though this can be adjusted to allow for other optimization solvers as well. The code itself has two main components:

1. Configuration file
2. Python script

The configuration file is a JSON file that allows for setting up the type of optimization problem to be solved (ILP or LP), what types of inputs to use from the RevRun resilience metrics, how the

weighting in the objective function is formulated, and the solver to utilize (only SciPy is currently supported for now). An example of this configuration file is shown below.

```
{
    "Solver": "Scipy",
    "Problem Type": "Integer",
    "Scoring Type": "group",
    "Weighting Type": "nonuniform",
    "Weights": {"network": 10, "physical_process": 1, "security": 1}
}
```

Figure 19: DRE Optimization Demonstration Configuration File

In the above configuration settings, we have specified as an example that we are solving the ILP problem to choose a best solution from a set. The objective function is using a non-uniform weighting where the weights are manually specified to have the network group resilience score weighted as ten times more important than the other group resilience scores. The scoring type can also be set to only use the final resilience score from RevRun. Note, this setup is fairly limited in that it currently only allows for utilizing the RevRun resilience metric outputs but can be easily extended to incorporate other tools, measurements, or scoring methods.

Additional configuration settings that have been considered for future extensions but are not currently implemented include logging settings, output file settings, constraint options, manual objective configuration, additional metrics to include in objective, cost settings for mitigations, ability to specify minimization versus maximization for the objective, and other integration options.

The Python script that runs the optimization itself is fairly simple but is built to allow for future extension. It pulls the configuration file noted above and then determines the corresponding objective function based on the configuration settings. If the problem type is set to "Integer", it will solve an ILP, otherwise if it is set to "Linear" it will solve a LP. The scoring type allows for choosing different levels of resilience scores from the RevRun output metrics, either "final", "group", or "subgroup". The weighting type can either be "uniform", in which case everything has the same weight in the objective function, or "nonuniform", where it will use the weights manually configured in the "weights" field of the configuration file. The script uses this information to build the objective function and then calls SciPy MILP solver to find a solution. Note that while we use the MILP solver, this is still an ILP in our formulation as we do not include real-valued decision variables (though those can be included in the future). A snippet of the code for this script is shown in Figure 20.

```
61
62      # Other cases can be added, depending on formulation and question to answer
63      match OPT_PROBLEM:
64          case "Integer": #Integer program (IP)
65              optimization_variables = []
66
67              # To add new solver, add new case here
68              match SOLVER:
69                  case "Scipy":
70                      # Using scipy.optimize.milp, w/0 linear variable terms
71                      c = -np.array(c_list)
72                      #print(c)
73                      bounds = scipy.optimize.Bounds(0,1)
74                      constraints = scipy.optimize.LinearConstraint(A=np.ones_like(c),lb=1,ub=1) # Choose 1
                        result
75                      result = scipy.optimize.milp(c=c, integrality=np.ones_like(c), bounds=bounds, constraints
                        =constraints)
76                      # result.x gives model that has best metric score, pull name for results
77                      result_list = result.x.tolist()
78                      model_index = result_list.index(1)
79                      model_name = dre_models[model_index]
80                      print(result.x, model_name)
81                      #print(result.fun)
82
```

Figure 20: Part of Optimization Python Script, Showing Integration with SciPy MILP solver

Note in Figure 20 that at code line 74, a linear constraint is included that specifies only one solution is recorded. This fits into the ILP standard form noted earlier in this section. Note also that at line 71, there is a minus sign included in the cost for the objective function, which can be used to switch between minimizing or maximizing the objective function (depending on problem needs). Since the SciPy MILP solver is actually configured to minimize the problem, and we are defaulting to maximizing resilience scores, it is set with a minus sign here to convert to a maximization problem.

From one of the example datasets of surrogate modeling resilience scores that was run in the spring of 2025, we can show an example of what the above script code looks like when looking at maximizing resilience score from the surrogate model outputs. In a Linux terminal, that output looks as follows:

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0.] sur118
```

Figure 21: Optimization Script Output Example

Figure 21 shows the results of line 80 in Figure 20. Here, we see that one of the surrogate models was chosen as the best, and specifically in this set of runs, it was surrogate model 118. The configuration settings here used the final resilience scores, but when separately weighting the group scores for "network", "physical_process", and "security", we see the results change. Note that since this is an early prototype, the output and user interface here is purely through the configuration settings and python script, so things have to be done manually. This can be extended in the future to make more user friendly and integrate better with the overall pipeline.

This page left blank

6. OTHER ADVANCEMENTS

Aside from the mainline research efforts, the DRE team was able to make contributions to other aspects of the technical capability and methodology involved in rigorous cyber experimentation with DRE.

6.1. Parallelization

Uncertainty quantification, optimization, and creating training sets for surrogate development were expected to require running a relatively large number of experiments, perhaps on the order of hundreds or thousands. Because each experiment can take tens of minutes or more to complete, running them serially would have been impractical.

While RevRun and the underlying tooling were capable of running individual experiments that span multiple nodes, at the beginning of the DRE project, no convenient way existed to parallelize across multiple experiments. One of the accomplishments of the DRE project is the parallelization of RevRun to address this limitation. Currently, parallelization is implemented using Parsl [36], although other viable solutions exist.

The parallelization strategy is simple, enabled by the fact that the problem of running multiple experiments is itself "embarrassingly parallel.". RevRun is configured with a list of experiments and a list of worker nodes. As it runs, it dynamically allocates experiments to workers. The input files for each experiment are sent to a worker, the experiment is remotely executed on the worker, and output files are copied back.

For convenience and robustness, RevRun has several other new features. In addition to output files, RevRun receives logging and other diagnostic information from workers to aid troubleshooting. In the event of random failures, RevRun automatically retries experiments. And RevRun helps automate initial set up of workers by copying user-specified files, creating phenix topologies and experiments, and other necessary operations. Worker node set-up is itself parallelized.

The most important limitation of parallelized RevRun is its dependence on the Parsl parallel scripting library. Parsl is an open-source tool developed by a collaboration of national labs and universities. Although it works with a variety of resource management software, first-class support for compute clusters which lack a resource manager has been deprecated. In the short to medium term, RevRun will continue to function with an older version of Parsl. However, as RevRun's other dependencies and the Python language itself evolve, it is anticipated that incompatibilities will emerge, and an alternative approach to managing remote execution of experiments will need to be sought.

6.2. Multi-Replicate Runs

During data collection from the HARMONIE environment, the team used various graphing tools to help visualize the input space being sampled and the resulting output space produced by the model. While investigating the output surface, the team noticed "spikes" in the surface, where the results were very discontinuous. The question arose – are these points in the space actually discontinuous, or are these data points representative of experiments which, for one reason or another, failed to run and thus produced outlier data?

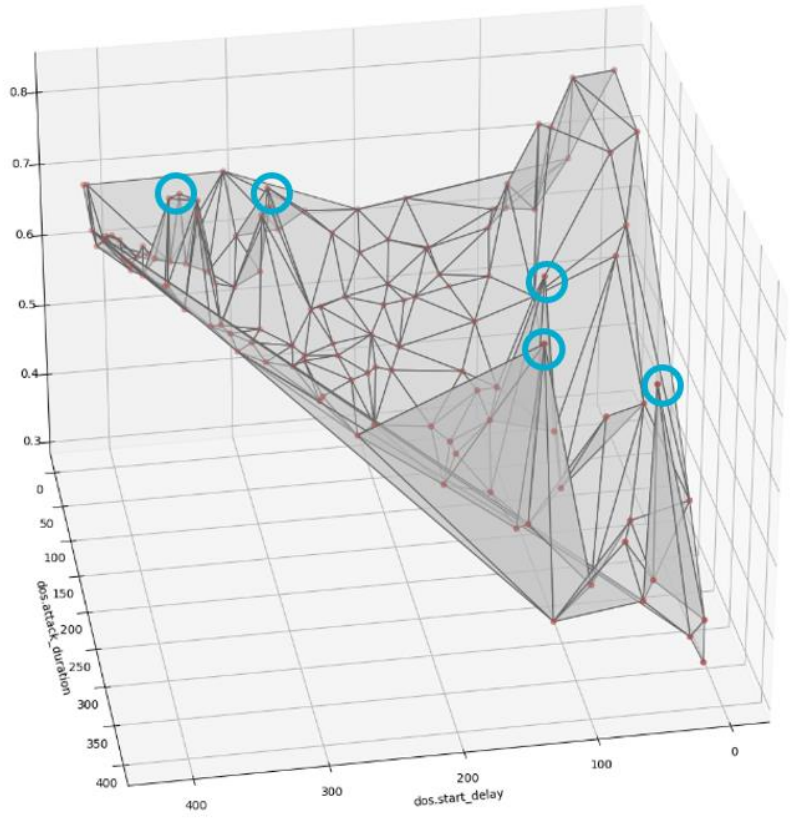

Figure 22: An example output surface with several "spikes" circled in blue.

The way to answer this question was to run multiple experiments (replicates) at each sampled point in the input space. This sampling pattern is something that, in theory, could be easily supported by RevRun, but which would require some modification of the DRE pipeline. An early solution has been implemented, but further work would be required to better integrate this option into the DRE pipeline. Future implementations may also provide options for selecting which replicate to use for calculating metrics for a particular sample point (e.g., automatically detect and discount outliers, select a representative replicate, and/or take an average of all replicates).

6.3. Expanding RevRun Tool Support

For several years, the only tool with RevRun support for collecting network traffic data was packetbeat [30]. However, while performing initial noise studies with the FY24/FY25 use case (HARMONIE), we determined that packetbeat was not sufficient for our needs.

- When packetbeat collects a packet that it cannot parse, it discards that packet from the dataset entirely. This can be a particular challenge when any communication protocols in the network model are atypical or newer protocols. For instance, there is not built-in support for detecting CALDERA command and control beacons. In fact, during FY23 experimentation, an additional tool needed to be run out of sync with the rest of the DRE pipeline in order to collect CALDERA traffic.
- When packetbeat collects TCP traffic, it attempts to automatically detect the sender and receiver and keep these roles consistent throughout the data. This can make consistent

filtering by RevRun difficult, as the roles may change from experiment to experiment depending on when packetbeat was started relative to the start of the TCP "conversation".

- When packetbeat collects TCP traffic, it accumulates packet count over the course of what it considers to be the full TCP "conversation". However, it is difficult to naively decumulate the data, so RevRun must work with the already-accumulated numbers. This can make certain processed data streams and metrics seem higher than they actually are.

In the past, projects using RevRun have often had to implement work arounds to address issues with packetbeat data during post-processing. The DRE project instead chose to build better support for tcpdump via wireshark (or tshark) [37] and filebeat [38] into RevRun so that these tools could instead be used to collect network traffic data.

In general, the use of tshark to collect network traffic data should not replace packetbeat. Both have their place in RevRun's set of supported data collection tools. One of the notable drawbacks to using tshark (or anything using tcpdump) is that strict packet filters are required to limit collected data to manageable volumes.

Integration of filebeat opens up a lot of new possibilities for RevRun data collection, particularly host data which is typically stored in log files with a range of formats and parsing needs.

Currently, tshark and filebeat have a moderate level of support within RevRun. Telling RevRun to collect data using these tools is pretty seamless. However, RevRun cannot yet automatically generate configurations for the in-experiment instances of tshark or filebeat using information gathered from the RevRun master_config.json, like it can with packetbeat and other historically supported tools.

This page left blank

7. DRE APPLICATION

If readers are interested in learning more about DRE, results related to this work have been published externally:

- [14] *Toward Rigorous Experimentation and Analysis in Cyber-Physical System Emulation.* SAND2024-04810C, https://doi.org/10.1109/KPEC61529.2024.10676298, published Sept. 2024.
- [39] *CS 2: Lessons Learned for Study of Uncertainty Quantification in Cyber-Physical System Emulation.* DATAWorks 2024, Institute for Defense Analyses. [Video.] https://www.youtube.com/watch?v=_CoP-1zxPZo&list=PLeZrxAVa0tJk0vNH1wKG3d-GtOjIsbIOv&index=5

Additional resources are available to the Sandia National Labs community, including DRE pipeline demos and tutorial materials. These materials may also be available to those outside Sandia on a case-by-case basis. For access to internal materials, please reach out to the authors.

The remainder of this section gives recommendations for using DRE.

7.1. Example Use Cases

This section is intended to spark ideas and discussion about potential applications for DRE.

Some example questions that DRE could help answer:

- (Validation) Does a given component of the model behave as expected under different parameter conditions?
- (Verification) Can the experimentation infrastructure (e.g., data collection tools) support running experiments under different parameter conditions? Or does the model break down at a certain point?
- How deterministic is the behavior of a given simulated physics model? An emulated cyber system model? An emulated threat scenario? Etc.
- Given the overall variability of a model, can we attribute variability to different potential sources?
- Suppose we understand how a system behaves under normal conditions. How much would conditions need to deviate before we saw "abnormal" behaviors?
- How continuous is system behavior over a range of input parameters?
- Which system configuration is most robust across variations of a particular scenario? How robust is a particular detection tool to variations in threat behavior?
- In X trials, how many times does the modeled system "succeed" in its goals?
- Given a set of uncertainties in a scenario, which uncertainties most significantly impact results? Where would resources to buy down uncertainties be most effective?
- Suppose we are using a system model to help answer some key question about the system. How confident are we in our answer? What is the range of potential answers?

Example cases where DRE would not be as useful:

- Models which do not allow for parallelized runs will limit the number of samples that DRE can collect.
- Environments with very limited data storage and processing capabilities may not be able to use DRE effectively.
- DRE has only been tested with continuous or pseudo-continuous variables. Truly discrete or categorical variables may need to be sampled using different approaches.
- Suppose you want to study the effects that different implementations of a tool will have on your system. If inherent variability in the base system model is very high, it may be difficult to detect impacts caused by variations in tool implementation.

7.2. Configuring the Pipeline

The following section describes, at a high level, the work that a user must perform in order to configure the DRE pipeline. There is more extensive written and audio/visual instruction available – please reach out to the authors for access.

7.2.1. Analysis Plan

At its core, DRE is a method for completing a large number of RevRun experiments. The same process for planning any RevRun-based analysis is applicable here. The RevRun User Guide [13] includes an appendix with a series of questions useful to planning an end-to-end RevRun analysis. It is recommended that the user review this planning guide, particularly when it comes to considering the goals of the analysis and any data collection needs. At this time, you will also want to consider which parameters will be varied by DRE, usually as part of a noise study, sensitivity study, or uncertainty quantification study.

7.2.2. SCEPTRE Model

The design of the SCEPTRE model itself is left to the reader, with a couple special notes. These notes assume the reader is at least somewhat familiar with SCEPTRE model building and configuration.

1. The point of using DRE is to vary certain input parameters in the experiment environment. Make sure these parameters are appropriately exposed such that they can be configured pre-experiment. For example, they may be part of a script that can be written pre-experiment and runs during the experiment, part of a ScOrch (SCEPTRE scenario orchestrator) component or phenix app that can be configured pre-experiment, etc.
2. As noted in Section 7.2.1 above, RevRun is an integral part of the DRE pipeline, especially for data collection and analysis. Make sure that the SCEPTRE environment is configured such that RevRun can collect any data relevant to the overall goals of the analysis. This may involve special care to expose certain sources of data that may not otherwise be exposed.
3. If you are already using ScOrch as part of your SCEPTRE environment, be sure that you are using one run only, and no loops. Typically, these options in ScOrch are used for efficiently running multiple experiments, but RevRun will already be managing this for you, so there is no reason to configure this yourself. Thus, any of your own components that are currently under the "loop" level may be brought out to the "run" level.

7.2.3. RevRun Setup

You can largely follow the RevRun User Guide for how to setup a new RevRun Project folder for your experiment, with a couple special notes. These notes assume the reader is at least somewhat familiar with RevRun configuration.

1. You do not need to manually set baseline_id and scenario_id in the master config file for your project. The DRE driver will take care of this for you.
2. You do not need to manually configure every individual RunID folder. Again, the DRE driver will take care of this for you.
3. You do need to set up a folder in your project called "baseline". The DRE driver will assume that this folder is configured to support a baseline experiment (i.e., optimal operations, or the base scenario against which all other experiments will be compared). But the DRE driver will also assume this folder contains all other files which may be required for non-baseline experiments. For example, the baseline folder may include an attack script which is not actually run in the baseline scenario, but which will be run by other experiments.

7.2.4. Dakota Configuration

Follow the Dakota Reference Guide to write a Dakota input file for your project. Much of this file can be copied from past analyses and tweaked for your own use.

1. Set the number of samples you want to generate. Remember that you will end up with this number of experiments plus one (the baseline).
2. Configure your variable distributions. The Dakota guide has many options here. Pay special attention to the names given to these parameters.
3. Set the command that you wish Dakota to run in order to start the DRE driver. The main driver should be the absolute path to driver_one_phase.py in a local clone of the DRE repo. The additional path given should be the path to your RevRun project folder.

7.2.5. DRE Driver

The DRE driver will be used to tie together the Dakota configuration and the RevRun project you set up. There are three main pieces of the driver: the main driver code, the base Configuration Manager, and the custom Configuration Manager. Section 3.1 described the purpose of each of these pieces in general, but here is some additional practical detail:

1. The main driver code is driver_one_phase.py, this is what Dakota will actually run. The only thing you'll need to modify here usually is to import the appropriate class from your custom Configuration Manager file.
2. The Configuration Manager Base is boilerplate – you should not need to ever change anything here.
3. The custom Configuration Manager is primarily where you need to write a custom "setup project" function. The purpose of this function will be to take the parameters generated by Dakota and put them where they need to go within the RevRun project folder. This typically includes writing the parameters into a script or configuration file to be read or otherwise used by some part of the configured RevRun project. Your custom function can look like anything and does not need to be generalized in any way if you do not want it to be.

7.3. Ready to Run

Before running any analysis, ensure there are no other experiments in phenix (use the phenix GUI or command line interface to verify). This is because the data extraction process relies on there being only one active minimega namespace – otherwise image mounting and management network reading can get somewhat confused.

If you plan to setup a long set of experiments and just let them go, you may need to consider tools like screen or tmux in order to keep your session alive for the duration of the experiments.

If you haven't already, create a directory somewhere with the purpose of holding Dakota-related files. Dakota will generate a bunch of files as it runs, so it's useful to run Dakota from within a unique directory to keep everything in one place. Put your created Dakota input file in this directory.

Start the DRE pipeline by running Dakota and pointing to your input file. Then, you should be able to just walk away.

When the analysis is complete, results can be found in your Dakota analysis directory at dakota_tabular.dat .

8. FUTURE WORK

A variety of future efforts have been identified throughout this work. Several have already been alluded to elsewhere in this document, but this section will summarize them.

First, although the DRE project made good strides toward an approach to surrogate modeling of cyber-physical emulation environments, some big questions remain.

- Are there other approaches to surrogate modeling that could be considered that might work better in this context?
- How would someone go about identifying a "good" surrogate model architecture based upon their data and needs?
- How simple or complex should that architecture be?

Several other questions arise related to data generated to train the surrogate model. For the purpose of demonstration, in this work, the team used considerable a priori knowledge to tune our data collection and metrics in order to create a smooth response surface for the surrogate models to learn. This approach might represent a best-case scenario, where modelers have a priori system knowledge available to heavily guide configuration choices. However, in the average use case, this kind of data may not be available.

- How would a user approach surrogate model development differently?
- Is there an efficient way to determine whether a usable surrogate model is even possible for the given system/scenario model and use case?
- How would a user select the "right" data streams and/or configure RevRun metrics on which to train the surrogate model?
- Is there a generalizable way to approach this problem, or will it be very model-specific?

There is quite a bit of opportunity here for further research on surrogate modeling.

Second, integration of optimization capabilities into the DRE pipeline are still in very early stages. At the close of the DRE project, there is a simple proof-of-concept application of an optimization process to some of the data collected during the project. This application is highly extensible, and future work should investigate how to integrate it more completely into the rest of the DRE pipeline. This may lead to additional development on the pipeline itself in order to support efficient optimization use cases.

Third, the DRE project identified several improvements that could be made to the RevRun software, which is a key piece of the DRE pipeline. Some of these improvements were actually implemented during the DRE project. However, several other recommended improvements require larger architectural updates to the RevRun software. It is outside the scope of this document to discuss these recommendations in detail, but some future directions for the RevRun software include:

- Improved flexibility of RevRun to leverage arbitrary data collection tools. The RevRun software was originally developed to only use a handful of very specific data collection tools for all experiments. Some extensions have been made over time to support additional specific tools. In fact, such extensions were made as part of the FY25 work under DRE. However, a better future state would be a "bring your own tools" model, where RevRun

might have a limited set of baked-in tools but would also allow users to easily integrate their own unique tools if needed.

- Improved support for parallelization and other efficiency tools and frameworks to wrap around RevRun. Under the DRE project, Parsl was integrated with RevRun to enable more coordinated parallelization. However, the team is aware that this may not be a permanent solution, and future work may need to identify other methods for coordinating parallelization of RevRun experiments. In addition, there may be other tools and frameworks to help support more efficient execution of the DRE pipeline, and future work may need to consider whether specific improvements need to be made to RevRun (or the rest of the DRE pipeline) in order to be amenable to these kinds of frameworks.
- Now that RevRun is being run at a much larger scale compared to historical use cases, there is much more data to process and calculate metrics on. This has highlighted the need to reevaluate the RevRun processing and metrics code to look for more efficient architecture and implementation choices to shorten runtimes. In addition, developers should consider options for more threading/parallelization at later points in the RevRun process (not only during initial experimentation, but also during data processing).
- Several challenges arose toward the end of the project related to large data storage and movement. While implementing these solutions may be outside the scope of the RevRun software, it may be useful for developers to consider some approaches to these problems which could be recommended to future users, or whether additional helper tools need to be developed in order to better support large-scale data storage and movement for use in RevRun.

Finally, the DRE pipeline does not need to be used in isolation. Other work has been ongoing at Sandia in the area of Data-Driven Modeling and Experimentation. LDRDs such as MARIAH (PI Meg Sahakian, joint funded by NSP Cyber and RES FY24-FY26) and MMAREJBLIGE (PI Ryan Adams, funded by NSP Cyber FY23-FY24, SAND2024-12736) could potentially be integrated with DRE as part of a larger experimentation pipeline. Future work should consider larger, multi-tool demonstrations to solve new types of questions that might be posed about cyber experimentation environments.

9. CONCLUSION

Over the past three years, the DRE team has made significant strides toward supporting more scientifically rigorous emulation-driven cyber experimentation. Historically, Emulytics™ has not been broadly used to perform large experiments with highly statistically precise results. DRE and projects like it are beginning to build the capabilities and methodologies needed to support this sort of use case. While this may not often be the desired use case for an Emulytics™ environment, this project did highlight the risk of limiting experimentation to only a small handful of experiments or considering the results from only one angle. Without providing additional context to results (e.g., data distributions, confidence levels, noise studies), analysts run the risk of missing a great deal of context when interpreting results from Emulytics™ environments, and this context is crucial for certain use cases, such as decision support and cost-benefit analysis.

Many of the capabilities described in this paper are ready to be integrated today, and the DRE team remains interested in working with existing modeling and experimentation efforts to help enhance analysis plans. In addition, the DRE project identified several avenues for future research and development, both on the DRE pipeline and on other enabling tools in the tool stack. Continued work in this area will only improve our ability to make better model-informed decisions, with the potential to have significant impact on future planning and experimentation across a variety of critical systems.

This page left blank

REFERENCES

[1] Dragos, Inc. (2017). CRASHOVERRIDE Analysis of the Threat to Electric Grid Operations. Technical Report Version 2.20170613, Dragos Inc., Hanover, MD. [Online]. Available: https://www.dragos.com/wp-content/uploads/CrashOverride-01.pdf

[2] Department of Energy Office of Cyberseucirty, Energy Security, and Emergency Response (CESER). (2021). Colonial Pipeline Cyber Incident. DOE CESER. Washington, DC. [Online]. Available: https://www.energy.gov/ceser/colonial-pipeline-cyber-incident

[3] Dragos, Inc. (2022, April). CHERNOVITE's PIPEDREAM Malware Targeting Industrial Control Systems (ICS). Dragos Inc., Hanover, MD. [Online]. Available: https://www.dragos.com/blog/industry-news/chernovite-pipedream-malware-targeting-industrial-control-systems/

[4] Colonial Pipeline, "Media statement update: Colonial pipeline system disruption," May 2021. [Online]. Available: https://www.colpipe.com/news/press-releases/media-statementcolonial-pipeline-system-disruption

[5] Center for Strategic & International Studies (CSIS). "Significant Cyber Incidents". May 2025. [Online]. https://www.csis.org/programs/strategic-technologies-program/significant-cyber-incidents

[6] Sandia National Laboratories. Emulytics™. https://www.sandia.gov/emulytics/

[7] Sandia National Laboratories. minimega. https://www.sandia.gov/minimega/

[8] Sandia National Laboratories. SCEPTRE: A Cyber-Physical Emulation Capability. Paper in preparation.

[9] Sandia National Laboratories. SCEPTRE | An emulation capability for Industrial Control Systems. 2023. https://sandialabs.github.io/sceptre-docs/

[10] M. Galiardi, A. Gonzales, J. Thorpe, E. Vugrin, R. Fasano, and C. Lamb. (2020, August). Cyber Resilience Analysis of SCADA Systems in Nuclear Power Plants. Presented at 28th Conference on Nuclear Engineering and Joint with the ASME 2020 Power Conference. [Online]. Available: https://asmedigitalcollection.asme.org/ICONE/proceedings-abstract/ICONE2020/83778/V002T08A003/1088626

[11] A. Pinar, T. Tarman, L.P. Swiler, J. Gearhart, D. Hart, E. Vugrin, G. Cruz, B. Arguello, G. Geraci, B. Debusschere, S. Hanson, A. Outkin, J. Thorpe, W. Hart, M. Sahakian, K. Gabert, C. Glatter, E. Johnson, S. Punla-Green. (2021). Secience & Engineering of Cyber Security by Uncertainty Quantification and Rigorous Experimentation (SECURE) Handbook. Sandia Report SAND2021-11459R.

[12] B.M. Adams, W.J. Bohnhoff, et al. (2021, November). Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.15 User's Manual. Sandia Report SAND2021-5822.

[13] RevRun team. RevRun: The REsilience VeRification UNit. User Guide. September 2024.

[14] J.E. Thorpe, N.J. Krakauer, J.A. Stephens, H. Stroble, B. Williams. Toward Rigorous Experimentation and Analysis in Cyber-Physical System Emulation. SAND2024-04810C, https://doi.org/10.1109/KPEC61529.2024.10676298, published Sept. 2024.

[15] Malwarebytes. (2024) Computer worm. [Online]. Available: https://www.malwarebytes.com/computer-worm

[16] MITRE Corporation, "CALDERA," 2021. [Online]. Available: https://www.mitre.org/research/technology-transfer/open-sourcesoftware/caldera

[17] M. Liljenstam, D.M. Nicol, V.H. Berk, R.S. Gray. Simulating Realistic Network Worm Traffic for Worm Warning System Design and Testing. *WORM'03: Proceedings of the 2003 ACM workshop on Rapid Malcode*. ACM. 27 October 2003.

[18] D.M. Nicol, W.H. Sanders, K.S. Trivedi. Model-Based Evaluation: From Dependability to Security. In *IEEE Transactions on Dependable and Secure Computing*. Vol 1, No 1. IEEE. January-March 2004.

[19] D. Jin, D.M. Nicol, G. Yan. An Event Buffer Flooding Attack in DNP3 Controlled SCADA Systems. In *Proceedings of the 2011 Winter Simulation Conference*. IEEE. 2011.

[20] S. Jauhar, B. Chen, W.G. Temple, X. Dong, Z. Kalbarczyk, W.H. Sanders. D.M. Nicol. Model-Based Cybersecurity Assessment with NESCOR Smart Grid Failure Scenarios. In *IEEE 21st Pacific Rim International Symposium on Dependable Computing*. IEEE. 2015.

[21] L. Jia, R. Alizadeh, J. Hao, G. Wang, J.K. Allen, F. Mistree. A rule-based method for automated surrogate model selection. *Advanced Engineering Informatics*. Vol 45. 2020.

[22] C. Menghi, S. Nejati, L. Briand, Y.I. Parache. Approximation-Refinement Testing of Compute-Intensive Cyber-Physical Models: An Approach Based on System Identification. *IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE/ACM. 2020.

[23] S. Asgari, H. Moazamigoodarzi, R.J. Tsai, S. Pal, R. Zheng, G. Badawy, I.K. Puri. Hybrid surrogate model for online temperature and pressure predictions in data centers. *Future Generation Computer Systems*. Vol 114. 2021.

[24] H.H. Nguyen. Modeling and Analysis of Trustworthy Systems using Extensions of Network Reliability. *Doctoral dissertation*. University of Illinois Urbana-Champaign. 2022.

[25] X. Qin, Y. Xian, A. Zutshi, C. Fan, J.V. Deshmukh. Statistical Verification of Cyber-Physical Systems using Surrogate Models and Conformal Inference. *ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. ACM/IEEE. 2022.

[26] M. Rahman, A. Khan, S. Anowar, M. Al-Imran, R. Verma, D. Kumar, K. Kobayashi, S. Alam. Leveraging Industry 4.0 – Deep Learning, Surrogate Model and Transfer Learning with Uncertainty Quantification Incorporated into Digital Twin for Nuclear System. *Preprint*. 30 September 2022. https://doi.org/10.48550/arXiv.2210.00074

[27] 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.

[28] A. Delavari, I. Kamwa and P. Brunelle, "Simscape Power Systems Benchmarks for Education and Research in Power Grid Dynamics and Control," *2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)*, Quebec, QC, Canada, 2018, pp. 1-5, doi: 10.1109/CCECE.2018.8447645.Other surrogate modeling references

[29] Elastic. (n.d.). Elasticsearch. Elastic. https://www.elastic.co/guide/en/elasticsearch/reference/index.html

[30] Elastic. (n.d.). Packetbeat. Elastic. https://www.elastic.co/docs/reference/beats/packetbeat

[31] H. Phillips, "An Introduction to Machine Learning in Python: Simple Linear Regression," Medium. Accessed: Sep. 18, 2025. [Online]. Available: https://medium.com/@hunter-j-phillips/a-simple-introduction-to-regression-and-machine-learning-in-python-5e6bd76b0bf8

[32] H. Phillips, "A Simple Introduction to Gradient Descent," Medium. Accessed: Sep. 18, 2025. [Online]. Available: https://medium.com/@hunter-j-phillips/a-simple-introduction-to-gradient-descent-1f32a08b0deb

[33] G. B. Dantzig, "Origins of the simplex method," in *A History of Scientific Computing*, New York, NY, USA: Association for Computing Machinery, 1990, pp. 141–151. [Online]. Available: https://doi.org/10.1145/87252.88081

[34] "SciPy MILP API Reference." Accessed: Sep. 18, 2025. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.milp.html

[35] Huangfu, Q., Galabova, I., Feldmeier, M., and Hall, J. A. J. "HiGHS - high performance software for linear optimization." https://highs.dev/

[36] Yadu Babuji, Anna Woodard, Zhuozhao Li, Daniel S. Katz, Ben Clifford, Rohan Kumar, Luksaz Lacinski, Ryan Chard, Justin M. Wozniak, Ian Foster, Michael Wilde and Kyle Chard. "Parsl: Pervasive Parallel Programming in Python." 28th ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC). 2019. 10.1145/3307681.3325400

[37] The Wireshark Team. Wireshark Network Protocol Analyzer. [Online]. Available: https://www.wireshark.org/

[38] Elastic. (n.d.). Filebeat. Elastic. https://www.elastic.co/docs/reference/beats/filebeat/

[39] *CS 2: Lessons Learned for Study of Uncertainty Quantification in Cyber-Physical System Emulation.* DATAWorks 2024, Institute for Defense Analyses. [Video.] https://www.youtube.com/watch?v=_CoP-1zxPZo&list=PLeZrxAVa0tJk0vNH1wKG3d-GtOjIsbIOv&index=5

This page left blank

DISTRIBUTION

Email—Internal

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

This page left blank

This page left blank

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.