SANDIA REPORT
SAND2021-11719
Printed September 2021

Sandia National Laboratories

Science and Engineering of Cybersecurity by Uncertainty quantification and Rigorous Experimentation (SECURE) Final Report

Ali Pinar, Thomas Tarman, Laura Swiler, Jared Gearhart, Derek Hart, Eric Vugrin, Gerardo Cruz, Bryan Arguello, Gianluca Geraci, Bert Debusschere, Seth Hanson, Alexander Outkin, Jamie Thorpe, William Hart, Meghan Sahakian, Kasimir Gabert, Casey Glatter, Emma Johnson, and She'ifa Punla-Green

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.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

This report summarizes the activities performed as part of the Science and Engineering of Cybersecurity by Uncertainty quantification and Rigorous Experimentation (SECURE) Grand Challenge LDRD project. We provide an overview of the research done in this project, including work on cyber emulation, uncertainty quantification, and optimization. We present examples of integrated analyses performed on two case studies: a network scanning/detection study and a malware command and control study. We highlight the importance of experimental workflows and list references of papers and presentations developed under this project. We outline lessons learned and suggestions for future work.

ACKNOWLEDGEMENTS

This work was funded by the Laboratory Directed Research and Development (LDRD) program at Sandia National Laboratories. The authors gratefully acknowledge the support of the LDRD program and Leigh Cunningham. We acknowledge the vision of Zach Benz, the initial program manager who has been a strong advocate and visionary for SECURE. We grateful for the support of Jennifer Gaudioso, Richard Griffith, and Heidi Ammerlahn for serving as Director Champions for this project. We acknowledge the help of our colleagues supporting related cyber emulation initiatives, including Michael Stickland, Steven Elliott, David Fritz, and Chris Symonds. We also acknowledge the support of various team members who contributed to this project over the years, including Anya Castillo, Jonathan Crussell, Christian Reedy, Erin Acquesta, Trevor Rollins, Tim Schulz, Patricia Schulz, Justin Doak, Vincent Urias, Cynthia Phillips, Carl Laird, John Siirola, Matthew Todd Farrell, Evan De La O, Nicholas Pattengale, Bethany Nicholson, Fulton Wang, Bei-Bei Chen, Caleb Morse, and Daniel Mayes. Our external partnerships with Umit Catalyurek, Santanu Dey, Yan Wang, Yusuf Ozkaya, Luka Malashkhia from Georgia Institute of Technology, John Mitchell from Rensselaer Polytechnic Institute, Jonathan Eckstein from Rutgers University, Kate Davis, Ana Goulart, Patrick Wlazlo, Hao Haung, and Abhijeet Sahu from Texas A&M University, and Matt Bishop, Miriam Heller and Karl Levitt from University of California at Davis have been a big part of our success. We also thank members of the Internal Advisory Board and other Sandians, including Brian Van Leeuwen, Todd Jones, Patty Hough, Habib Najm, Michael Eldred, Philip Kegelmeyer, Bill Miller, John Naegle, Scott Stephens, Kevin Hulin, Jason Stamp, and Jean-Paul Watson. Finally, we thank the members of the SECURE External Advisory Board, including William Pike, Stephen Schwab, Terry Benzel, Brian Gattoni, Herb Klumpe, Xiaochuan Luo, the late Eugene Litvinov, Steven Gabriel, and Youssef Marzouk.

CONTENTS

1. Introduction 13
   1.1. Overview and evolution of research 13
   1.2. Research Elements 14
      1.2.1. Research Element 1: Predictive Cyber Emulation 14
      1.2.2. Research Element 2: Uncertainty Quantification 14
      1.2.3. Research Element 3: Adversarial Optimization 14
   1.3. Integration exemplars 14
   1.4. Workflows 15
   1.5. Outline of report 15
2. Emulytics Thrust Area 16
   2.1. Emulation and Simulation Models 16
   2.2. Markov Models 17
   2.3. Mathematical Models 19
      2.3.1. Traffic Generation 22
3. Uncertainty Quantification Thrust Area 23
   3.1. Dimension Reduction 23
   3.2. Discrete PCE 23
   3.3. Multifidelity UQ 24
   3.4. Multifidelity Estimation with Replicates 25
4. Adversarial Optimization 29
   4.1. *N-k* Worst Case Analysis 29
   4.2. Network segmentation 31
   4.3. Optimal Sensor Placement 32
   4.4. Robust Optimization 33
5. End-to-End Exemplar 35
   5.1. Case Study: Command and Control (C2) 35
   5.2. Case Study: Scanning/Detection 36
   5.3. Case Study: SCADA Network/Power Grid Impacts 37
   5.4. Integrating the various pieces: Markov Model 39
6. Cross-cut: Verification and Validation 43
   6.1. Verification 43
   6.2. Validation 44
7. Cross-cut: Tools 46
   7.1. SCORCH 46
   7.2. Minimega/SCEPTRE 46
   7.3. Elasticsearch 46
   7.4. PAO/Pyomo 47
   7.5. Dakota 47
8. Recommended Workflow 49
9. Project Metrics 55
   9.1. Publications & Presentations 55
      9.1.1. Papers completed 55
      9.1.2. Papers under review 56
      9.1.3. Technical Presentations 56

9.2. Conference & Workshop Organization 57
9.3. Mentoring and Training 57
9.4. Team Building & Partnerships 58
10. Summary and project legacy 59
11. References 62
Appendix A. Command and Control (C2) Case Study 65
A.1. Overview 65
A.2. Analysis Scenario 65
A.3. C2 Emulation Environment 67
A.4. Emulation Verification using Telemetry 70
A.5. Mathematical Model 71
A.5.1. Comparison of Mathematical Model and Emulation Model Results 74
A.6. Analysis and Uncertainty Quantification 76
A.6.1. PCE Sampling 76
A.6.2. Multi-Fidelity UQ 77
A.7. Conclusions 81
A.8. Appendix A References 81
Appendix B. Scanning and Detection Case Study 82
B.1. Overview 82
B.2. Scenario 82
B.3. Topology 83
B.4. Nmap 83
B.5. Detection 85
B.6. Tools 85
B.6.1. Mathematical model 85
B.6.2. ns-3 87
B.7. Emulation experiments using Scorch 88
B.7.1. Data collection 89
B.8. Experimental methods 89
B.8.1. Experiment reproduction 89
B.9. Verification 94
B.10. Validation 94
B.11. Optimal segmentation 95
B.12. Appendix B References 95
Appendix C. SCADA Network/Power Grid Impacts 97
C.1. Overview 97
C.2. CRASHOVERRIDE 97
C.2.1. CRASHOVERRIDE Configuration 97
C.3. TAMU Topology 97
C.4. Power Grid Impact Studies 99
C.4.1. UQ Study 99
C.5. Segmentation Study 100
Appendix D. Optimal Sensor Placement Model 103
D.1. Model Elements 103
D.1.1. Attack Graphs 103
D.1.2. Defender Model 104

D.1.3. Attacker Model 104
D.1.4. Physical System Model 105
D.2. Math Model 105
D.2.1. Model Reformulation 107
D.3. Example 109
D.4. Appendix D References 112

LIST OF FIGURES

Figure 2-1: End-to-end threat scenario based on APT-3 and CRASHOVERRIDE 16
Figure 2-2: Translating the end-to-end threat scenario to a Markov state transition diagram 17
Figure 2-3: Markov analysis results showing attacker time to success and success probabilities, depending on defender capabilities 18
Figure 2-4: Intrusion detection comparison between C2 mathematical and emulation models 20
Figure 2-5: State transition diagram for scanning/detection mathematical model 21
Figure 2-6: Port discovery comparison between mathematical and emulation models 21
Figure 3-1. Cost ratio between MF estimator and MC as a function of the low-fidelity number of replicas 26
Figure 3-2. Confidence intervals for several multifidelity estimators with and without optimization of the number of low-fidelity replicas 27
Figure 4-1. Comparison of load shed for the IEEE-118 bus system for random and worst-case attacks. 30
Figure 4-2. Worst-case attack before network segmentation and after network segmentation. Segmentation allowances: 1 extra balancing authority segment, 2 extra control center segments, 5 extra substations segments. Attacker budget: 5 subnets 31
Figure 4-3. Notional attack graph for the WECC 9 bus system. Optimal placement decisions for a budget of 7 and a load shed threshold of 1.25MW are shown by orange hourglass icons. The red arrows show the path that has the highest probability of evading detection (24 percent). 33
Figure 5-1: End-to-end threat scenario based on APT-3 and CRASHOVERRIDE 35
Figure 5-2: Notional C2 exemplar system representation 36
Figure 5-3: Typical SCADA network topology for scanning and detection 37
Figure 5-4: Optimal SCADA network segmentation workflow 38
Figure 5-5: Results showing benefit of optimal segmentation 38
Figure 5-6: UQ study, showing regression for 95th percentile (representing worst-case tail outcomes) 39
Figure 5-7: Baseline Markov model of APT-3 threat scenario 40
Figure 5-8: Change in Ready state (state 9) residence probability vs. Ready state detection probability 41
Figure 5-9: Sensitivity analysis, showing greatest benefit in Ready state detection relative to detection in other states 41
Figure 8-1. Spectrum of cyber model fidelity, ranging from actual system to simulation testbeds. 50
Figure 8-2. Recommended workflow for cyber modeling suggested by the SECURE project 51
Figure 8-3. Hierarchical validation for a cyber system, starting with validation of individual attack steps at the bottom and proceeding to validation of the full attack at the top. 53

APPENDIX A

Figure A 1. Multi-stage attack considered by SECURE 65
Figure A 2: Notional C2 exemplar system representation 66
Figure A 3: C2 Exemplar Emulation Environment 68
Figure A 4. Mapping between water filtration and intrusion detection systems. 71
Figure A 5. Average number of alerts over time, for the emulation and mathematical models (non-rate limited case). 73
Figure A 6. Probability of having at least k alters by time period 16, for the emulation and mathematical models (non-rate limited case). 73

Figure A 7. Average number of alerts over time, for the emulation and mathematical models (rate limited case). Note how the number of alerts levels off. 74
Figure A 8. Comparison of the emulation and mathematical models CDFs for the probability of exceeding a given number of alerts by time period 9. 75
Figure A 9. p-values for the K-S by time period. 75
Figure A 10. C2 MFUQ estimator. 77
Figure A 11. Scatterplots of total number of alerts at timesteps 1, 5, and 10 for 40 parameter samples for the emulation and mathematical models. 79
Figure A 12. Correlation squared between the emulation and mathematical models at time steps 1, 5, and 10. 79
Figure A 13. Variance (left) and coefficient of variation (right) for the total number of alerts. 80
Figure A 14. Prediction of mean number of alerts and associated confidence interval for single (MC) and multi-fidelity (MF) estimators 80

APPENDIX B
Figure B 1. SECURE end-to-end threat scenario, with SCADA network studies highlighted 82
Figure B 2. Notional SCADA network topology for scanning/detection study 83
Figure B 3. Nmap protocol operations while scanning open, closed, and filtered hosts 84
Figure B 4. Snort "sfportscan" rule 85
Figure B 5. Mathematical state transition diagram 86
Figure B 6. Port discovery analysis (mathematical model and `minimega` emulation) 86
Figure B 7. Detection times 87
Figure B 8. ns-3 model for scanning/detection 88
Figure B 9. Port Discovery Statistical Test Results for Deterministic Case 91
Figure B 10. Port Discovery Statistical Test Results for Deterministic Case 93

APPENDIX C
Figure C 1. TAMU cyber topology 98
Figure C 2. 2000-bus power model 99
Figure C 3. UQ Experiment Results with Quantile Lines for Normalized Loss of Load 100
Figure C 4. Segmentation Results 101

APPENDIX D
Figure D 1. Test case attack graph. Solid lines indicate attack edges. Dashed lines indicate reward edges. 109
Figure D 2. Optimal attack with no interdictions ($\pi_{OPT} = 0.629$). 111
Figure D 3. Optimal interdiction strategy and attack for a defender budget $b = 12$ ($\pi_{OPT} = 0.140$). Interdicted edges are shown in green. Attacked edges are bold. 111
Figure D 4. Optimal probability of evasion versus defender budget. 112

LIST OF TABLES

Table 2-1: C2/Markov state 6 transition probabilities (from emulation and mathematical models) ..17
Table 2-2: ID RTUs/Markov state 8 transition probabilities (from emulation and mathematical models) 18
Table 2-3: Defender capabilities 19
Table 3-1. Comparison between the single fidelity MC estimator based on `minimega` data only and the MF estimator based on the additional math model evaluations. 25
Table 3-2. MF estimator data for the cases with and without optimization of the low-fidelity replicas. 28

Table A 1. Key variables of interest for the C2 study. 68
Table A 2. Main effects from PCE analysis for the number of total alerts and false positives at time period 5. 77
Table A 3. Key parameters of interest for MFUQ study 78

Table C 1. Parameters of UQ Study 1 100

Table D 1. Input data for example network. 110

This page left blank

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
ATT&CK	A framework and knowledge base developed by MITRE Corporation for adversary tactics and techniques based on real-world observations. ATT&CK is used as a foundation for the development of specific threat models.
C2	Command and control. Used to refer to communications between malware that is installed on a compromised network and an Internet-connected server that is used to issue commands to control the malware.
CRASHOVERRIDE	A malware framework that attacks RTUs on power grid networks; presumed to have been used in the 2016 cyberattack on the Ukraine power system.
CSE	Computational Science and Engineering
DC-OPF	DC Optimal Power Flow
Emulytics	A holistic approach to system emulation and analytics: https://www.sandia.gov/emulytics/
EBM	Emulation-Based Model
IDS	Intrusion Detection System
Nmap	An open-source utility to scan for and find hosts and services
PAO	Python Adversarial Optimization
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition system. Typically refers to an industrial control system. In our use cases, it is the control system managing the power grid network.
SECURE	Science and Engineering of Cyber security through Uncertainty quantification and Rigorous Experimentation
SNL	Sandia National Laboratories
Snort	An intrusion detection system
TAMU	Texas A&M University
UQ	Uncertainty Quantification
V&V	Verification & Validation
VM	Virtual Machine

1. INTRODUCTION

1.1. Overview and evolution of research

Securing cyber systems is paramount, but cyber defenders lack evidence-based techniques, which employ principled and rigorous measurements and models. The 2016 Federal Cybersecurity R&D Strategic Plan [32] states: "Most [cybersecurity] techniques are domain- and context-specific, often not validated as mathematically and empirically sound, and rarely consider efficacy and efficiency. Thus, the state of the practice consists of heuristic techniques, informal principles and models of presumed adversary behavior, and process-oriented metrics." Such rigorous, principled methods become critically important for high-consequence systems that support national security missions.

Through a lab-wide initiative, Sandia has been investing in research project, SECURE: Science & Engineering of Cyber Security by Uncertainty Quantification and Rigorous Experimentation. The goal of SECURE is to discover and develop techniques for evidence-based cybersecurity, leveraging the cyber experimental foundation provided by Emulytics (a scalable, virtualized environment for modeling cyber systems) to produce quantitative knowledge concerning a target system, estimate cybersecurity risks, and identify defensive strategies. Specifically, we have integrated Emulytics, uncertainty quantification and adversarial optimization into workflows, enabling evidence-based risk assessment and risk mitigation.

Our approach is inspired by the success of computational science and engineering (CSE) in our nuclear stockpile stewardship programs. Without physical experiments, we assess the readiness of the nuclear stockpile by *computational* experiments. Can we use similar computational experiments to secure our cyber systems? While this is possible in principle, cyber systems are drastically different than physics-based systems, requiring novel techniques for rigorous cyber experimentation. SECURE is built on three pillars 1) *Emulytics* to create detailed, quantitative knowledge concerning a target system; 2) *data analysis and uncertainty quantification* (UQ) techniques that will use information from emulations to develop rigorous reduced-order models that capture key features of these systems; and 3) *adversarial stochastic optimization* that will analyze these reduced-order models to optimize cyber defenses, which are validated and refined using Emulytics.

Cybersecurity experimentation on live environments is costly, time consuming, and disruptive (if not impossible). Thus, these tests provide sparse knowledge about complex cyber systems, and provide limited ability to answer "what if" questions: "What is the best way to defend our network?" "In creating defenses, which attacks should concern us as being maximally disruptive to this system?" Consequently, enabling technologies for Emulytics in virtualized environments are beginning to coalesce to vastly improve our ability to develop, test, and deploy cybersecurity strategies. This capability enables an experimental approach to evidence-based cybersecurity, where computational experiments provide insight into the dynamics and interactions in a cyber system. In simple systems, results of these experiments can directly answer "what if" questions. In complex cyber systems, novel statistical methods for UQ are needed to understand complex interactions. Such statistical characterizations can then be used to explore alternative defense strategies.

This report documents the methods, tools, and case studies developed in SECURE and demonstrates that cyber experimentation can be the foundation of principled cyber security. We claim rigorous cyber experimentation can be a pillar of science of cyber security to ensure security of high-consequence cyber systems, just as CSE is now a pillar of science for our nuclear stockpile stewardship.

1.2. Research Elements

A systematic approach to cyber analytics requires efforts from varying disciplines in close coordination. We need to understand (a) our predictive capabilities and limitations with emulation, (b) how to analyze uncertainty to produce meaningful results for real world systems when there are uncertainties concerning the nature of threats, and (c) demonstrate an ability to quantifying confidence and value across competing risk management strategies, which lead to our three research elements.

1.2.1. Research Element 1: Predictive Cyber Emulation

Developed at Sandia, Emulytics is a state-of-the-art tool set to define cyber-experiment models and testbeds at scale for complex, distributed systems. These systems present challenges related to high-dimensionality, sparseness of data, and expensive forward models. Thus, it is still poorly understood how representation fidelity impacts predictive capabilities in real-world cyber systems, especially in situations with unknown/unobserved or pervasive threats where only the effects are observable. SECURE developed Emulytics methods and mathematical models that scientifically address the fidelity of our models and testbeds under deep uncertainty in the threat space. Our in-silico laboratory enabled reproducible and replicable results for a variety of testbed states and threats.

1.2.2. Research Element 2: Uncertainty Quantification

Our UQ capabilities assess the confidence in computational predictions given a variety of information streams, including models, experimental data, boundary conditions, and expert opinion. Cyber systems present unique research challenges in terms of model validation due to the presence of discontinuous and discrete outputs, the necessity for effective network inference for unknown network structures and topologies, and the tractability of high-dimensional structural and model uncertainties. We developed a set of capabilities to perform validation and forward propagation of uncertainties—including configuration parameters and threats—to handle discreteness and discontinuities, dimension reduction, and multi-fidelity representations.

1.2.3. Research Element 3: Adversarial Optimization

We developed scalable, general-purpose decision-making capabilities for the risk management of both known and unknown cyber threats. The current state-of-the-art in adversarial optimization consists of domain specific models and algorithms that generally assume perfect knowledge on the part of the adversary, perfect execution of adversarial attacks, simultaneous attack vectors, known outcomes of specific attacks, and perfect execution of defender response. The simplest problems are strongly NP-hard, and there is a current lack of well-established solution procedures even for simplified models. We developed a suite of scalable adversarial optimization techniques to address uncertainties such as parameter or structural uncertainties in a network. We identified worst case attackers and attacks that are represented as alternatives against which we can devise and evaluate threat mitigation strategies. Our main objective was to determine optimal investment and runtime defense strategies for interdicting future, possible adversarial threats. We evaluated the performance of proposed optimal solutions with emulation.

1.3. Integration exemplars

Exemplar problems provide a large R&D project such as SECURE a means to focus research efforts onto a useful, visible outcome. Moreover, it ensures integrations among research elements, which is essential and challenging for interdisciplinary efforts, especially when the research elements do not have a history of working together. Although there are a number of potential high-consequence cyber

systems that SECURE could leverage as an exemplar, the team chose cyber-controlled power systems because they are generalizable to a large number of other high consequence systems (including other cyber-physical systems, enterprise networks, etc.). The power grid threat scenario considered by SECURE starts with an initial infection in a grid operator's enterprise network, where malware establishes a command and control (C2) channel to an Internet server. Next, the malware pivots into the grid operator's control center, where it scans the Supervisory Control and Data Acquisition (SCADA) network to identify vulnerable field devices. It then runs the CRASHOVERRIDE malware on the field devices, effecting consequences on the power grid (measured in terms of load lost due to malware actions). This exemplar drove a number of studies, which are described in detail later in this report.

1.4. Workflows

A primary product of SECURE is definition of a cyber experimentation workflow that follows the principles of a scientific process that relies on computation. We documented our approach in detail in Section 8, and prepared a handbook of cyber experimentation that can guide practitioners through the process. This handbook also outlines the algorithmic challenges behind the process providing a useful resource for algorithms developers to enhance the theoretical foundations that support rigorous cyber experimentation.

1.5. Outline of report

The three thrust areas (Emulytics, Uncertainty Quantification, and Adversarial Optimization) are described in Chapters 2-4. Chapter 5 discusses the overall integrated exemplar with detailed case studies on scanning/detection, command and control, and the SCADA network to support the exemplar. The integration of the entire attack chain in a probabilistic approach using a Markov model is also discussed in Chapter 5. Chapter 6 discusses the cross-cutting themes of Verification and Validation. Chapter 7 describes the software tools used and/or developed as part of SECURE. Chapter 8 presents a recommended cyber experimentation workflow. Chapter 9 lists project accomplishments and Chapter 10 provides a summary of the project and its legacy. The Appendices provide details on the case studies in the exemplar we developed for this research, which is an end-to-end cyber attack on a SCADA network.

2. EMULYTICS THRUST AREA

The Emulytics team had two roles on the SECURE project; it had a support role and it had a research role. In its support role, the SECURE Emulytics team developed exemplar scenarios emulation-based models and other models to support the other SECURE teams. In its research role, the Emulytics team developed novel R&D in the areas of mathematical modeling of end-to-end threat scenarios and cybersecurity scenarios that support the end-to-end analysis.

2.1. Emulation and Simulation Models

In its support role, the Emulytics team developed an end-to-end cybersecurity exemplar based on the Advanced Persistent Threat 3 (APT-3) scenario[1] coupled with the 2016 CRASHOVERRIDE attack on the Ukraine power grid. This scenario is depicted in Figure 2-1. Each stage in the exemplar scenario provides transition probabilities to an end-to-end Markov model, which is used to determine attacker (or defender) success metrics such as probability of successful attack and time to successful attack. For some stages (indicated in the figure as blue dots), this information is determined from MITRE ATT&CK vendor evaluation data or through other sources (e.g. publications, SME judgment, etc.). For other stages (indicated as purple dots) these transition probabilities are determined through SECURE modeling, using emulation-based models, mathematical models, and ns-3 discrete event simulation models.

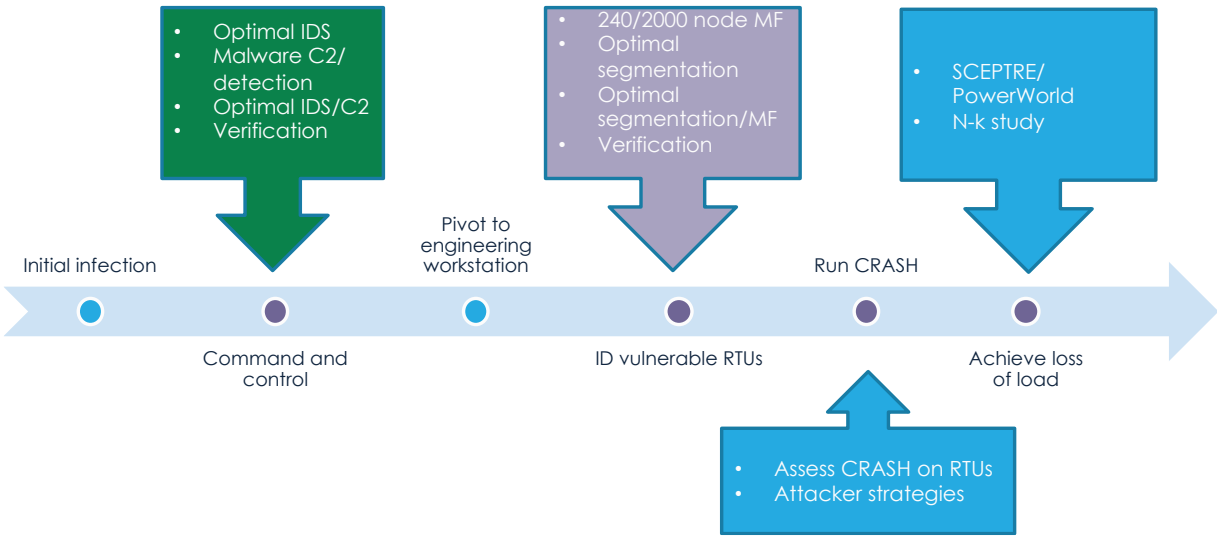


Figure 2-1: End-to-end threat scenario based on APT-3 and CRASHOVERRIDE

For the model-informed parts of the threat model the Emulytics team developed `minimega` and SCEPTRE emulation models (see Chapter 7.2 for more detail about `minimega` and SCEPTRE). `Minimega` was used in the "Command and control" (C2) study to provide a high fidelity, controlled environment to assess intrusion detection performance and validate a mathematical model describing the stochastic nature of malware C2 detection. `Minimega` was also used in the "ID vulnerable RTUs" study, again to assess intrusion detection performance and validate a mathematical model that describes the discovery and detection processes. Finally, SCEPTRE models were developed to couple emulation-based models of cyber assets to a synthetic power grid topology depicting a hypothetical

[1] https://attack.mitre.org/groups/G0022/

2000-bus model of the Texas power grid. This coupled model was used to assess power grid impacts due to CRASHOVERRIDE malware actions.

More details regarding the C2 and scanning/detection models are found in the Appendix A and Appendix B.

2.2. Markov Models

The Emulytics team used Markov analysis to assess attacker/defender performance relative to the end-to-end scenario, answering questions regarding an attacker's probability of successfully performing a power grid attack, and the time required for an attacker to traverse all of the steps required to reach this state. The process starts with translating the end-to-end scenario to a Markov state transition diagram, as shown in Figure 2-2.

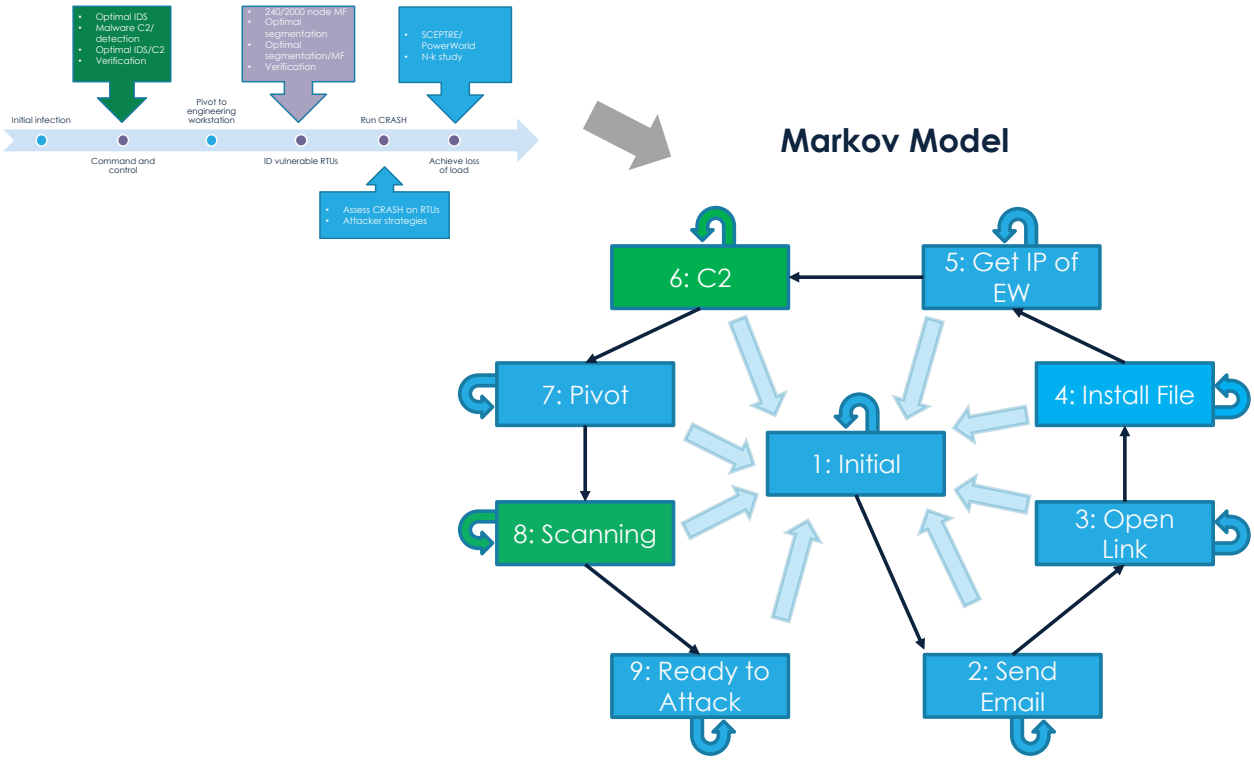


Figure 2-2: Translating the end-to-end threat scenario to a Markov state transition diagram

Once the state transition diagram is constructed, the task shifts to populating the model with transition probabilities. These transition probabilities can be determined via a number of means: through data collected from the MITRE ATT&CK evaluations, through subject matter expert (SME) judgment, or through cyber experimentation. In this study, we used cyber experimentation (i.e. emulation-based modeling and mathematical modeling) to calculate transition probabilities for both the "Command and control" (Markov state 6) and the "ID vulnerable RTUs" (Markov state 8) steps highlighted in green in Figure 2-2. These transition probabilities are shown in Table 2-1 and Table 2-2.

Table 2-1: C2/Markov state 6 transition probabilities (from emulation and mathematical models)

Snort condition	Timestep value	Detection probability	Next state transition probability	Same state transition probability
Unstressed	16 s	0.565	0.435	0.0
Stressed (dropping packets)	16 s	0.372	0.628	0.0

Table 2-2: ID RTUs/Markov state 8 transition probabilities (from emulation and mathematical models)

Attacker scanning strategy	Timestep value	Detection probability	Next state transition probability	Same state transition probability
Fast	30 s	0.69	0.31	0.0
Slow	61 s	0.70	0.30	0.0

An example of an analysis using the experimental and MITRE ATT&CK transition probabilities is shown in Figure 2-3.

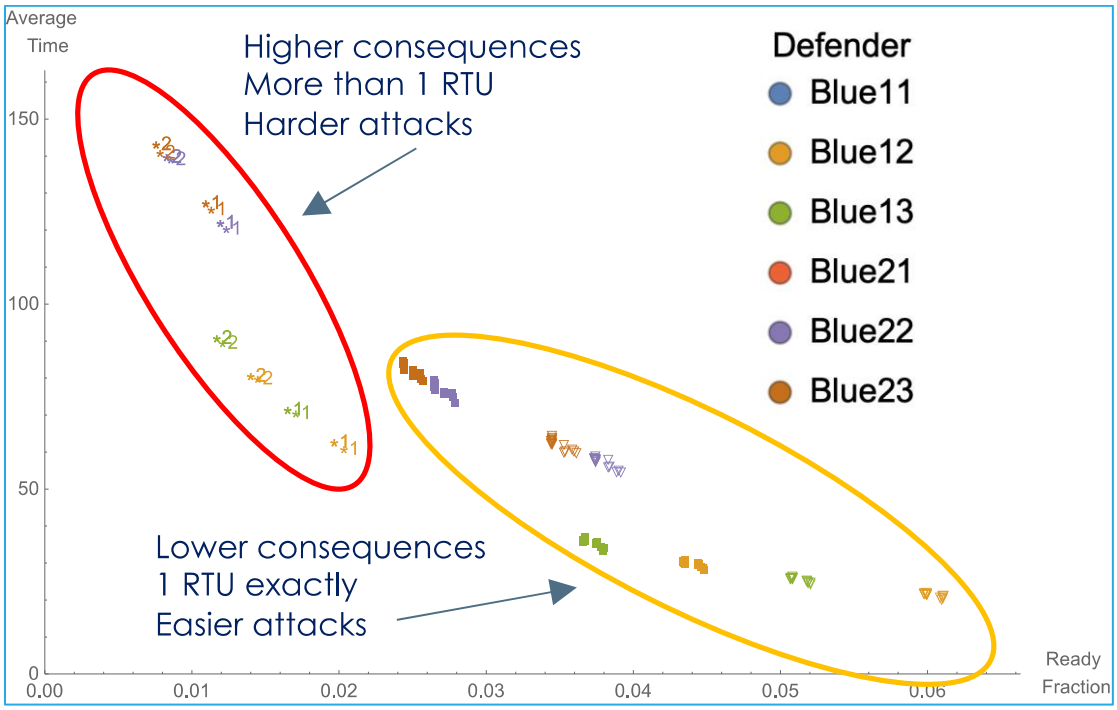


Figure 2-3: Markov analysis results showing attacker time to success and success probabilities, depending on defender capabilities

This analysis assumes a set of different defender (blue team) capabilities denoted B_{ij}, depending on the specific MITRE ATT&CK tactics employed by the attacker (denoted by subscript i), and the defender's ability to handle increasing levels of ambiguity in attack indications (denoted by subscript j, and summarized in Table 2-3).

Table 2-3: Defender capabilities

Defender name	Level of ambiguity	Detection capabilities
$\mathbf{B_{i,1}}$	None	Indicators of compromise (IOC)
$\mathbf{B_{i,2}}$	Medium	IOC, specific alerts
$\mathbf{B_{i,3}}$	High	IOC, specific alerts, general alerts

Figure 2-3 shows the mean time it takes an attacker to transition from state 1 (initial state) to state 9 (ready to attack state) in the Markov chain on the Y axis, and the steady state probability of the attacker residing in state 9 on the X axis. These results are collected into two sets, indicated by the ovals, with the yellow oval indicating results if the attacker only needs to discover exactly one RTU to proceed, and the red oval indicating results if the attacker needs to discover more than one RTU. In cases where the attacker must find more than one RTU in order to continue with its attack, the probability of success is lowest and the time to success is longest (as shown in the set surrounded by the red oval). This makes intuitive sense, since the criteria are more difficult than in the other set, where the attacker only needs to find one RTU.

Within each set there are two arcs: one arc (green and orange points) is for attacker i=1, and the other arc (dark red and purple points) is for attacker i=2. Recall that each of these attackers is distinguished by the particular MITRE ATT&CK tactics that the attacker employs. As can be seen in Figure 2-3, attacker i=1appears to use tactics that do a better job of evading detection than attacker i=2. Details regarding the Markov model formulation and the tactics used by both attackers can be found in [34].

Within each arc there are two groups. In one group, denoted by triangles and *1 markers, the intrusion detection system is stressed by the volume of network data, and is dropping packets as a result. In the other group, denoted by squares and *2 markers, the intrusion detection system is able to process every packet. As the results show, when the C2 intrusion detection system is stressed and dropping packets, the attacker's time to success decreases and its ready fraction increases, indicating that the attacker is more likely to achieve its object more quickly, which makes intuitive sense.

Within each group points are classified according to defender capability. Orange and purple markers represent a defender of Medium capability ($B_{i,2}$), and green and dark red markers represent a defender of High capability ($B_{i,3}$). As Figure 2-3 shows, when the defender's capability increases from j=2 to j=3, the attacker's time to success increases and its ready fraction decreases, indicating that it becomes harder for the attacker to achieve its objectives, which also makes intuitive sense. It should be noted that defender j=1 is not shown in this figure because the results are off the scale of the plot at Ready Fraction = 1.0, meaning that the attacker is certain to succeed against defender j=1.

2.3. Mathematical Models

Mathematical models provide useful insights into the dynamics of a cybersecurity scenario, particularly during model development. In addition, mathematical models can provide a computationally-efficient surrogate for emulation-based modeling, and depending on the analysis question, at sufficient fidelity. During the SECURE project two mathematical models were developed: one that evaluated an attacker's ability to communicate to a command and control server on the Internet without detection (the C2 step, or step #6 in the Markov model), and another that evaluated an attacker's ability to

identify vulnerable RTUs in a SCADA network without detection (the Identify Vulnerable RTUs step, or step #8 in the Markov model).

The C2 mathematical model was based on a stochastic Poisson arrival model and captured different rates for malware C2 traffic (modeled after the Emotet malware) and for benign user background traffic. The model considered intrusion detection and considered cases where intrusion detection is overwhelmed and unable to process every packet. The model was calibrated using data from an initial set of emulation runs and validated against additional runs to assess the mathematical model's predictive value as well as its correlation to emulation (for later use in multi-fidelity studies). The validation results, shown in Figure 2-4, show good agreement between the mathematical model results and the experimental mean, and the model results fall within the emulation experiments' 95% confidence interval. More detail about the C2 model can be found in [43].

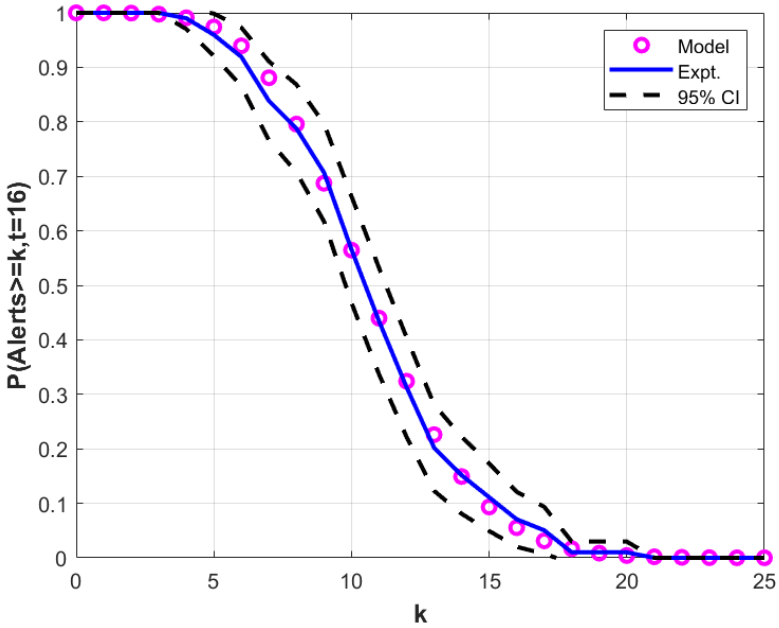


Figure 2-4: Intrusion detection comparison between C2 mathematical and emulation models

The model for assessing an attacker's discovery of vulnerable RTUs, and a defender's ability to detect such scanning, was also developed. Unlike the C2 model, the scanning/detection model is a state-based model that tracks the attacker's discovery process, as shown in Figure 2-5. Using this discovery model (specifically the number of ports discovered vs. time) and a model of the intrusion detection system's alerting algorithm, one can calculate the likelihood of a scan triggering an intrusion detection alert. Similar to the C2 model, the scanning/detection model was also validated against emulation results, and as shown in Figure 2-6, the mathematical model for node discovery agreed with the emulation results relative to the 95% confidence interval. More detail can be found in [44].

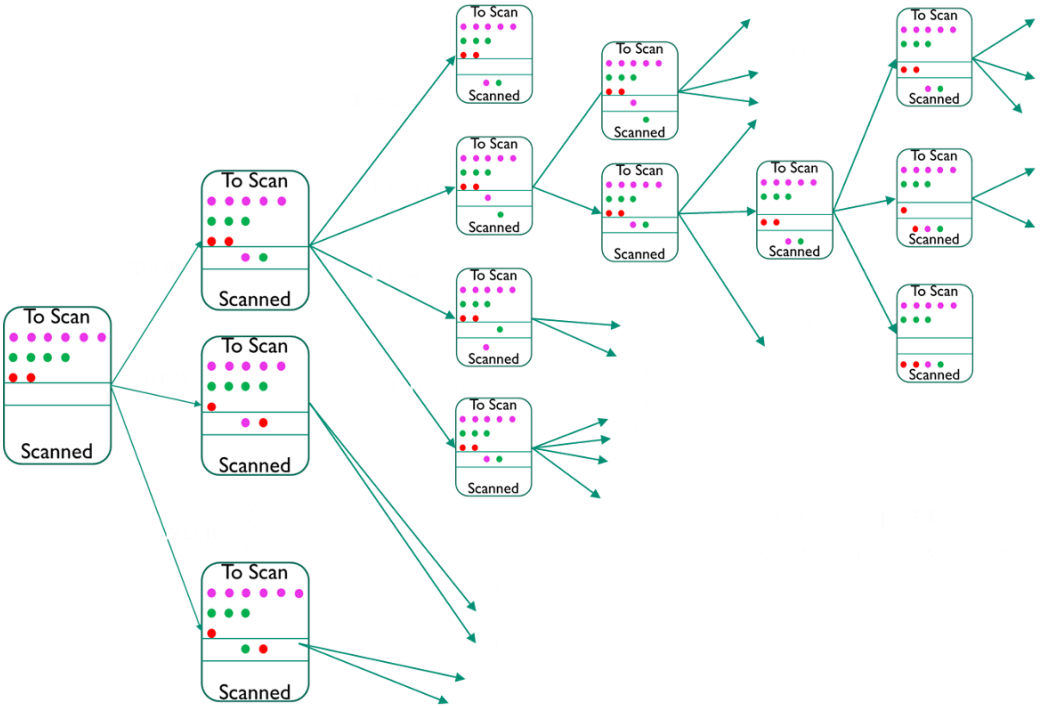


Figure 2-5: State transition diagram for scanning/detection mathematical model

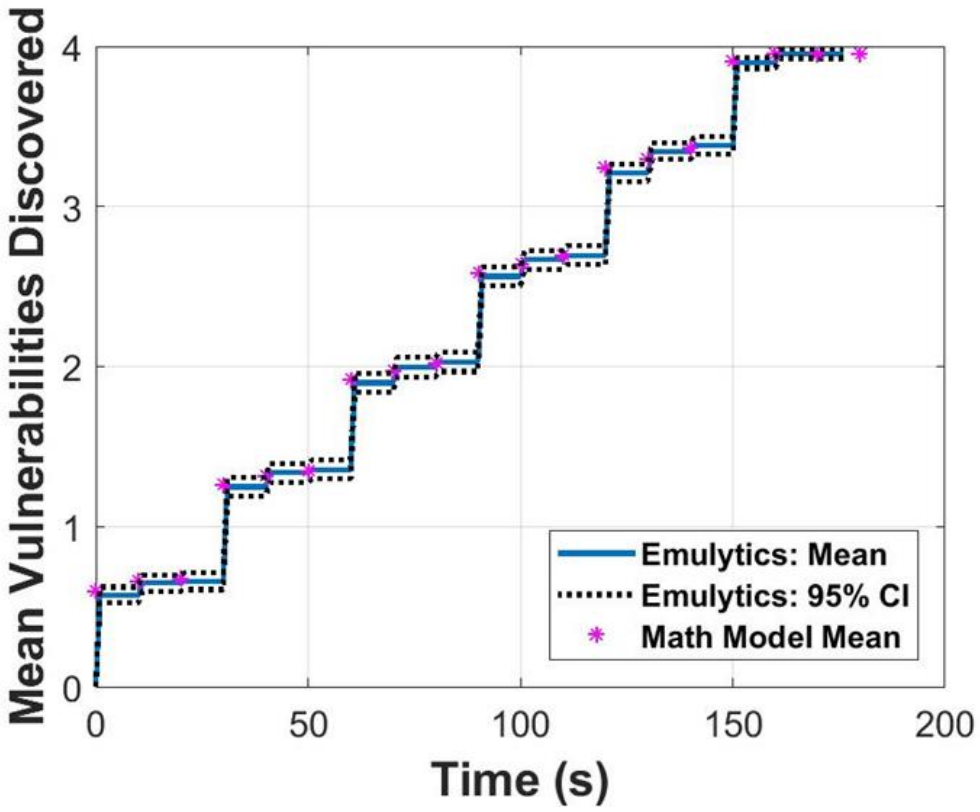


Figure 2-6: Port discovery comparison between mathematical and emulation models

2.3.1. Traffic Generation

In this study, background traffic was generated with the specific question in mind. This approach, tailoring solutions specifically for the experimental goal, can be more effective and efficient. At the same time, there is merit in general purpose approaches that can provide a basis to be tailored. In collaboration with Prof. Catalyurek at Georgia Institute of Technology, we investigated temporal graph generation. First. part of the work focused on topology and generating different graphs with a specified k-core structure [46]. The second part of the work focused on the temporal structure. Specifically, we investigated modeling how frequently the interactions between two entities are repeated, changing. Patterns for the frequencies. We proposed how to build a model from a given data and generating graphs from a given model. For future work, we plan to extend this work to specifically cyber traffic generation [47].

3. UNCERTAINTY QUANTIFICATION THRUST AREA

Uncertainty quantification refers to characterizing input uncertainties and propagating them through a model (e.g. a cyber simulation or emulation model) to obtain the resulting uncertainties on the output quantities of interest. Uncertainty analysis can be used to assess the likelihood of typical or extreme outputs, determine the mean or median performance, understand the variability in the responses, and find probability of failure. A related activity to UQ is sensitivity analysis, which is the identification of the most important variables affecting the response. It involves understanding how model outputs vary as the inputs vary.

In SECURE, we studied three areas supporting UQ. *Dimension reduction* identifies the most important components of a high-dimensional space, allowing uncertainty analysis to focus only on the important components, thus helping tractability. *Discrete polynomials* are an example of a surrogate model, which serve as a "surrogate" or proxy for the computationally expensive simulation or emulation. Surrogate models are used extensively in UQ and optimization of computational models because they are fast to evaluate. However, the accuracy of the surrogate approximation must be determined. *Multifidelity UQ* is another area of UQ which attempts to improve efficiency of sampling by incorporating samples from both low and high-fidelity models.

3.1. Dimension Reduction

In monitoring the behavior of physical or emulated computer experiments, the number of certain events that occur in a given timeframe can be highly significant. Thus, recording these quantities at some frequency (e.g. every second) creates useful time-series data, although that data may be inherently stochastic (due to randomness in timings of initializations, small changes in orderings of system calls, etc.). The challenge is to understand how much of the inherent randomness observed in time series vectors of quantities from cyber experiments can be explained by a few underlying components (i.e. reducing the dimensionality of the data while retaining as much of its variability as possible).

In this work, we examined Principal Component Analysis (PCA) on cyber experiment time-series and compared with a discrete version of PCA called XPCA. We studied XPCA because the Nmap port discovery results are discrete values: 1, 2, 3, etc. ports found. We applied PCA and XPCA to several datasets involving 1000 replicates of port scanning results. Our main finding of this work is that PCA performs better than XCPA with respect to variance explained but worse with respect to reconstruction error on these discrete time series data sets. This is due to the discrete nature of the port discovery time series. The full results are described in the paper below.

- "Time Series Dimension Reduction for Surrogate Models of Port Scanning Cyber Emulations." Erin C.S. Acquesta, Laura P. Swiler, and Ali Pinar. SAND20-10617.

3.2. Discrete PCE

Uncertainty quantification is often accomplished via computationally expensive Monte Carlo sampling. However, less costly stochastic expansion methods can approximate the functional dependence of the simulation response on uncertain model parameters by expansion in a polynomial basis. The polynomials used are tailored to the characterization of the uncertain variables. Polynomial chaos expansion is based on orthogonal polynomials.[10,45] The goal of PCE is to construct a more efficient and accurate estimate of the uncertain response distribution than would be obtained from Monte Carlo sampling.

In this research, we investigated the use of discrete orthogonal polynomials for constructing polynomial chaos expansions to build a response approximation of the results from cyber experiments. One unique feature of the work is the presence of replicates (replicated data points) from the cyber emulations. The references below discuss how samples are chosen in input space and presents an analysis of "best practice" approaches for constructing stochastic expansions based on data one might obtain from a cyber experiment.

- Bert J. Debusschere, Gianluca Geraci, John D. Jakeman, Cosmin Safta, and Laura Swiler, "Polynomial Chaos Expansions for Discrete Random Variables in Cyber Security Emulytics Experiments", SIAM CSE 2021 presentation, March 1, 2021. SAND2021-2270C.
- Bert J. Debusschere, John Jakeman, Eric Vugrin, Gianluca Geraci, Laura Swiler. "Sensitivity Analysis for Cyber Security Scenarios Using Mixed Discrete - Continuous Polynomial Approximations." Manuscript in preparation.

3.3. Multifidelity UQ

Often, uncertainty quantification is challenging to perform because of the large number of samples that must be run through a cyber model, which can be computationally expensive. However, in multi-fidelity (MF) uncertainty quantification, many samples from one or more low-fidelity models (such as a mathematical model or a network simulator like NS-3) are fused with a few runs of a high-fidelity cyber model (e.g. actual software run on real or virtualized hardware) to decrease the estimator variance and obtain more reliable statistics. While we may only be able to run a few dozen samples of a high-fidelity model, we assume the cost of the low-fidelity model is much cheaper and so we can generate many low-fidelity samples for the cost of one high-fidelity model evaluation. The papers [9,8] present the theory behind multifidelity UQ. Additionally, [8] presents several network problems of increasing difficulty, and demonstrates that the multifidelity estimator demonstrated increased efficiency with respect to Monte Carlo sampling.

The MF estimator for a mean of response Q can be built starting from the single fidelity MC for the high-fidelity model and adding a weighted unbiased term to it:

$$\widehat{Q^{MF}} = \frac{1}{N}\sum_{i=1}^{N} Q_{high}^{(i)} + \alpha\left(\frac{1}{N}\sum_{i=1}^{N} Q_{low}^{(i)} - \frac{1}{r\times \mathrm{N}}\sum_{j=1}^{r\times N} Q_{low}^{(j)}\right) \qquad (3.1)$$

$$\widehat{Q^{MF}} = \widehat{Q_{high}} + \alpha\widehat{\Delta_{low}},$$

In Equation 3.1, N is the number of high-fidelity runs, and *r* is the oversampling ratio that allows for a maximization of the efficiency of the estimator by defining the optimal number of low-fidelity model evaluations as (N+1) x *r*. The first term on the right-hand side is just the usual mean estimate from the high-fidelity model. The second term is the low fidelity estimate "corrected" so that it is unbiased. Note that the second term has many more samples: this contributes to the variance reduction of the MF estimator. For a MF estimator with a single low-fidelity model, the coefficient α is obtained in closed form as function of the correlation and estimated variance of the two models.

For the application of MF UQ to the command and control study that is discussed in Chapter 5, we started with 40 evaluations of the high-fidelity model, which in this case was a `minimega` emulation. The low-fidelity model in the C2 case was a mathematical model. For C2, Equation 3.1 becomes Equation 3.2:

$$\widehat{Q^{MF}} = \frac{1}{40}\sum_{i=1}^{40} Q_{minimega}^{(i)} + \alpha\left(\frac{1}{40}\sum_{i=1}^{40} Q_{math}^{(i)} - \frac{1}{2171\times 40}\sum_{j=1}^{2171\times 40} Q_{math}^{(j)}\right) \qquad 3.2$$

$$\widehat{Q^{MF}} = \widehat{Q_{minimega}} + \alpha\widehat{\Delta_{math}},$$

Given the particular cost ratios and correlations of this problem, further described in Appendix A, the estimate of the mean number of alerts (Q) as a function of three time point (t=1, 5, and 10 seconds) is shown in Table 3-1. Comparison between the single fidelity MC estimator based on `minimega` data only and the MF estimator based on the additional math model evaluations. Note that two estimates of the mean response are provided: one based on 40 `minimega` runs and one based on a multifidelity estimate that incorporated both the `minimega` runs and the math model runs as shown in Equation 3.2. In Table 3-1, it is possible to observe how, for larger times, the value of the α coefficient approaches 1, which corresponds to the case of perfect correlation and ratio one between the variances of the two models.

Table 3-1. Comparison between the single fidelity MC estimator based on `minimega` data only and the MF estimator based on the additional math model evaluations.

t [s]	$\hat{Q}_{minimega}$	ρ	α	$\hat{\Delta}_{math}$	$\hat{Q}^{MF}$
1	0.570	0.8049	-0.7056	-0.0798	0.626
5	2.802	0.9725	-0.9568	0.0921	2.714
10	5.695	0.9832	-0.9868	0.3908	5.309

3.4. Multifidelity Estimation with Replicates

A unique feature of the cyber emulations that we studied under SECURE was their stochastic nature. That is, if we ran the emulation model multiple times with the same configuration, we would obtain results that were different. This can be due to timing and ordering of processes that are spawned as well as differences in the host or VM state, for example. This means that the quantity of interest listed as Q above needs to be evaluated some number of times (replicas) for each experiment to obtain a mean value under the stochastic conditions.

In this section, we consider the possibility to optimize the number of replicas in a low-fidelity model in order to speed-up the multifidelity computations. We might want to perform fewer replicas of the high-fidelity model than the low-fidelity model, for example, because of the higher cost. This section examines the ramifications of these replicas.

This scenario is based on the C2 example described in the previous section and Appendix A, although we here consider that the mathematical model, serving as low-fidelity model, is indeed characterized by the distribution of the number of alerts at each time step. Therefore, for each time location, we can query the model to obtain a single prediction/replica. In the limit of infinite number of replicas,

the average would correspond to the deterministic mean value predicted by the model, but, in all other cases, the predicted value will show a stochastic error with respect to the true system's response. We consider the cost of a replica to be 0.001s, while the runtime for `minimega`, the high-fidelity, is unchanged at 162s. We note here that, as done in the previous scenario, `minimega` will need the average of 10 replicas to get a quantity of interest.

The effect of the stochastic noise in the low-fidelity is to decrease the correlation between the high- and low-fidelity. The true correlation can be recovered only in the limit of infinite replicas, however increasing the number of replicas for the low-fidelity increases, linearly, the computational cost.

We want to address here the following question: given an assigned high-fidelity model and dataset, what is the best configuration for the low-fidelity for both the number of UQ parameters and replicas? In this context, the best estimator is the one that minimizes the estimator variance with the minimum cost.

As a first result we show the optimal cost ratio between a MF UQ estimator and a single fidelity estimator, i.e. `minimega` simulation only for different time steps.

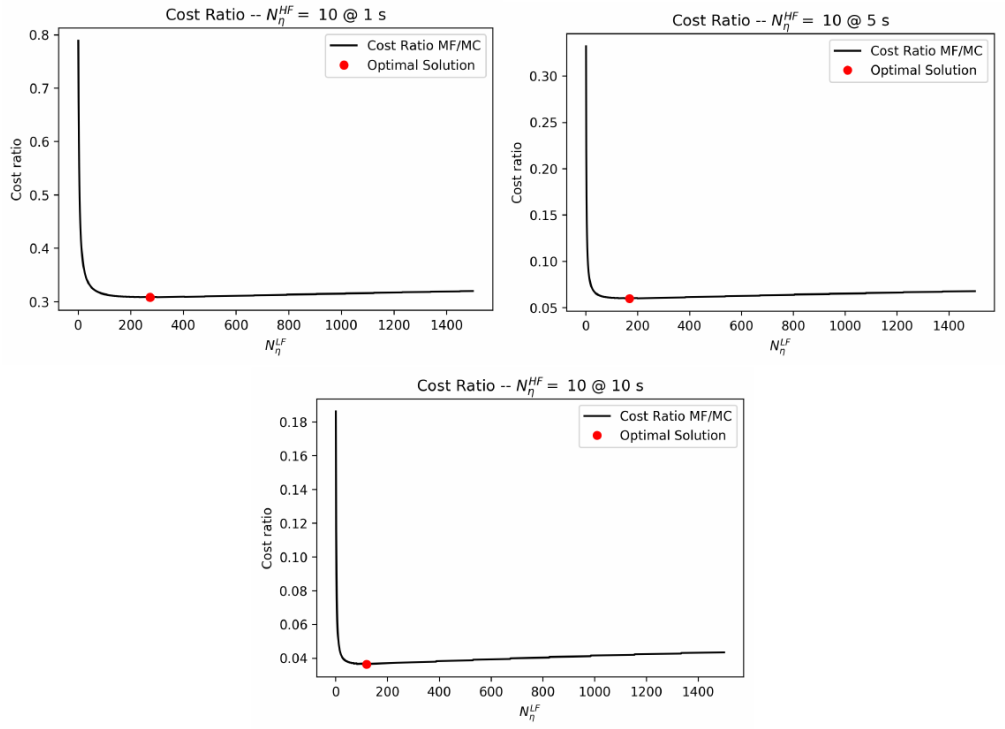


Figure 3-1. Cost ratio between MF estimator and MC as a function of the low-fidelity number of replicas

In Figure 3-1, the cost ratio is reported for all time instances considered in the study, 1, 5 and 10 seconds, as a function of the number of replicas for the low-fidelity. We observe that the minimum cost ratio, which indicates the maximum estimator efficiency, is obtained with a number of replicas between 100 and 300 replicas. The most restrictive case occurs for the time instant 1s, which requires 274 replicas. We will use this number of replicas for the estimators at all time instances.

We consider and compare several estimators in the numerical experiment. All of them use the 40 `minimega` runs, each with 10 replicas, and they differ only for how the low-fidelity correction is handled:

- The first MF estimator uses the optimal number of replicas (274) and an oversampling ratio of 421. The total cost of the estimator is then equivalent to 43 high-fidelity runs;
- The second MF estimator has a total equivalent cost of 43 high-fidelity simulations, by design in order to be compared with the previous one, but it uses only 10 replicas for both high and low-fidelity. This estimator corresponds to a MF estimator that does not exploit the optimization in the low-fidelity replicas. In order the keep total cost constant, the oversampling ratio is much larger (11536.8), since for each UQ samples only 10 replicas are evaluated, compared to 274 replicas of the previous estimator;
- The MC estimator with an equivalent cost of 43 high-fidelity samples is estimated in order to provide a fair comparison among estimators;
- The MF that we used in the previous section is also reported for comparison, although that estimator has a lower equivalent computational cost of 41.

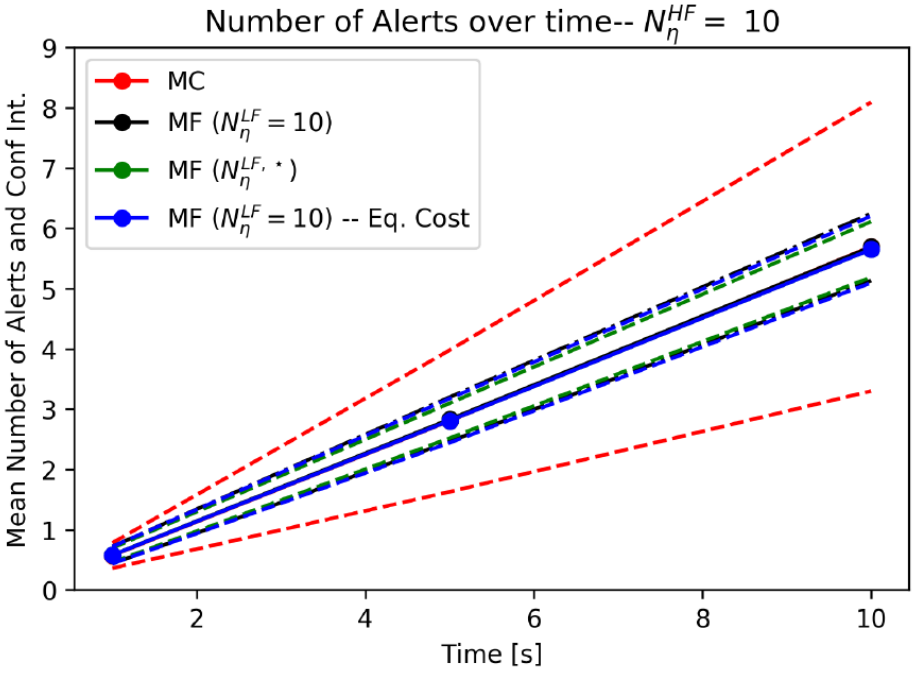


Figure 3-2. Confidence intervals for several multifidelity estimators with and without optimization of the number of low-fidelity replicas

In Figure 3-2, all the multifidelity estimators, with and without optimization of the number of low-fidelity replicas, are reported. As expected, the MF estimator with the optimal number of low-fidelity replicas, green lines, is the one with the smallest confidence interval, whereas the estimators with 10 low-fidelity replicas have a similar confidence interval.

The comparison is even more clear by considering the data reported in Table 3-2.

Estimator	N_ξ	$\tilde{r}$	N_η^{LF}	$\tilde{\Lambda}$			$N_\xi^{HF,eq}$
				$t = 1$s	$t = 5$s	$t = 10$s	
MF	40	1762.34	10	0.425	0.091	0.050	41
MF $(N_\eta^{LF,\star})$	40	421	274	0.297	0.056	0.034	43
MF (C_{eq})	40	11536.8	10	0.424	0.090	0.049	43
MC	43	-	-	-	-	-	43

Table 3-2. MF estimator data for the cases with and without optimization of the low-fidelity replicas.

In Table 3-2, the colors of the estimators correspond to the color of the lines in Figure 3-2. In addition to the oversampling factor and the number of low-fidelity replicas, we also report the variance reduction, for all the MF estimators, achieved at the different time instances. The best MF estimator, which uses the low-fidelity optimization, achieved a variance reduction ranging from 70% to 97%, approximately. On the contrary, without low-fidelity replicas optimization, the variance reduction achieved ranges from 60% to 95%. We further note here that the optimal performance of the estimators is indeed obtained for the time instance of 1s, which also corresponds to the time for which the 274 replicas is optimal. Later time steps, would require, in principle, lower number of replicas, although we have fixed the number of 274 replicas to reflect the practical constraint behind the selection of an optimal allocation for different quantities of interest.

This study of replicas in multifidelity uncertainty estimation is a unique contribution that has been supported by the SECURE project.

4. ADVERSARIAL OPTIMIZATION

Another focus of the SECURE project was the use of adversarial optimization to model the interactions between cyber defenders and attackers. Standard optimization models aim to identify a solution that maximizes or minimizes a given function, subject to a collection of mathematical constraints. For example, consider the DC optimal power flow (DC-OPF) model. This model, which approximates AC power flow, identifies how a grid should be operated to eliminate or minimize unmet demand. Since electric power grids are governed by the laws of physics and capacities of equipment like lines and generators, a collection of constraints must be satisfied by any solution to ensure that it is feasible.

Adversarial optimization extends standard optimization methods by embedding optimization models within other optimization models. For example, consider a version of the DC-OPF model which we refer to as the "*N-k* DC-OPF" problem. In this example, we assume that there is a power grid with *N* components and an attacker who can disable *k* of those components. The attacker aims to find the set of components to attack so that unmet demand or load shed on the system is maximized. However, once the grid operator observes the attack, they will update how their system is being operated to minimize load shed. This problem belongs to class of adversarial optimization problems called bi-level programs, since there is an outer optimization problem, an attacker who wants to maximize load shed, and an inner optimization problem, a grid operator who wants to mitigate the effect of an attack until the affected components can be restored.

These methods are of particular interest to SECURE because they provide a means of finding worst-case attacks against a system. In the case of the *N-k* DC-OFP problem, these methods give a bound on the maximum amount of load shed that can be caused for an attack of a given size. Without these methods, some type of sampling-based or heuristic search would be required to find bad attacks, but it would be difficult to prove that the worst-case attack had been found. The bi-level framework can also be extended to consider tri-level level problems. One use of this class of problems is to determine how to best protect a system. In the tri-level case, a defender could first determine how a system should be protected (the first level). The attacker will then find the worst-case attack against the fortified system (the second level) knowing that the grid operator will try to minimize the impact of the attack (the third level).

The adversarial optimization work on SECURE had two main focuses. The first was developing a toolkit to express and solve adversarial optimization problems. While there is a large body of published literature on adversarial optimization algorithms, there are few general-purpose tools available to write and solve these types of problems. In practice, this means that applying these methods typically requires custom solutions. To address this, the SECURE team developed the Python Adversarial Optimization (PAO) toolkit, which contains both a modeling language for expressing adversarial problems and algorithms for solving them (see Section 7.4 for addition details). The second focus was on developing adversarial optimization models to address cyber-physical security problems. The remainder of this section describes each of the adversarial optimization models developed under SECURE.

4.1. *N-k* Worst Case Analysis

The first optimization capability developed under SECURE was the *N-k* DC-OPF problem described above. To begin, we implemented an existing version of this model [30]. A key feature of this model is that it does not make any assumptions about how the *k* components on the system are disabled. For example, it could be from either a physical or a cyber-attack. This is useful because this capability

can be used to bound the damage that can be caused for a wide variety of threats, without having to model the specific threat. Figure 4-1 shows an example of this for the IEEE-118 bus test network. In this example, we consider the load shed that is caused by attacking a given number of buses. We assume that when a bus is attacked the associated loads, generators, and lines are disabled. We compare random attacks generated via one thousand Monte Carlo samples (the box-and-whisker plots) to the worst-case attacks found using the N-k model (red dots). Observe that the worst-case attacks can be significantly more severe than even the tails of the randomly generated attack distributions.

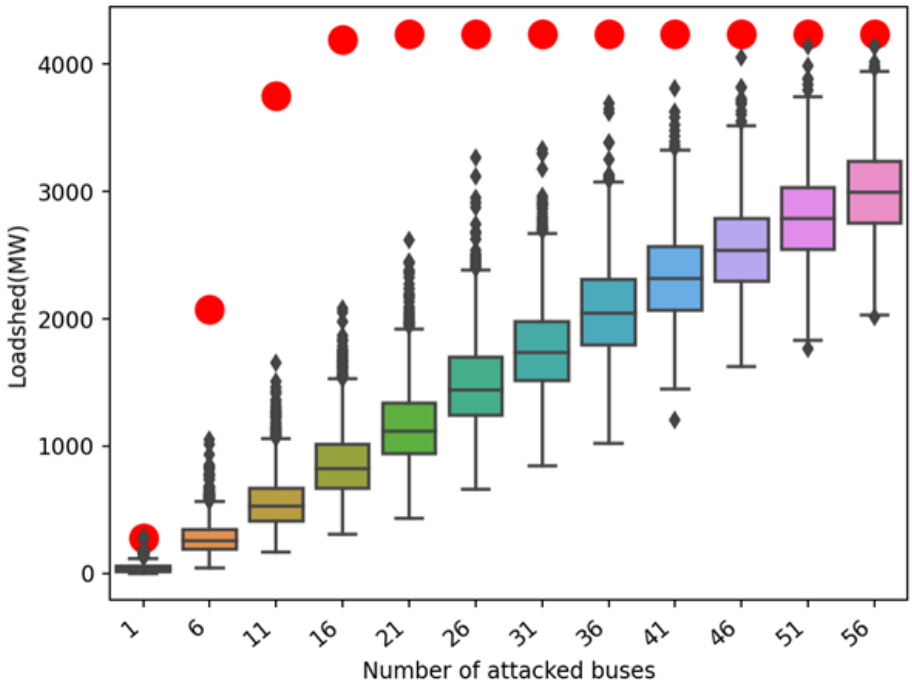


Figure 4-1. Comparison of load shed for the IEEE-118 bus system for random and worst-case attacks.

SECURE utilized the *N-k* model in the following two research thrusts:

First, it was coupled with a cyber-physical emulation to better understand the impact of a CrashOverride malware attack on a notional electric system. More details on this work can be found in the paper below:

A. Castillo, B. Arguello, G. Cruz and L. Swiler, "Cyber-Physical Emulation and Optimization of Worst-Case Cyber Attacks on the Power Grid," 2019 Resilience Week (RWS), 2019, pp. 14-18, doi: 10.1109/RWS47064.2019.8971996. SAND2019-12468C.

Second, we explored a simplified version of the model to speed up solution times. While the *N-k* model is a powerful capability for finding worst-case attacks, it can be difficult to solve for large attack budgets, even for networks with a few hundred buses. This difficulty further increases with the number of buses in the network. To address this challenge, the simplified version of the model removes a complicating constraint (Ohm's law) from the DC-OPF model, reducing it to capacitated network flow. Solving the *N-k* model without this constraint gives a lower bound on the worst-case attack and leads to a significant improvement in performance. Analysis and experiments showed that in certain regimes, the results from the simplified model are often as good or nearly as good as the original DC-OPF formulation. The details of this approach can be found in the reference below:

E.S. Johnson and S.S. Dey, "A scalable lower bound for the worst-case relay attack problem on the transmission grid," in first round of revisions at *INFORMS Journal on Computing,* available at arXiv.2105.02801, SAND2021-10211.

4.2. Network segmentation

As an extension of the *N-k* DC-OPF model, the power grid cyber-physical network segmentation model was developed under SECURE to improve grid resiliency to SCADA cyber-attacks. The model assumes a three-tier SCADA system where an attacker must start attacks from balancing authorities, the first tier. Attacks must then pivot to control centers to reach substations. Once a substation has been infiltrated, all grid components at that substation are disabled by the attacker to damage the grid and cause loss of power to customers. A network designer can segment networks within each tier a pre-determined number of times to restrict possible attack vectors, with anticipation of the worst possible attack on the segmented SCADA system. See Figure 4-2 for an example of a cyber-physical system before and after network segmentation, considering a worst-case attack where an attacker is limited to attacking 5 subnets.

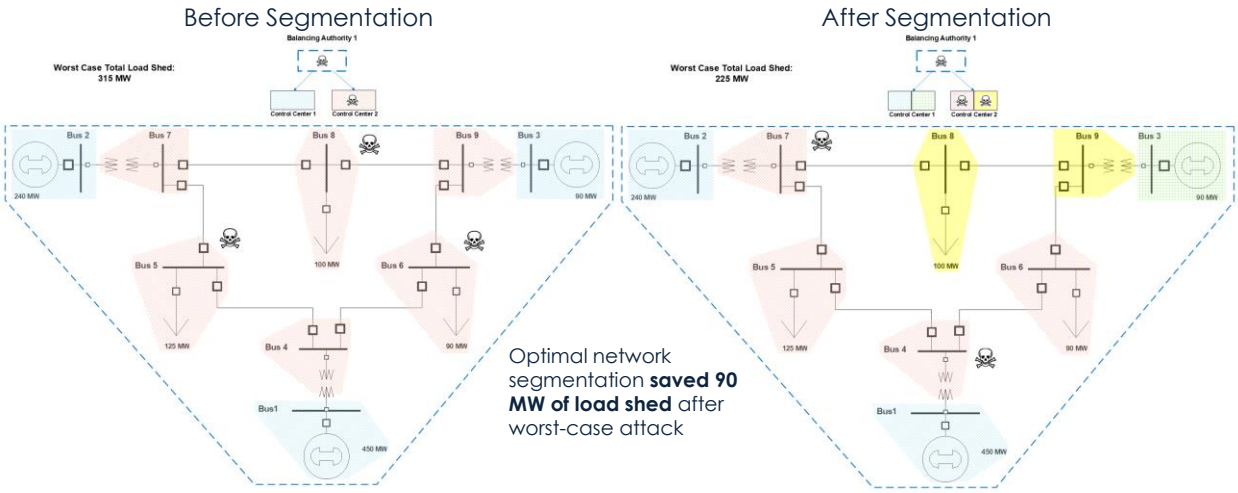


Figure 4-2. Worst-case attack before network segmentation and after network segmentation. Segmentation allowances: 1 extra balancing authority segment, 2 extra control center segments, 5 extra substations segments. Attacker budget: 5 subnets

In this example of optimal network segmentation, we show a baseline SCADA configuration for the 9-bus WSCC test system along with an optimally segmented network. Before network segmentation, there is one balancing authority network, two control center networks, and 9 substation networks. An attacker with five units of attack budget can cause a load shed of 315 MW–the full system load. After network segmentation, the balancing authority is segmented into two subnets, each control center is segmented into two subnets, and five substations are each segmented into two subnets. This configuration only allows an attacker with the same attack budget to shed 215 MW of load.

Network segmentation under worst-case attacks is performed via a mixed-integer trilevel interdiction model. The three players in this model are a network designer, an attacker, and a grid operator who runs a DC-OPF model to redispatch the grid after a worst-case attack on the optimally segmented SCADA system. The model was solved using bilevel branch-and-bound, as well as a trilevel cut-generation approach. Algorithm details and results can be viewed in the following publications

B Arguello and E.S. Johnson and J.L. Gearhart, "A Trilevel Model for Segmentation of the Power Transmission Grid Cyber Network", submitted to IEEE systems, available at arXiv.2108.10958. SAND2021-10208O.

4.3. Optimal Sensor Placement

The sensor placement optimization model was developed to identify where sensors should be placed in a cyber network to maximize the probability that attacks are detected, knowing that an attacker will aim to evade detection after the sensors are placed. This model uses attack graphs, derived from the threat modeling work described in Section 2.2, as the "game board" where attackers and defenders interact. An example of an attack graph is shown in Figure 4-3. In this example, the boxes represent the state that an attacker is in (e.g. root permission on a SCADA workstation) and the arcs represent attacks (actions) that are taken to move between states. The example shown below is an attack graph for the WECC 9 bus power system. In this example, attacks begin at the remote station (at the top) and terminate at one or more of the nine PLCs associated with the nine buses on the power system.

Each arc has a baseline probability of detecting an attack and a modified detection probability if the defender chooses to install (or upgrade) a sensor on that arc. The attacker wants to cause at least some amount of load shed. To achieve this, they determine which PLCs to attack and how to attack them, in a manner that obtains this load shed with the lowest probability of being detected. Note that attacks can take the form of a path if only one PLC is attacked or a tree if multiple PLCs are attacked. The defender has a budget for the number of sensors that can be installed. Their goal is to determine how to place sensors to maximize the probability that the attack is detected.

In the example shown below, we assume that the attacker wants to cause at least 125MW of load shed, about 40 percent of the total load. Using notional baseline detection probabilities, they can achieve this about 72 percent of the time. If seven new sensors can be installed, the evasion probability is reduced to 24 percent. The orange hourglass icons show the locations where the optimization model chose to place sensors in this example. After the sensors are placed, the attacker identifies the best attack that is available to them. This is shown by the arcs highlighted in red in Figure 4-3. In this case the attacker can achieve the target load shed by attacking PLC 5. However, doing so requires them to pass by two new sensor installations, in addition to the existing detection capabilities.

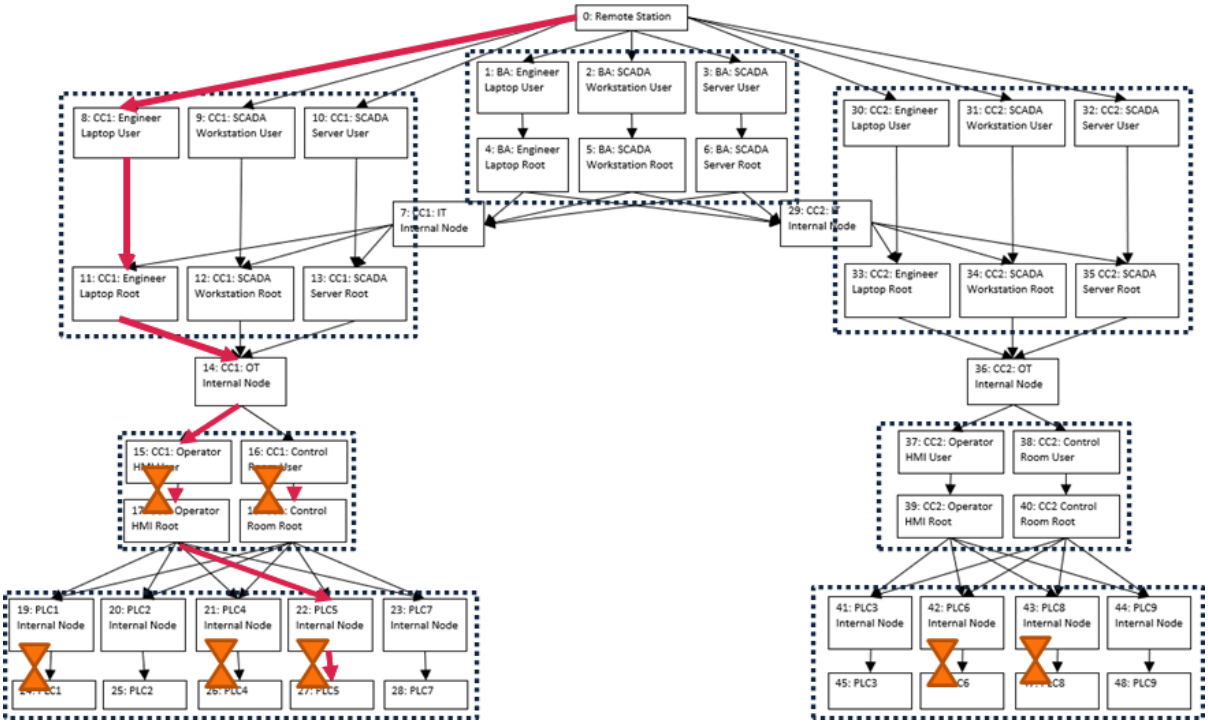


Figure 4-3. Notional attack graph for the WECC 9 bus system. Optimal placement decisions for a budget of 7 and a load shed threshold of 1.25MW are shown by orange hourglass icons. The red arrows show the path that has the highest probability of evading detection (24 percent).

This model is formulated as a bi-level problem where the defender first installs sensors and the attack finds the best available attack. In the case described above where the attackers want to cause a specified amount of load shed, there is an implicit third stage for the DC-OPF problem. Unlike the previous examples, the attack is not looking to maximize load shed. Instead, they want to ensure that the grid operator cannot avoid shedding the specified amount of load.

4.4. Robust Optimization

SECURE also developed methods to incorporate robustness into multi-level adversarial optimization problems. In their standard form, optimization models use constraints that are parameterized by known values. However, in practice uncertainties can exist in the parameters used by the model. When distributional information on these parameters is available, approaches like stochastic programming can be used to account for decision making under uncertainty. When distributional information is not available, robust optimization methods offer an alternative approach for dealing with uncertainty. Robust optimization assumes that parameters are not fixed but are instead constrained to take values within some uncertainty set. When robust models are solved, the solutions that are generated are guaranteed to perform well over all parameter values in the uncertainty set.

Under SECURE, these approaches were applied in the context of sensor placement on networks, such as the attack graphs shown in the previous section. In a cyber setting, the sensor model focuses on placing sensors to maximize the probability of detecting an attack. As sensors are placed, the attacker may alter their path to minimize the probability that they are detected. One potential issue with this model is that the sensors that are placed on the network might not perform as expected or advertised.

Given this, the robust version of this model helps ensure that the placement decisions guard against some amount of sensor failure or degradation.

5. END-TO-END EXEMPLAR

The SECURE project developed an end-to-end cybersecurity exemplar based on the Advanced Persistent Threat 3 (APT-3) scenario[2] coupled with the 2016 CRASHOVERRIDE attack on the Ukraine power grid. This scenario is depicted in Figure 5-1. For each stage in the exemplar scenario that the team evaluated experimentally (depicted as purple dots in the arrow), the SECURE team selected a set of attack tactics (e.g. from the MITRE ATT&CK database) that were representative of real attack tactics, amenable to experimental implementation, but generalizable to other tactics. For other stages, the SECURE team either used experimental data from the MITRE ATT&CK evaluations or used data from the literature and/or subject matter judgment. In all cases, data from each stage informed transition probabilities for a Markov model in order to assess end-to-end performance [see Section 2.2].

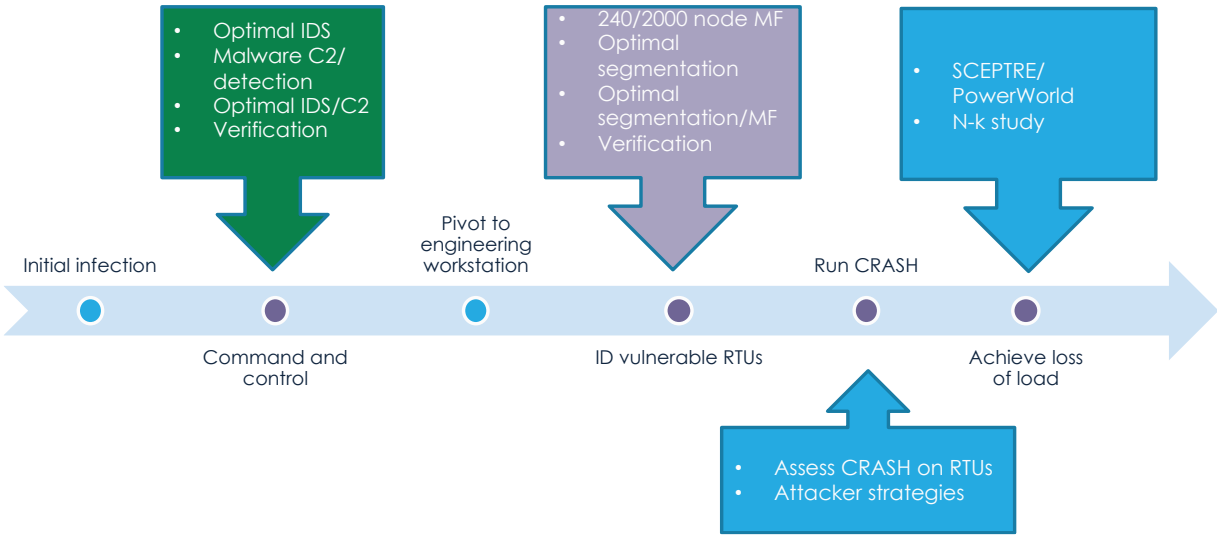


Figure 5-1: End-to-end threat scenario based on APT-3 and CRASHOVERRIDE

5.1. Case Study: Command and Control (C2)

After initial infection, the next attack step in the exemplar scenario is focused on Command and Control (C2). In this step, an attacker aims to establish a malicious C2 channel between one or more infected hosts and a C2 server. To counter this, the system defender uses an intrusion detection system (IDS) to monitor network traffic and detect malicious C2 traffic, as illustrated in Figure 5-2. In this example there is both benign (green) and malicious (red) traffic on the network. While there are many types of C2 malware and IDS systems that could be considered, we selected Emotet and SNORT, respectively, for this case study.

[2] https://attack.mitre.org/groups/G0022/

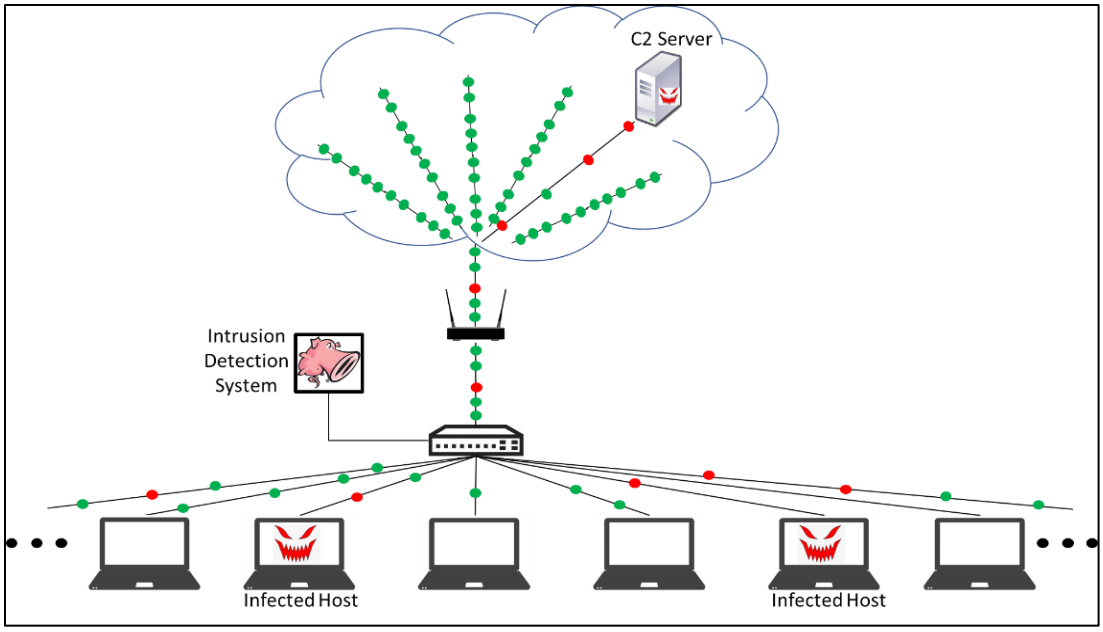


Figure 5-2: Notional C2 exemplar system representation.

The aim of this case study was to use the SECURE experimentation methodology to rigorously analyze this system. Specifically, we were interested in understanding the number of alerts (both true and false positives) that would be generated by the IDS over time under various conditions and settings. To accomplish this, we developed an emulation model and a mathematical model for this system. The emulation model provided a high-fidelity representation but was costly to run since it requires specialized computing resources and runs in real time. The math model has lower-fidelity but can be run significantly faster than the emulation model, using desktop computing resources. Next, validation and verification activities were performed on these models to build confidence in the results they provide. Finally, analysis was performed using these models to understand which parameters have the largest impact on this system. A key focus of the analysis was optimally using both models in tandem to efficiently perform the analysis, while ensuring the accuracy of the results. See Appendix A for a detailed description of this study.

5.2. Case Study: Scanning/Detection

In the APT-3 threat scenario, when an attacker gains a presence on a control center machine (e.g. an engineering workstation), the attacker performs reconnaissance on the SCADA network to identify vulnerable RTUs that are susceptible to CRASHOVERRIDE attack. In this case, without loss of generality, the SECURE team selected Nmap as the network reconnaissance tool because it is simple, open source, and generalizable to other scanning tools. Using Nmap, the attacker probes the network address space to 1) find active IP addresses, and 2) determine which ports are open on those nodes. In most of our scenarios, as shown in Figure 5-3, the attacker resides on a control center workstation and runs Nmap to identify 24 RTUs across eight substations. Vulnerable RTUs are modeled as nodes with an open Secure Shell (SSH) port and secure nodes are those whose SSH port is closed or those who do not respond to Nmap probes. In addition, the scenario includes a Snort intrusion detection node that is capable of seeing all traffic traversing the router that connects the control center to the substations and is configured with the 'sfportscan' Snort rule to detect scanning activity.

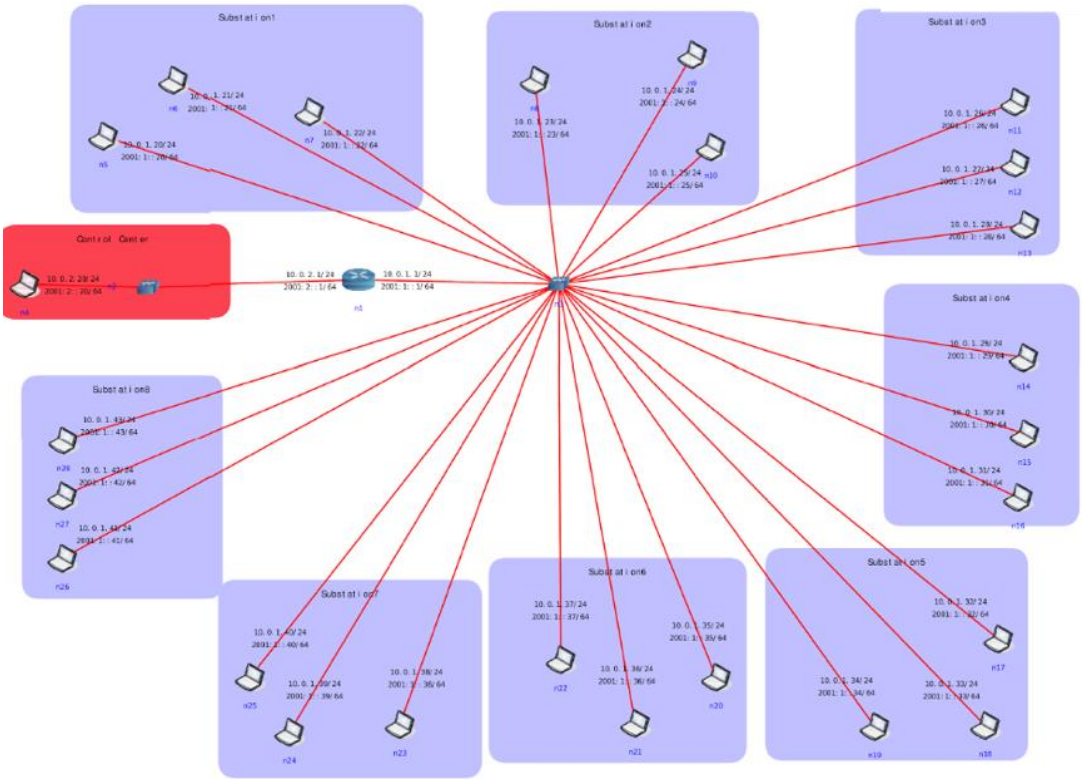


Figure 5-3: Typical SCADA network topology for scanning and detection

More detail regarding the scanning/detection scenario may be found in [44] and [41] as well as Appendix B.

5.3. Case Study: SCADA Network/Power Grid Impacts

The CRASHOVERRIDE malware was integrated into the SCEPTRE emulation environment to study the impact of CRASH actions on portions of the synthetic Texas 2000-bus power grid. SCEPTRE is an ideal environment for these kinds of studies because it couples emulated cyber models (using `minimega`) with simulated power grid models (using PowerWorld). In addition, tools were developed to import combined grid/cyber topologies from Texas A&M University into SCEPTRE, which eliminates the need to generate these topologies by hand.

One study considered a scenario where a portion of the synthetic Texas 2000-bus power grid is controlled by a subset of nine substations and 49 field devices, as shown in Figure 5-4. In this study, an initial network topology and subnetting scheme was provided to the SECURE optimization team to determine a worst-case attack for a given attack budget. ("Attack budget" is the number of subnetworks an attacker can subvert.) The resulting worst-case attack identifies the substations that are subverted, and the SCEPTRE emulation team assumes that all field devices within the substation are compromised.

Next the optimization team computes a new subnetting topology that optimally minimizes the effect of a worst-case attacker and calculates the resulting worst-case attack. The new topology is imported back into SCEPTRE (using VLANs, firewall rules, or other configurations to enforce the new subnetting scheme), along with the new worst-case attack.

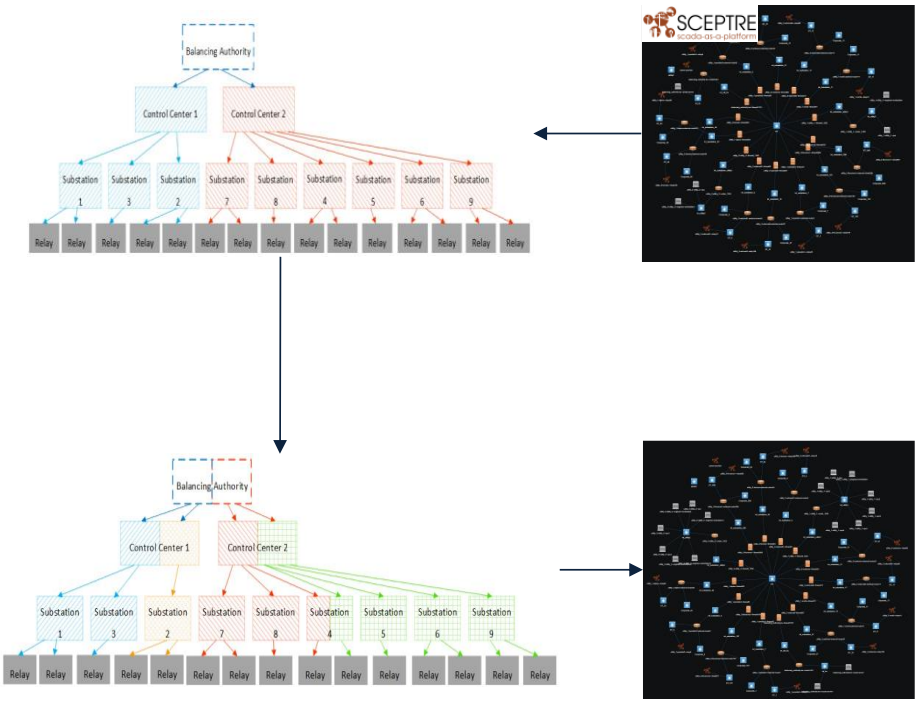


Figure 5-4: Optimal SCADA network segmentation workflow

In both cases, the topologies and worst-case attacks are evaluated by coupling Dakota to Scorch, using Dakota to specify CRASHOVERRIDE parameters. 100 combinations of CRASH parameters are evaluated in both cases, and the results are shown in Figure 5-5. The figure clearly shows a difference between the original segmentation scheme (in orange) and the optimal segmentation scheme (in blue). Future work should quantify this difference for additional attacker budgets and at larger scales and validate the optimality of the computed solutions.

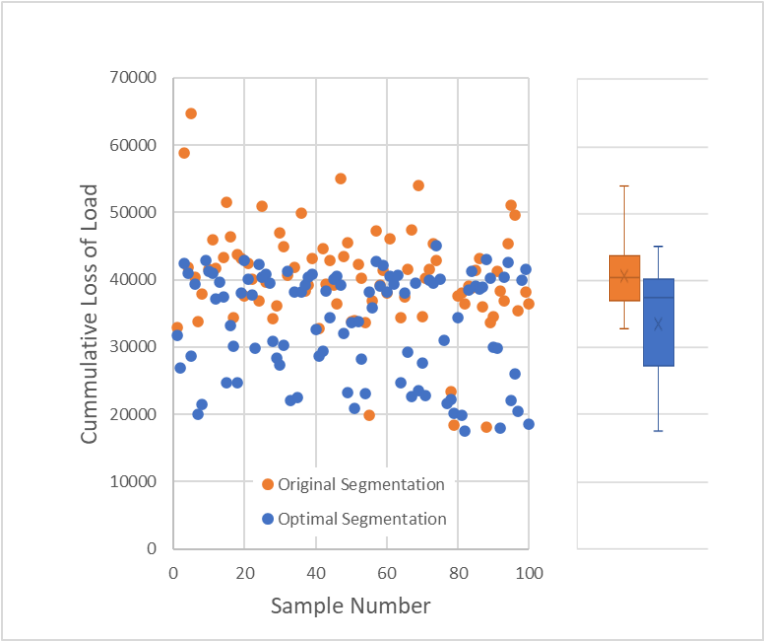


Figure 5-5: Results showing benefit of optimal segmentation

The SECURE team also conducted an uncertainty quantification (UQ) study to forward propagate uncertainty in the number of RTUs affected by CRASHOVERRIDE to uncertainty in the resulting loss of load. The study was conducted on the topology described above, with 49 field devices and the synthetic 2000-bus Texas power grid model, but with no network segmentation.

The results, shown in Figure 5-6, show a general trend toward increasing load loss with increasing numbers of RTUs targeted by CRASH, as indicated by the mean and median regression lines indicated in red and green, respectively. However, there is a significant amount of variance, particularly for smaller groups of RTUs, which makes good predictive regression difficult. However, if one were to consider regression on higher quantiles of results (say, 95th quantile), one would see less variance around these regressions, as shown in the top regression line in the figure. Furthermore, a 95th quantile estimation carries more meaning in terms of identifying, predicting, and planning for worst-case or tail events.

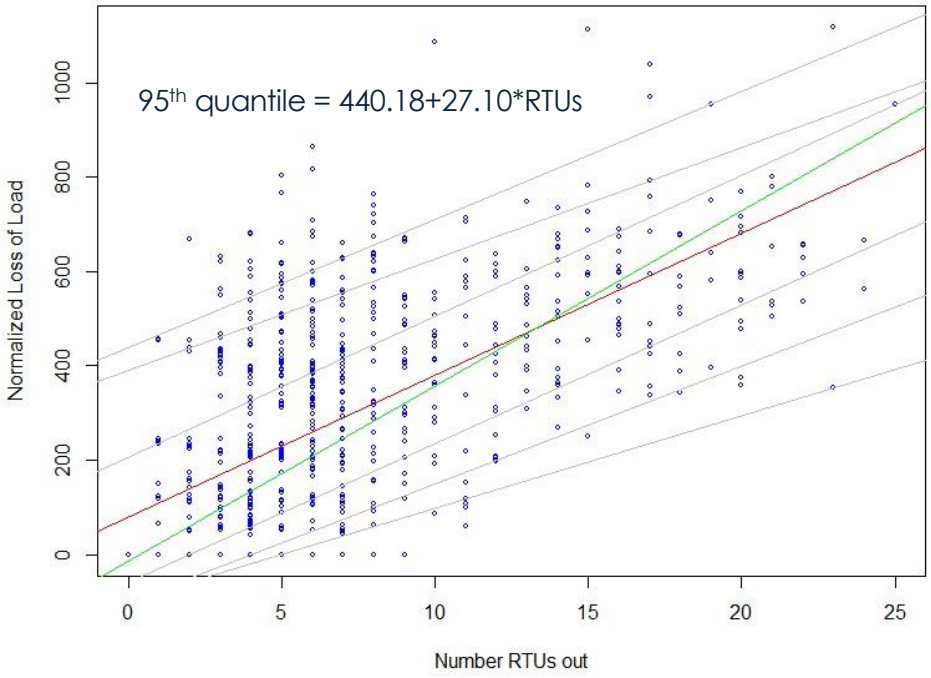


Figure 5-6: UQ study, showing regression for 95th percentile (representing worst-case tail outcomes)

More details about the SCADA network and power grid studies can be found in Appendix C.

5.4. Integrating the various pieces: Markov Model

The Markov threat modeling technique described in Section 2.2 provides a useful end-to-end analysis of attacker success relative to defender effectiveness. One example of the utility of Markov threat analysis was demonstrated midway in the SECURE project, when the Markov model shown in Figure

5-7 was constructed using SME- and literature-informed transition probabilities. (At the time, the SECURE project had little experimental data to populate in the Markov model.)

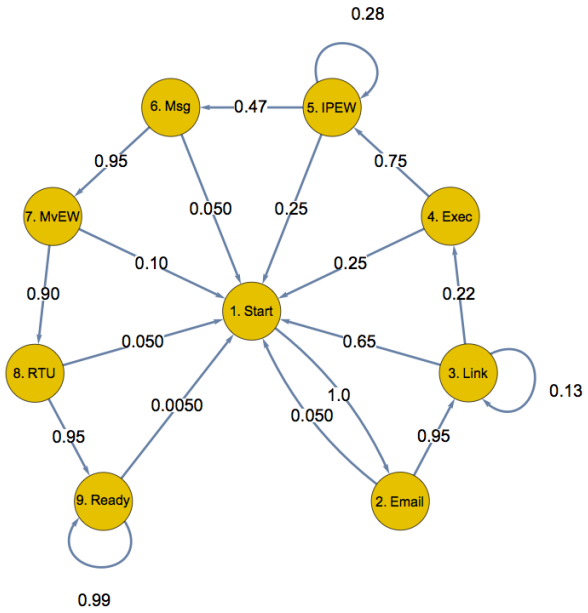


Figure 5-7: Baseline Markov model of APT-3 threat scenario

The baseline model in Figure 5-7 was analyzed to understand the effect of varying defender capabilities (modeled in the transition probabilities) on the attacker's ability to reach the Ready state (state 9), where the attacker could inflict power grid damage. For example, Figure 5-8 shows how a doubling in the Ready state detection probability, from 0.005 in panel (a) to 0.01 in panel (b), dramatically reduces the time the attacker can spend in that state. This effect is more comprehensively shown in panel (c).

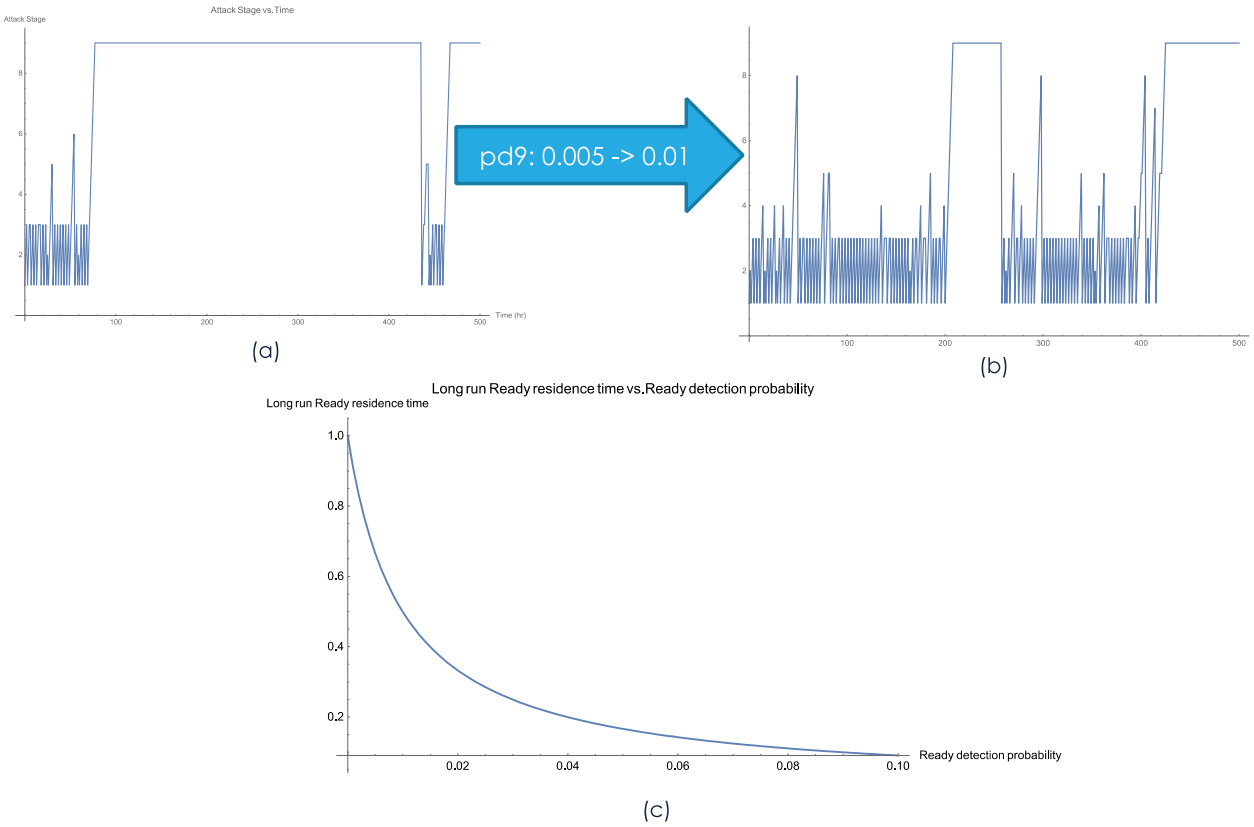


Figure 5-8: Change in Ready state (state 9) residence probability vs. Ready state detection probability

Next, the team considered a sensitivity analysis in order to understand where increases in detection capability provide the greatest benefit (to the defender). A similar analysis of Ready state residence time versus detection probability ***for all states*** was conducted, and the results are shown Figure 5-9.

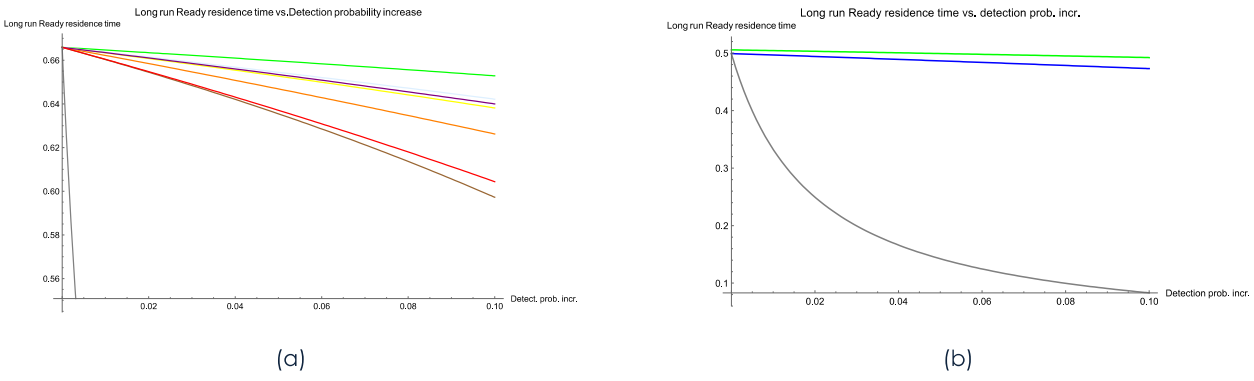


Figure 5-9: Sensitivity analysis, showing greatest benefit in Ready state detection relative to detection in other states

Panel (a) shows similar reductions in an attacker's Ready state residence time as the defender's detection probability increases for each state, except for detection in the Ready state (which shows a dramatic decrease). Panel (b) shows more detail for detection in the Ready state, relative to the other

states. From these results, a decision maker can conclude that it would be most cost-effective to focus defender resources on detection in the Ready state, i.e. detecting malware presence in the SCADA network.

More detail regarding Markov threat analysis and using it to guide defensive investments can be found in [34].

6. CROSS-CUT: VERIFICATION AND VALIDATION

Verification and validation (V&V) are critical activities performed when using computational simulations for the purposes of predicting high-consequence events. Under SECURE, we took V&V concepts from the physics and engineering community [5,14,29,31,1,2,33] and applied them to cyber emulation experiments. This is summarized below.

6.1. Verification

An important part of using emulation is verifying whether the emulation environment is working as intended, also called verification [22,33]. Part of verification involves software testing and quality assurance. A unique aspect of cyber emulation involves assessing the performance of the emulation running in the virtualized environment and determining whether there are sufficient resources to properly handle the scenario that is being run. If there are not, the virtualized components may produce experimental artifacts and behavior that result in the experimental outcomes being unrepresentative or incorrect.

Under SECURE, we focused on determining whether there are sufficient virtualized resources to support the emulation experiment and whether we can identify metrics that indicate when the results of an emulation experiment are unreliable. This work is inspired by previous work by Heller described in [13], but whereas Heller's work was with the Mininet experimentation environment (which uses Linux namespaces to model individual nodes), we considered virtual machines running in `minimega`. Furthermore, where Heller considered invariants (e.g. network timing characteristics that change predictably with changing experimental conditions), we consider a wider variety of host and virtual machine metrics. We refer to these metrics as telemetry metrics, following the usage of this phrase from Google [11], Microsoft [26], Intel [23] and others [38]. These companies use telemetry in the context of network monitoring metrics (e.g. monitoring traffic to and from VMs, including number and size of packets; round trip time for TCP flows), virtual machine resource usage (e.g. number of system processes, thread counts, memory committed and available, physical disk read and write time), and application monitoring (e.g. CPU utilization as a measure of performance).

We studied telemetry metrics relating to the performance of virtual machines which are used in a cyber emulation study and the physical machine hosting that study. We ran studies with two different scenarios under various levels of over-subscribed resources.[42] This first study was performed on a scanning/detection scenario, and then on a command and control scenario. Both studies involved `minimega` which was deployed on experiments with increasing numbers of namespaces using a single physical node on a High Performance Computing (HPC) system at SNL. A namespace is an experiment that is isolated in its own VLAN or set of VLANs. Each namespace has its own copy of each machine in the scenario networked through a unique set of VLANs. Thus, we were able to run multiple namespaces (e.g. namespaces = 1, 2, 4, ,5 …50) in parallel while the experiments remained isolated within their own namespace, effectively increasing the load on the HPC node in a well controlled fashion.

For the scanning/detection scenario, we found that the alert time distribution changed in a discernible way as a function of namespace: the distribution of alert times was much wider with a higher mean for namespaces 20, 33, and 50 [42]. We also found that the telemetry metrics of system load and throughput can be used to filter out replicates that had statistically different results than the one-namespace results which were used as the gold standard. The same telemetry metrics of system load and stolen cycles were also used to identify oversubscription problems in the C2 scenario.

6.2. Validation

Validation addresses the question of adequacy of the model: is the model accurate enough and appropriate to be used for a prediction? [33] Typically, validation involves the comparison of the model with observational or experimental data using statistical metrics called validation metrics [33, 25].

Under SECURE, we performed two research studies relating to validation:

1. Reproducibility. The main question in reproducibility is can one reproduce cyber emulation results generated on one testbed (e.g. Sandia's `minimega` emulation running on an HPC using the SCORCH orchestration tool) on another testbed (e.g. Texas A&M University's CORE testbed running in their Resilient Energy Systems Laboratory, RESLab). The two emulation environments are built on fundamentally different technology: `minimega` uses VMs but CORE uses containers.

Reproducible cyber experimentation is essential to assure valid, unbiased results across cyber testbeds. Even minor differences in setup, configuration, and testbed components can have an impact on the experiments, and thus, reproducibility of results. In collaboration with TAMU, we performed a set of reproducibility experiments for the scanning/detection scenario.[41] The details are in the paper presented at the 14th Cyber Security Experimentation and Test workshop[41]; we note that as part of this study, we examined four statistical metrics: the t-test, the Kolmogorov-Smirnov (K-S) test, the area metric, and the Relative Hausdorff metric.

The following summarizes the lessons learned from the reproducibility study:

- Even after providing a comprehensive writeup and details of the experiment, both teams still required significant coordination to reproduce the experiment.
- It can be challenging to determine if small differences are due to differences in the hardware/emulation platform OR due to an implementation detail that is not correctly reproduced. Therefore, subject matter expertise is critical.
- Statistical tests and ensembles of replicate results can help in this comparison as they provide some estimate of the uncertainty inherent in the results on one platform.
- We recommend public repositories for experimental artifacts. One example is the SEARCCH project: Sharing Expertise and Artifacts for Reuse through Cybersecurity Community Hub project (https://searcch.cyberexperimentation.org/)
- We need consensus in artifacts and how testbed technologies use them
- We need to understand differences between common cyber experimentation platforms, to account for these differences when determining whether an experiment is reproduced
- Appropriate distance metrics should be developed, depending on the experiment question and objective

2. Physical validation.

We conducted two sets of physical validation experiments. The first set used physical nodes in a HPC cluster at Sandia, and compared the experimental results from these physical nodes with the results from the `minimega` virtual machine testbed. These results compared very well, in part because both experiments used the exact same operating systems. Therefore, we conduced a subsequent series of physical validation experiments on physical hardware at TAMU's RESLab. For these experiments, we again used the scanning/detection scenario and used the same statistical metrics we found most

useful from the reproducibility study: the K-S test and the area metric. Instead of comparing `minimega` to CORE (reproducibility study) or physical HPC nodes, we compared `minimega` to RESLab. The RESLab experiments involved physical RTUs modeling the open ports in the scanning/detection scenario, but used CORE emulation components modeling the rest. Thus, it was not a purely physical system used for validation, but the physical units were modeling the most important components of interest: the open, vulnerable RTUs. The preliminary results from this exercise, which involved 1000 runs from TAMU and 1000 runs from SNL under a variety of conditions, show that we can validate Sandia's `minimega` emulation against a hybrid physical system at TAMU.[40]

7. CROSS-CUT: TOOLS

Several software tools were developed and/or used as part of the SECURE project. A short description of these is given below.

7.1. SCORCH

SCORCH is a software tool that manages the deployment of cyber experiments. The key benefits of SCORCH are that 1) it will configure the experiments and 2) it is able to collect and store the outputs, thereby speeding up analysis time and reducing manual error. SCORCH is an automated scenario orchestration framework for emulation-based models that also utilizes `minimega`.

7.2. Minimega/SCEPTRE

`minimega` (https://minimega.org/) is an open source distributed Virtual Machine (VM) management tool used for launching and managing virtual machines locally or across a cluster [4]. `minimega` is fast, easy to deploy, and can scale to run on massive clusters with virtually no setup. It is scalable and able to support studying both small and very large VM networks. `minimega` is designed to give you low-level control of all the fine details when it comes to setting up and running VMs and has now been pulled into other tools, e.g. SCEPTRE, to take care of the low-level features of spinning up VMs.

SCEPTRE is an application that uses an underlying network emulation and analytics platform to model, simulate, emulate, test, and validate control system security and process simulations. Traditionally, tools and techniques for simulating and emulating control system field devices have been limited because the physical processes being monitored and controlled are omitted. SCEPTRE leverages proven technologies and techniques to integrate the end device and process simulations, with control hardware-in-the-loop (HIL), providing an integrated system capable of representing realistic responses in a physical process as events occur in the control system, and vice versa. SCEPTRE is a proven control system environment platform, having been fielded for many R&D applications, operational joint tests, and exercises supporting testing, training, validation, and mission rehearsal.

SCEPTRE is comprised of simulated control system devices, such as remote terminal units (RTUs), programmable logic controllers (PLCs), protection relays, and simulated processes, such as electric power transmission systems, refinery processes, and pipelines. The simulated control system devices are capable of communicating over Internet Protocol (IP) networks using standard SCADA protocols such as Modbus, DNP3, IEC 61850, and others. SCEPTRE also includes support for HIL, wherein real field devices under study (i.e. a specific model of PLC) can be connected to and interact with the physical process being simulated. This allows the user to include high fidelity systems where they are needed without sacrificing scalability. SCEPTRE provides an analysis capability for assessing and improving the cyber security of control systems used in the energy sector and DoD. The SCEPTRE platform provides an environment where hardware and software upgrades and new mitigations can be evaluated before installation in an operational environment.

7.3. Elasticsearch

Elasticsearch (https://www.elastic.co/elasticsearch/) is an open source tool for storing large amounts of data in a highly searchable way that is amenable to a variety of data types and structures. Under SECURE, Elasticsearch was leveraged for data storage and retrieval during the Validation and Verification studies. These studies required large amounts of data to be stored, sorted, and easily

searchable. Using Elasticsearch allowed for storage of varied data types and structures, easy conversion of data to and from JSON format, and simple querying.

7.4. PAO/Pyomo

PAO is a Python-based package for Adversarial Optimization. The goal of this package is to provide a general modeling and analysis capability for bilevel, trilevel and other multilevel optimization forms that express adversarial dynamics. Many planning situations involve the analysis of a hierarchy of decision-makers with competing objectives. For example, the cyber-grid applications developed in the SECURE Grand Challenge consider the behavior of attackers and defenders, where defenders wish to protect their cyber infrastructure and execute power grid operations to meet expected energy demands, and attackers wish to maximally disrupt grid operations. Thus, these cyber-grid applications can be naturally modeled as bi-level and tri-level optimization problems, where decision-makers need to account for the behavior of adversaries at a lower-level.

SECURE researchers developed tailored optimization solutions for cyber-grid applications using the Pyomo modeling environment, which are analyzed with commercial and open source optimization solvers. Concurrently, PAO was developed to automate these tailored solutions to future applications that share similar structure. PAO extends the modeling concepts in the Pyomo algebraic modeling language to express problems with an intuitive algebraic syntax. Additionally, PAO supports compact problem representations that simplifies the implementation of solvers for bilevel, trilevel and other multilevel optimization problems. PAO currently includes four solver interfaces that are applicable to different classes of adversarial optimization problems.

- **Pyomo**
  - GitHub repository: https://github.com/Pyomo/pyomo
  - Online documentation: https://pyomo.readthedocs.io/en/latest/
  - Bynum, M., G. Hackebeil, W. E. Hart, C. Laird, B. Nicholson, J. Siirola, J.-P. Watson, and D. L. Woodruff. (2021) Pyomo: Optimization Modeling in Python. 3rd. Springer.
- **PAO**
  - GitHub repository: https://github.com/or-fusion/pao
  - Online documentation: https://pao.readthedocs.io/en/latest/
  - Hart, W. E., A. Castillo, E. S. Johnson, and S. Punla-Green (2021). PAO 1.0: A Python Library for Adversarial Optimization. Tech. rep. SAND 2021–6720. Sandia National Laboratories.

7.5. Dakota

Dakota is a suite of iterative mathematical and statistical methods that interface to computational models or simulations (https://dakota.sandia.gov). Dakota's goal is to make parametric explorations of models practical to support design, analysis, or test cycles. Dakota is an open-source software toolkit and has algorithms to enable design exploration, model calibration, risk analysis, and quantification of margins and uncertainty with computational models. Dakota seeks to enhance the use of computational models with a variety of iterative analyses (running the model multiple times depending on the objective of the study) so that models may be used not just for single-point solutions, but also achieve broader impact in the areas of credible prediction and optimal design.

Related to SECURE, there is an extensive suite of uncertainty analysis methods in Dakota, including a variety of sampling methods (Monte Carlo, Latin Hypercube Sampling, quasi-Monte Carlo methods, design of experiments, fractional and full factorial designs), sensitivity analysis methods, reliability methods, stochastic expansion methods such as polynomial chaos, epistemic uncertainty approaches including interval analysis and Dempster-Shafer evidence calculations, and Bayesian calibration methods, and multifidelity uncertainty methods. These are summarized in:

- L. P. Swiler, B.M. Adams, and M.S. Eldred, "Dakota: Bridging Advanced Scalable UQ Algorithms with Production Deployment." In Springer Handbook on Uncertainty Quantification, Ghanem R., Higdon D., Owhadi H. (eds) (2015). https://doi.org/10.1007/978-3-319-11259-6_52-1.

8. RECOMMENDED WORKFLOW

Given the variety of tools that can be used to assess cyber systems, experimentalists might be tempted to dive right into a study. However, an analysis rigorous enough for use in high-consequence cyber systems requires a carefully thought-out experimental design. This section describes the experimentation workflow developed and used by the SECURE research team while conducting its studies of power grid cyber effects.

The workflow presented in this document is primarily focused on emulation testbed modeling, although it may be employed for other types of cyber models. Thus, to facilitate the discussion, we define the following terms:

- *Cyber Model* – a generic term that can apply to any methods (or combinations of methods) used to assess cyber systems
- *Cyber Testbed* - the hardware platform and software framework used to run a cyber model or combination of cyber models.
- *Physical Model* - Cyber models that run real software on a representative hardware platform to model the actual system in full fidelity.
- *Emulation Model* – Cyber models that run real software in real time on a computing cluster, using hardware abstractions such as virtual machines and/or containers to represent individual nodes, and virtual networking technologies such as Virtual Local Area Networks (VLANS) to interconnect VMs or containers.
- *Emulation Testbed* – (also known as "virtual testbed") Resources (e.g. computing cluster, virtualization technologies, and experimentation/orchestration software) used to instantiate emulation models.
- *Simulation Model* – primarily discrete event simulators (e.g. OMNET++ [18] or ns-3 [21]), which run abstract representations of software and hardware. These models can run faster than real time.
- *Mathematical Model* – Mathematical formulas that capture dynamic and/or steady-state values of a quantity of interest and can be solved using mathematical analysis tools such as Matlab or Mathematica.

Figure 8-1 shows a spectrum of testbeds employed in the modeling of cyber systems and associated tradeoffs in terms of realism vs. cost.

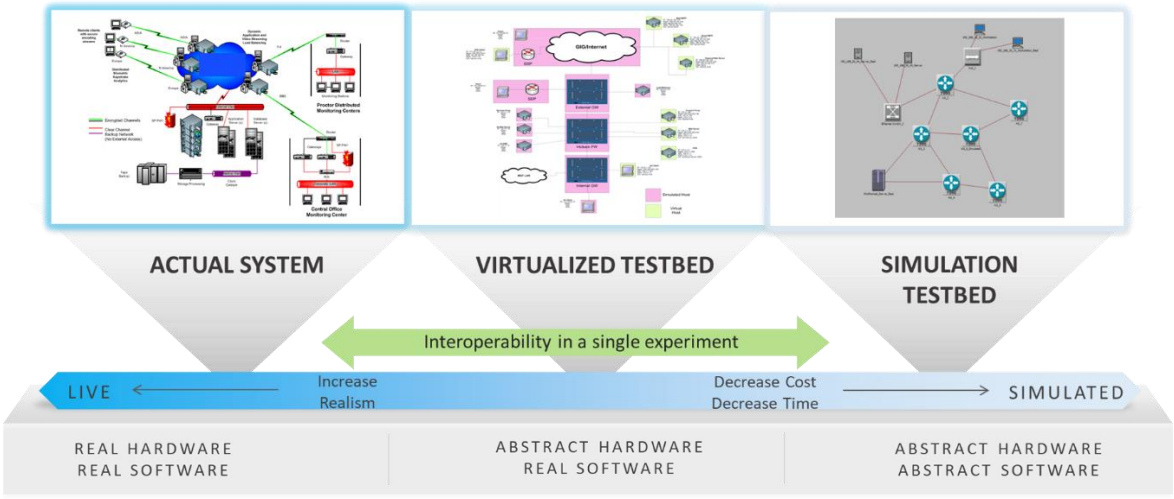


Figure 8-1. Spectrum of cyber model fidelity, ranging from actual system to simulation testbeds.

Because the topic of experimental design for emulation models is an active area of investigation in the cyber-security research community, several frameworks have been developed to help facilitate sound experimental practices and generate reproducible results. For example, the DEWs (Distributed Experiment Workflows) [27] provide generic descriptive language to encode the scenario and topology for an experiment. Likewise, DARPA's National Cyber Range [7], Emulab [37], and DETER [28] are cyber testbeds that can be used for research and experimentation on networks. Reference [25] also examines how platform variations affect emulation models, using carefully structured experiments and statistical analysis. Although these tools exist and work well for experiments, methods for using them rigorously to provide comprehensive evidence to answer questions about high-consequence systems have not been developed and characterized. For example, reproducibility in cyber experiments remains a challenge, due to small timeframes, implementation differences, and differences in platform configurations. Therefore, to facilitate the achievement of reproducible, unbiased results and methods that may be readily applied in other contexts (e.g. on other cyber testbeds with differences in operating systems, software and hardware, kernels, system resources, etc.), the SECURE project developed the following workflow to help guide future studies, as shown in Figure 8-2. We acknowledge that this workflow was designed for an experimental model (to study sensitivity and uncertainty analysis) but note that it can be applied more generally to generate ensembles of runs that can support optimization studies or other studies. Further detail and a description of SECURE's experimental design (especially the design of experimental runs) can be found in "Design of Experiments for Cyber Emulation" [39].

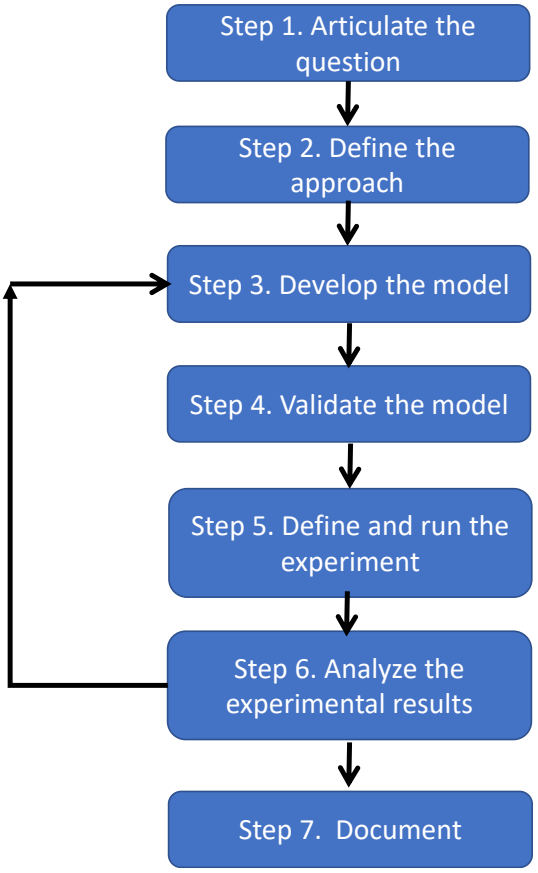


Figure 8-2. Recommended workflow for cyber modeling suggested by the SECURE project

When performing cyber modeling experiments, we recommend that the following workflow be used:

1. **Clearly articulate the question.** Be specific. (e.g. "If an attacker uses port scans and a given configuration of the Nmap scanning tool, how many alerts will our intrusion detection software identify in a 60-second window?" NOT "Will our intrusion detection software work efficiently?") If possible, identify what statistics are of interest (e.g. the average number of alerts in a time window, the probability that there will be more than 10 alerts, or the full distribution of alerts).

2. **Define the approach that best answers the question.** Scope the problem, identify inputs and outputs, and consider your modeling options.
    a. Identify your requirements (e.g. fidelity, scale, size of parameter space, desired variance in outputs, time per replicate, number of replicates). Most cyber models, require multiple runs per model configuration setting (i.e. multiple replicates), because there is inherent variability or stochastic behavior in each replicate, due to small timing differences, ordering of various events happening on the system, etc.
    b. Choose your modeling domain(s) (e.g. emulation, mathematical), noting that your choice of modeling domain should depend on the model requirements identified in Step 2(a), as shown in Figure 8-2. For example, if a large scale is required, scalable modeling technologies (e.g. emulation, simulation, or mathematical modeling) would

be more practical than physical testbed modeling; however, if high fidelity is required, then physical or emulation testbed modeling would be more effective than simulation or mathematical modeling. Of course, a combination of technologies can be used to maximize outcomes (e.g. a coupled model or models at multiple levels of detail in a multifidelity modeling study).

c. Define how each modeling activity contributes to the answer.

3. **Develop the model, depending on the modeling domains.** The developmental activities for different types of models will vary by model:
    - *Mathematical models* develop equations that will be solved, typically as a function of time (e.g. traffic might be modeled with a Poisson arrival rate distribution to calculate the expected number of packets arriving in a particular time step).
    - *Simulation models* use discrete event network simulators, which often have simulation examples and model libraries (e.g. with different routing protocols, network traffic, etc.) that can be used as building blocks; however, the configuration of the simulation must typically be customized for the scenario of interest to the study.
    - *Emulation models* bear some similarity to simulated models, but the actual software components and virtualized hardware components (e.g. routers, servers, workstations, NIC cards, etc.) must be explicitly identified. The emulation platform we used for SECURE was `minimega` [17]. Below we specify steps that are fairly general and need to be customized for a particular emulation platform and experiment.
        i. Define or import the topology
        ii. Develop the application components, if needed
        iii. Define the experimental behaviors that will be investigated
        iv. Develop a data collection strategy
        v. Set up and verify the configuration
        vi. Obtain the resources to run the model

4. **Validate the model.** Compare the model to higher fidelity representations and/or to independently developed models of similar fidelity, to assess the degree of agreement between your model and the benchmark. Choose the comparison metrics that best expose the statistics of interest (e.g. differences due to virtual machine artifacts). A high-fidelity model (e.g. simulation or emulation) should ideally be benchmarked against an actual physical system, as in [24]. However, lower-fidelity models (e.g. mathematical) might be benchmarked against higher-fidelity models. Any large and/or systemic differences between your modeled data and the benchmark data should be investigated before the experiment progresses.

At present, there is no standard for benchmarking cyber emulations; the current best-practice is a hierarchical validation, which occurs in stages, as shown in Figure 8-3. First, the components and/or attack steps are validated individually, then larger groupings or components are validated, and then the entire system is validated. Figure 8-3 depicts the validation of a cyber-attack model, but a similar validation process could be applied to any kind of performance issue or behavior.

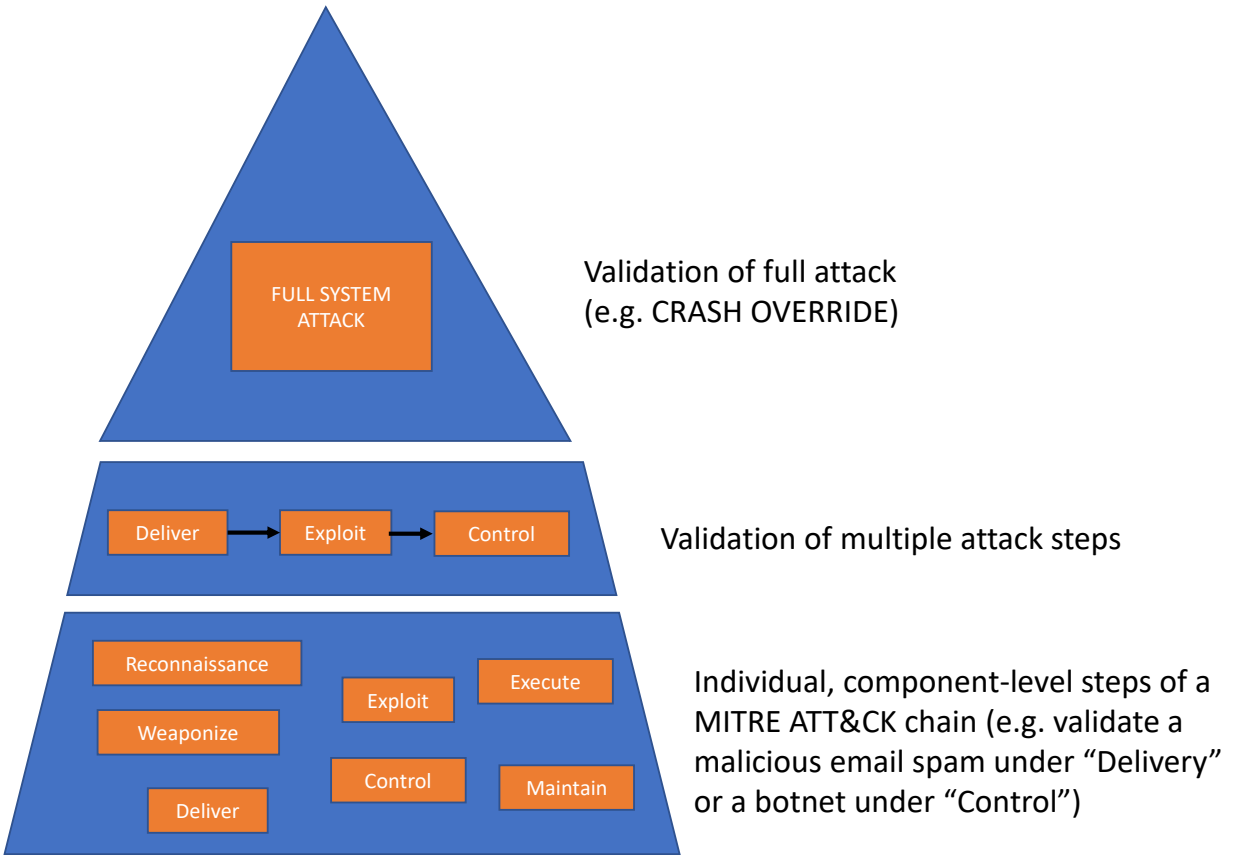


Figure 8-3. Hierarchical validation for a cyber system, starting with validation of individual attack steps at the bottom and proceeding to validation of the full attack at the top.

5. **Define and run the experiment.** Define the inputs/outputs for your model and specify them in a configuration file for an experimental orchestrator (e.g. Scorch or Dakota [16]). Choose an experimental design that will produce an appropriate list of input/output parameter settings:
   a. Define the inputs that will be varied in the experiment and specify the distribution of possible values for each input (e.g. discrete bandwidth values, uniformly distributed traffic generation rates between upper/lower bounds, etc.). Each input that will be varied in the experiment should have a specification of its distribution in a parametric or empirical distribution form.
   b. Define the outputs that will be extracted from the experiment. These outputs can take the form of detailed experimental data (e.g. packet captures and logfiles sent to an Elasticsearch/Logstash/Kibana (ELK) data collection node [19]), and/or summarized experimental outputs calculated within the experiment as it executes (e.g. the time at which an intrusion detection system generates an alert).
   c. Develop the experimental design. This can be done in a variety of ways [39, 36]. If the number of inputs is small (1-5) and each input has only 2 or 3 levels, a full factorial design can be run involving all combinations of input parameter levels. If the inputs are specified with continuous distributions, Monte Carlo sampling or more efficient alternatives such as Latin Hypercube sampling or quasi-Monte-Carlo space-filling methods can be used to

generate samples. In each of these cases, the number of samples should typically be at least 10x the number of input parameters.

d. Define the number of replicates per design point. At each point in the experimental design space (e.g. input 1 is at value A, input 2 is at value B, etc.), it may be necessary to run the model multiple times, where each model run is a replicate. If the model is deterministic (e.g. running at one setting of parameter inputs always gives the same results), then it is only necessary to run the model once per parameter setting. However, many cyber models are stochastic due to slight variations in timings of processes and order of operation executions. In this case, one setting of the parameter inputs should be run with replicates to obtain statistics on the response for that parameter setting.

e. Run the model. Once the experimental design is identified, it produces a list of input parameter settings at which the cyber model should be run. This list is given to the experiment orchestrator (e.g. Scorch, Dakota). The next step is to run the cyber model at these settings. For each parameter setting, the model may be run once or some number of times (multiple replicates), depending on whether the model is deterministic or stochastic.

6. **Analyze the experimental results.** Use your data to generate a table (as an Excel spreadsheet, a data structure in a Python analysis script, a table in Elasticsearch, a table in Minitab [20], etc.) and organize the results (where the rows are each run of the cyber model, the first set of columns are the input parameters, and the second set of columns are the outputs) for further analysis.
   a. Verify results. Depending on the experimental design and the available benchmarks, choose the most appropriate validation method (e.g. scatterplots of inputs v. outputs, calculation of basic statistics on the outputs, etc.).
      i. (Optional) If the values obtained in Step 6(a) are orders of magnitude different from the benchmark values, revisit Step 3.
   b. Assess convergence
      i. (Optional) If the values obtained in Step 6(b) are orders of magnitude different from the benchmark values, revisit Step 3.
   c. Determine conclusions/insights. Employ statistical analysis methods appropriate to the experimental design (e.g. main effects analysis for full factorial designs with discrete input levels, correlation analysis, standardized regression analysis, and/or Sobol variance-based indices for designs with continuous input distributions). Statistical tests (e.g. t-tests or Kolmogorov-Smirnov tests) can be used to compare the results gathered from different tests, scenarios, platforms, or emulators.

7. **Document.** Document your results comprehensively so that they will be fully useful and reproducible for subsequent researchers.
   a. Question(s). List the question(s) addressed in the study.
   b. Methods. Define each step of the methodology, with enough detail that the study can be easily replicated.
   c. Analysis. Describe the analyses performed.
   d. Results. Report the complete results, including tables of raw data.
   e. Conclusions/Insights. Highlight the conclusions/insights gained from the study.

9. PROJECT METRICS

9.1. Publications & Presentations

A list of publications and presentations from the SECURE project is listed below. One special item that we wish to highlight: a web-based Handbook has been developed to archive some of the research concepts and results of SECURE. The Handbook was created in part from feedback from the External Advisory Board. A goal of the Handbook is to highlight the SECURE project and results to the external community.

9.1.1. Papers completed

1. Castillo, B. Arguello, G. Cruz and L. Swiler, "Cyber-Physical Emulation and Optimization of Worst-Case Cyber Attacks on the Power Grid," 2019 Resilience Week (RWS), 2019, pp. 14-18, doi: 10.1109/RWS47064.2019.8971996. SAND 2019-12468 C.
2. Pinar, Z. Benz, A. Castillo, W. Hart, L. Swiler, T. Tarman, "SECURE: An Evidence-based Approach to Cyber Experimentation," IEEE Resilience Week: https://ieeexplore.ieee.org/document/8971976 ,
3. Vugrin, J. Cruz, C. Reedy, T. Tarman, and A. Pinar "Cyber Threat Modeling and Validation: Port Scanning and Detection," *Proceedings of the 7th Annual Hot Topics in the Science of Security (HoTSoS) Symposium.*
4. Geraci, G., L.P. Swiler, J. Crussell, B. Debusschere. "Exploration of Multifidelity approaches for Uncertainty Quantification in network applications." *Proceedings of 3rd International Conference on Uncertainty Quantification in Computational Sciences and Engineering* in Crete, Greece, June 2019. SAND2019-3274C.
5. Acquesta, L. P. Swiler, and A. Pinar ,"Time Series Dimension Reduction for Surrogate Models of Port Scanning Cyber Emulations.". SAND20-10617.
6. Geraci, G., Crussell, J., Swiler, L.P. and Debusschere, B. J. "Exploration of Multifidelity UQ Sampling Strategies for Computer Network Applications." *International Journal of Uncertainty Quantification*, 2021. Pp. 93-118. DOI: 10.1615/Int.J.UncertaintyQuantification.2021033774. SAND2021-1221J.
7. Gabert, A. Pınar, and U. Catalyurek, "Computing Hierarchical Dense Structures on Dynamic Graph Streams,"ACM Intl. Conf. on Web Search and Data Mining (WSDM) 2021
8. Tarman, T. Rollins, L.P. Swiler, J. Cruz, E. Vugrin, H. Huang, A. Sahu, P. Wlazlo, A. Goulart, and K. Davis. Comparing reproduced cyber experimentation studies across different emulation testbeds. *USENIX 14th* Cyber Security Experimentation and Test (CSET) Workshop. Aug. 9, 2021. SAND2021-5696C.
9. Gabert, Y. Ozkaya, K. Sancak, A. Pinar, and U. Catalyurek, "ElGA: Elastic and Scalable Dynamic Graph Analysis," to appear in SC'21.
10. Gabert, A. Pinar, and U. Catalyurek, "Shared-Memory Scalable k-Core Maintenance on Dynamic Graphs and Hypergraphs," in *IEEE ParSocial* 2021.
11. Ozkaya, F. Balın, A. Pınar, and U. C ̧atalyu ̈rek, SSGG: A scalable graph generation algorithm to sample over a given shell distribution in Proc. IPDPS Workshops, W. Graph Learning.
12. Stickland, J. Li, T.D. Tarman, L.P. Swiler. Uncertainty Quantification in Cyber Experimentation. SAND2021-5710C.
13. Malashkhia, L. Swiler, A. Pinar, and Y. Wang, "A Robust Control Scheme for Time Delay Switch Attacks," to appear in AMSec'21 Workshop on Additive Manufacturing (3D Printing) Security.

9.1.2. Papers under review

1. Arguello and E. Johnson and J. Gearhart, "A Trilevel Model for Segmentation of the Power Transmission Grid Cyber Network." SAND 2021-10208 O, submitted for journal publication, also available as arXiv.2108.10958: https://arxiv.org/abs/2108.10958 and Optimization Online: http://www.optimization-online.org/DB_HTML/2021/08/8562.html
2. Outkin, T. Schulz, T. Tarman, P. Schulz, A. Pinar. Defender Policy Evaluation and Resource Allocation against MITRE ATT&CK Data and Evaluations, submitted for journal publication.
3. Johnson and S.S. Dey, "A scalable lower bound for the worst-case relay attack problem on the transmission grid," submitted for journal publication, available at arXiv.2105.02801. SAND 2021-10211 O.
4. Ozkaya, A. Pinar, and U. Catalyurek, "TRIGGER: TempoRal Interaction Graph GenEratoR," submitted for conference publication.
5. Cheramin, J.Cheng, R. Chen, and A.Pinar, "Data-Driven Robust Optimization Using ScenarioInduced Uncertainty Sets," submitted for journal publication.
6. Vugrin, E. and S. Hanson, J. Cruz, C. Glatter, T. Tarman, and A. Pinar. "Detection of command and control traffic: model development and experimental validation." Submitted for conference publication.
7. Hanson, S. and G. Cruz. "SCORCH User Guide." In preparation as a SAND report, 2021.
8. Gabert, A. Pinar, and U. Catalyurek, "Coreness to Cores: Batch Dynamic Algorithm to Efficiently Find k-Cores," submitted for conference publication, submitted for conference publication.
9. Emma Johnson, Santanu Dey, Jonathan Eckstein, Cynthia Phillips, John Siirola, "A Covering Decomposition Algorithm for Power Grid Cyber-Network Segmentation," submitted for journal publication.
10. She'ifa Punla-Green, John Mitchell, Jared Gearhart, William Hart, Cynthia Phillips, "Shortest Path Network Interdiction with Asymmetric Uncertainty,: submitted for journal publication.

9.1.3. *Technical Presentations*

1. Tom Tarman, SECURE overview, Texas A&M University,. December 2018
2. Ali Pinar, "SECURE: An Evidence-based Approach to Cyber Experimentation," IEEE Resilience Week, October 2019
3. Bryan Arguello, "Cyber-Physical Emulation and Optimization of Worst-Case Cyber Attacks on the Power Grid," IEEE Resilience Week, October 2019. SAND 2019-12468 C.
4. Tom Tarman , "Cyber Experimentation," University of Texas at San Antonio, March 2019
5. Bryan Arguello, "Talk: Bilevel Optimization of Cyber Physical Models for Power Grid Resilience," INFORMS Annual Meeting, October 2019. SAND 2019-12885 C.
6. Laura Painton Swiler, "Uncertainty Quantification in Cyber Emulation," INFORMS Annual Meeting, October 2019
7. Eric Vugrin, Gerardo Cruz, Christian Reedy, Alexander Outkin, Vincent Urias, Thomas Tarman, "Cyber Threat Modeling And Validation," INFORMS Annual Meeting, October 2019
8. Ali Pinar, "Rigorous Cyber Experimentation for Security of Cyber Physical Systems," INFORMS Conference on Security, February 2020.

9. Bryan Arguello, Emma Johnson, Jared Gearhart, "A Trilevel Cyber-Physical Power System Network Segmentation Model," INFORMS Annual Meeting, November 2020. SAND 2020-11077 C.
10. Bert J. Debusschere, Gianluca Geraci, John D. Jakeman, Cosmin Safta, and Laura Swiler, "Polynomial Chaos Expansions for Discrete Random Variables in Cyber Security Emulytics Experiments", SIAM CSE 2021 (virtual), March 1, 2021. SAND 2021-2270 C.
11. Tom Tarman, Comparing reproduced cyber experimentation studies across different emulation testbeds. *USENIX 14th Cyber Security Experimentation and Test (CSET) Workshop.* August 2021.
12. Ali Pinar, "Cyber Security: A new frontier for computational science and engineering," Institute for Mathematics and its Applications (IMA), U. Minnesota, February 2021
13. Ali Pinar, "SECURE: An Evidence-based Approach to Cybersecurity," 2019 Graph Exploitation Symposium, MIT Lincoln Labs, MA, April 2019.
14. Ali Pinar, "Rigorous Cyber Experimentation for Science of Security," Lab Research Technical Exchange, May 2021.
15. Bryan Arguello, Jared Gearhart, Emma Johnson, Santanu Dey, "Trilevel Programming for Network Segmentation of Power System Cyber-Physical System," INFORMS Annual Meeting, October 2021. In R&A.
16. Emma Johnson, Santanu Dey, "A Scalable Lower Bound for the Worst-Case Relay Attack Problem on the Transmission Grid," INFORMS Annual Meeting, October 2021. SAND 2021-11154 C & SAND 2021-11196 V.
17. Ali Pinar, "SECURE: Science of Security by Rigorous Experimentation," October 2021
18. Ali Pinar, "Principled Methods for Quantified Security for High-Consequence Cyber Physical Systems," Workshop on Military Communications, December 2021.

9.2. Conference & Workshop Organization

Despite the limitations imposed by the COVID-19 pandemic, our work has been presented in many conferences and workshops and the team members have contributed to organizations of these meetings in leadership roles. Of note here, is the Cyber Experimentation and Science of Security Workshop (CESoS'21), which will be held in November 2021. This planned workshop will bring together leading researchers in the field and will give Sandia the opportunity to present the SECURE work and introduce its new SCIRE institute: "Sandia's Cyber security Institute for Rigorous Experimentation."

9.3. Mentoring and Training

SECURE is an interdisciplinary project. Cross-training team members on other disciplines to form a well-functioning team has been a big part of our effort and a bigger part of our success. We believe this interdisciplinary culture built under SECURE will have an enduring effort at the Lab.

The team comprised eight early career staff and many more senior researchers stepped in for higher roles. The project also supported dissertation studies of 3 PhD students. Two of these students are already Sandians and we are optimistic that the third will join Sandia upon graduation.

9.4. Team Building & Partnerships

The project helped with building many external partnerships. In addition to connecting with many individuals, we expect that our partnerships with Georgia Institute of Technology, Texas A&M University, University of Southern California Information Sciences Institute, and University of California at Davis, as well as at Pacific Northwest National Laboratory and other national laboratories, will be long-lasting productive partnerships which can position Sandia as a leader in rigorous cyber experimentation.

10. SUMMARY AND PROJECT LEGACY

Judgments about the security of high-consequence cyber systems require hard evidence, quantified uncertainties around the evidence, and rigorous experimental methods to produce that evidence. The SECURE Grand Challenge LDRD project was proposed to address a significant cybersecurity experimental gap between rapidly maturing testbed technologies (e.g. `minimega`, DETER, amd cloud computing technologies) and emerging R&D in the "science of cybersecurity" [3,6,15]. This is work is not being performed elsewhere, and is appropriate for Sandia National Laboratories to pursue, based on its multi-decade heritage as a nuclear weapons engineering laboratory responsible for the HPC codes and simulations that inform decisions regarding high-consequence nuclear weapon systems.

The inspiration behind our approach is the success and impact of computational science and engineering (CSE), specifically on Sandia's nuclear stockpile stewardship mission, and broadly on the scientific community. Cyber experimentation can provide the predictive capability as the scientific computing models do for physics-based systems, and thus there are a lot of parallels that we can draw inspiration from. However, cyber systems are much different than physics-based systems, due to lack of closed-form equations, discrete nature of systems and extreme nonlinearities. Due to these differences, we cannot expect traditional CSE methods to work well on cyber systems; we promised to invent new methods that can provide similar capabilities for cyber systems.

SECURE successfully delivered on this promise. The SECURE external advisory board (EAB) acknowledged that SECURE addressed this important niche with rigor, and that only a national laboratory could credibly take on this challenge. As evidence of the success of SECURE, several "firsts" were realized during this project, including:

- Multifidelity uncertainty quantification in a cyber experiment context.
- The development of an integrated software package (PAO) for expressing and solving adversarial optimization models.
- End-to-end cyber exemplars that integrate emulation, modeling, and uncertainty quantification to rigorously analyze cyber security problems.
- Developed methodology for tri-level interdiction, allowing tractable solutions for optimal segmentation at large scales
- Developed automated methods to import synthetic (and potentially real) cyber power grid topologies into SCEPTRE experiments, enabling scaled studies.
- Developed integrated workflow of orchestration (Scorch), design of experiments (Dakota), and emulation (SCEPTRE) to enable rigorous cyber experimental studies.
- Use of optimization to focus (computationally) expensive emulation experiments on optimal parameter regimes. For example, optimal segmentation was paired with emulation to identify optimal topologies and adversary scenarios.
- Development of a novel telemetry-based verification approach for emulation-based testbeds, inspired by previous work with the Mininet experimentation environment [13].

- Polynomial chaos expansions over "mixed" discrete and continuous variables in a cyber experiment context, enabling efficient sensitivity analysis.
- Use of statistical tests at each time step of an experiments with large number of experimental replicates to carefully assess similarity between cyber experiments run on different emulation testbeds or to perform emulation vs. physical testbed comparisons.
- Applied Markov modeling as an integrated framework for end-to-end integration of attack success probabilities at each step. We extended it to analyze attacker/defender capabilities as well as to incorporate information from both MITRE ATT&CK framework and emulations in the following ways:
    a. Understanding attack evolution over time and handling time-unbounded attacks,
    b. Using the Markov model as good approximation of system security, even if one doesn't know attacker strategies,
    c. Allowing single- and muli-step attacks within an integrated framework

In addition, several important observations about rigorous cyber experimentation were encountered during this project, including:

- Verification efforts helped find very subtle bugs when deploying emulation experiments on large numbers of namespaces on one host.
- Cyber experimentation is best conducted (in terms of efficiently generating statistics and mapping the response space) when augmenting emulation models with results from other models:
    a. Mathematical modeling, where feasible, provides important insight into attacker/defender dynamics, and provides a supplemental source of metrics useful for cross-validation with emulation models and efficiently generating statistics.
    b. Likewise, discrete event simulation efficiently provides results that can be highly correlated with emulation, making it useful for multifidelity studies.
    c. While benefits can be gained from simplified system models, creating these representations currently requires significant levels of collaboration between modelers and cyber experts.
- Emulation models have large inherent variability, even for fixed parameters. Including parameter uncertainty further complicates analysis of these systems. This highlights the need for methods like MFUQ and efficient experimental designs.
- Sensitivity analysis on attack chains is critical for providing quantitative (and sometimes counter-intuitive) evidence to support defender investments.
- Rigorous, quantitative attack analysis can expose counter-intuitive relationships between attack success metrics (not always unique solutions; Pareto frontier), informing defender decisions regarding tradeoffs in defensive investments.
- Careful attention is required when translating information between models (mathematical, emulation, Markov attack model, etc.)

The impact of SECURE goes beyond these individual contributions. SECURE demonstrated how we can bring rigor into cyber experimentation, an elusive goal that has been chased by many for a long time. We built our analyses on reproducible processes, verified experiments, and validated models. Our algorithms provide provably accurate results, state-of-the-art statistical analysis, and take into account the underlying uncertainties. As a result, we can objectively assess and quantify security. We claim that quantifiable security measures will be a game-changer in cyber security.

While SECURE has drastically advanced the state of the art, we recognize there is a long way in front of us. We aim to pursue this line of work through externally funded research, new internal collaborations, and external partnerships with academia and other national laboratories. To this goal, we are starting a new institute Sandia's Cyber security Institute on Rigorous Experimentation (SCIRE). This institute will enable continued interdisciplinary collaborations across Sandia, connect us with the external researchers, strengthen Sandia's position as a leader and visionary in rigorous cyber experimentation. What has started under SECURE will continue under SCIRE, and SECURE's legacy will live on.

We started the project by drawing inspiration from CSE. And we are ending the project having demonstrated that rigorous cyber experimentation can be a pillar of the science of cyber security, just as CSE is a pillar of science.

11. REFERENCES

NOTE: Reports listed as "in preparation" were in preparation at the time this report was written. They may be available by contacting the authors.

1. AIAA Standards: *Guide for the Verification and Validation of Computational Fluid Dynamics Simulations* (AIAA G-077-1998(2002)) Computational Fluid Dynamics Committee. https://doi.org/10.2514/4.472855
2. ASME V&V 20-2009 *Standard for Verification and Validation in Computational Fluid Dynamics and Heat Transfer.* https://www.asme.org/products/codes-standards/v-v-20-2009-standard-verification-validation
3. Balenson, David, Laura Tinnel, and T Benzel. "Cybersecurity Experimentation of the Future (Cef): Catalyzing a New Generation of Experimental Cybersecurity Research." *SRI International, Tech. Rep.* (2015).
4. Crussell, Jonathan, Erickson, Jeremy, Fritz, David, and Floren, John. minimega v. 3.0, Computer software. Sandia National Laboratories, Albuquerque, NM, December 18, 2015. https://www.osti.gov//servlets/purl/1312788.
5. Diegert, K., Klenke, S., Novotny, G., Paulsen, R., Pilch, M. and T. Trucano. *Toward a More Rigorous Application of Margins and Uncertainties within the Nuclear Weapons Life Cycle – A Sandia Perspective*. Sandia Technical Report SAND2007-6219.
6. Dykstra, Josiah. Essential Cybersecurity Science: Build, Test, and Evaluate Secure Systems. " O'Reilly Media, Inc.", 2015.
7. Ferguson, B., A. Tall, and D. Olsen. 2014. National cyber range overview. In 2014 IEEE Military Communications Conference. IEEE, 123–128.
8. Geraci, G., Crussell, J., Swiler, L.P. and Debusschere, B. J. "Exploration of Multifidelity UQ Sampling Strategies for Computer Network Applications." *International Journal of Uncertainty Quantification*, Jan. 2021. Pp. 93-118. DOI: 10.1615/Int.J.UncertaintyQuantification.2021033774
9. Geraci, G., L.P. Swiler, J. Crussell, and B. Debusschere. *Exploration of Multifidelity Approaches to Uncertainty Quantification in Network Applications.* UNCECOMP ECCOMAS 2019. Thematic Conference on Uncertainty Quantification in Computational Sciences and Engineering. SAND2019-3274C.
10. Ghanem, R. And P. Spanos, *Stochastic Finite Elements: A Spectral Approach.* New York, New York: Springer Verlag (2002).
11. Google. Network telemetry. 2021. https://cloud.google.com/network-telemetry.
12. Hedayat, A.S., Sloane, N.J.A., and J. Stufken. *Orthogonal Arrays: Theory and Applications.* Springer Series in Statistics, 1999.
13. Heller, Brandon. "Reproducible Network Research with High-Fidelity Emulation." Ph.D. dissertation, Stanford University, 2013.
14. Helton, J.C. *Conceptual and Computational Basis for the Quantification of Margins and Uncertainty*. Sandia Technical Report 2009-3005.
15. Herley, C., and P. C. van Oorschot. "Science of Security: Combining Theory and Measurement to Reflect the Observable." *IEEE Security & Privacy* 16, no. 1 (2018): 12-22. https://doi.org/10.1109/MSP.2018.1331028.
16. https://dakota.sandia.gov

17. https://minimega.org/
18. https://omnetpp.org
19. https://www.elastic.co/what-is/elk-stack
20. https://www.minitab.com/
21. https://www.nsnam.org
22. IEEE. IEEE Standard for System and Software Verification and Validation. IEEE STD 1012-2012 (Revision of IEEE STD 1012-2004), pages 1–223, 2012. Doi: 10.1109/IEEESTD.2012.6204026.
23. Intel. Cloud telemetry: Advancing your it strategy. 2021 https://www.intel.com/content/www/us/en/cloud-computing/telemetry.html.
24. Jones, S.T., Gabert, K. G., and T. D. Tarman. *Evaluating Emulation-based Models of Distributed Computing Systems*.SAND2017-10634. https://www.osti.gov/biblio/ 1398865-evaluating-emulation-based-models-distributed-computing-systems.
25. Maricq, A., Duplyakin, D., Jiminez, I., Maltzahn, C., Stutsman, R. and R. Ricci. *Taming Performance Variability*. 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI). 2018
26. Microsoft. Azure monitor. 2021. https://docs.microsoft.com/en-us/azure/azure-monitor/autoscale/autoscale-common-metrics.
27. Mirkovic, J., Bartlett, G. and J. Blythe. *DEW: Distributed Experiment Workflows*. USC Information Sciences. Proceedings from USENIX/CSET 2018 Conference.
28. Mirkovic, J., T.V. Benzel, T. Faber, R. Braden, J.T. Wroclawski, and S. Schwab. The DETER project: Advancing the science of cyber security experimentation and test. In 2010 IEEE International Conference on Technologies for Homeland Security (HST), pages 1–7. IEEE, 2010.
29. Morgan, M. G. and M. Henrion. *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*. Cambridge University Press, 1990.
30. Motto, A.L., J. M. Arroyo and F. D. Galiana, "A mixed-integer LP procedure for the analysis of electric grid security under disruptive threat," in *IEEE Transactions on Power Systems*, vol. 20, no. 3, pp. 1357-1365, Aug. 2005, doi: 10.1109/TPWRS.2005.851942.
31. National Research Council. *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification*. Washington, DC: The National Academies Press, 2012. https://doi.org/10.17226/13395.
32. National Science and Technology Council, "Federal Cybersecurity Research and Development Strategic Plan." 2016. https://www.nitrd.gov/pubs/2016-federal-cybersecurity-research-and-development-strategic-plan.pdf.
33. Oberkampf, W.L. and C.J. Roy. Verification and Validation in Scientific Computing. Cambridge University Press, 2010.
34. Outkin, A.V., T. Schulz, T.D. Tarman, P.V. Schulz, A. Pinar. Defender Policy Evaluation and Resource Allocation against MITRE ATT&CK Data and Evaluations. SAND2021-7713. https://arxiv.org/abs/2107.04075
35. Saltelli, A., Tarantola, S., Campolongo, F., Ratto, M. *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. New York: Wiley; 2004.
36. Santner, T., B. Williams, and W. Notz, *The Design and Analysis of Computer Experiments*. New York, New York: Springer (2003).

37. Siaterlis, C., A.P. Garcia, and B. Genge. On the use of Emulab testbeds for scientifically rigorous experiments. IEEE Communications Surveys & Tutorials, 15(2):929–942, 2012.
38. Sumologic. What is telemetry? the guide to application monitoring. 2021. https://www.sumologic.com/insight/what-is-telemetry/.
39. Swiler, L., Stickland, M, and T. Tarman. Design of Experiments for Cyber Emulation. Sandia National Laboratories Technical Report SAND2019-5640C. May 2019.
40. Tarman, T.D., L.P. Swiler, E. Vugrin, H. Huang, A. Sahu, P. Wlazlo, A. Goulart, and K. Davis. Validation of cyber experiments: comparing emulation results against a physical testbed. In preparation, 2021.
41. Tarman, T.D., T. Rollins, L.P. Swiler, J. Cruz, E. Vugrin, H. Huang, A. Sahu, P. Wlazlo, A. Goulart, and K. Davis. Comparing reproduced cyber experimentation studies across different emulation testbeds. *USENIX 14th* Cyber Security Experimentation and Test (CSET) Workshop. Aug. 9, 2021. SAND2021-5696C.
42. Thorpe, J., L.P. Swiler, J. Cruz, S. Hanson, T. Tarman, T. Rollins, and B. Debusschere. "Verification of Cyber Emulation Experiments Through Virtual Machine and Host Metrics." In preparation, 2021.
43. Vugrin E., S. Hanson, J. Cruz, C. Glatter, T. Tarman, and A. Pinar. Detection of command and control traffic: model development and experimental validation. in preparation, 2021.
44. Vugrin, E., J. Cruz, C. Reedy, T. Tarman, and A. Pinar "Cyber Threat Modeling and Validation: Port Scanning and Detection," *Proceedings of the 7th Annual Hot Topics in the Science of Security (HoTSoS) Symposium.*
45. Xiu, D., *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press (2010).
46. Y. Ozkaya, F. Balın, A. Pınar, and U. C ̧atalyu ̈rek, SSGG: A scalable graph generation algorithm to sample over a given shell distribution in Proc. IPDPS Workshops, W. Graph Learning.
47. Y. Ozkaya, A. Pinar, and U. Catalyurek, "TRIGGER: TempoRal Interaction Graph GenEratoR," submitted for conference publication

APPENDIX A. COMMAND AND CONTROL (C2) CASE STUDY

A.1. Overview

Over the last few decades, a variety of emulation tools have been developed to perform cyber experimentation. Despite this progress, relatively little attention has been devoted to developing methods that ensure the quality of experiments based on these capabilities. In this article, we demonstrate how the mathematical modeling, verification, validation, and uncertainty quantification methods, developed under SECURE, can be used in combination with emulation modeling to perform rigorous experimentation for a Command and Control (C2) cyber-attack. To our knowledge this exemplar demonstrates a level of experimental rigor and detail that has not been previously done for this kind of case study.

Recall that the full end-to-end exemplar studied in SECURE considers a multi-stage attack in which an attacker attempts to access a power utility's cyber control network and ultimately disrupt operations by causing load shed using the attack stages shown in Figure A 1. Here we focus on the second step where an attacker aims to maintain C2 communications between an infected host and C2 server in order to pivot to other hosts and/or the ICS network. To counter this, the system owner uses an intrusion detection system (IDS) to identify malicious C2 traffic and take steps to remediate the infection to prevent disruption of physical operations.

Figure A 1. Multi-stage attack considered by SECURE

The goals of this study fall into the following two categories: *application objectives* related to analyzing malicious C2 traffic in a cyber system and *SECURE research objectives* related to methods for cyber experimentation. Given this, we consider the following:

- Application objectives:
  - How long does it take to detect a C2 channel?
  - How does background traffic affect detection?
  - Which factors have the largest impact on the performance of an IDS system?
- SECURE research objectives:
  - What emulation capabilities are required to adequately represent this scenario?
  - Can we develop an approximate mathematical model of the emulation to analyze this scenario?
  - How can we validate the math model against the emulation?
  - What is the benefit of a math model?
  - Can the emulation and math model be used in conjunction to support analysis?

A.2. Analysis Scenario

In this study, we focus on detecting C2 malware traffic within the enterprise network portion of an electrical power utility. Figure A 2 illustrates the system being analyzed. We assume that one or more hosts within the network have been infected and are communicating with an external C2 server. The

internal network contains both benign and malicious network traffic, all of which is sent through a single router and switch. An IDS that monitors traffic to and from the network. The IDS performs packet inspection and issues an alert if the contents of an individual packet appears suspicious, according to one or more of its rules. We assume that it is possible that benign traffic may cause the IDS to issue an alert (i.e., a false-positive). In instances where there are large packet flow rates, the IDS may not have sufficient capacity to scan all packets [1]. In this case, unscanned malicious packets will still reach their destination without causing an alert.

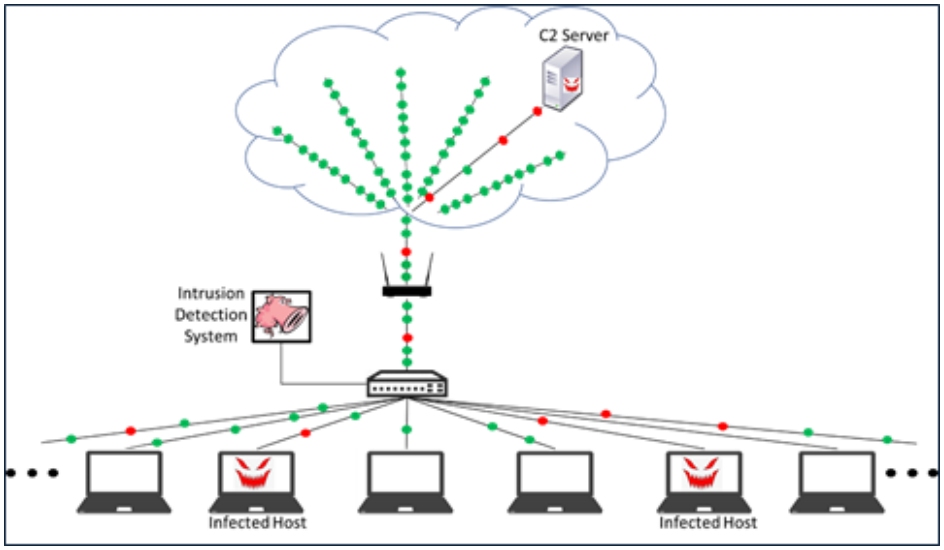


Figure A 2: Notional C2 exemplar system representation

For this study we analyze C2 communication from the Emotet malware and its detection by the Snort IDS. Emotet was first discovered in 2014 as a banking Trojan. Since its initial discovery, Emotet has infected more than 1 million computers and caused hundreds of millions of dollars in damage [2]. Most antivirus and IDS programs have some sort of mechanism to detect an Emotet infection. For the Snort IDS alone, dozens of rules have been written to detect Emotet.

Though this study is motivated by and focuses on specific Snort and Emotet features, the work discussed below is not unique to this IDS or malware. Rather, we believe the capabilities presented below could be generally applicable to any IDS and malware combination in which the IDS generates alerts based on individual packet inspection. Consideration of different IDSs and alerts would merely require alternate parameterizations.

Given the goals of the attacker and the defender, the key Quantities of Interest (QoIs) are the alert rates (i.e., number of alerts issued at a point in time) for both malicious and benign traffic, under various network, attack, and IDS configurations. We recognize that issuance of an alert does not necessarily equal detection; detection generally requires a combination of alerts and human recognition that the alerts are indicative of a problem. Modeling the human element of detection is beyond the scope of this work, so, instead, we assume that a detection occurs when a large enough number of alerts are issued that network administrators would reasonably determine that the anomalous traffic is malicious. Hence, the primary focus of this work is accurately modeling alerts over time and not establishing detection thresholds.

The remainder of this article provides an overview of how the SECURE experimentation methodology was applied to the C2 malware problem. The following summarizes the process that we used to analyze the C2 problem. For each of steps described, detailed tutorials and technical documentation are also available.

1. Emulation model development: Create a high-fidelity "ground truth" model using emulation.
2. Emulation model verification: Build confidence that the emulation models are working as intended.
3. Mathematical model development: Create a low-fidelity statistical model surrogate for the emulation model.
4. Mathematical model validation: Assess the validity of the low-fidelity model using statistical tests for discrete, time-series data to ensure that the inexpensive mathematical model can be used as a proxy for the more costly high-fidelity emulation model.
5. Analysis and Uncertainty quantification:
    1. Efficient sampling: Use Polynomial Chaos Expansion (PCE) to efficiently sample the input parameter space using the mathematical model to identify which input parameters have the largest effect on the QoIs.
    2. Multi-fidelity uncertainty quantification: Integrate results from low- and high- fidelity models to improve the accuracy of the QoIs with minimal experimentation costs, for the key parameters identified using PCE.

A.3. C2 Emulation Environment

We model the C2 environment using emulation, a capability primarily used to model distributed communication networks. As the name implies, emulation models aim to replicate high-level functionality of target networks using emulated hardware components. Abstraction of the hardware layer serves to facilitate implementation of these "logical network replicas" at reduced costs. A typical emulation environment consists of a set of virtual machines that are networked together using virtual switching. The entire environment is supported by a cluster of hardware servers. Emulation environments serve a variety of purposes such as testing, evaluation, training, and experimentation. Because of their heavy use of virtualization, large network environments can be deployed, torn down, and redeployed to an original state with relatively little effort. This makes emulation environments particularly well-suited for repeatable and reproducible experimentation of distributed communication networks. There are several tools available for creating, deploying, and managing emulation environments, including two created at Sandia National Laboratories: `minimega` and SCORCH. Sandia's `minimega` tool is used for launching and managing virtual machines locally or across a cluster. SCORCH is an automated scenario orchestration framework for emulation-based models that utilizes `minimega` to deploy and instrument experiments.

We created the emulation model for this study using `minimega` and SCORCH. The environment model is comprised of the following primary components, as shown in Figure A 3: a malware traffic generator (attacker), an IDS (defender), and the background traffic generator (environment). Each component has parameters that can be adjusted and tuned for various experiment iterations. The malware traffic is generated via custom Python code that enables researchers to modify the message features, size, and frequency of the generated packets. Rather than represent each machine with an individual host, we use a single device to generate "aggregate traffic" representative of the total traffic we would see from multiple hosts. For this scenario, the malware traffic generator is calibrated to mimic the packet structure of the Emotet malware message format, encrypted structure, and C2 timing

(using the 2018/2019 variant of Emotet). The signature of the Emotet network traffic has been previously researched and captured in detection rules [2,3]. Snort is used as the IDS and implements Emotet-specific detection rules to alert on Emotet-based packet signatures. The IDS component can be tuned for different detection algorithms/rules, memory availability, and processing speed. To increase the scenario's fidelity and provide a realistic network for experiments, background packets are created and sent from a client to a server via a custom Python script. The background traffic message format, packet size and frequency can be modified per experiment.

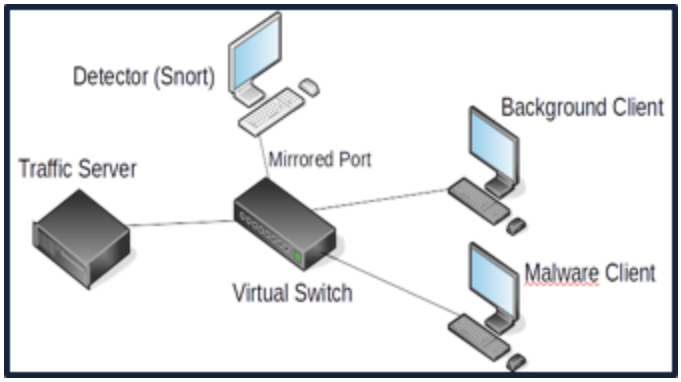


Figure A 3: C2 Exemplar Emulation Environment

For this study, we focus on the parameters shown in Table A 1. These parameters can be binned into four groups. The *general parameters* describe basic parameters of the test environment. The *IDS parameters* define the capacity and characteristics of the IDS. The *background traffic parameters* specify the intensity of the background traffic and the false-positive rate. *The malicious traffic parameters* specify the intensity of the malware traffic and the false-negative rate. For each of these parameters, we indicate the value or the range of the values that the parameter can take. For those values that are uncertain, we assume they follow a continuous or discrete probability distribution, as indicated in the Distribution column. Even for this relatively modest sized problem, many parameter configurations can be explored. Note that some of parameters listed in Table A 1cannot directly be controlled in the emulation environment, as specified in the Comments column.

Table A 1. Key variables of interest for the C2 study.

Parameters	**Units**	**Value**	**Distribution**	**Comments**
General Parameters				
Total number of workstations	No units	10	Fixed	Variable type: input parameter Basis: selected to represent "moderately" sized portion of a corporate network
Average packet size	Bytes	150-250	Continuous uniform	Variable type: observed quantity Basis: packets observed in the experiments had an average size of 200 bytes in experiments; +/-50 bytes is selected to permit variability across experiments

IDS Parameters				
Snort capacity	Bytes per second	1e5, 2e5, 5e5, or 1e6	Discrete with equal probability	Variable type: input parameter to emulation model Basis: selected to represent "moderately" sized portion of a corporate network
Number of CPUs	No units	8	Fixed	Variable type: input parameter Basis: expert judgement and known hardware configurations
Number of CPUs to maximize Snort	No units	1-8	Discrete with equal probability	Variable type: input parameter Basis: positive integers bounded by total # of CPUs
CPUs running other (non-Snort) processes	No units	0-7	Discrete with equal probability	Variable type: input parameter Basis: positive, integers bounded by total # of CPUs
Drop rate multiplier	No units	0.9-1.1	Symmetric continuous triangular distribution	Variable type: observed quantity Basis: expert judgment used to assess the actual drop rate, which could be +/- 10% difference from the calculated rate
Background Traffic Parameters				
Benign traffic per host	Packets per sec	5-100	Continuous log-uniform	Variable type: input parameter Basis: 100 pps per host (with 20 hosts) results in 2000 pps for total traffic. This amount represents the upper limit on the traffic generator's capacity and is comparable to (and may exceed) congested TCP traffic conditions used in other IDS evaluation literature (e.g., [4] and [5]). The lower bound was selected to represent a minimal level of traffic for evaluation.
Fraction of benign packets with Emotet signatures.	Fraction of packets per sec	1e-5-1e-3	Continuous log-uniform	Variable type: input parameter Basis: expert judgment because published values were not available; selected values are relatively small to indicate the small probability that the Emotet signature would occur due to spurious conditions

Detection rate for signatures in regular, benign traffic (if signature is present)	No units	0.9-0.99	Continuous uniform	Variable type: observed quantity Basis: we observed an average detection rate of 0.95 when we used the Snort rule to evaluate actual Emotet traffic packet captures (pcaps) and simulated Emotet traffic in emulation experiments; range was expanded to 0.9-0.99 to permit variability across experiments
Malicious Traffic Parameters				
Number of infected workstations	No units	0-10	Discrete with equal probability	Variable type: input parameter Basis: non-negative integer, bounded by total number of hosts
Malware traffic per infected host	No units	4-10	Continuous uniform	Variable type: input parameter Basis: published observations and analysis of actual Emotet traffic pcaps
Fraction of malware packets with Emotet signatures	No units	0.1-0.2	Continuous uniform	Variable type: input parameter Basis: analysis of Emotet traffic pcaps and structure of TCP traffic
Detection rate of signatures for malware traffic (if signature is present)	No Units	0.9-0.99	Continuous uniform	Variable type: observed quantity Basis: we observed an average detection rate of 0.95 when we used the Snort rule to evaluate actual Emotet traffic pcaps and simulated Emotet traffic in emulation experiments

A.4. Emulation Verification using Telemetry

An important aspect of using emulation is verifying whether the emulation environment is working as intended. For this study, we approach the verification problem using the same strategy that was employed in the SCADA study. The core idea of this approach is to monitor performance metrics while intentionally stressing the emulation environment to identify potential issues. This monitoring process is called telemetry [6-9], which includes metrics like server load and availability, disk space usage, memory consumption, performance, etc. Though many aspects of the emulation could be verified, we focused on determining whether sufficient virtualized resources are available to support the scenario because insufficient resources can cause experimental outcomes to be unrepresentative or incorrect.

In this study, we run the C2 scenario under various levels of over-subscribed resources. We start with a baseline scenario where there is only one namespace running on a physical host. We then consider four scenarios where an increasing number of namespaces (2, 5, 10, and 20, and 40) are run in parallel on the same physical host. In this case, analysis of QoIs and telemetry data indicates that resources

were being oversubscribed with the 20 and 40 namespace cases. The collected telemetry showed a clear point where resources were oversubscribed. Further research is needed to develop general methods for verification of emulation models.

A.5. Mathematical Model

Emulation testbeds provide a safe, high-fidelity environment for conducting cyber experiments. However, since these testbeds run real software and protocols, the experiments typically need to be executed in real-time. This can be time-prohibitive in instances where:

- Scenarios evolve over long time periods.
- Analyses include features of the system may be unknown or vary, or in which the analyst aims to characterize a potentially wide range of possible outcomes.
- Analyses consider stochastic behaviors and thus require many experiments to suitably characterize the relevant statistics.

Given the potential number of parameter setting that could be explored (see Table A 1), we developed a low-fidelity statistical model that can be run significantly faster than the real-time emulation model. The model can be most easily described through an analogy, as depicted in Figure A 4. Consider a water contamination sensor system that receives flows from various sources across a water transportation network. The flows may contain benign or beneficial matter like fluoride (normal network traffic) and also toxins like lead (malicious C2 messages). Water containing both good and bad matter flows into a reservoir tank and passes through a filter (IDS) that removes the toxic particles. The "cleaned" water is then distributed throughout the system. The filter may fail to catch some portion of the toxins (false negatives); it may also remove benign materials (false positives). The filtration system is rate-limited and has a finite reservoir capacity (memory). If the inflow rate exceeds the capacity of the sensor system, a bypass valve is activated, permitting the unfiltered water to circumvent the filter and pour directly into the system without filtration. See [11] for a full description of the mathematical model.

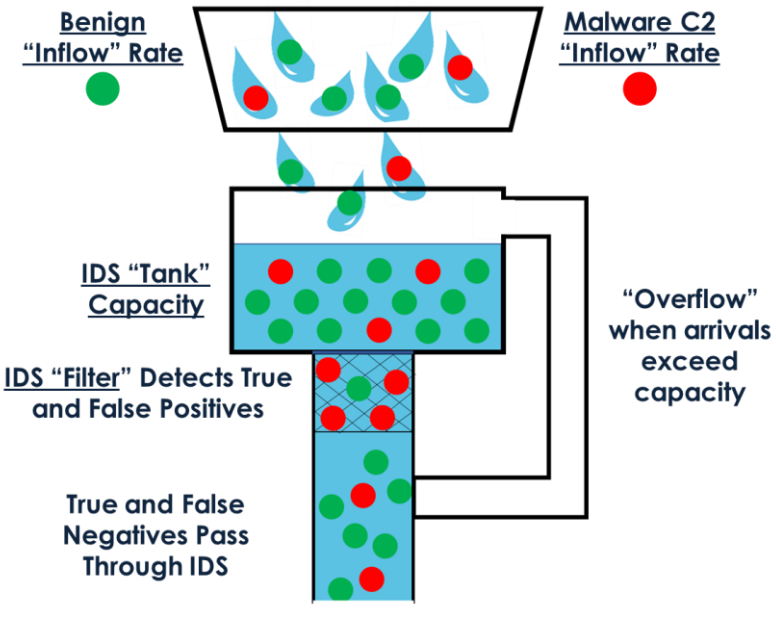


Figure A 4. Mapping between water filtration and intrusion detection systems.

The mathematical model of the IDS builds on the flow/filter concepts to represent network traffic as an influx of packets from various hosts (flow) and the detection of C2 traffic by an IDS (filter). Most of the hosts are not infected with the malware, so the packets in their traffic is benign. Some hosts are infected by malware and generate packets that contains malicious C2 traffic. All packets are routed through a device running an IDS, whose signature-based rules act as a filter: if the rule identifies the malware signature from a malicious packet (true positive), the IDS issues an alert. Detection of malicious traffic is not perfect, so some malicious packets pass through without an alert being issued (false negative). In some instances, the IDS may issue an alert for a benign packet (false positive), but most benign packets result in no alert (true negative).

The IDS is rate-limited in its capacity to process network traffic (i.e., the IDS has a threshold measured in packets/bytes per second) within a set time period. Hardware characteristics (e.g., number of CPUs, memory available), software features (e.g., types of detection rules being used by the IDS, computing requirements for individual rules, number of rules being used, degree of parallelization), and the number of other processes being run on the device (and computational requirements for the processes) all affect the IDS's capacity. In the most extreme cases (when network traffic rates far exceed the IDS's capacity), the IDS may eventually stop issuing alerts altogether until the memory buffer is cleared. In these instances, all packets will pass to their destinations without being inspected by the IDS, including any malicious C2 packets; because they are dropped, alerts are not generated for these packets, resulting in universal negatives (false and true).

The mathematical model integrates these concepts into a probabilistic, discrete-time representation to describe the C2 traffic and detection by the IDS. The key model inputs include:

- Packet arrival rates at the IDS for both benign and malicious traffic
- True and false positive rates (on a per-packet basis) for the IDS's signature-based rule
- Average packet size
- IDS capacity

When specific values are assigned to these inputs, the model produces the following two primary outputs: the average number of alerts that are expected over time (Figure A 5), and the probability that at least N alerts will be registered by a point in time (Figure A 6). Results can be produced for non-rate limiting (Figure A 5 and Figure A 6) and rate-limiting scenarios (Figure A 7). Observe that in the latter case, the number of alerts levels off as the IDS reaches capacity. The total alert results can also be separated into false positive alerts and true positive alerts.

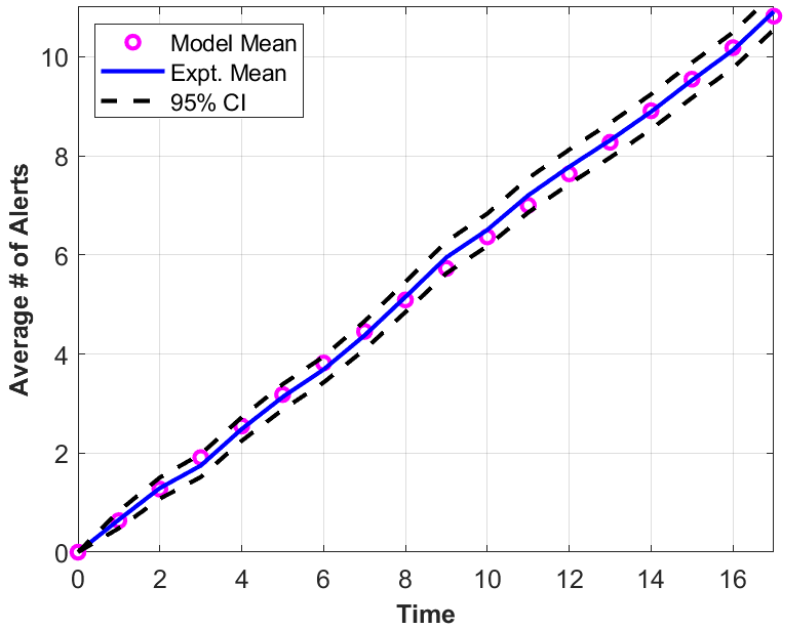


Figure A 5. Average number of alerts over time, for the emulation and mathematical models (non-rate limited case).

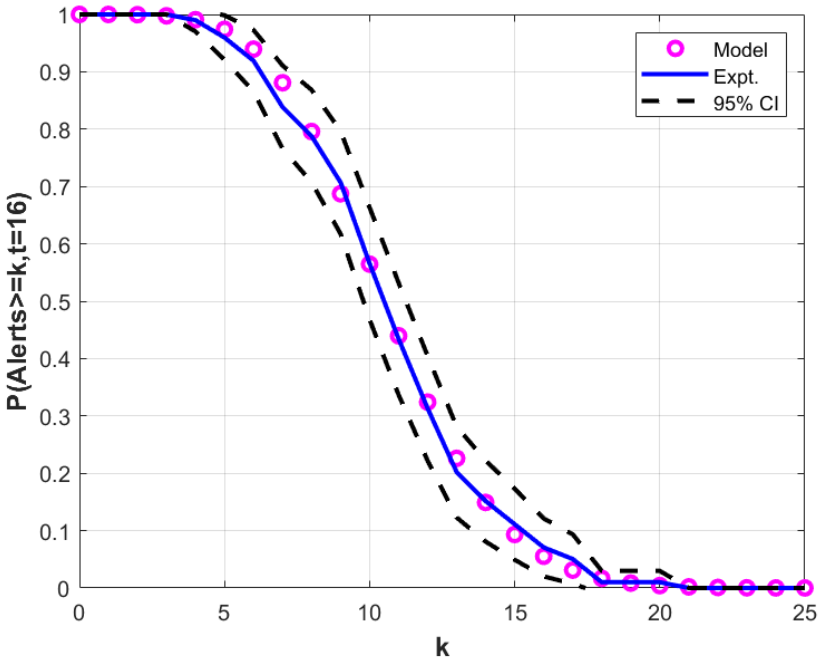


Figure A 6. Probability of having at least k alters by time period 16, for the emulation and mathematical models (non-rate limited case).

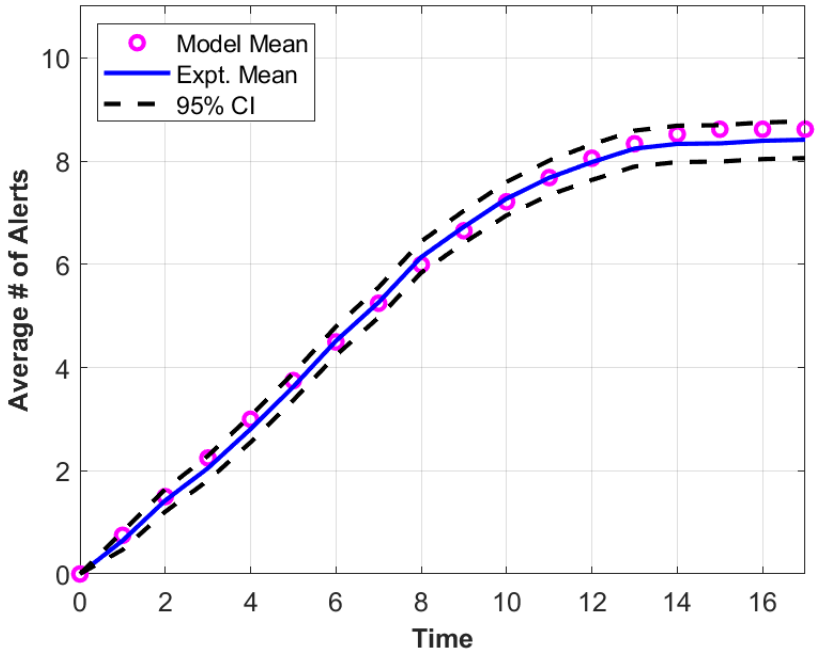


Figure A 7. Average number of alerts over time, for the emulation and mathematical models (rate limited case). Note how the number of alerts levels off.

A.5.1. Comparison of Mathematical Model and Emulation Model Results

The mathematical model is validated against the results generated by the emulation model. Figure A 5, Figure A 6, and Figure A 7 show the results for both models. For these particular results, a visual inspection shows a strong level of agreement between the two models, with the mathematical model results generally falling within the 95% confidence intervals of the mean value from the emulation model results. The data generated by the emulation model is both discrete (number of alerts triggered) and time-series (number of alerts per time-step). For example, a particular run might have 0 alerts triggered in the first second, 3 alerts after 5 seconds, and 7 alerts after 10 seconds. After enough of these emulation runs are collected, we can generate a cumulative distribution function (CDF) at each time step on the number of triggered alerts. In other words, we have a curve representing the probability that more than k alerts are generated by a given point in time. We compare the CDFs from both models using a more rigorous, statistical approach than visual comparison. We do this by using the Kolmogorov-Smirnov (K-S) test, a standard statistical test for comparing two distributions. Figure A 8 shows an example of the model and experimental CDF curves at time period 9 for a particular C2 scenario.

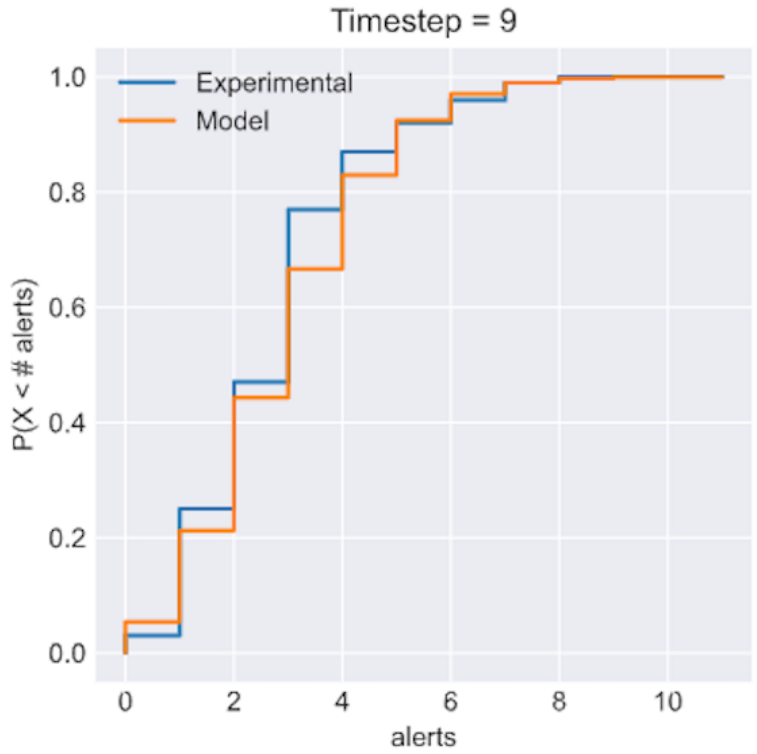


Figure A 8. Comparison of the emulation and mathematical models CDFs for the probability of exceeding a given number of alerts by time period 9.

Using the K-S test, we can calculate a p-value for each time period, as shown in Figure A 9. Observe that for time period 9, the p-value is about 0.2. A high p-value indicates that the null hypothesis, that the two CDFs are statistically similar, cannot be rejected. While the p-value dips around time period 9, it is still above 0.1 even at its lowest point. Given this, we would not reject the null hypothesis in this example.

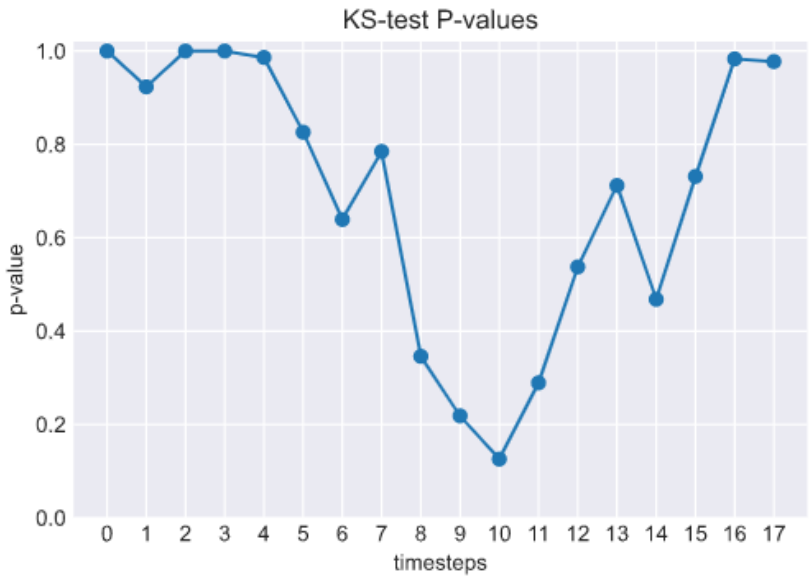


Figure A 9. p-values for the K-S by time period.

In addition to the results shown above, we have compared the emulation and mathematical model results across a variety of parameter combinations. Though the results may not be perfectly identical, the combination of visual inspection and statistical comparisons provide confidence that the mathematical model is a reasonable proxy for the actual system and that it can provide reasonable estimates of alert statistics for the C2 scenario under consideration.

A.6. Analysis and Uncertainty Quantification

Given the high- and low- fidelity models, we next focus on uncertainty quantification (UQ) to understand how uncertainty in the input parameters propagates to the QoIs. We do this using two analysis methods: polynomial chaos expansion (PCE) and multi-fidelity UQ (MFUQ). We first use PCE to screen the 12 uncertain parameters shown in Table A 1 to determine which parameters are the most important for more detailed study. The screening is done using the low-fidelity mathematical model to avoid the computational costs of using the emulation model. Once the key parameters are identified, MFUQ is used to analyze the QoI using a combination both models.

A.6.1. PCE Sampling

In the UQ community, QoIs are commonly represented as a polynomial function of the uncertain inputs; this approach is referred to as a Polynomial Chaos Expansion (PCE) of the QoI. Provided that a QoI is a smooth function of the inputs, the smoothness in the polynomial representation can give an accurate representation with fewer samples than would be required with a Monte Carlo (MC) approach. Once a PCE is constructed, it can be used to determine the mean, variability, or other moments of the QoI. PCEs can also be used to perform a Global Sensitivity Analysis (GSA) of the QoI with respect to each of the inputs. In other words, it can tell us which inputs contribute the most to the variability in the output.

One of the challenges of applying the PCE approach to cyber security experiments is that many of the input variables are discrete. For example, the number of infected nodes on a network, the number of CPUs on the host that runs an IDS, or the nominal network bandwidth of the node connections are all discretely valued. Therefore, we employ PCEs that have been tailored to discrete random variables and their probability masses. These tools have been implemented in PyApprox, a Sandia open source software package for uncertainty quantification [10].

We applied this approach to the QoIs of total alerts and false positives at time period 5, for the parameter distributions as in Table A 1. This corresponds to a case with 12 uncertain parameters, 5 of which are discrete in nature. A third order PCE was trained on random samples of the QoI that were obtained with the C2 math model. Table A 2 shows the main effect indices for both QoIs for the 12 uncertain parameters.

Table A 2. Main effects from PCE analysis for the number of total alerts and false positives at time period 5.

Parameters	Total Alerts, t = 5 sec.	False Positives, t = 5 sec.
Number of infected workstations	**0.87**	0.00
Fraction of benign packets with Emotet signatures	0.00	**0.51**
Benign traffic per host	**0.01**	**0.20**
Malware traffic per infected host	**0.05**	0.00
Fraction of malware packets with Emotet signatures	**0.03**	0.00
Snort capacity	**0.01**	**0.01**
Other CPU Processes	**0.01**	0.00
Number of CPUs to maximize snort	0.00	0.00
Average packet size	0.00	0.00
Detection rate for signatures in benign traffic	0.00	0.00
Detection rate of signatures for malware traffic	0.00	0.00
Drop rate multiplier	0.00	0.00

Based on these results, the main parameter that impacts the value of total alerts is the number of infected hosts, with lesser contributions from the amount of malware traffic per infected host and the fraction of malware packets that show the Emotet signature. The number of false positive alerts is most sensitive to the amount of benign traffic per infected host and the fraction of benign traffic packets that show the Emotet signature.

A.6.2. Multi-Fidelity UQ

Next, we explore the use of MFUQ to make optimal use of the emulation model which has high fidelity but is expensive to run and the lower-fidelity mathematical model which can be evaluated quickly. MFUQ estimator is built starting from the single fidelity MC results ($Q_{minimega}$) and adding a weighted unbiased term which involves the lower-fidelity math model (Q_{math}). The benefit of this additional term is that it can reduce the variance of the QoI (see [12] for the technical details of this approach). Using this approach many samples from the low-fidelity mathematical model can be combined a relatively small number of high-fidelity emulation model results to decrease the estimator variance and obtain more accurate and reliable statistics, with reduced computational costs.

$$\hat{Q}^{MF} = \frac{1}{N}\sum_{i=1}^{40} Q_{minimega}^{(i)} + \alpha\left(\frac{1}{N}\sum_{i=1}^{40} Q_{math}^{(i)} - \frac{1}{r\times 40}\sum_{j=1}^{r\times 40} Q_{math}^{(j)}\right)$$
$$= \hat{Q}_{minimega} + \alpha\hat{\Delta}_{math},$$

Figure A 10. C2 MFUQ estimator.

Based on the screening results from the PCE analysis, we focus on the five parameters shown in Table A 3. A total of 40 samples of these parameters was used for this study. The emulation model required 18 hours (plus additional processing time) to perform a total of 400 emulation runs (the 40 unique

parameter combinations with 10 iterations each). In contrast, the mathematical model required less than 1 second total for all 40 of the parameter combinations (0.4 s for all the samples). We note that the mathematical model is able to provide statistics for the QoI without being affected by any stochastic noise; therefore, we will compare the average from 10 emulation model replicas with the values from the mathematical model.

Table A 3. Key parameters of interest for MFUQ study

Parameters Varied in Experiment	Units	Value	Distribution
Aggregate Benign traffic rate	Packets per sec	100-3000	Continuous log-uniform
Fraction of benign packets with Emotet signatures.	No units	1e-5 to 8e-4	Log-uniform
Aggregate malware traffic rate	Packets per sec	10-20	Uniform
Fraction of malware packets with Emotet signatures	No units	0.01-0.025	Uniform
RAM assigned to the 1 CPU running SNORT	Mbytes	128, 256, 512, 1024	Discrete with equal probability

For this study, we consider the total number of alerts at time periods1, 5, and 10. We begin by performing a pilot study to compare the total number of alerts generated from both models. From Figure A 11, we note that the correlation between both models is high, as confirmed in Figure A 12 which shows the estimated squared correlation between the models at the time steps considered.

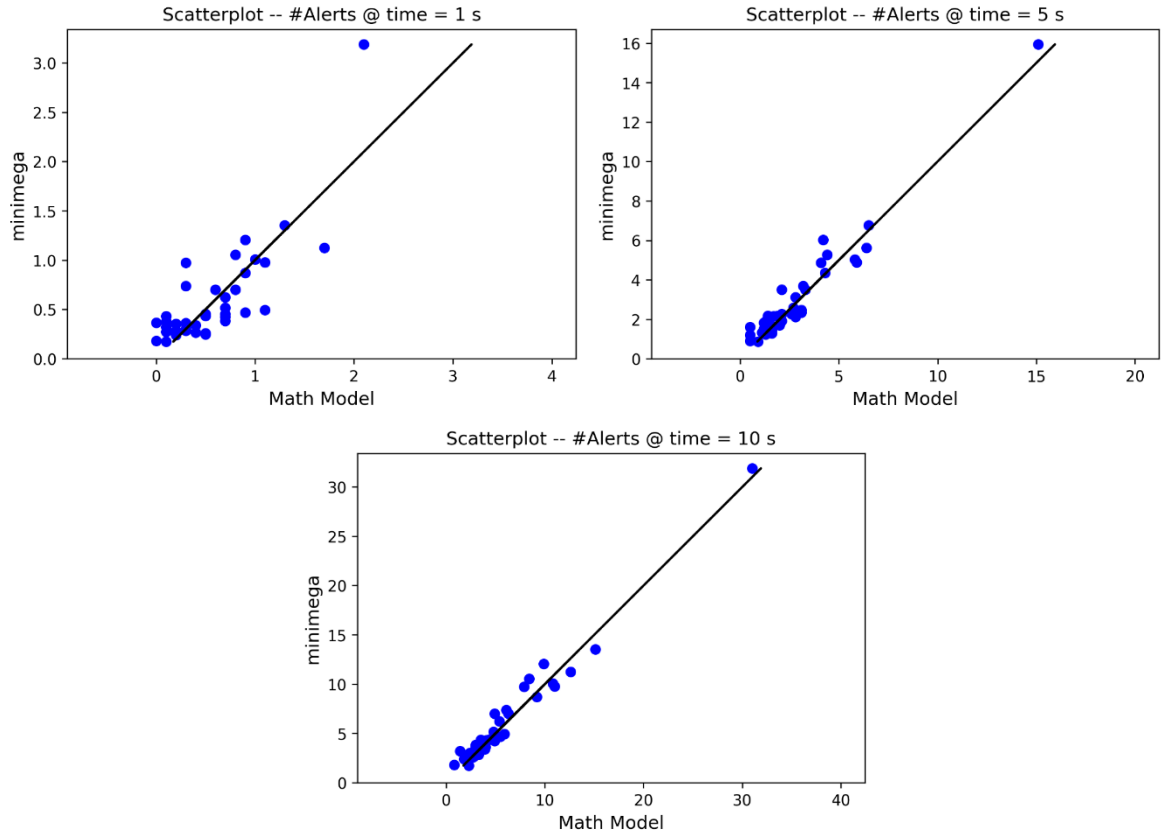


Figure A 11. Scatterplots of total number of alerts at timesteps 1, 5, and 10 for 40 parameter samples for the emulation and mathematical models.

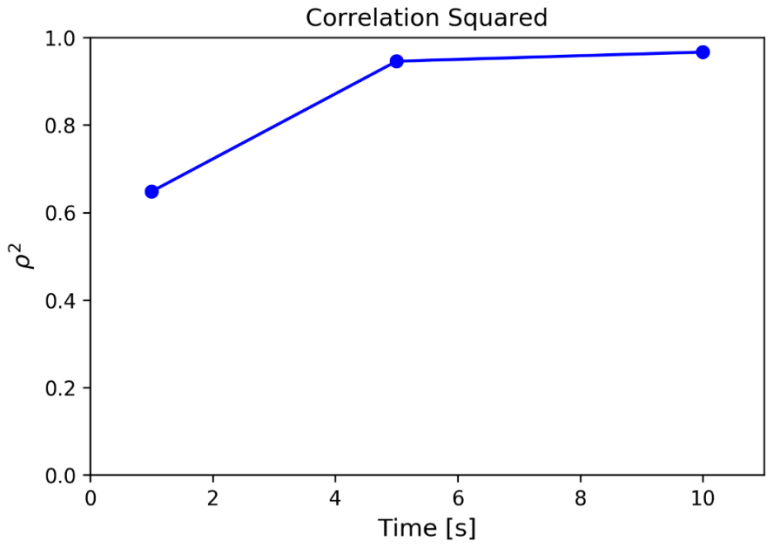


Figure A 12. Correlation squared between the emulation and mathematical models at time steps 1, 5, and 10.

From the pilot study, it is possible to estimate the variance of the number of alerts, which is reported, along with the coefficient of variation, in Figure A 13. We note that the variance of the number of alerts increases with time (as expected), and that the coefficient of variation (defined as the ratio between the standard deviation and the mean) approaches a value of 92%. By leveraging this information (relative to the computational costs of the two models and their correlation), we obtained

the optimal number of mathematical model replicates that would be required to minimize the estimator variance for a fixed number of emulation model experiments. Due to the increase in variance with time, the most restrictive condition is obtained for the time of 10 seconds. At this time, the optimal estimator is obtained by using a total of 86840 mathematical model samples. By adding samples to the original 40 samples from the emulation model, we obtain an estimator with a total cost of 40.53 equivalent emulation model runs. It follows that we can reduce the variance of the estimator by only adding a fraction of the cost of a single emulation model run (0.53).

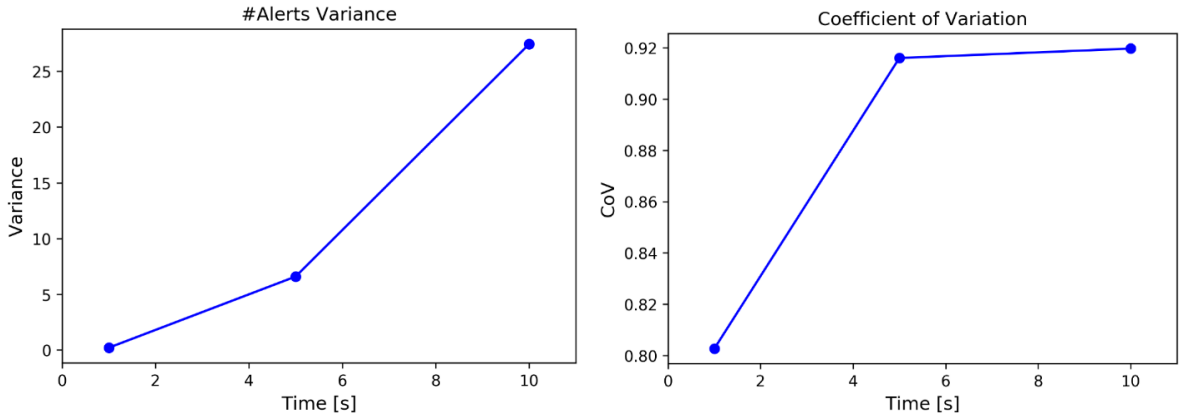


Figure A 13. Variance (left) and coefficient of variation (right) for the total number of alerts.

In Figure A 14, we report the mean number of alerts and the associated 99.7% confidence interval for the MFUQ estimator and the single-fidelity MC estimator. From the experiments, we can also evaluate the estimator variance, which was used to calculate the confidence intervals. We note that the variance reduction that the MFUQ estimators attains increases with time since the Multi-Fidelity estimator can maintain a high variance reduction with respect to MC, thanks to the increasing correlation between the models. The single MC estimator is not able to compensate for the increase in variance over time, and consequently, its confidence intervals grow more rapidly with progressively less accurate estimation for the mean number of alerts.

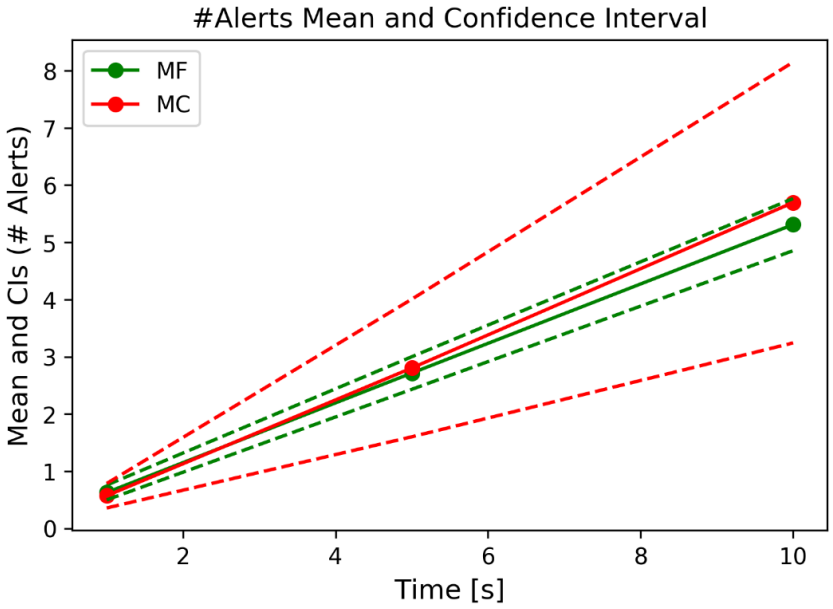


Figure A 14. Prediction of mean number of alerts and associated confidence interval for single (MC) and multi-fidelity (MF) estimators.

A.7. Conclusions

This exemplar demonstrates how the capabilities developed under SECURE can be used to support rigorous cyber experimentation. Specifically, it shows:

- How experiments and metrics can be used to verify the behavior of emulation models.
- How to develop low-fidelity models to approximate high-fidelity models and how to validate the outputs of these models.
- How UQ methods can be used to efficiently explore input and output uncertainty.
- How high- and low-fidelity models can be combined to effectively utilize the experimentation budget.

A.8. Appendix A References

1. Karim I, Vien Q-T, Le TA, Mapp G. A Comparative Experimental Design and Performance Analysis of Snort-Based Intrusion Detection System in Practical Computer Networks. *Computers*. 2017; 6(1):6. https://doi.org/10.3390/computers6010006
2. US CERT (2018). "Alert (TA18-201A) Emotet Malware." accessed October 21, 2020 at https://us-cert.cisa.gov/ncas/alerts/TA18-201A
3. "Snort Rules." accessed October 21, 2020 at https://snort.org/downloads/#rule-downloads
4. S. A. R. Shah and B. Issac. "Performance comparison of intrusion detection systems and application of machine learning to Snort system, *Future Generation Computer Systems,* 80(2018), 57-170.
5. W. Bul'ajoul, A. James, and M. Pannu. "Improving Network Intrusion detection system performance through quality of service configuration and parallel technology". *J of Computer and System Sciences,* 81 (2015), 981–999.
6. https://docs.microsoft.com/en-us/azure/azure-monitor/autoscale/autoscale-common-metrics
7. https://cloud.google.com/network-telemetry
8. https://www.sumologic.com/insight/what-is-telemetry/
9. https://www.intel.com/content/www/us/en/cloud-computing/telemetry.html
10. https://github.com/sandialabs/pyapprox
11. Vugrin, E., J. Cruz, C. Reedy, T. Tarman, and A. Pinar. "Cyber Threat Modeling and Validation: Port Scanning and Detection," *Proceedings of the 7th Annual Hot Topics in the Science of Security (HoTSoS) Symposium*. Sept. 2020. https://doi.org/10.1145/3384217.3385626
12. Geraci, G., Crussell, J., Swiler, L.P. and Debusschere, B. J. "Exploration of Multifidelity UQ Sampling Strategies for Computer Network Applications." *International Journal of Uncertainty Quantification,* 2021. Pp. 93-118. DOI: 10.1615/Int.J.UncertaintyQuantification.2021033774. SAND2021-1221J

APPENDIX B. SCANNING AND DETECTION CASE STUDY

B.1. Overview

This section discusses scanning for vulnerable RTUs (remote terminal units) and the detection of scanning activity within the SCADA network. This activity is part of the end-to-end threat scenario, depicted in the purple box in Figure B 1.

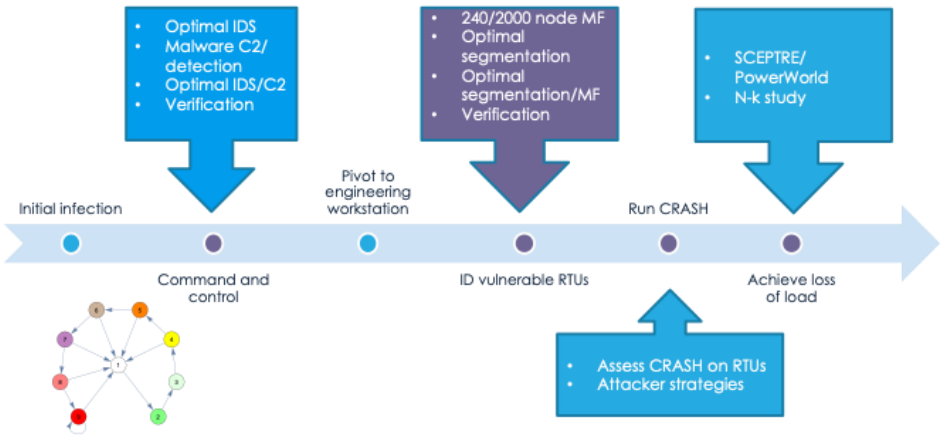


Figure B 1. SECURE end-to-end threat scenario, with SCADA network studies highlighted

In this scenario, when the attacker lands on an engineering workstation in the power grid control center, it doesn't know the IP addresses of RTUs that are vulnerable to the CRASHOVERRIDE malware, so it must scan for them. However, as the attacker is scanning, the defender is monitoring SCADA network traffic and examining it using an intrusion detection system (IDS). One method used by IDS to detect scanning activity is to look for network packets that might indicate such activity, and when these packets are received with an intensity above a certain threshold, the IDS signals an alert. This detection approach guides an attacker's strategy: it can attempt to run slowly "below the radar" of IDS detection (at the expense of launching its attack later), or it can run quickly (at a higher risk of detection).

The following sections describe the mathematical modeling, the ns-3 simulation, and the emulation-based experimentation that were applied to model this step in the attack timeline.

B.2. Scenario

The scenario addressed in the emulation, simulation, and mathematical models assumes the attacker uses Nmap to scan for vulnerable RTUs, and the defender uses Snort (with the sfportscan module) to detect scanning activity. Both tools were selected for these models because they are commonly used, open source, and familiar to the experimental team. In particular, the fact that these tools are open source means that the experimental team can better understand how these tools work "under the hood," which is especially important when developing simulation and mathematical models. However, it's important to emphasize that, although these specific tools were selected for the studies, the methodologies (and, in some cases, the results) are generalizable to other scanning and IDS tools.

B.3. Topology

The topology studied in the emulation, mathematical, and simulation models is shown in the following Figure B 2.

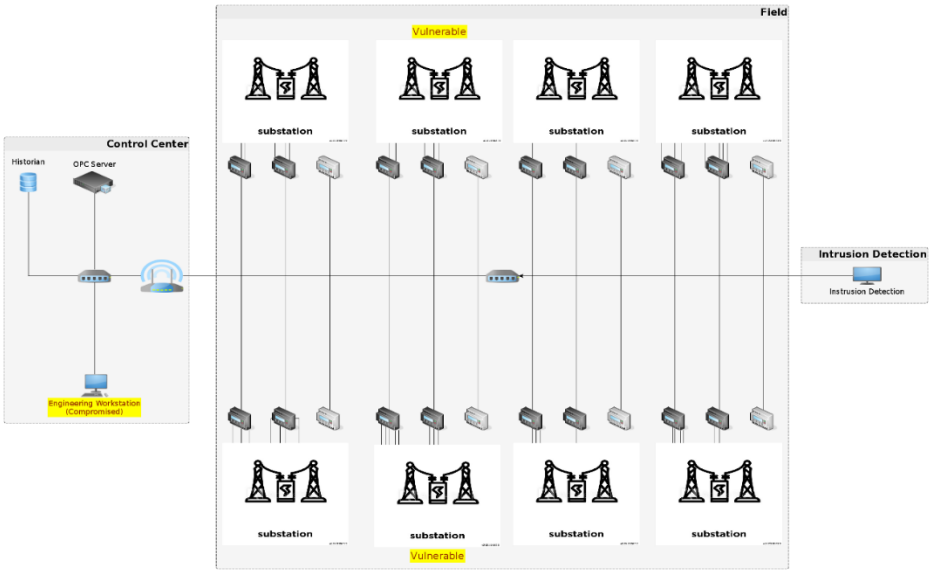


Figure B 2. Notional SCADA network topology for scanning/detection study

This topology (which does not reflect a particular real-world SCADA/ICS network, but is meant to be representative), consists of the following components:

- An engineering workstation in a control center network that represents the attacker's current location, from which it scans the SCADA network for vulnerable devices;
- A router that separates the control center IP subnet from the SCADA network IP subnet;
- An IDS that listens to all traffic on the SCADA network IP subnet;
- 8 SCADA substations, all on the same IP subnet; and
- 24 hosts, distributed across the SCADA substations, configured as follows:
  - 4 hosts are vulnerable to CRASHOVERRIDE,
  - 8 hosts are not vulnerable, but are discoverable,
  - 12 hosts are neither vulnerable nor discoverable.

B.4. Nmap

As described earlier, in our modeled scenarios we configure the attacker node to use Nmap to scan for and find vulnerable nodes. Nmap performs its scan using the Transmission Control Protocol (TCP) connection establishment protocol to look for active IP addresses with open ports, as shown in the following Figure B 3:

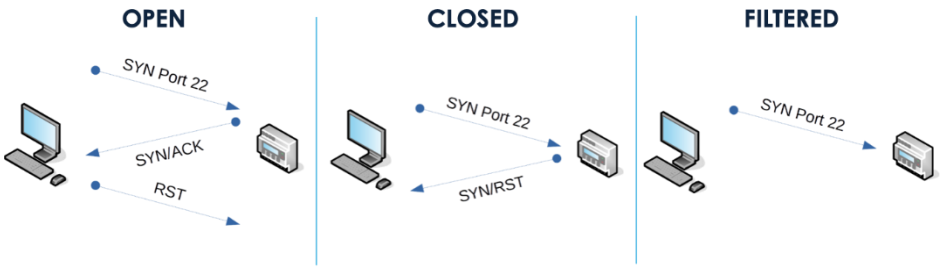


Figure B 3. Nmap protocol operations while scanning open, closed, and filtered hosts

In our scenarios, we model "vulnerable" hosts (see previous section) as hosts that have a particular port in the "open" state, which represents a vulnerable application. When Nmap scans a host on an active IP address with an open port (i.e. an application listening on that port), Nmap sends a TCP SYN (synchronization) packet to that host IP/port combination, and the host responds with a SYN/ACK (acknowledgement). Normally the initiator would acknowledge the connection with a third message, ACK; however, Nmap does not want to maintain an open connection, so it responds with an RST (reset). If Nmap receives a SYN/ACK from a remote host, then it knows two things: that the IP address is valid, and that an application is listening on that port.

Our scenarios model non-vulnerable but discoverable hosts as hosts that have that particular port in the "closed" state (meaning that these hosts are not running the vulnerable application). When Nmap scans a host on an active IP address with a closed port, Nmap sends a TCP SYN packet to the host IP/port combination, and the host responds with a SNY/RST message. Therefore, if Nmap receives a SYN/RST from a remote host, it knows that the IP address is valid, but there is no application listening on that port.

Hosts that are neither vulnerable nor discoverable are modeled as hosts with the IP address/port combination that are "filtered." In this case, when Nmap sends a TCP SYN message these hosts, there is no reply back to the Nmap host, meaning that the host either does not exist, or chooses not to reply.

Intrusion detection systems (IDS) will observe these connection request/response packets and use them to determine whether a scanning attack is occurring, as described in the next section. To counter IDS, Nmap has a couple of command line configurations that can be used. To reduce the scanning traffic intensity, Nmap allows the user to increase the delay between scanning probes (the "delay" parameter) and decrease the number of hosts that are probed in each attempt (the "host group" parameter). By default, Nmap scans hosts in sequence by IP address; however, that approach could tip off an IDS, so Nmap has a command line parameter to randomize the sequence in which the hosts IPs are probed.

In our studies, we varied parameters related to attacker strategy (i.e. "fast" vs. "slow") and randomness (i.e. "sequential" vs. "random"). In addition, we also configured our experiments to allow random packet drop (i.e. "no drop" vs. "drop"), to determine the effect of imperfect packet transfers on results. The combination of the randomness order and random packet drop parameters are organized into two formulations: a *deterministic* formulation (i.e. sequential ordering, no packet drop) and a *stochastic* formulation (i.e. random ordering, random packet drop). The plots shown later in this section show results from both formulations.

B.5. Detection

Our scenarios assume intrusion detection using Snort (Ref. 2). Snort is a very flexible IDS framework that uses signature definition files and rules to identify traffic as malicious. In this example we use the "sfportscan" rule to detect Nmap scanning traffic using the technique identified in the previous section. As shown in Figure B 4, the sfportscan rule looks for SYN/RST traffic from "closed" (i.e. non-vulnerable, but discoverable) hosts, which is indicative of a scanning attack. If Snort/sfportscan counts five or more SYN/RST packets within a 60 second window, then it generates an alert. Our models and scenarios consider two attacker strategies: a "fast" strategy where the attacker attempts to discover as many vulnerable nodes as quickly as possible, and a "slow" strategy where the attacker attempts to stay within the 5 SYN/RST packets within a 60 second threshold. The results shown later in this section account for both strategies.

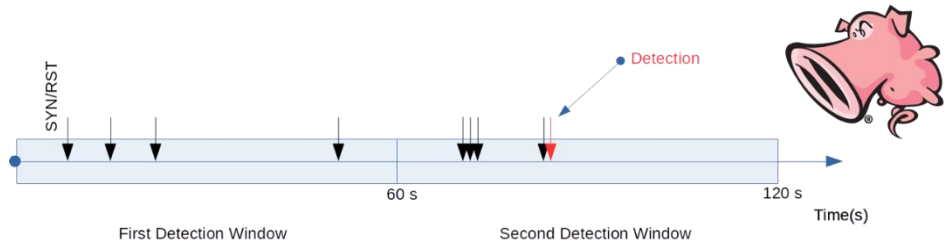


Figure B 4. Snort "sfportscan" rule

B.6. Tools

B.6.1. Mathematical model

We developed a mathematical model to assess the port discovery process. The model describes the stochastic state transitions that occur within the Nmap protocol that occur over time during the scanning process. This mathematical model is described in detail in (Ref. 1) and summarized below and in Figure B 5:

1. The model states (illustrated in Figure B 5) are defined by the progress that Nmap makes scanning the nodes. The initial state at time 0 (indicated in the state on the far left of Figure B 5) contains key model parameters provided to the model. Each state consists of three lists that track the nodes that have yet to be scanned (topmost list in the state figure), the nodes that are being actively scanned (middle list), and the nodes that have already been scanned (bottom list). Furthermore, the color of the dots in the lists indicates the scanned nodes' status - magenta for filtered (inconclusive), green for closed (secure), and red for open (vulnerable). All nodes begin in the first "To Scan" list in the initial state.
2. The model describes the transition from the initial state to subsequent states (in the second column in Figure B 5). The transition probabilities Pr{# filtered, # closed, # open} are determined by the number and type of nodes that have yet to be scanned and the probability that combinations of nodes are selected for scanning.
3. The third step the model consists of a third set of states (third column) that describe which nodes have been discovered (i.e. TCP SYN/RSTs occurred) and which ones timed out. The transition probabilities are conditioned on the current (second) state and depend on which nodes have been discovered so far. That is, the transition probability is Pr{# filtered_to_scan, # closed_to_scan, # open_to_scan | # filtered, # closed, # open}
4. If timeouts occurred, steps 2 and 3 are repeated.
5. Steps 2-4 are repeated until all nodes are moved to the Scanned list.

The steps in the model are implemented to effectively create a probability tree that lists the probability of discovering open, closed, and filtered nodes at each time step.

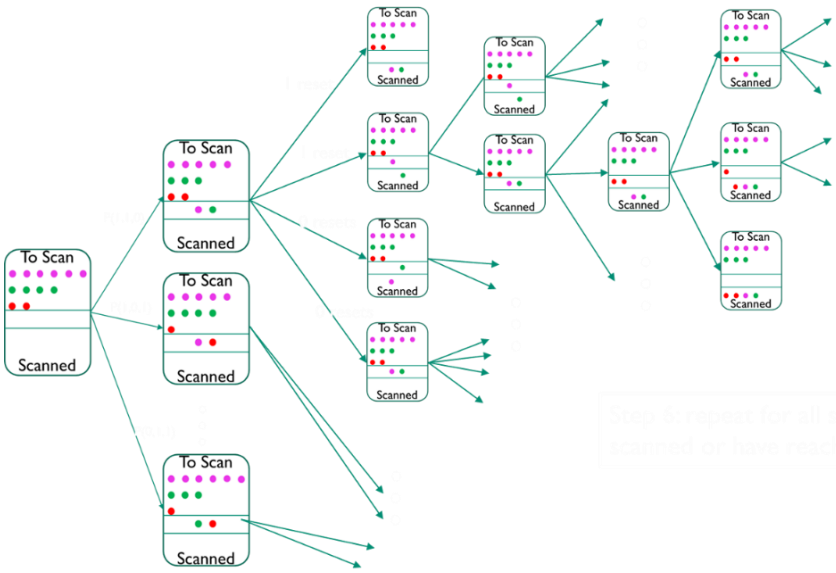


Figure B 5. Mathematical state transition diagram

We use the model results to compute the statistics of port discovery. Figure B 6 shows the open port discovery process. The magenta stars represent the mean number of open ports discovered, as calculated with the math model. The blue line represents the mean number of open ports discovered from 1000 runs of the `minimega` emulation model, and the dashed black lines represent the 95% confidence intervals on the emulation means. The plot shows the mathematical results tracking the mean of the `minimega` runs and falling within the 95% confidence interval of these runs. This agreement validates the predictive value of the mathematical model, which, for small topologies, can run more quickly than the emulation model, making it more suitable for more widely evaluating the effect of configuration parameters (e.g. host group size and delay) on the results.

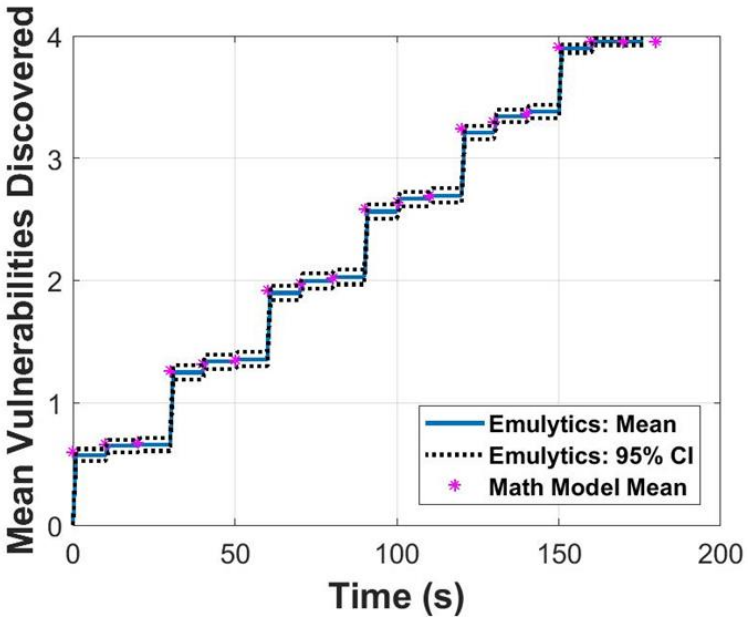


Figure B 6. Port discovery analysis (mathematical model and `minimega` emulation)

The model results were also processed to determine when and if detection would have occurred using the logic in the Snort sfportscan algorithm. These times were compared against the detection times that were experimentally determined using the `minimega` topology. The mathematical results, shown in Figure B 7, also closely track the results from the emulation runs and, again, validate the mathematical model's predictive ability.

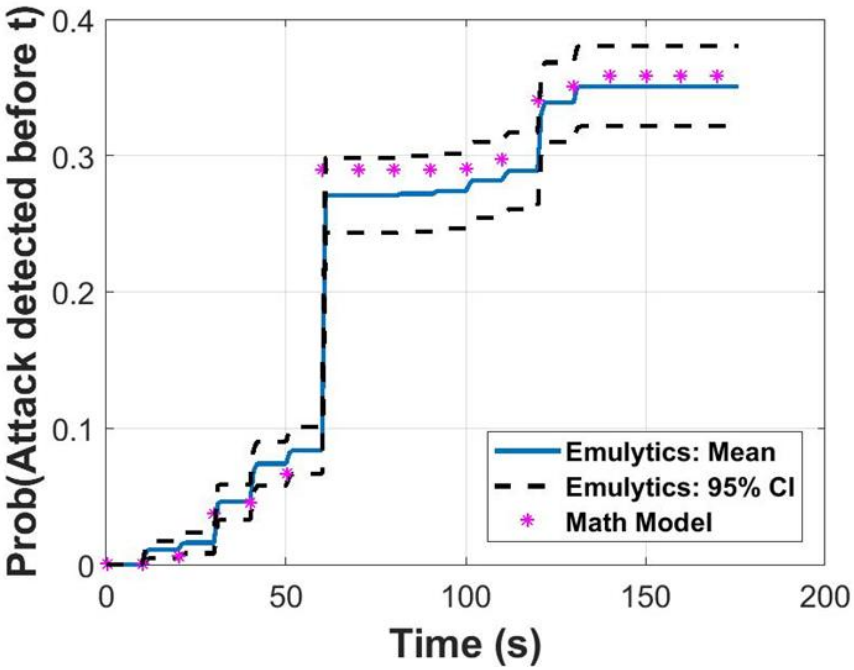


Figure B 7. Detection times

B.6.2. ns-3

Ns-3 (Ref. 3) is a discrete event simulator that is used for network simulation and has an extensive model library for various network links, devices, and applications. Because it is a simulation, the components are abstracted objects and it does not run real implementations of applications and protocols. However, ns-3 simulations can run much more quickly when compared to emulations because discrete event simulations are event-driven rather than time-driven and can run faster than real time. This makes an ns-3 simulation particularly useful for serving as the low fidelity model in multi-fidelity modeling studies because it is much more efficient, and if implemented correctly, well correlated with emulation runs.

The Nmap ns-3 model developed in this work implements two major components - a topology and an Nmap application simulation model. The topology, shown in Figure B 8, corresponds to the SCADA network topology described earlier, but is different from the emulation model topology in a couple of ways:

- The ns-3 simulation topology has each SCADA device on its own subnet:
  This design choice is an artifact of how the example ns-3 star topology code does subnetting, and should not appreciably affect packet timings and results. Nevertheless, it does affect scalability of the topology because the subnetting schemed used in the model only allows up to 255 subnets (and with one host per subnet, 255 hosts).

- Different mechanisms are used to implement closed and filtered nodes:
  Whereas the emulation uses iptables filtering to implement closed and filtered nodes, the ns-3 model does not install a packet sink on closed nodes, and causes Nmap to scan unused IP addresses for filtered nodes.

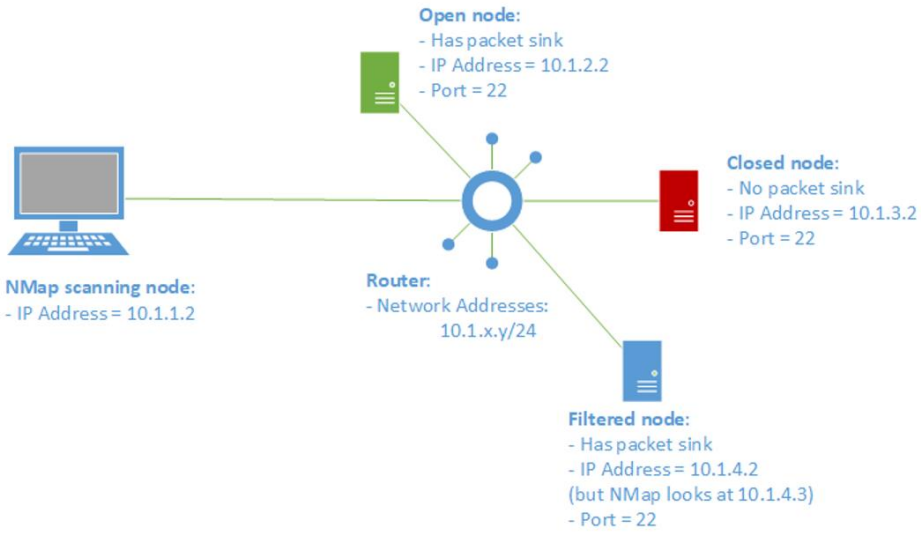


Figure B 8. ns-3 model for scanning/detection

The Nmap application running on the scanning node functions similarly to the real Nmap application running in the emulation. Also, the ns-3 model implements packet dropping using a similar mechanism that is used in the emulation model.

B.7. Emulation experiments using Scorch

The name SCORCH comes from the terms SCenario ORCHestration. It is primarily an automated scenario orchestration framework for emulation-based models, where a scenario is a specification of high-level experimental behaviors for a given experimental goal. Concretely, SCORCH is implemented as a python package that interfaces with `minimega` to run experimental scenarios on and collect data from emulation-based models (EBMs) managed by `minimega`.

At a high-level, basic SCORCH usage is as follows. First, a *scenario configuration file* is created that defines a scenario (experimental behaviors), model parameters, and output parsing. This file describes the "what" of the experimental scenario. The scenario is defined in terms of modular scenario *components* which represent re-usable experiment primitives. The code implementing components describes the "how" of the experimental scenario.

Secondly, a `minimega` topology is deployed on a hardware cluster (or single machine). This is the EBM to which the experimental scenario will be applied. This step highlights a degree of separation between structure and function of the experiment. The `minimega` topology represents the structure of the experiment while the SCORCH scenario represents the function. This separation enables efficiency in experimentation by, for example, enabling the user to apply the same scenario to a variety of topologies without the need to re-create the scenario for each topology, or enabling the user to apply a variety of scenarios to the same topology without having to tear down the topology.

In this study, the SCADA network topology is deployed within `minimega` where each virtual machine (VM) receives the necessary software and model parameters to execute the scanning/detection scenario. For example, the scanning VM includes Nmap and a list of parameters such as: number of IP addresses and ports to scan, specific port number to scan, time to wait between scans (delay), etc. This set is subsequently used to scan the SCADA network. Each time a port is scanned, the metadata associated with the scan is logged to an Nmap.out file. To counter the adversarial scanning VM, the detector VM runs snort and its configuration parameters capable of sensing the syn packets used in Nmap probing. If snort notices a packet that aligns with criteria in one of its rules, it will signal an alert and append all such to an alert file. During this reciprocal exchange, tcpdump captures all traffic on the network by way of a port mirror residing on the `minimega` virtual LAN hosting the SCADA network. This data is saved as a PCAP file.

B.7.1. Data collection

Input/output to and from the live virtual network is handled by the individual components as facilitated by the framework. Here, the `minimega` command and control agent, miniccc, handles the data input and output process, in tandem with the snort, tcdump, and filebeat components. During SCORCH execution, the Nmap and snort components call miniccc to signal that their respective model parameters and other supporting data, be added to the model. This occurs during EBM setup, where miniccc copies the data from the hardware cluster node to the respective VM within the `minimega` topology. After the experiment has completed, each component initiates an exfil process where it again calls the miniccc agent to extract any logging data accrued by Nmap, snort or tcpdump. This data is then written to the host cluster node for analysis. If enabled, SCORCH interfaces with Filebeat to push the collected experimental data and artifacts to a specified Elasticsearch server.

Following data collection, post processing scripts run against the PCAP and snort alert files to derive the time delta (in seconds) between the 1st packet captured and the 1st alert instance captured, for every Nmap portsweep occurrence. If any time format discrepancies exist between the PCAP and alert file, the scripts will convert the packet time to reflect seconds since Unix epoch time (Jan 1, 1970). Once the initial alert time values have been calculated, the post processing scripts aggregate the initial alerts times for every experiment and log to a metrics.txt file. This is done for each snort sensitivity level (low, medium, high).

B.8. Experimental methods

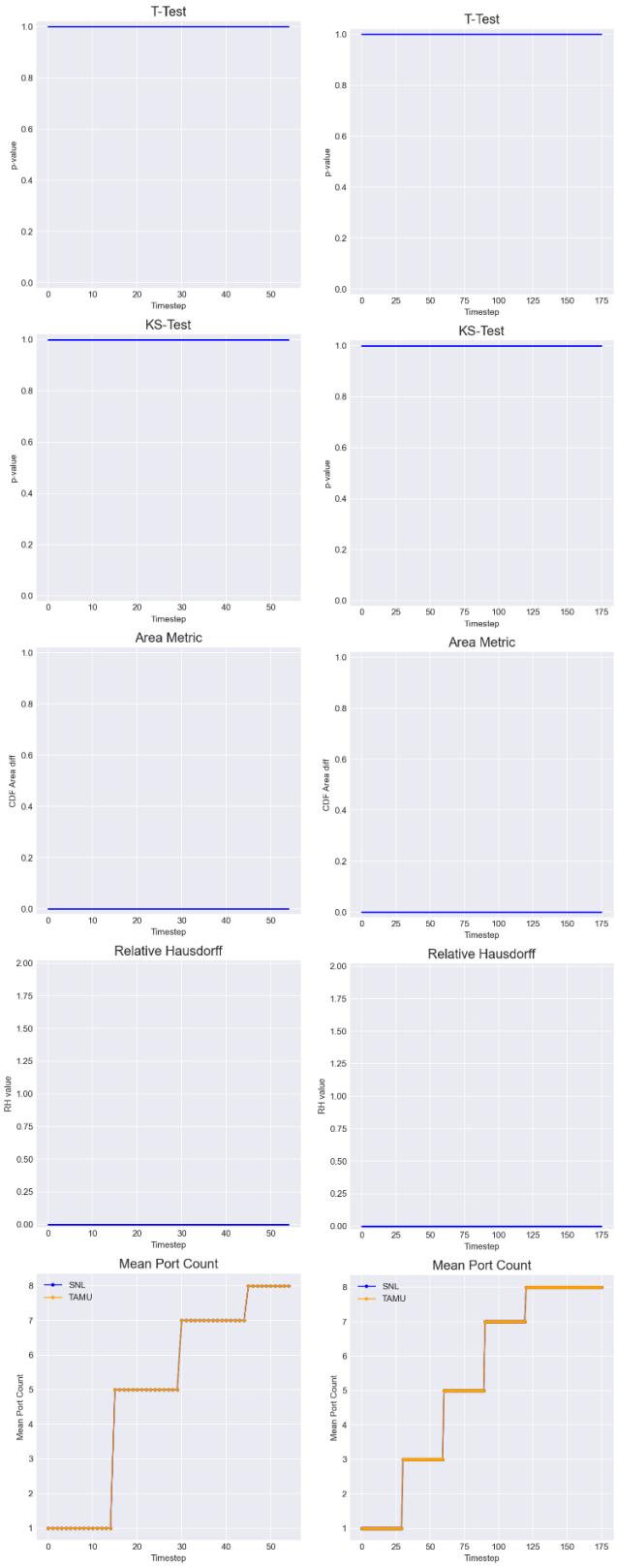
B.8.1. Experiment reproduction

Reproducibility is essential to science because it ensures results are not biased according to overt or hidden desires for a particular outcome. The SECURE team, working with our collaborators from Texas A&M University (TAMU), wanted to see understand the degree to which the results published in (Ref. 1) can be reproduce by a research team that did not contribute to the original paper. In the process of reproducing this study (which is described in detail in Ref. 4), the team not only considered the methods for reproducing the results, but also the metrics by which the results from Sandia and TAMU should be compared. The comparison metrics used during this study were:

- t-test: the t-test is a widely-used test for determining if there is a statistically significant difference between the means of two data sets,

- Kolmogorov-Smirnov Test: the KS-test is a non-parametric statistical test for equality of distributions, based on the maximum difference between the cumulative distribution functions (CDFs),
- Area Test: the area test also compares CDFs, but accounts for the entire difference between CDFs rather than the maximum difference, and
- Relative Hausdorff Distance: originally developed for graph analysis, the Relative Hausdorff Distance can also be used to compare distributions

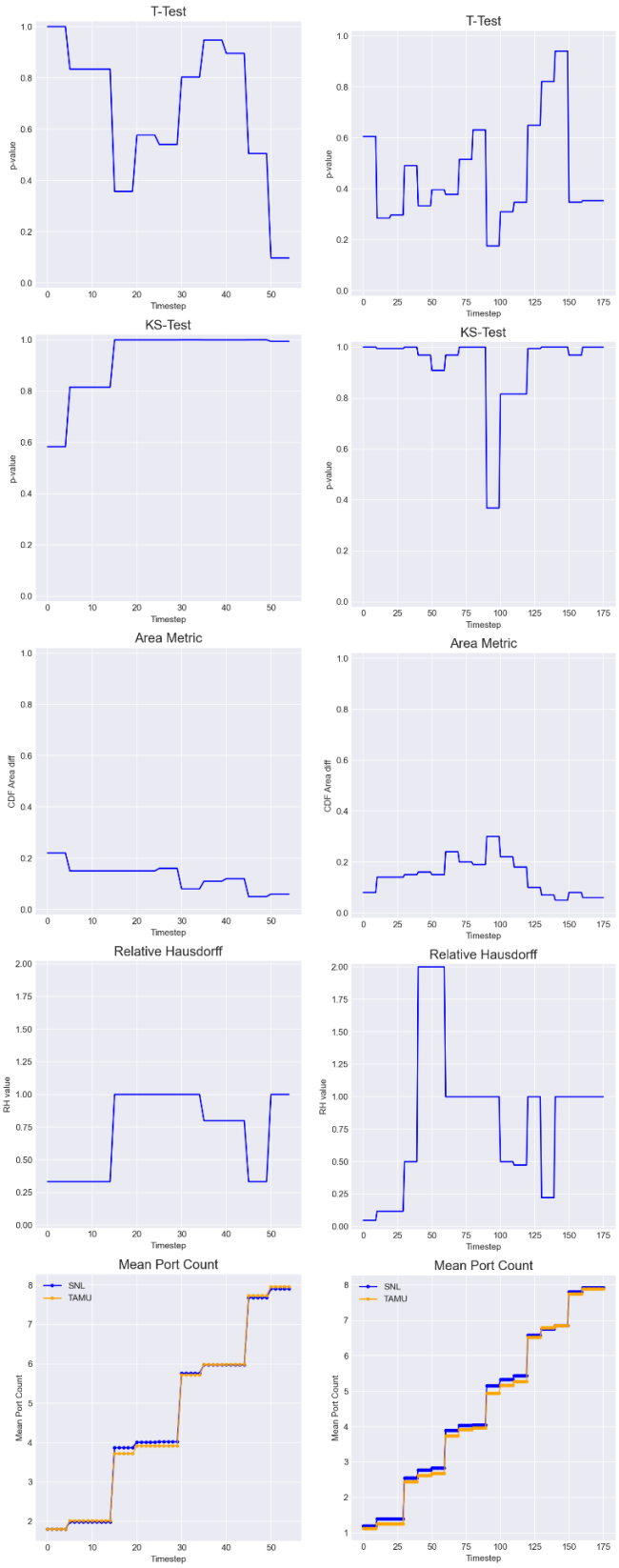
The plots in Figure B 9 show the application of these metrics to compare Sandia and TAMU port discovery results in the case where there is no added randomness (i.e. deterministic formulation):



Fast, deterministic Slow, deterministic

Figure B 9. Port Discovery Statistical Test Results for Deterministic Case

The results above show perfect agreement between the Sandia and TAMU results, as evidenced by all four metrics, indicating that TAMU correctly set up the experiment for the deterministic formulation. The plots in Figure B 10 show the application of these metrics to compare Sandia and TAMU port discovery results in the case where there is there is added randomness in the Nmap search order and in packet loss (i.e. stochastic formulation):



Fast, stochastic Slow, stochastic

Figure B 10. Port Discovery Statistical Test Results for Deterministic Case

From these comparisons we find that the KS Test shows good agreement between the Sandia and TAMU results, as evidenced by the p values > 0.05. The Area Metrics for all cases also show good agreement as evidenced by the consistently low area values. However, we find that the Relative Hausdorff metric does not seem to be a suitable metric for comparing results, as seen in the plots above.

B.9. Verification

An important part of using emulation is verifying whether the emulation environment is working as intended, also called verification [Ref. 5]. Part of verification involves software testing and quality assurance. A unique aspect of cyber emulation involves assessing the performance of the emulation running in the virtualized environment and determining whether there are sufficient resources to properly handle the scenario that is being run. If there are not, the virtualized components may produce experimental artifacts and behavior that result in the experimental outcomes being unrepresentative or incorrect.

Under SECURE, we focused on determining whether there are sufficient virtualized resources to support the emulation experiment and whether we can identify metrics that indicate when the results of an emulation experiment are unreliable. We refer to these metrics as telemetry metrics, following the usage of this phrase from Microsoft [6], Google [7], Intel [9] and Sumo Logic [9]. We studied telemetry metrics such as system load and CPU utilization relating to the performance of virtual machines which are used in the scanning/detection scenario and the physical machine hosting that study. We ran experiments with various levels of over-subscribed resources.

In these experiments, we purposefully put more and more strain on the physical resources available to the emulation experiments. We accomplished this by forcing the physical host to do more and more work in parallel through the concept of a namespace, which is an isolated copy of the experiment environment running on its own VLAN. For the purposes of this study, we ran several iterations of the same experiment with increasing numbers of parallel namespaces. By increasing the number of namespaces, we hoped to reach a point of resource over-subscription, where the results of the experiments run are affected by emulation artifacts caused by this over-subscription. We saw evidence of oversubscription at 20 namespaces and greater [Ref. 10].

We found that statistical tests such as the Tukey multiple mean comparison test was useful to identify anomalies in results as we increased the number of parallel namespaces running in the experiments. For scanning/detection, as we increased namespaces, we found that the alert time distributions shifted upward and became much more diffuse with longer tails. We also found that the telemetry metrics of system load and throughput were effective at filtering out replicates which had statistically significantly different results than the baseline case with one namespace [10].

B.10. Validation

Validation is the process of verifying that the model is correct with respect to the questions that it is intended to answer. Validation can be done in several ways; it can be performed on multiple models and compared (i.e. cross-validation), and validation experiments can be conducted in physical testbeds and compared with models. In the SECURE project we performed two different kinds of physical experiments and compared the results with the `minimega` scanning/detection model:

1. Validation experiments on physical hosts in the Sandia Carnac cluster. The physical hosts used for these experiments were all identical, but configured differently to assume different roles in the validation experiment, and
2. Validation experiments using physical and virtual devices in the Texas A&M testbed. Physical relay devices were used to model vulnerable hosts (open ports) in the scanning detection scenario, however, due to limited numbers of physical devices, closed ports were modeled using the CORE virtual machine testbed, and filtered devices were modeled using firewall rules in the network switch.

The Carnac validation experiments utilized the same software (applications and operating systems) that was used in the `minimega` virtual machine-based experiments. The primary differences between the `minimega` and Carnac experiments were 1) `minimega` used KVM-based virtual machines whereas the Carnac experiments were run on physical hosts, and 2) a few configuration differences due to differences in networking between the virtual and physical experiments. We found the port scanning and Snort detection time results between the `minimega` and Carnac experiments matched up very well.

The Texas A&M University (TAMU) physical testbed experiments used a mixture of physical and virtual hosts in order to achieve the scales that were needed to conduct the validation experiment. The TAMU team used four field devices to implement vulnerable hosts with open ports, eight virtual machines running in the CORE virtual testbed environment to represent secure hosts with closed ports, and used firewall rules in the network switch to represent 12 secure hosts that are filtering inbound TCP connection requests. The TAMU physical testbed configuration used the same scanning and detection software used in the `minimega` experiment, however, because the TAMU testbed was very different from the `minimega` testbed, a number of custom scripts were written to orchestrate the experiment and collect data. These scripts required some amount of debugging, resulting in some back-and-forth between the Sandia and TAMU teams to make sure the physical experiment was producing correct validation data. Due to limitations in available time, the two teams were able to validate port discovery but did not have an opportunity to assess validation with respect to detection times. Detailed port discovery validation data are presented in Ref. 11.

B.11. Optimal segmentation

Network segmentation is a strategy used by the network designer to limit the scope of what an attacker may see if they are able to achieve a malware presence on the network. However, network segmentation has costs and constraints on the network design - too much segmentation will incur excessive costs and exceed the defender's budget. Therefore, a tri-level optimization formulation was developed to account for 1) network designer's budget, 2) attacker's budget (in terms of the number of networks that the attacker can compromise), and 3) the network operator's response to an attack (e.g. re-dispatching generation resources to loads). This optimization model and results are documented in Ref. 12.

B.12. Appendix B References

1. Vugrin, Eric D., Jerry Cruz, Christian Reedy, Thomas Tarman, and Ali Pinar. "Cyber Threat Modeling and Validation: Port Scanning and Detection." *Proceedings of the 7th Symposium on Hot*

Topics in the Science of Security, Lawrence, Kansas, Association for Computing Machinery, 2020. https://doi.org/10.1145/3384217.3385626
2. https://www.snort.org
3. https://www.nsnam.org
4. CSET: T. D. Tarman, T. Rollins, L.P. Swiler, J. Cruz, E. Vugrin, H. Huang, A. Sahu, P. Wlazlo, A. Goulart, and K. Davis. Comparing reproduced cyber experimentation studies across different emulation testbeds. USENIX 14th *Cyber Security Experimentation and Test (CSET) Workshop*. Aug. 9, 2021. SAND2021-5696C.
5. Oberkampf, W.L. and C.J. Roy. Verification and Validation in Scientific Computing. Cambridge University Press, 2010.
6. https://docs.microsoft.com/en-us/azure/azure-monitor/autoscale/autoscale-common-metrics
7. https://cloud.google.com/network-telemetry
8. https://www.intel.com/content/www/us/en/cloud-computing/telemetry.html
9. https://www.sumologic.com/insight/what-is-telemetry/
10. Thorpe, J., Swiler, L.P., Cruz, G., Tarman, T.D., Rollins, T., and Debuscherre, B. "Verification of Cyber Emulation Experiments Through Virtual Machine and Host Metrics." Manuscript in preparation.
11. T. D. Tarman, L.P. Swiler, E. Vugrin, H. Huang, A. Sahu, P. Wlazlo, A. Goulart, and K. Davis. "Validation of cyber experiments: comparing emulation results against a physical testbed." Manuscript in preparation.
12. B Arguello and E.S. Johnson and J.L. Gearhart, "A Trilevel Model for Segmentation of the Power Transmission Grid Cyber Network", arXiv.2108.10958: https://arxiv.org/abs/2108.10958. SAND2021-10208O

APPENDIX C. SCADA NETWORK/POWER GRID IMPACTS

C.1. Overview

This section demonstrates how the methods developed under SECURE can be used to analyze the power grid impacts of the larger attack chain. Recall that the full end-to-end exemplar considered under SECURE describes a multi-stage attack in which an attacker attempts to access a power utility's corporate enterprise network, pivot to the ICS network, identify vulnerable RTUs, run the CRASHOVERRIDE malware and ultimately disrupt operations by causing load shed. The focus of this article is the power grid impacts caused by the CRASHOVERRIDE malware.

C.2. CRASHOVERRIDE

CRASHOVERRIDE was malware designed to attack power grids and was used in the 2016 cyber attack on the Ukrainian electric grid. Unlike the previous attack on the Ukrainian grid in 2015 in which attackers manually switched off power to electrical substations, the CRASHOVERRIDE attack was fully automated and could perform attacks much more quickly and with less preparation. Once the malware had infected the system, CRASHOVERRIDE could launch four payload modules. This study focuses on the module that communicates directly with grid equipment and switches breakers within the power grid. https://www.dragos.com/resource/crashoverride-analyzing-the-malware-that-attacks-power-grids/

In power systems, field devices (such as relays, RTUs and PLCs) monitor and control the power grid. CRASHOVERRIDE understands how to enumerate and discover the inputs and outputs to field devices and leverages this to open circuit breakers in the power system. Additionally, CRASHOVERRIDE can force the field devices into an infinite loop thus continually opening the circuit breakers even if operators are dispatched to re-close them.

In our multi-stage attack, Nmap is used to scan the network for vulnerable RTUs. CRASHOVERRIDE will then target only those RTUs and open the breakers associated with those RTUs. The power grid impacts of this CRASHOVERRIDE attack will highly depend on the identification of vulnerable RTUs.

C.2.1. CRASHOVERRIDE Configuration

CRASHOVERRIDE modules were designed to be used with configuration files specifying various parameters of the attack. This section focuses on the configuration associated with the module that targets the protocol payload. In this configuration, a set of stations are specified for an attack. Each targeted station has the following configuration options:

- target ip - specifies the IP address of the targeted field device
- first action - specifies the first action (on or off) used to switch grid components
- change - specifies whether to continually toggle power grid equipment (1) or only change once (0)
- interval - specifies the time interval in between toggles

C.3. TAMU Topology

Power grid impact experiments were all performed on a synthetic cyber-physical topology of the Texas power grid developed by Texas A&M University's (TAMU) Cyber Physical Resilient Energy Systems (CyPRES) project. https://cypres.engr.tamu.edu/test-cases/

This topology consists of both cyber and physical components. The cyber model shown in Figure C 1 has three main sets of components: (1) balancing authorities, (2) utility control centers, and (3) substations. The primary and secondary balancing authorities are responsible for managing the flow of electric power among the utilities. The utility control centers are responsible for monitoring multiple substations and contain networking equipment, a demilitarized zone, and SCADA software. The substations are responsible for monitoring and controlling the power grid and contain networking equipment, relays, as well as corporate devices such as PCs, security cameras, phones, and card readers. The relays in each substation are mapped to busses and branches of a synthetic 2000-bus power model of the Texas grid shown in Figure C 2. Overall, this topology contains 2 balancing authorities, 150 utility control centers, and 1251 substations.

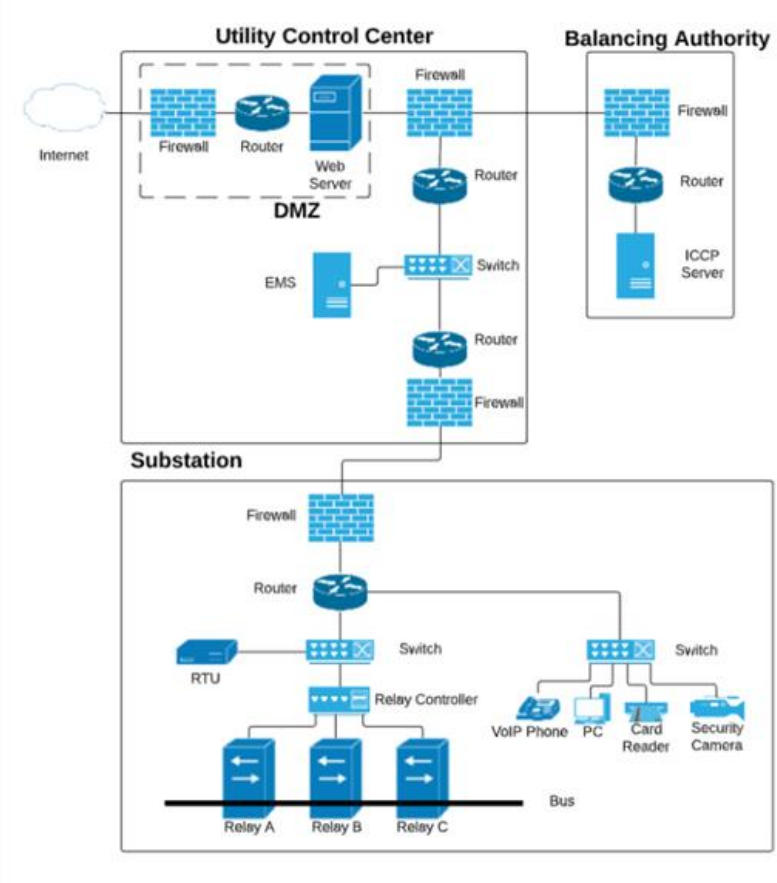


Figure C 1. TAMU cyber topology

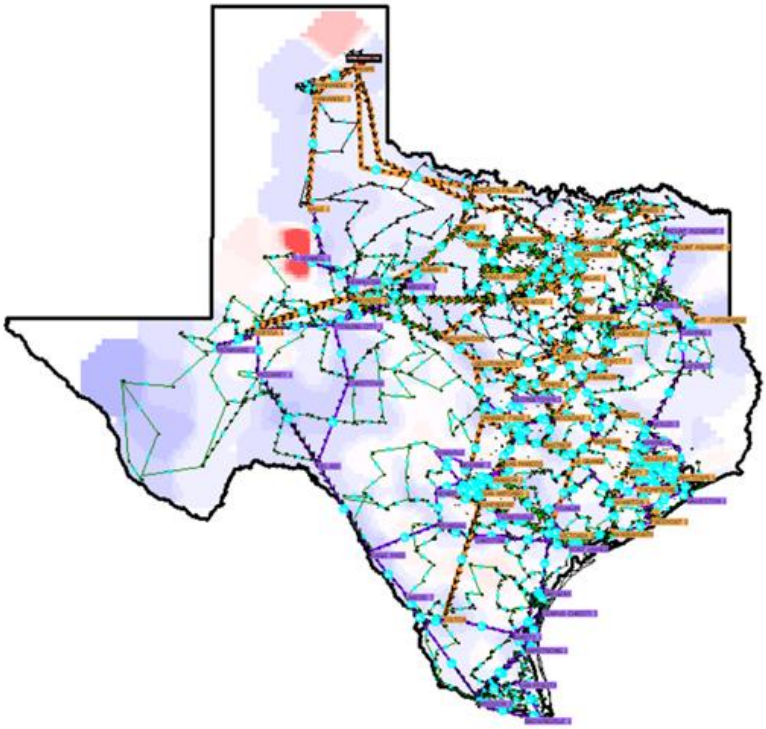


Figure C 2. 2000-bus power model

C.4. Power Grid Impact Studies

The CRASHOVERRIDE malware and the TAMU topology were used for two main studies: an uncertainty quantification study and an optimal segmentation study.

C.4.1. UQ Study

A workflow was developed for the UQ study that leverages both traditional UQ tools and emulation tools. Dakota provides a means to sample CrashOverride parameters and generates a CrashOverride configuration file. For each sample of parameters, Scorch then injects the new CrashOverride configuration file into the SCEPTRE experiment, runs the CrashOverride malware in SCEPTRE, collects physical process data from the power model, and then resets the SCEPTRE emulation. Dakota then chooses the next sample and the process repeats. The data is then post processed and can then be further analyzed.

The UQ study was performed on a small subset of the TAMU topology consisting of 1 balancing authority, 2 control centers, and 11 substations. All protections on relays in the topology were disabled so that the effects of CRASHOVERRIDE could be clearly identified. 800 experiments were run sampling the parameters in Table C 1. The overall timing of each experiment was 150s; the first 30 seconds of each experiment was normal operations. CRASHOVERRIDE was executed at the 30s mark and was run for an additional 2 minutes. The physical process data was post-processed to calculate loss of load for each experiment.

Parameter	Values
target ip	set of 49 relay IPs
first action	[off]

change	[0, 1]
interval	[10, 11, 12, ..., 60]

Table C 1. Parameters of UQ Study 1

Figure C 3 shows results of the UQ study. Each point on the plot shows the loss of load results for a single experiment. The red line shows the mean regression line. the green line shows the median regression line while the black lines show the regression lines for the 0.05, 0.1, 0.25, 0.75, 0.9, 0.95 quantiles respectively.

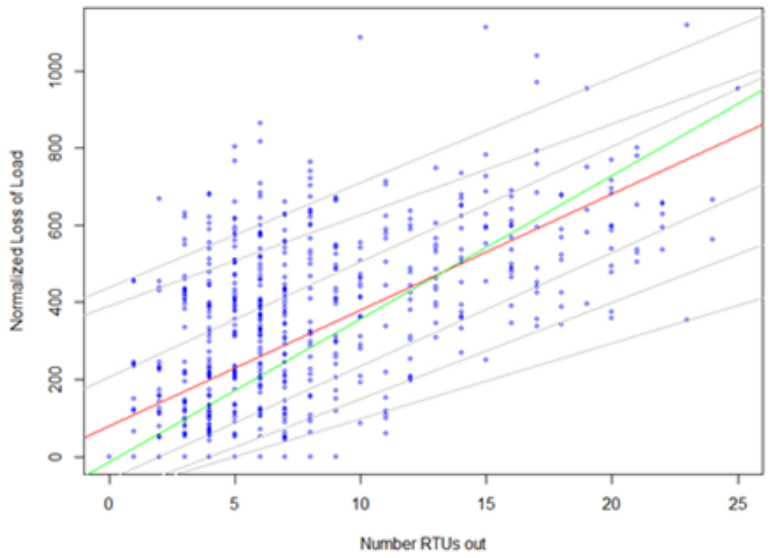


Figure C 3. UQ Experiment Results with Quantile Lines for Normalized Loss of Load

For a given number of RTUs out (such as 4), there is a huge spread in the loss of load based on which four RTUs are targeted. This variance makes it hard to get a good regression model: the regression captures the mean trend but does not capture the variance well. If we instead look at the quantile regression lines, there is a better trend than with the mean regression line. Each quantile regression also gives us an analytic formula for a tail probability of normalized loss of load. For example, the 95th quantile = 440.18+27.10*RTUs_out. This formula can be used in end-to-end CRASH studies, where we want to couple upstream attack uncertainties to a tail probability loss of load (instead of worst case).

Future studies are planned, to increase complexity of the model by scaling the size of the topology as well as reimplementing the relay protections. However, due to the large variability of results present in the small topology, future work will first include more analysis of the current results such as worst-case analyses.

C.5. Segmentation Study

The second study using the TAMU topology and CRASHOVERRIDE malware was a segmentation study. The optimal segmentation work determined optimal segmentation of a network using mathematical optimization. This study applied the mathematical results to the TAMU topology and investigated the impacts the CRASHOVERRIDE malware would have on this new, segmented topology. We hypothesized that using the mathematical results would decrease the impact of the

CRASHOVERRIDE malware since optimal segmentation would force the attacker to pivot more within the network to deliver the CRASHOVERRIDE payload to specific relays.

A workflow was designed that interfaces emulation with mathematical optimization for network segmentation. The workflow starts with an initial SCADA network implemented in SCEPTRE. The design of the topology (i.e. current network segments) is input to the mathematical optimization. The mathematical optimization then does two things. 1) identifies the worst-case attacker on the original topology and 2) identifies a new optimally segmented network topology along with the worst-case attack for this new topology. The SCEPTRE topology is then updated with the new segmented topology. Theoretically speaking, this is done by re-subnetting and applying new firewall rules. However, for our example, we wanted to investigate the effects of the CrashOverride malware against the optimal and non-optimal network topologies. So practically speaking, we investigated this by simply changing the potential targets of CrashOverride based on the segmentation that came from the optimization.

To gather results, the worst-case attacker (specific to each topology) was used to identify the set of RTUs that CrashOverride would target. The CrashOverride malware was implemented and for each topology, 100 experiments were run varying the other parameters of CrashOverride. Figure C 4 shows results of this study. The results show that the optimal segmentation of the network lowered the cumulative loss of load for the scenario.

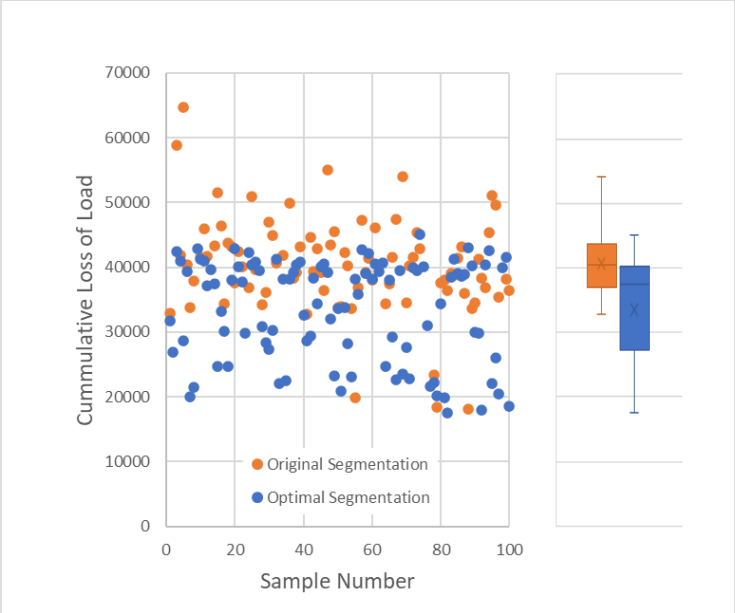


Figure C 4. Segmentation Results

Moreover, this study shows the value of coupling mathematical optimization with emulation. Determining an optimal segmentation in emulation is usually SME driven and would require full enumeration or brute force to determine a true optimal solution. This is infeasible in emulation so practically heuristics would normally be used, but these do no guarantee an optimal or even near optimal solution. By coupling the emulation with mathematical optimization, most of the burden is done by the lightweight mathematical model.

Beyond this initial study, other questions we want to answer about this study are:

1. Can we use emulation to show that the mathematical result is better than SME design?
2. Can we use emulation to explore the robustness of the mathematical abstraction?

3. Does the incorporation of other real-world parameters (such as scanning and detection probabilities) affect the optimality of the segmentation?
4. What are the tradeoff costs between cost to implement segmentation versus benefit the segmentation provides against an attacker?

APPENDIX D. OPTIMAL SENSOR PLACEMENT MODEL

This appendix describes the optimal sensor placement model, which is a network interdiction model that identifies where a defender should place intrusion detection systems (IDSs) in a cyber system to minimize the probability that attackers can successfully achieve a given attack objective. For example, an attack objective could be to cause a specific amount of load not served by the power grid.

This model combines four key elements to create a realistic representation of a cyber-physical system, and attacker and defender behaviors. First, it uses cyber-attack graphs to represent the landscape where attackers and defenders make decisions and interact. Attackers must move across these graphs from the initial point of compromise to locations where they can produce a cyber or physical consequence. At each step of the attack they will select subsequent actions so that the probability of being detected is minimized. Second, it captures the constraints that defenders must consider when installing IDS in a cyber network. Third, it uses threat information about known cyber threat groups to model attacker behaviors. Finally, when applicable, it uses physical system models to ensure that the attacks that occur on the cyber network produce the desired effect on the physical system.

This model uses bi-level programming to capture the interactions between the defender and the attacker. The defender makes the first move by selecting which IDSs should be installed in the cyber network and how they should be deployed. The attacker can see the defender's decisions before deciding which actions to take and will change their attack strategy based on where sensors are placed. While the defender must move first, they are able to anticipate the attacker's response to their actions and incorporate that information into their decision. Bi-level programming makes certain assumptions that are likely unrealistic in practice. For example, in its standard form it assumes that the attacker has complete knowledge of the system that is being attacked. However, by assuming the attacker has complete information the worst-case attacks can be bounded. This information can help cyber analysts focus on the most important attack and defense strategies.

The remainder of this paper is organized as follows. Section D.1 describes the four elements of the model in more detail. Section D.2 provides the details of the math model formulation and techniques used to solve the model. Section D.3 demonstrates the application of this model to a small attack graph.

D.1. Model Elements

This section describes the approach for modeling each of the four major elements of this model. A common theme throughout each of these sections is to make the model as realistic as possible for modeling and protecting cyber-physical systems.

D.1.1. Attack Graphs

For this model, we use attack graphs to represent the "game board" where the attacker and defender interact. Attack graphs were developed and first applied to cyber application in the 1990's and various methods have been developed for generating attack graphs [1,2,3]. In a cyber-attack graph, nodes represent different states of the system. The state of system can include information that describes whether an attacker has access to a component of the cyber system, their permission levels, and trust relationships that exist between components. The edges represent the actions taken by the attacker to move from one system state to another.

D.1.2. Defender Model

The defender model describes the decision space available to the defender for installing and configuring IDS. It consists of the defender's objective, the available decisions, and constraints on which combinations of decisions are allowable.

The defender's objective is to minimize the likelihood that an attacker can achieve a given consequence on the physical system without being detected. This is different that the objective functions employed in many interdiction models that typically focus on minimizing the worst consequence that at attacker can achieve. One advantage of focusing on the worst possible outcome does provide a bound on how much damage the attacker could cause to the physical system. The disadvantage of this approach is that the mitigation strategy for the worst-case scenario may not protect against other attack scenarios that cause less damage to the system. Furthermore, attackers may have specific objectives in mind that have a much smaller impact than the worst-case attack. Similarly, defenders may have a threshold for system failure that is less than the worst-case attack. The approach employed by this model allows the defender to define the threat and then determine the necessary steps to protect against it.

In this model, we consider two types of defender decisions for IDS. The first type is *edge interdiction* decisions. These are binary choices indicating whether an edge on the attack graph should be protected. Each edge is assumed to have a baseline probability of detection if no action is taken and an increased probability of detection if the defender chooses to protect the edge. The second type is *enabling capability* decisions. These are decisions that do not directly map to the attack graph but enable one or more *edge interdiction* decisions. These types of decisions provide for more flexibility when modeling the defender's decision space. For example, if an IDS system includes a fixed and variable cost component, an *enabling capability* decision can be used to represent the upfront acquisition cost and *edge interdiction* decision variables can be used to represent the cost of interdicting each edge.

The model includes three types of constraints on the defenders IDS decision. The first is a cost constraint that prevents the combined costs of all IDS decisions from exceeding the available budget. The second is a constraint on the false alarm rate of a given protection strategy. False alarms consume cyber analysts time, erode confidence in the system, and increase the likelihood that actual attacks are missed, therefore it is desirable to keep the rate of alarms to a manageable level. Finally, the model includes constraints that limit the burden that can be placed on users of the system due to IDS placement. Disallowing users from performing certain tasks or onerous authentication requirements make it more difficult for users to perform legitimate function, so a balance must be struck between security and usability.

D.1.3. Attacker Model

We use the threat group information in the MITRE ATT&CK™ database as the basis for representing the attacker behavior in this model [4]. ATT&CK contains a list of known cyber-threat groups. For each group, it identifies the techniques that the group has been observed to use. When executing an attack, the attacker is only able to use edges on the attack graph that are within their list of observed capabilities. The only other constraint we place on the attacker is that they must execute an attack that achieves at least the target level of disruption. The objective for the attacker is to select the attack with the lowest probability of detection.

By using this threat information, we aim to limit the assumptions and data requirements for the attacker. A common approach for modeling attackers in the interdiction literature is to assume that capabilities of the attacker are limited by a knapsack constraint. This requires establishing a "budget" on the attacker's ability to acquire new capabilities and a "cost" for each capability under consideration.

In practice, it is difficult to determine values for each of these terms. Given this, we favor the approach of using only the techniques that these groups have been observed using.

One potential criticism of our approach is that the observed capabilities of these groups may not represent their full set of capabilities. While this is likely the case, there are a few observations that mitigate some aspects of this concern. First, the data in ATT&CK is compiled from many cyber security groups around the world that monitor the various threat groups and report on the methods they use. Second, even though threat groups have the advantage of being able to operate remotely and in relative secrecy, they are still organizations that face many of the same challenges as any technical organization. Learning new capabilities takes time and resources, and organizational momentum can be difficult to change. Given this, it is more likely that organizations incrementally add or pivot to new capabilities instead of fundamentally changing the way they execute attacks. In future work, we can also consider an "$N + k$" extension to the model where attackers can acquire up to k additional capabilities.

D.1.4. Physical System Model

The final element of the model is a representation of consequence on the physical system that would result from a specific cyber-attack. Any attack that is selected by the attacker must meet or exceed the threshold value for the consequence. Given this, the model needs some method for mapping a given cyber-attack to damage on the system. For this model, we provide a generic structure for connecting attacks to consequences. For example, certain edges on the attack graphs can be assigned a numerical prize and the attacker can aim to collect enough of these prizes to meet the threshold value. Alternatively, the system model can be separate optimization problem, e.g. an D.C. optimal power flow (DCOPF) model, to calculate the consequence resulting from an attack.

D.2. Math Model

We formulate the math model as a bi-level interdiction problem. The first stage consists of the defender and the second stage consists of one or more attackers. Let $a \in A$ represent each attacker threat group. Given a specific cyber-physical system configuration and the capabilities of the different threat groups, we let G represent the resulting directed attack graph where N and E represent the nodes and edges, respectively. Each threat group a has an associated attack graph $G_a \subseteq G$ where N_a and E_a represent the nodes and edges that the group can include in their attacks based on their capabilities.

We assume that each of these attack graphs share a common root node s which represents the remote station where attacks originate. We also assume that there are a set of terminal edges $T \subset E$ that represent connections between the cyber and physical system. Once the attacker traverses these edges they can produce an effect on the physical system. The set $T_a = E_a \cap T$ represents the terminal nodes available to attacker a. Finally, we assume that there is at least one path between the root node s and each edge $t \in T_a$ for each attacker a.

For each edge $e \in E$ there is a baseline probability $r_e \in (0,1]$ that attacker would evade detection if they attacked that edge based on the existing security infrastructure. Let $D \subseteq E$ represent the edges that can be interdicted by installing an IDS. For each edge $e \in D$ there is a reduced probability of evasion for the attacker $q_e \in [0, r_e)$. We assume that these parameters depend only on the action (attack) being taken and do not depend on the specific attacker performing the action. Therefore, attackers who could use the same technique at the same step in an attack will share a common edge. Since ATT&CK does not provide data on how stealthy each group is at evading detection we do not

specify these inputs by attacker. If this data were available, parallel edges could be created to represent different skill levels for the same type of attack.

The defender also has a set of *enabling capabilities* C that they can choose to invest in. These capabilities do not directly impact the attack graphs but may be required before specific interdictions from the set D can be selected by the defender. Let P_c be the set of edges e that are enabled by capability c. The binary parameter I_C^P takes a value of zero when capability c just allows the associated edges $e \in P_c$ to be mitigated, and a value of one when selecting capability c forces the associated edges to be mitigated. This model assumes that each edge mitigation is associated with at most one *enabling capability*.

The defender must ensure that the investments they select do not exceed their available cost budget, false-positive rate threshold, and user impact threshold. Let $B = \{cost, false\text{-}positive, impact\}$ be the collection of the defender "budget" categories. For each category $b \in B$ there is an upper limit $m_b > 0$. For each category b and *enabling capability* C and edge interdiction $e \in D$ there is an associated cost $c_{bc}^C \geq 0$ and $c_{be}^E \geq 0$, respectively. For the cost category the units of these parameters are dollars. For false-positives it is number of events per some unit of time (e.g. false-positives per day). For the user impact, the units are more subjective measurements of how much a given mitigation would impact users (e.g. a 0 to 10 scale).

The defender sets a threshold v that represents the smallest disruption on the physical system that they would like to protect against. The units for this term depend on the system being modeled. For example, in the case of a power grid the units could be load not served. Given this, each attacker will attempt to find a valid subgraph of G_a. A sub-graph is valid if it starts at node s, is connected, and contains a subset of nodes $T_a' \subseteq T_a$ that generate a consequence of at least v on the physical system. Let $G_{av} \subseteq G_a$ represent the family of valid attack subgraphs for attacker a that meet or exceed the threshold disruption v.

Let $S \subseteq D$ represent the edges that the defender chooses to protect. The following function calculates the probability that attack $G \in G_{av}$ evades detection, given a collection of interdictions S. The probability of an attacker evading detection on different edges is assumed to be independent.

$$h_G(S) := \left(\prod_{e \in G} r_e\right)\left(\prod_{e \in G \cap S} \frac{q_e}{r_e}\right) \tag{1}$$

Let x_e^E and x_c^C be binary variables that represent the edge and capability interdictions selected by the defender, respectively. Given this, the probability of evasion, in terms of these binary variables, is defined as follows.

$$\bar{h}_G(x^E) := \left[\prod_{e \in G} r_e\right]\left[\prod_{e \in G \cap D} \left(\frac{q_e}{r_e}\right)^{x_e^E}\right] \tag{2}$$

The sensor placement math model is given below. The defender's objective function is given by (3). The aim is to minimize the probability that a collection of attackers will successfully execute an attack that achieves a consequence of at least v. The defender can select a combination of *enabling capabilities* and edge interdictions subject to the three "budgets" under consideration. The budget constraints are modeled using a knapsack constraint (4). Constraints (5) and (6) connect the *enabling capabilities* to the

edge interdictions. The first constraint is used when an *enabling capability* allows an edge to be mitigated, whereas the second constraint forces the edge to be mitigated. Once an interdiction strategy has been realized, each attacker then finds the best available attack G_a the maximizes their chances of evading detection.

$$min_{x^E,x^C} \quad max\{\bar{h}_G(x^E) : a \in A, G_A \in \mathcal{G}_{av}\} \tag{3}$$

$$s.t. \quad \sum_{c\in C} c^C_{bc}x^C_c + \sum_{e\in D} c^E_{be}x^E_e \leq m_b \quad \forall b \in B \tag{4}$$

$$x^C_c \geq x^E_e \quad \forall c \in C, e \in P_c : I^P_c = 0 \tag{5}$$

$$x^C_c = x^E_e \quad \forall c \in C, e \in P_c : I^P_c = 1 \tag{6}$$

$$x^E, x^C \in \{0,1\} \tag{7}$$

D.2.1. Model Reformulation

The model in described by Expressions (3) through (7) is not conducive to being solved in the form presented. In this section, a mixed-integer linear programming formulation of the model and decomposition strategy is derived. Let π represent the probability that an attacker evades detection. Given this, the defender's decision model can be described as follows.

$$min_{x^E,x^C,\pi} \quad \pi \tag{8}$$

$$s.t.$$

$$\sum_{c\in C} c^C_{bc}x^C_c + \sum_{e\in D} c^E_{be}x^E_e \leq m_b \quad \forall b \in B \tag{9}$$

$$x^C_c \geq x^E_e \quad \forall c \in C, e \in P_c : I^P_c = 0 \tag{10}$$

$$x^C_c = x^E_e \quad \forall c \in C, e \in P_c : I^P_c = 1 \tag{11}$$

$$\pi \geq \bar{h}_G(x^E) \quad \forall a \in A, G \in \mathcal{G}_{av} \tag{12}$$

$$x^E, x^C \in \{0,1\}, \pi \in \mathbb{R} \tag{13}$$

Towle and Luedtke have shown that (1) and equivalently (2) are supermodular set functions and that the feasible region for this model can alternatively be defined using the set HGa, which is defined below (see Proposition 2 [5]). The set HGa has been studied by several researches and can be described by linear equalities [6,7]. This creates the potential to solve this model using a delayed constraint generation strategy. The idea of this approach is to generate an optimal interdiction strategy x^{E^*} and then solve an auxiliary problem to see if a new attack Ga can be found that violates Constraint (12) for any of the threat groups. If such an attack can be found, new linear inequalities can be generated and added to the master problem. Since Constraint (12) decomposes by threat group the auxiliary problem for each of the threat groups can be solved in parallel.

$$H_{G_a} := \left\{(x^E, \pi) \in \{0,1\}^{|D|} \times \mathbb{R} : \pi \geq \bar{h}_{G_a}(x^E)\right\} \tag{14}$$

To leverage this, we develop the auxiliary problem as follows. Assume that there is a fixed interdiction strategy S_F. For a given threat group a, let y_e be a binary variable that indicates that edge e is part of attack graph G_a. Since a separate auxiliary problem can be written for each threat group, we drop the index a from the notation. Let z_e represent the probability of evasion on edge e for the given interdiction strategy S_F, as shown below.

$$z_e = \begin{cases} q_e, & e \in S_F \\ r_e, & otherwise \end{cases} \tag{15}$$

The attacker's objective is to find an attack G that maximizes the probability of evasion, as shown by the objective below.

$$max \prod_{e \in G} z_a \tag{16}$$

This objective can be converted to a linear expression by taking the logarithm of the z_a parameter, as shown below.

$$max \sum_{e \in E_a} log(z_e) y_e \tag{17}$$

The following constraints ensure that the attack graph that is generated meets the requirements described previously. For this model, assume that each edge $t \in T$ has a numerical "prize" p_e attached to it. Constraint (18) is a covering constraint that requires that the attack graph that is selected by the attacker must be able to generate at least the minimum reward v. Constraint (19) enforces precedence relationships between attacks across the network. It states that an attack on an edge e defined by nodes (i,k) is only allowed if one of edges that immediately proceeds edges e (i.e. an edge (j,i)) has been attacked. Note: This formulation assumes the graph is directed and acyclic.

$$s.t.$$

$$\sum_{e \in T_a} p_e y_e \geq v \tag{18}$$

$$\sum_{j \in N_a : (j,i) \in E_a} y_{(j,i)} \geq y_{(i,k)} \quad \forall (i,k) \in E_a : i \in N_a / s \tag{19}$$

$$y_e \in \{0,1\} \quad \forall e \in E_a \tag{20}$$

If $z_a = 0$ (i.e. the interdiction detects attacks with probability 1) the logarithm will be undefined. In these instances, any edges that meet this condition can be removed from the network since the attacker

should not want to pursue an attack that has no chance of succeeding. If all the sub-problems for each threat group are not feasible without these edges, then the current interdiction strategy S is optimal.

Constraint (18) represents one strategy for ensuring that the generated attack is valid. One alternative approach is to use disjunctions to define a collection of subsets of T_a that would meet the threshold v, if disrupted. Let T^v represent the set of sets of edges $e \in T_a$ that achieve the desired disruption. Let $j = 1, \ldots, |T^v|$ be an index on this set and Y_j be an indicator that the set $T_j^v \in T^v$ is active. Given this, the constraint below can be used in place of (18) to ensure that a valid combination of edges is attacked. This constraint does not force edges that are not part of a set T_j^v to be equal to zero, though this can be added if needed.

$$\bigvee_{j \in \{1,\ldots,|T^v|\}} \begin{bmatrix} Y_j \\ y_e = 1 \quad \forall e \in T_j^v \end{bmatrix} \tag{21}$$

For this initial version of the model, cuts of the form shown in Constraint (22) (from Nemhauser and Wolsley [6]) are used. The function $\rho_G^a(S)$ represents the marginal difference in h_G obtained by adding a to S and is calculated as follows: $\rho_G^a(S) \coloneqq h_G(S \cap \{a\}) - h_G(S)$. Researchers have noted that these cuts tend to be very weak and present methods for deriving much stronger cuts [5,7]. These improved cuts can be used in a future version of the algorithm.

$$\pi \geq h_G(S) - \sum_{a \in S} \rho_G^a(D/\{a\})(1 - x_a) + \sum_{a \in D/S} \rho_G^a(S) x_a \tag{22}$$

D.3. Example

To demonstrate this model, consider the attack graph shown in Figure D 1. This graph contains 30 edges, 25 of which can be interdicted. The remaining 5 edges are *reward edges*. Table D 1 summarizes the input for each edge. In this example, assume that the defender is concerned about attacks that would produce a reward of at least 3 units for the attacker.

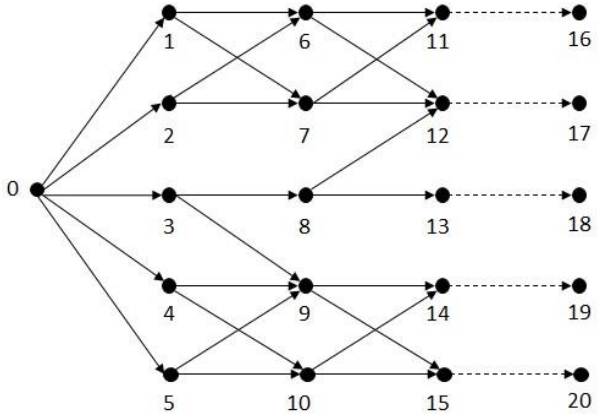


Figure D 1. Test case attack graph. Solid lines indicate attack edges. Dashed lines indicate reward edges.

Table D 1. Input data for example network.

From Node	To Node	Interdiction Cost	Reward	Baseline Evasion Probability	Interdicted Evasion Probability
0	1	1.41		0.81	0.42
0	2	1.10		0.86	0.49
0	3	1.42		0.90	0.51
0	4	1.38		0.89	0.45
0	5	1.18		0.91	0.39
1	6	1.21		0.87	0.51
1	7	1.02		0.84	0.35
2	6	1.09		0.85	0.40
2	7	1.11		0.92	0.45
3	8	1.45		0.88	0.52
3	9	1.12		0.82	0.35
4	9	1.06		0.86	0.49
4	10	1.31		0.83	0.43
5	9	1.16		0.81	0.43
5	10	1.19		0.81	0.40
6	11	1.29		0.84	0.48
6	12	1.33		0.94	0.53
7	11	1.04		0.94	0.38
7	12	1.35		0.84	0.41
8	12	1.45		0.85	0.42
8	13	1.43		0.85	0.49
9	14	1.23		0.92	0.52
9	15	1.43		0.94	0.41
10	14	1.46		0.84	0.44
10	15	1.08		0.84	0.35
11	16		2.74	1	1
12	17		1.42	1	1
13	18		1.21	1	1
14	19		1.20	1	1
15	20		1.49	1	1

The baseline detection probably can be found by solving the attacker sub-problem with the baseline detection probabilities. With no interdictions the attacker has a 62.9 percent chance of successfully

executing the optimal attack without being detected. Figure D 2 shows the optimal attack in this scenario. The edges selected by the attacker are shown in bold. This attack includes edge (11,16), which is the most valuable *reward edge*, and the nearby edge (12,17). Note that there are no combinations of two *reward edges* that produce a reward of at least 3 and do not include edge (11,16). If this edge can be effectively defended, the attacker will have to attack three of the other edges to achieve a reward of 3.

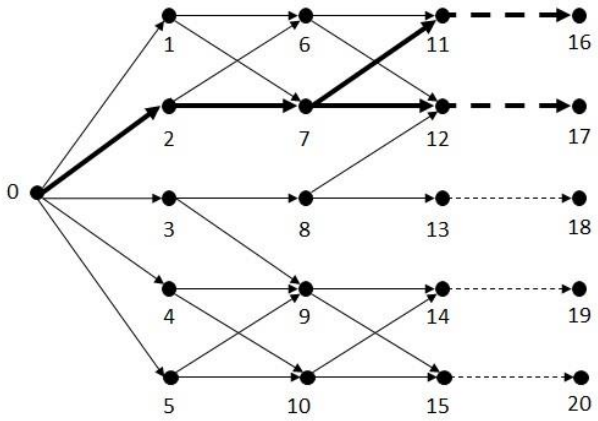


Figure D 2. Optimal attack with no interdictions ($\pi_{OPT} = 0.629$).

Next, consider the case where the defender has a budget of 12. For context, the defender would require a budget of 32 to protect all the edges in the network. In this case, the attacker has only a 14.0 percent chance of successfully executing the optimal attack. Figure D 3 shows the solution in this case. The interdicted edges are shown in green and the edges that are attacked are in bold. Observe that in this case edge (11,16) is no longer attacked. Furthermore, the selected interdictions appear to emphasize protecting this edge since all paths to node 16 include two interdicted edges. This has the effect of forcing the attacker to attack three edges to obtain the minimum reward.

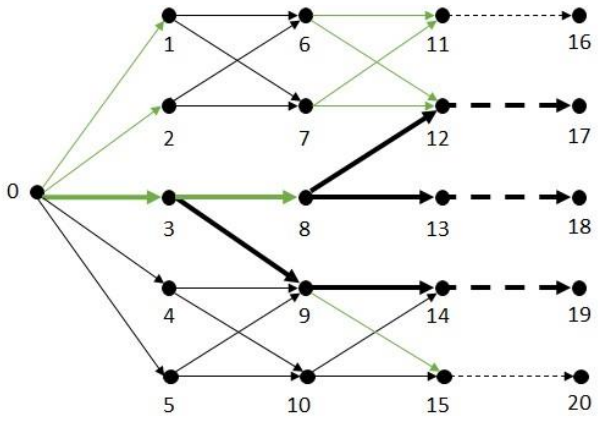


Figure D 3. Optimal interdiction strategy and attack for a defender budget $b = 12$ ($\pi_{OPT} = 0.140$). Interdicted edges are shown in green. Attacked edges are bold.

Finally, consider the tradeoff between the level of investment made by the defender and the probably that the attacker can evade detection. This was achieved by solving the model for all budgets between 0 and 32. Figure D 4 shows the probability that the attacker successfully evades detection as a function of the available defender budget. Once the defender budget reaches 22, no additional reduction in the attacker's probability is possible. This type of result can help decision makes understand the tradeoffs between cost and risk.

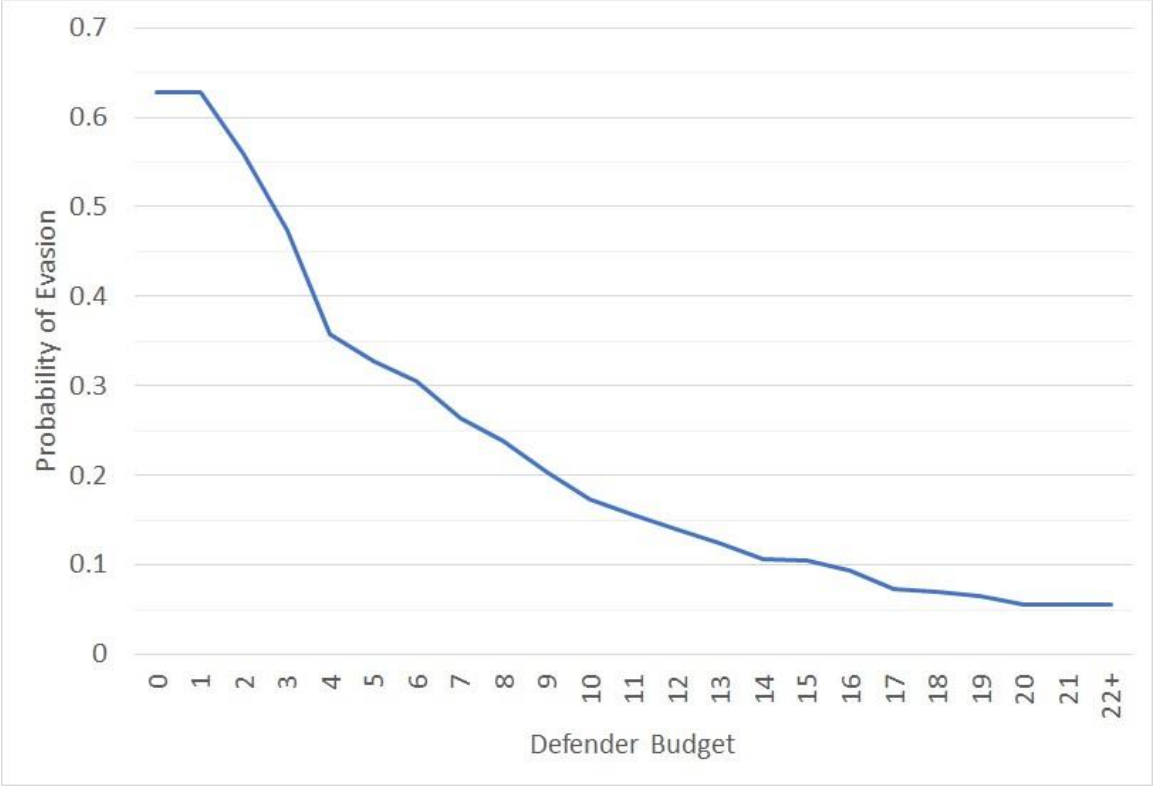


Figure D 4. Optimal probability of evasion versus defender budget.

D.4. Appendix D References

1. Steven Noel. A Review of Graph Approaches to Network Security Analytics. In: Samarati P., Ray I., Ray I. (eds) From Database to Cyber Security. Lecture Notes in Computer Science, vol 1117. Springer, 2018.
2. Cynthia Phillips and Laura Swiler. A graph-based system for network-vulnerability analysis. In New Security Paradigms Workshop, Charlottesville, VA, pages 71–79, 1998.
3. Oleg Sheyner and Jeannette Wing. Tools for generating and analyzing attack graphs. In Formal Methods for Components and Objects, volume 3188, pages 344–371. Springer, 2003.
4. MITRE ATT&CK. https://attack.mitre.org/
5. Towle, E., Luedtke, J. New solution approaches for the maximum-reliability stochastic network interdiction problem. Comput Manag Sci 15, 455–477 (2018). https://doi.org/10.1007/s10287-018-0321-1.
6. George L. Nemhauser and Laurence A. Wolsey. An analysis of approximations for maximizing submodular set function - I. Math Programming, 14(1):265–294, 1978.

7. Shabbir Ahmed and Alper Atamturk. Maximizing a class of submodular utility functions. Math Programming, 128(1-2):149–169, 2011.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Scott Collis	1400	sscoll@sandia.gov
Cynthia Phillips	1400	caphill@sandia.gov
John Feddema	1460	jtfedde@sandia.gov
Michael Eldred	1463	mseldre@sandia.gov
Gianluca Geraci	1463	ggeraci@sandia.gov
Laura Swiler	1463	lpswile@sandia.gov
Daniel Turner	1463	dzturne@sandia.gov
William Hart	1464	wehart@sandia.gov
Bethany Nicholson	1464	blnicho@sandia.gov
John Siirola	1464	jdsiiro@sandia.gov
Leigh Cunningham	1910	lcunning@sandia.gov
Bill Miller	5000	millerwm@sandia.gov
Heidi Ammerlahn	5100	hrammer@sandia.gov
Gerardo Cruz	5446	gcruz@sandia.gov
David White	5600	drwhite@sandia.gov
Jennifer Depoy	5620	jdepoy@sandia.gov
Casey Glatter	5621	cglatte@sandia.gov
Derek Hart	5621	derhart@sandia.gov
Meghan Sahakian	5621	mgaliar@sandia.gov
Jamie Thorpe	5621	Jthorpe@sandia.gov
Eric Vugrin	5621	edvugri@sandia.gov
Jason Stamp	5623	jestamp@sandia.gov
Dahlon Chu	5680	ddchu@sandia.gov
Evan Eric De La O	5682	eedelao@sandia.gov
Steven Elliott	5682	selliot@sandia.gov
Seth Hanson	5682	derhart@sandia.gov
Nicholas Pattengale	5682	ndpatte@sandia.gov
Christian Reedy	5682	creedy@sandia.gov
Michael Stickland	5682	mgstick@sandia.gov
Thomas Tarman	5682	tdtarma@sandia.gov
Jorge Urrea	5682	jmurrea@sandia.gov

Name	Org.	Sandia Email Address
Todd Jones	5689	stjones@sandia.gov
Kasimir Gabert	5689	kggaber@sandia.gov
Erin Acquesta	5954	eacques@sandia.gov
Andrew McIlroy	8000	amcilr@sandia.gov
Habib Najm	8300	hnnajm@sandia.gov
Bert Debusschere	8351	bjdebus@sandia.gov
Jennifer Gaudioso	8700	jmgaudi@sandia.gov
Phillip Kegelmeyer	8700	wpk@sandia.gov
Berlinda Eras	8710	bbaca@sandia.gov
Dean Jones	8720	dajones@sandia.gov
Bryan Arguello	8721	barguel@sandia.gov
Sean DeRosa	8721	sderosa@sandia.gov
Jared Gearhart	8721	jlgearh@sandia.dov
Emma Johnson	8721	esjohn@sandia.gov
Alexander Outkin	8724	avoutki@sandia.gov
Susanna Gordon	8740	spgordo@sandia.gov
Patty Hough	8754	pdhough@sandia.gov
Cindy Veitch	8760	ckveitc@sandia.gov
Jonathan Crussell	8762	jcrusse@sandia.gov
Matthew Farrell	8762	mtfarre@sandia.gov
Ali Pinar	8762	apinar@sandia.gov
Gayle Thayer	8762	gthayer@sandia.gov
Kevin Hulin	8765	kjhulin@sandia.gov
Justin Doak	8765	jedoak@sandia.gov
Glory Avina	8766	gremman@sandia.gov
Chris Symomnds	8766	cjsymon@sandia.gov
Charles J. Hanley	8810	cjhanle@sandia.gov
Raymond H. Byrne	8813	rhbyrne@sandia.gov
Richard Griffith	8850	rogrif@sandia.gov
Lon Dawson	8851	ladawso@sandia.gov
Robert Leland	8900	leland@sandia.gov
John Zepper	9000	jdzeppe@sandia.gov
Marcus Chang	9300	mchang@sandia.gov
John Naegle	9300	jhnaegl@sandia.gov

Name	Org.	Sandia Email Address
Zachary Benz	9310	zobenz@sandia.gov
Brian Gaines	9366	bgaines@sandia.gov
Vince Urias	9373	veuria@sandia.gov
Brian Van Leeuwen	9373	bpvanle@sandia.gov
Scott Stephens	9750	sasteph@sandia.gov
Technical Library	01977	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.