SANDIA REPORT
SAND2021-11609
Printed September 2021

Sandia National Laboratories

Design Considerations for Distributed Energy Resource Honeypots and Canaries

Jay Johnson, Louis Jencka, Timothy Ortiz, C. Birk Jones, Adrian Chavez, Brian Wright, Adam Summers

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

There are now over 2.5 million Distributed Energy Resource (DER) installations connected to the U.S. power system. These installations represent a major portion of American electricity critical infrastructure and a cyberattack on these assets in aggregate would significantly affect grid operations. Virtualized Operational Technology (OT) equipment has been shown to provide practitioners with situational awareness and better understanding of adversary tactics, techniques, and procedures (TTPs). Deploying synthetic DER devices as honeypots and canaries would open new avenues of operational defense, threat intelligence gathering, and empower DER owners and operators with new cyber-defense mechanisms against the growing intensity and sophistication of cyberattacks on OT systems. Well-designed DER canary field deployments would deceive adversaries and provide early-warning notifications of adversary presence and malicious activities on OT networks. In this report, we present progress to design a high-fidelity DER honeypot/canary prototype in a late-start Laboratory Directed Research and Development (LDRD) project.

ACKNOWLEDGEMENTS

The team would like to thank Vince Urias, Abraham Clements, and William Stout for their valuable contributions to the project.

CONTENTS

1. Introduction 8
2. Prior Work 10
3. DER Canary and Honeypot Design 11
   - 3.1. DER Simulator Software 12
     - 3.1.1. DER Clock 12
     - 3.1.2. DC Simulation 13
     - 3.1.3. AC Simulation 14
     - 3.1.4. IEEE 1547-Compliant Interfaces 16
     - 3.1.5. Non-standardized Communication Interfaces 16
     - 3.1.6. Monitoring and Test Environment 17
   - 3.2. Intrusion Detection and Alerting 19
     - 3.2.1. Behavior-based Intrusion Detection 22
   - 3.3. Honeypot and Honeynet Networking 24
   - 3.4. Hosting Challenges 24
   - 3.5. Threat Sharing 24
4. DER Honeypot Demonstrations 26
   - 4.1. Stand-Alone DER Honeypot Implementation 26
   - 4.2. Cloning an Energy Storage System for a DER Canary Application 29
5. Alternative DER Simulator Applications 34
   - 5.1. Training Tools for Cyber Defenders 34
   - 5.2. Real-Time Grid Integration Studies 34
6. Conclusion 35

Appendix A. Example DER Honeypot Output 36

Appendix B. SunSpec Model Definitions 37
   - B.1. SunSpec Model 30003 37
   - B.2. SunSpec Model 30004 38

Appendix C. CYBER Monitoring Software Installation 41
   - C.1. Zeek Installation 41
   - C.2. Snort Installation 41
   - C.3. Filebeat Installation 41
   - C.4. Logstash Installation 41
   - C.5. Logstash Installation 42
   - C.6. Grafana Installation 43

LIST OF FIGURES

Figure 1: Different deployment options for the DER Simulator. 11
Figure 2: Design of the DER simulation with inputs and communication interfaces. 12
Figure 3: Pre-recorded 1-second solar irradiance profile. 13
Figure 4: Photovoltaic current-voltage (I-V) curves with changing irradiance and temperature. 14
Figure 5: AC grid inputs. 14
Figure 6: DER measurement visualization dashboard. 17
Figure 7: Live dashboard graphs displaying the PQ plane (top left), DC I-V curve (top right), and the volt-var curve configuration (bottom) 18

Figure 8: Interface for live configuration of the DER Simulator. 18
Figure 9: An example deployment of a bump-in-the-wire IDS monitoring and alerting system in a DER environment. 20
Figure 10: The process in which alerts are identified and presented to system administrators when anomalies or attacks are detected....20
Figure 11: A Grafana dashboard displaying network activity along with system alerts stored within the Elasticsearch database. 21
Figure 12: Data features used by the Adaptive Resonance Theory algorithm to detect network anomalies. 22
Figure 13: Flow diagram for the online learning process. 23
Figure 14: ART results for (a) normal, (b) adversary reconnaissance, and (c) a denial-of-service attack. No false positives occurred when subjected to the normal data and each of the cyber-attacks were detected. 24
Figure 15: Honeypot Common Model data read using the SunSpec SVP Dashboard. 27
Figure 16: Honeypot DER Capacity Model data. 27
Figure 17: Multiple reads of the Honeypot DER AC Measurement Model. 28
Figure 18: Honeypot DER AC Controls Model showing the PF function is set to 0.90 with an underexcited excitation. 28
Figure 19: Raspberry Pi 3B+ and 4B computers configured with DER Simulator software. 29
Figure 20: DER ESS (left) vs DER Honeypot (right) for SunSpec Model 702 *DER Capacity*. 30
Figure 21: DER ESS (left) vs DER Honeypot (right) for SunSpec Model 701 *DER AC Measurement*. 31
Figure 22: DER ESS (left) vs DER Honeypot (right) for SunSpec Model 703 *DER DC Measurement*. 31
Figure 23: DER ESS (left) vs DER Honeypot (right) for SunSpec Model 704 *DER AC Controls*. 32
Figure 24: DER ESS (left) vs DER Honeypot (right) for SunSpec Model 705 *DER Volt-Var*. 33
Figure 25: Model 30003 in the SunSpec SVP Dashboard. 38
Figure 26: Model 30004 in the SunSpec SVP Dashboard. 40

ACRONYMS AND DEFINITIONS

Acronym	Definition
ACL	Access Control List
ANSI	American National Standards Institute
API	Application Programming Interface
CIP	Critical Infrastructure Protection
DER	Distributed Energy Resource
DERMS	Distributed Energy Resource Management System
ELG	Elasticsearch, Logstash, and Grafana
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IEC	International Electrotechnical Committee
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
NERC	North American Electric Reliability Corporation
NIST	National Institute of Standards and Technology
OEM	Original Equipment Manufacturer
OT	Operational Technology
PKI	Public Key Cryptography
PV	Photovoltaic
RTO	Regional Transmission Organization
SDO	Standards Development Organization
SP	Special Publication (from NIST)
TLS	Transport Layer Security
TSO	Transmission System Operator

1. INTRODUCTION

As the US transitions to more distributed, clean, interoperable power systems, it is critical to effectively defend these generation sources, loads, and storage devices—and associated networks—from intentional threats. Modern DER devices include one or more communication interfaces that provide a wide range of control and monitoring points to DER vendors, grid operators, aggregators, and 3rd parties. These DER interoperable capabilities give grid operators new visibility into power system operations and control capabilities to better optimize the system. Unfortunately, DER communications often run over the public internet, except in the cases of large installations where dedicated utility connections are required or cost-effective. Internet-connected DER equipment potentially represents a new attack vector for the power system and needs to be protected against compromise.

In fact, in March 2019, the first disruptive cybersecurity incident on record in the U.S. power industry was experienced by a renewables developer; sPower lost visibility into 500 MW of wind and solar assets due to a Denial of Service (DoS) attack on an unpatched Cisco firewall[1]. Distributed Energy Resources (DER), Inverter-Based Resources (IBRs), and vendor/aggregator cloud infrastructures are also vulnerable to attacks[2,3].

Deception technologies offer multiple cybersecurity defense functionalities to capture adversary tactics and techniques to expand our understanding of the threat landscape and DER vulnerabilities. At a practical level, virtualized DER devices can be configured to provide (a) protection by directing adversary focus away from critical assets and (b) detection by sending alerts when the adversary interacts with the artificial equipment. Once the underlying cyber-physical DER emulation capabilities are created, tools of deception can be stood up and replicated rapidly—or collocated on operator cloud networks and fielded systems. Virtual DER can be deployed in:

- Honeypots – internet-connected applications to capture adversary actions. When there is a collection of virtualized devices with internal networking it is sometimes called a honeynet.
- Canaries – virtualized device deployed alongside real DER equipment on an OT network to alert operators to adversary presence (i.e., fake devices in real systems).

There is currently an absence of well-developed cyber-physical deception elements and virtualization technologies for OT systems. This 6-month LDRD was designed to help defend US critical infrastructure by deceiving cyber-adversaries with realistic internet-connected DER environments. The team investigated what was needed to create a credible DER deception, and what intrusion data could be collected when deploying these technologies as honeypots and canaries on fielded systems. The team created a Python-based cyber-physical DER Simulator that included a SunSpec Modbus communication interface which exposed data points for environmental, power system, and power electronics behaviors that mirrored physical DER equipment. The DER simulator along with a network monitoring tool built from the Snort and Zeek Intrusion Detection Systems (IDS) coupled with an Elasticsearch, Logstash, and Grafana (ELG) stack for alerting was deployed on multiple Raspberry Pi systems. These small, modular compute systems could be easily installed on utility OT systems as canaries or connected to the internet to track adversary behaviors in the future.

---

[1] S. Lyngaas, "Utah renewables company was hit by rare cyberattack in March," CYBERSCOOP, Oct 31, 2019. URL: https://www.cyberscoop.com/spower-power-grid-cyberattack-foia/

[2] W. Westerhof, "Horus Scenario: Exploiting a weak spot in the power grid, URL: https://horusscenario.com/

[3] F. Bret-Mounet, "All your solar panels are belong to me", DEF CON 24, 2016.

This project determined that it was difficult to fully disguise and deploy authentic DER device technologies. State-of-the-art device emulation leaves artifacts that can be detected with patience and thorough analysis. To fool highly-sophisticated adversaries further research is needed on the significance and observability of such virtualization artifacts to increase realism in the communication systems, power conversion emulation, and unique environmental factors that dictate DER device behaviors.

2. PRIOR WORK

Currently, there are R&D and commercial honeypot technologies, including Sandia's R&D 100 award-winning High-Fidelity Adaptive Deception & Emulation System (HADES)[4] and canaries from companies like Thinkst[5] and Fortinet[6], but these focus on information technology (IT) environments containing corporate firewalls, computers, and networks. There are fewer deception technologies deployed on OT networks and none that represent DER networks and devices, despite the advantages these could provide network defenders. As an example, a 2014 SANS institute report covering a German Steel Mill hack recommended adding canaries to the system to help detect network presence[7].

In the last couple decades, some research groups and companies have started exploring the efficacy of OT canaries. Cisco created the SCADA HoneyNet that replayed to Nmap and Xprobe requests with pre-configured packet responses based on configuration files[8]. Attivo Networks' BOTsink[9] technology is designed to generate Windows and Linux devices that support ICS/SCADA protocols like DNP3, Modbus, and IEC 61850. Attivo is currently researching honeypots for distribution substations and other SCADA systems with PNNL[10], though it is unclear how well their physics engine(s) function or if they consider the risk to aggregates of cloud-connected equipment. CONPOT[11] is an ICS/SCADA honeypot, built on the Honeynet Project[12] source code, which provides a server-side environment a full protocol stacks of some ICS protocols. The problem with most of these tools is they do not represent the communications interfaces from real devices or respond to input variables or control setpoints as real equipment would.

What makes our research unique is focusing specifically on DER power devices and closely matching cyber-physical behaviors. In this work, we created a physical simulation of the DER that also includes a communication interface that dynamically changes based on a power electronics simulation and incorporates inputs from users. To our knowledge, a DER honeypot of this detail has never been created.

[4] HADES: High-Fidelity Adaptive Deception & Emulation System, Sandia White paper, SAND2017-3364 M. URL: https://ip.sandia.gov/techpdfs/HADES.pdf
[5] Thinkist Canary, URL: https://canary.tools/
[6] "FortiDeceptor Enables a New Breach Protection Approach," Fortinet White Paper, 2019.
[7] RM Lee, MJ Assante, T Conway, "German steel mill cyber attack," Industrial Control Systems SANS Report, 2014.
[8] Pothamsetty, Venkat and Matthew Franz. n.d. SCADA HoneyNet Project: Building Honeypots for Industrial Networks. URL: http://scadahoneynet.sourceforge.net/.
[9] K. Hiltpold, Threat Deception for SCADA Environments, Attivo Networks Confidential presentation, 2021.
[10] T. W. Edgar, W. Hofer, M. Feghali, Model Driven Deception for Defense of Operational Technology Environments, PNNL-30387.
[11] L. Rist, et al. 2018. Conpot: ICS/SCADA Honeypot. URL: http://conpot.org
[12] The Honeynet Project, URL: https://www.honeynet.org/

3. DER CANARY AND HONEYPOT DESIGN

There are several difficult design choices involved when creating DER canaries and honeypots. Some considerations include how to create the DER simulation, what monitoring and alerting tools to deploy on the host machine or as bump-in-the-wire devices, how to make the local network look "lived-in", and how to host these systems. In many situations it may be more convenient to build the DER Simulator and monitoring software on virtual machines (VMs) because they can be easily spun up based on the needs of the OT operator or researcher. This also makes the virtual environment highly-scalable when using a VM management software such as Sandia's Minimega tool.[13] But in other cases, especially for canary applications, it may be preferred to have a stand-alone device that can be turned over to network operators to install on the OT environment. Some example deployment scenarios are included in Figure 1. In each of these situations, the deployment strategy will need to be considered carefully by the team implementing the solution.

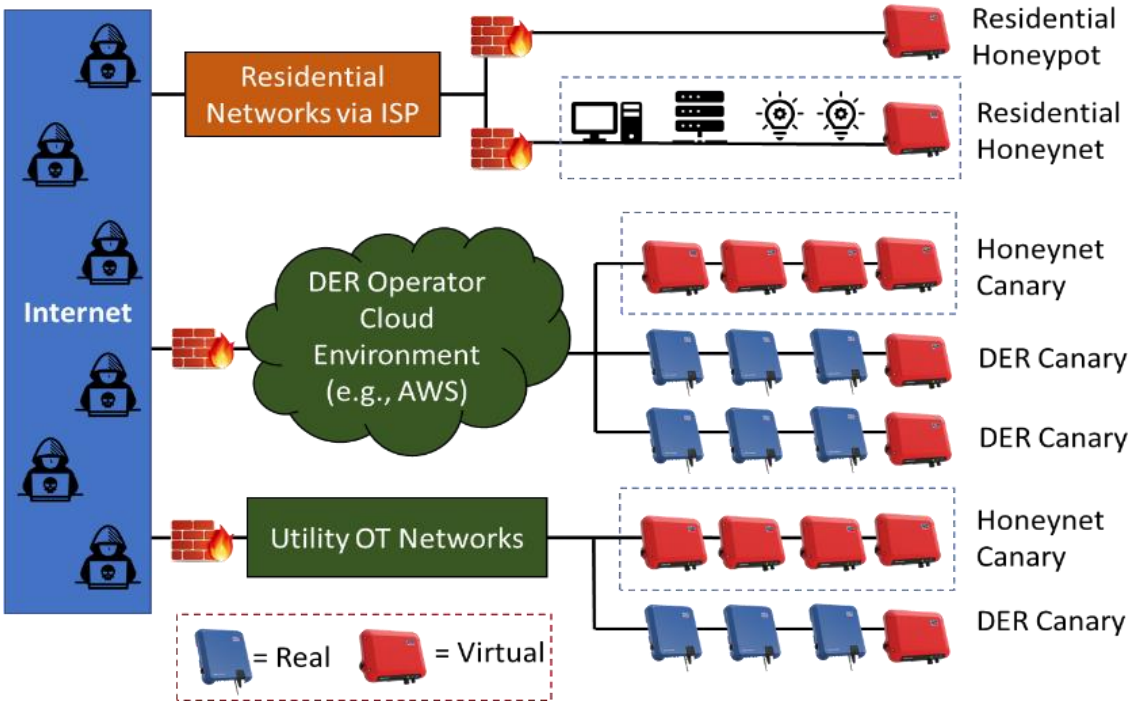

Figure 1: Different deployment options for the DER Simulator.

This team ultimately decided to create a Python-based DER Simulator that leveraged pySunSpec2 and other Python packages. The DER Simulator was deployed with Snort and Zeek IDSs along with an ELG stack for alerting on multiple low-power, single-board Raspberry Pi computers. These self-contained computers could be deployed as honeypots or handed off to partner organizations to create canaries. Creating equivalent Linux-based VMs would require little additional effort if the honeypot/honeynet needed to be deployed in the cloud or at a massive scale. Specific components of the DER Honeypot design are detailed in the following sections.

[13] Minimega, URL: https://minimega.org/

3.1. DER Simulator Software

In collaboration with the SunSpec Alliance, the team created a Python-based DER Simulator that represented DC inputs, power electronics, and communications that are nearly indistinguishable from physical equipment. An asynchronous I/O python package, asyncio[14], was used to execute concurrent code representing the DC and AC simulations, Modbus server, and other DER Simulator subprocesses.

As shown in Figure 2, the primary elements of this DER Simulator design were:

(a) A clock object to synchronize the asynchronous DER Python code with local clock time.
(b) An AC power electronics simulation that accounted for efficiencies of the equipment and grid-support functions.
(c) One or more DC power simulations that represented the photovoltaic system and maximum power point tracker.
(d) A SunSpec Modbus server with the 700-Series models that represent novel DER interoperability interfaces and protocol information models.
(e) DER device fingerprinting that allows the device to spoof operating system and additional networking interfaces.
(f) Network-based intrusion detection system that logged network traffic and generated alerts based on a set of rules.

Each of these simulator elements are described in the subsections below.

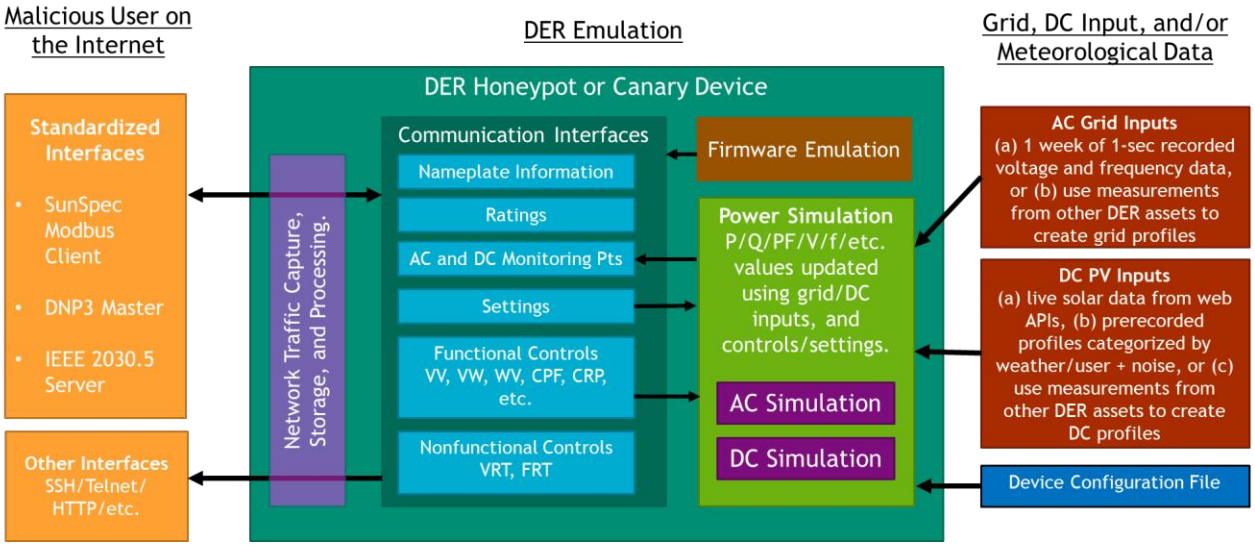

Figure 2: Design of the DER simulation with inputs and communication interfaces.

3.1.1. DER Clock

The clock object was designed to synchronize data from the AC and DC-sides of the power electronics device and either synchronize to the local operating system time or run the simulations from a fixed time for debugging purposes. The DER honeypot needed the ability to synchronized with the local time to:

[14] asyncio - Asynchronous I/O, URL: https://docs.python.org/3/library/asyncio.html, accessed Sept 13, 2021.

- track solar irradiance profiles for the DER equipment, and
- accurately reflect startup and shutdown times for the virtualized photovoltaic equipment.

As an example, if the DER was on and reporting full power production at midnight, this would expose the PV system honeypot. To solve that issue, the AC and DC simulators used the DER clock as the time reference for profile data used in the simulation.

3.1.2. DC Simulation

At this point, only photovoltaic DC simulation capabilities were constructed, but the simulation tool could be expanded to energy storage systems, synchronous generators, electric vehicle charges, and other DER/IBR equipment in the future with the addition of battery, EV, or other DC models.

The DC simulation was constructed with a maximum power level, maximum power point voltage (V_{MPP}), and user-selected number of DC ports. Each of the PV DC ports took these inputs and generated EN 50530[15] current-voltage (I-V) curves. The temporal I-V curves were updated based on the irradiance and temperature using one of three techniques:

1. using a repeating week-long irradiance/temperature profile with 1-second resolution,
2. using a predictive model based on NOAA cloud cover forecasts and the SNL-developed PV modelling library, pvlib[16,17], or
3. communicating to another physical DER on the same network and replicating its DC power.

An example of two days of pre-recorded solar irradiance with 1-second resolution, captured at DETL, is shown in Figure 3. The I-V curve was generated using irradiance and temperature, as shown in Figure 4. The curve is created with a user-selected number of points and maximum voltage. For these simulations, 1000 points were used for the I-V curve and the maximum voltage was set to open circuit voltage (V_{OC}) for the greatest irradiance and coldest temperatures expected for the simulation.

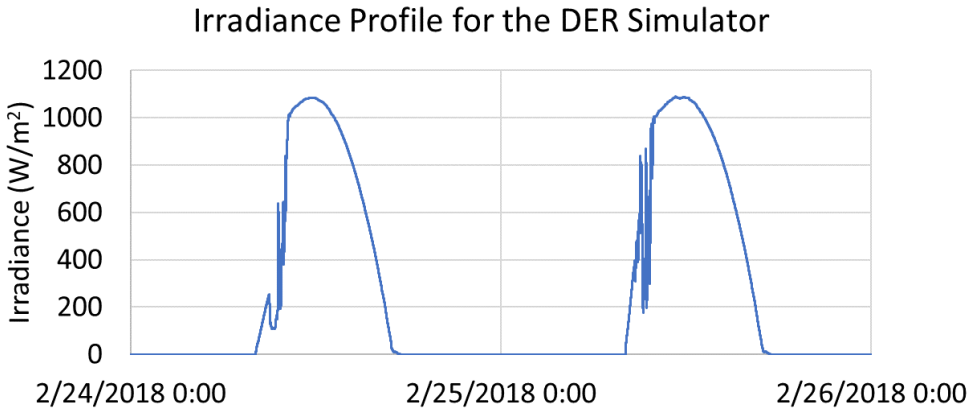

Figure 3: Pre-recorded 1-second solar irradiance profile.

[15] CENELEC - EN 50530, "Overall efficiency of grid connected photovoltaic inverters," 2010.
[16] W.F. Holmgren, C.W. Hansen, M.A. Mikofski. "pvlib python: a python package for modeling solar energy systems." Journal of Open Source Software, 3(29), 884, (2018). https://doi.org/10.21105/joss.00884
[17] pvlib python, URL: https://pvlib-python.readthedocs.io/en/stable/

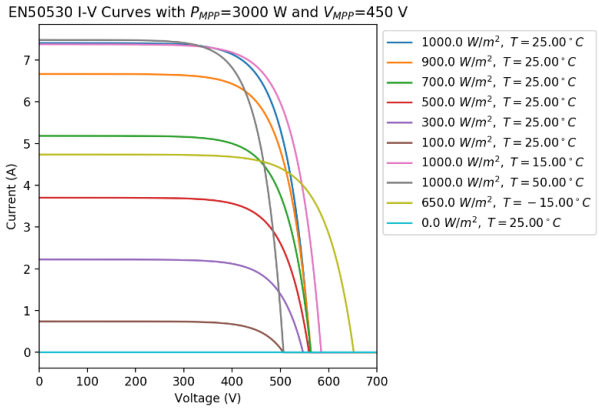

Figure 4: Photovoltaic current-voltage (I-V) curves with changing irradiance and temperature.

3.1.3. AC Simulation

The AC inputs were designed to be generated from a multi-day grid voltage and frequency profile recorded from Sandia's Distributed Energy Technologies Laboratory (DETL), like that shown in Figure 5, or from measurements from other DER equipment or sensors. In canary applications, there is likely similar DER equipment on the same subnet that could be queried to reproduce the current grid conditions in the canary. The realism of the DER simulation would be enhanced by reporting the local grid voltage and any changes to frequency the other grid-connected equipment experienced. In theory, this could also be done with grid measurement equipment on or near the DER canary/honeypot as well. For all simulation inputs and calculated values, there is a small amount of random noise (e.g., ±0.5%) applied to the signal to simulate measurement errors and ensure multiple DER Simulators do not have the exact same measurements.

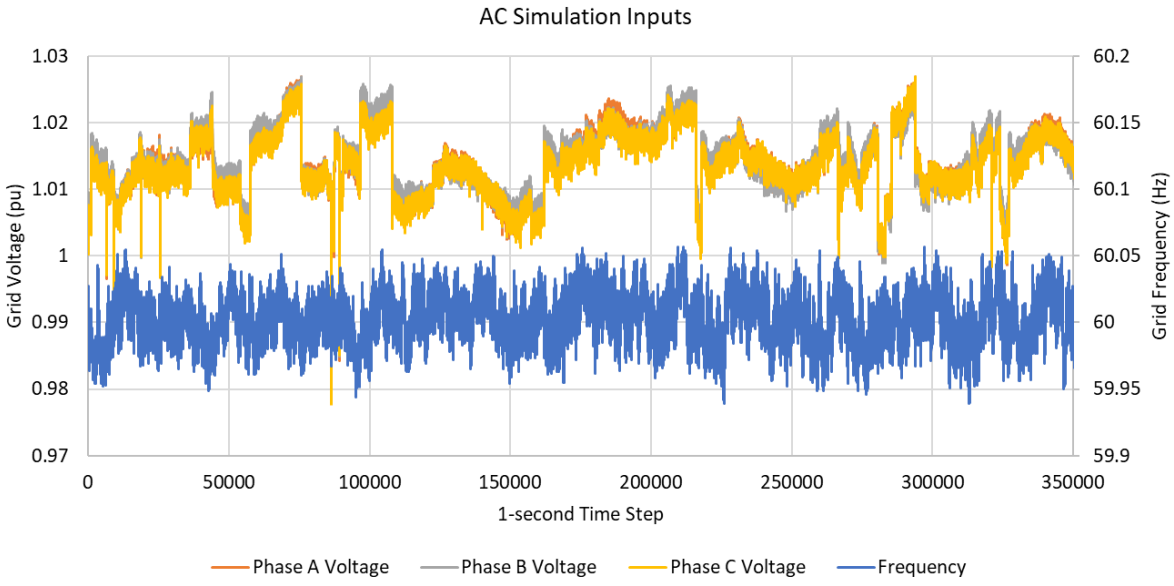

Figure 5: AC grid inputs.

The electrical behavior of the DER reflects changes in the DER grid-support function parameters and the local power system attributes. The steady-state functions like volt-var, frequency-droop, and volt-watt were built into the DER simulation, based on extensive interoperability and power testing at DETL over the last 5 years[18,19,20,21,22]. Transient functions like voltage and frequency ride through[23] and unintentional islanding protection were not included because these behaviors are not visible by attackers through the communication network. Transient functions operate during rare grid faults so these capabilities will not reveal the simulation unless they are configured outside their normal operating ranges.

The AC side tracks what grid-support functions are enabled at any given time. The reactive power target for the DER is calculated by working through these reactive power functions:

1. Constant power factor mode (PF)
2. Voltage-reactive power mode (VV)
3. Active power-reactive power mode (WV)
4. Constant reactive power mode (VarSet)

and determining which, if any, are enabled and calculating the associated reactive power setpoint. That setpoint is then updated on each of the phases of the DER AC-side simulator.

The active power for the DER is calculated based on status of the following control functions:

1. WMax – A power setting that derates the inverter
2. Frequency-Droop – Changes active power of the DER based on the grid frequency.
3. Volt-Watt - Reduces the power if the voltage is too high
4. WMaxLimPct - Curtailment based on percentage of nameplate capacity
5. WSet - Curtailment based on active power in watts
6. Other Var priority reactive power functions like VV, WV, PF, or VarSet that push the inverter of the maximum power point to produce the required reactive power

In the cases of some of the reactive power functions like fixed power factor, changes to these settings on larger DER will have impact on the local voltage. This is currently not included, but a simple power system model could be used to make changes to the voltage measurements to reflect changes in DER

[18] J. Johnson, R. Ablinger, R. Bruendlinger, B. Fox, and J. Flicker, "Interconnection standard grid-support function evaluations using an automated hardware-in-the-loop testbed," IEEE Journal of Photovoltaics, 8, 565–571. 2018.
[19] N. Ninad, E. Apablaza-Arancibia, et al. Development and evaluation of open-source IEEE 1547.1 test scripts for improved solar integration. In EU PVSEC 2019: 36th European Photovoltaic Solar Energy Conference and Exhibition, Marseille, France, 9-13 September 2019, 952–957, 2019.
[20] J. Johnson, E. Apablaza-Arancibia, N. Ninad, et al. "International development of a distributed energy resource test platform for electrical and interoperability certification," In 2018 IEEE 7th World Conference on Photovoltaic Energy Conversion (WCPEC), 2492–2497. 2018.
[21] J. Johnson, R. Brundlinger, C. Urrego, and R. Alonso, "Collaborative development of automated advanced interoperability certification test protocols for PV smart grid integration," In EU PVSEC 2014: European Photovoltaic Solar Energy Conference and Exhibition, Amsterdam, Netherlands, 2014.
[22] N. Ninad, E. Apablaza-Arancibia, M. Bui, J. Johnson, et al. "PV inverter grid support function assessment using open-source IEEE P1547.1 test package," In 47th IEEE Photovoltaic Specialists Conference (PVSC), Virtual Meeting, June 15-Aug. 21, 2020.
[23] N. Ninad, E. Apablaza-Arancibia, M. Bui, and J. Johnson, "Commercial PV inverter IEEE 1547.1 ride-through assessments using an automated phil test platform." Frontiers in Energy Research (submitted) 2020.

controls. This would help in convincing the adversary they are affecting the local power grid realistically and encourage them to keep testing their attacks.

3.1.4. IEEE 1547-Compliant Interfaces

IEEE 1547-2018[24] mandated that DER devices include one of three standardized interoperability interfaces: IEEE 2030.5, IEEE 1815, or SunSpec Modbus. DER products are starting to arrive on the market with these communication interfaces, so the DER Simulator was designed to include these capabilities as well as legacy Modbus interfaces that exists on many DER devices from Fronius, SMA, SolarEdge, etc. In the case of the IEEE 1815 (DNP3) and SunSpec Modbus communication interfaces, the DER will host a DNP3 Outstation interface or Modbus Server. In the case of the IEEE 2030.5 interface, the DER can be created with an IEEE 2030.5 client but there are two complications:

- The DER must be provisioned with a valid public key infrastructure (PKI) credential for that jurisdiction
- The DER client initiates the connection to the server, so it is not clear how the adversary would establish a valid IEEE 2030.5 connection.

At this time, only the Modbus server has been created in the DER Simulator. It is instantiated when the DER Simulator is created and populates the initial nameplate, settings, monitoring, and control data using either a JSON file[25] or python dictionary. The values are updated at a user-selected update rate. Once connected, client changes to the control parameters adjust the operation of the device which is then reflected in the power measurements of the AC and DC side. A major software engineering effort was undertaken to map hundreds of data points in the Modbus server to the appropriate control/monitoring functionality in the DER Simulator.

3.1.5. Non-standardized Communication Interfaces

Often DER equipment will reach out to DER vendor monitoring cloud environments or servers to request firmware updates, post solar generation summaries, or report monitoring or prognostics information. In rare cases, DER devices will include open ports and services, like ssh or telnet, for remote configuration or debugging. These non-standardized communications interfaces must be effectively mirrored to create realistic DER simulations and network behaviors.

When the honeypot is scanned, it is also important to include the correct number of traceroute hops, realistic nmap fingerprints, and the manufacturer MAC addresses, etc. To get the networking to match real systems, it was proposed to deploy the devices at representative locations with real internet service provider (ISP) connections to the internet. For residential DER devices, the honeypot would be connected to the internet via a residential ISP connection. For utility canaries, the DER Simulator would be connected to the OT network.

[24] IEEE Std 1547-2018, "IEEE Standard for Interconnection and Interoperability of Distributed Energy Resources with Associated Electric Power Systems Interfaces," pp.1-138, 6 April 2018, doi: 10.1109/IEEESTD.2018.8332112.
[25] For example: https://github.com/jayatsandia/svp_energy_lab/blob/dev3.7/Lib/svpelab/sunspec_device_1547.json

A python script was developed utilizing a popular packet manipulation library, Scapy, to recreate the nmap fingerprints and MAC addresses of our physical DER devices at DETL. When running, the script will intercept all incoming traffic over the specified network interface and do a couple things depending on what traffic is being intercepted. If the script detects an ARP scan, the program will generate a corresponding response with a spoofed MAC address (i.e., the MAC addresses of the DETL DER devices). The spoofing script will also detect and respond to standard ICMP requests so that if an adversary wanted to see what devices were active on the network it would respond back with a message saying, "I am here". The final capability of the script is that it can respond to a nmap scan by revealing any number of ports to be open on the honeypot—thereby, mimicking the chosen physical devices. This portion of the script can spoof many other ports such as 21 if the DER included FTP, 22 if the DER included SSH, 23 if the DER used Telnet, 80 or 443 if the DER had a web server, etc. This functionality will be particularly useful in deploying a DER Simulator that mirrors a specific physical device. For example, we may want to have a DER device that has the Modbus server operating on port 502 but also spoofs other ports/services, and the specific versions of these services, when port scanned. However, it should be noted that these services may need to also include some basic functionality if the adversary were to attempt to connect to them—an activity for future work.

3.1.6. Monitoring and Test Environment

To display the current and historical behavior of the DER equipment, a web application was created to run alongside the DER simulator. This web application was developed for visualizing the state of the simulated DER device, as well as for live configuration of the DER Simulator. This application has been useful in the development and debugging of the DER Simulator.

The web application is written in TypeScript, and it utilizes open-source libraries d3js and plot.ly for rendering interactive graphs. The application communicates with the DER Simulator via a simple protocol over websocket, with which it can read and write SunSpec models. It uses several standard SunSpec models for reading metrics on the device state, including SunSpec 701 (DER AC Measurement) and SunSpec 714 (DER DC Measurement). The web application also makes use of models that are specific to the DER Simulator, which were created to expose internal state and provide an interface for live configuration of the simulator. These models are detailed in Appendix B.

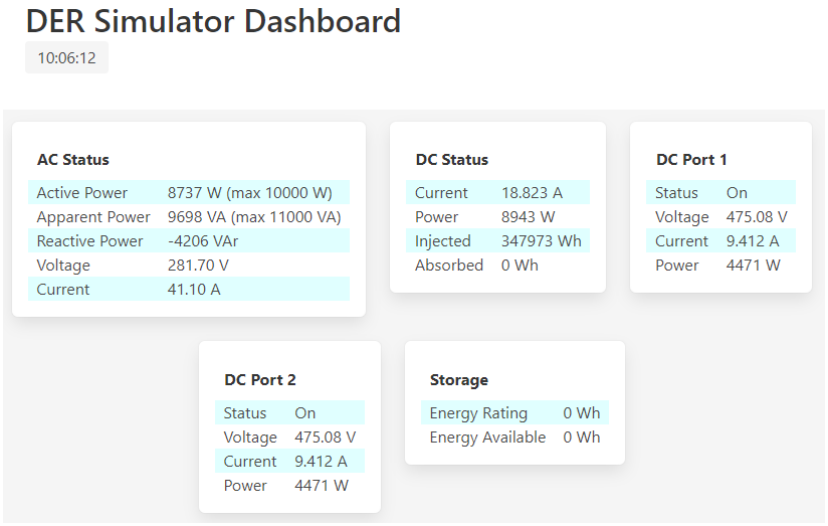

Figure 6: DER measurement visualization dashboard.

The web application's dashboard, shown in Figure 6, displays metrics from the AC, DC, and storage interfaces of the DER device. These metrics are all gathered from SunSpec models, which are polled every second by the web application to provide a live view of the simulator's state.

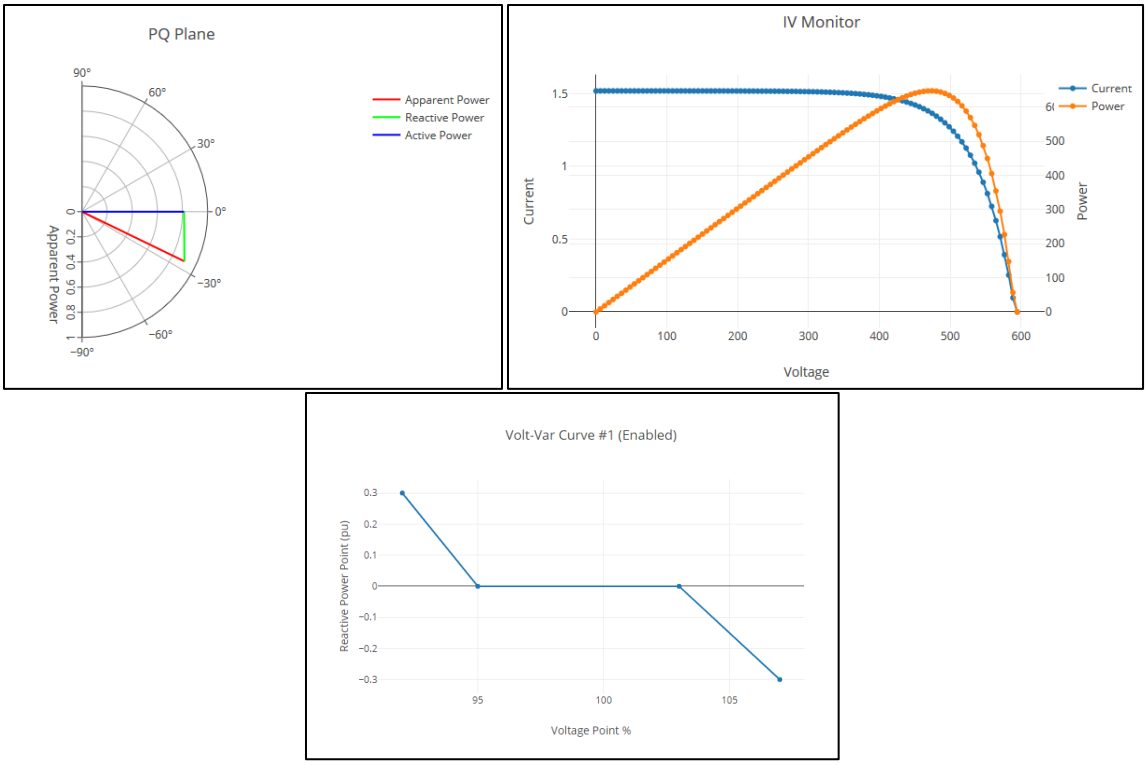

Figure 7: Live dashboard graphs displaying the PQ plane (top left), DC I-V curve (top right), and the volt-var curve configuration (bottom).

Displayed in the dashboard are plots of various power behaviors in Figure 7, including the PQ plane, the DC current-voltage and current-power curves, and grid-support curves such as Volt-Var. The dashboard also allows for live control of various parameters of the simulated DER device and its environment, as shown in Figure 8. These inputs can be a time-indexed data source such as a CSV, or user-defined constant values. This allows for easy testing/debugging of the simulator, e.g., changing the grid voltage to confirm that the volt-var function is operating as expected. Simulation properties including simulator clock time, irradiance, ambient temperature, and grid measurements. The AC grid model and the DC model are both synchronized with the simulator clock, so interesting time periods could be replayed simply by adjusting the simulation time.

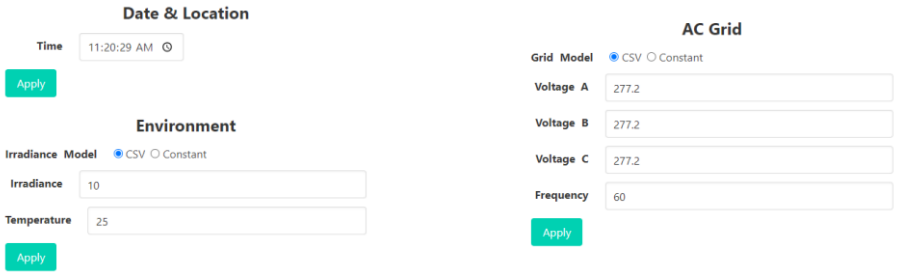

Figure 8: Interface for live configuration of the DER Simulator.

3.2. Intrusion Detection and Alerting

The objective of honeypot deployments is to gather actionable threat information and cataloging adversary objectives and Tactics, Techniques, and Procedures (TTPs). Example TTPs are included in the MITRE ATT&CK and ATT&CK for ICS frameworks[26]. Some of the adversary objectives could include making money (i.e., deploying ransomware, cryptojacking), hacktivism (i.e., DOS/DDoS, Modbus reads), national state attacks (i.e., Modbus manipulation, PII theft). The TTPs that enable those actions could include many things such as brute forcing weak/default username and passwords, zero-day software exploits, command injection, etc. The TTPs that enable those attacks includes exploiting stolen passwords, zero-day software exploits, and remote code injection, etc. To capture the latest adversary's TTPs, the honeypots can be equipped with continuous monitoring tools at both system- and network-level. Those tools generate raw data such as system/authentication/firewall logs, and network packet flows. To alert on suspicious interactions with the honeypot, these monitoring data will be coalesced with time-stamped Security Information and Event Management (SIEM) or network analysis tool like Splunk[27], OSSEC[28], Snort[29], or MALCOLM[30] for automated attack preemption driven by detection tools and supporting attack responses by human analysts. The main challenge is to identify real attacks from the background noise of mostly automated and naïve attack attempts by bots.

Alerts can be configured on the system for integrity violations (e.g., injecting backdoor into a system kernel module) or data breach (e.g., accessing a private secure shell key). A future host-based intrusion detection system could periodically create snapshots of the honeypot states, so all malicious payloads delivered into the honeypot are preserved for forensic investigation.

In the case of this demonstration, a network-based intrusion detection system (NIDS) was deployed on the same system as the DER Simulator to track network data. There are many different types of NIDS tools that are open-source or commercially available[31]. These tools fall broadly into two categories of anomaly detection:

- Signature-based NIDS tools detect malware using deep packet inspection. Depending on the sector that the IDS is being applied, the signatures will be tailored to detect threats specific to those systems.
- Behavior-based techniques detect known and unknown patterns through inference using statistical learning or classification techniques like supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning methods.

The network monitoring was implemented as a bump-in-the-wire solution to analyze all traffic communicated between systems[32]. As shown in Figure 9, the bump-in-the-wire security system can be

[26] MITRE, ATT&CK, URL:https://attack.mitre.org/
[27] Splunk, URL: https://www.splunk.com/
[28] OSSEC, URL: https://www.ossec.net/
[29] Snort, URL: https://www.snort.org/
[30] MALCOLM, URL: https://github.com/idaholab/Malcolm
[31] C. Lai, A. Chavez, C. B. Jones, N. Jacobs, S. Hossain-McKenzie, J. Johnson, A. Summers, "Review of Intrusion Detection Methods and Tools," Sandia Technical Report, SAND2021-1737, February 2021.
[32] S. Hossain-McKenzie, A. Chavez, N. Jacobs, C.B. Jones, A. Summers, & B. Wright, "Proactive Intrusion Detection and Mitigation System: Case Study on Packet Replay Attacks in Distributed Energy Resource Systems" 2021 IEEE Power and Energy Conference at Illinois (PECI), April 2021.

configured between individual systems or networks communicating with one another. The Snort IDS was installed along with a collection of OT protocol rules[33] for Modbus and DNP3 on the bump-in-the-wire device of the DER Simulator itself. In addition to monitoring the OT protocols, Snort has been configured to monitor all other network traffic passing through as well. Snort comes pre-populated with signatures for many IT focused protocols and attacks that are also used within OT environments, such as ICMP, Telnet, Denial-of-Service (DoS), network scans, etc.

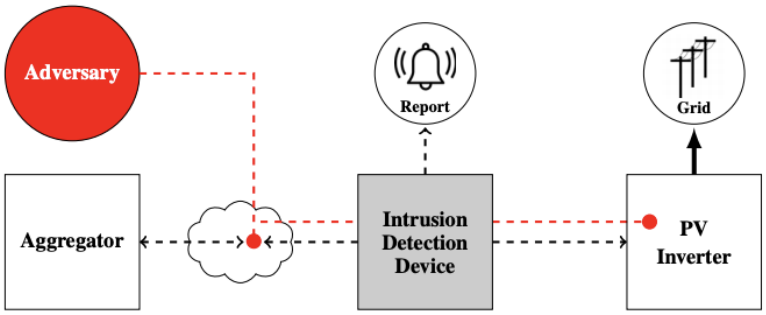

Figure 9: An example deployment of a bump-in-the-wire IDS monitoring and alerting system in a DER environment.

Additional monitoring and cybersecurity tools have also installed on the bump-in-the wire system as shown in Figure 10. Zeek has also been installed to capture network statistics specific to each protocol observed on the bump-in-the-wire device. Any alerts or logs generated are immediately ingested by Filebeat[34] which monitors user-defined log files. Once Filebeat reads and parses each log file configured by the end user, the results are sent to Logstash[35] which listens on a network port for alerts or events that can then be sent to an Elasticsearch [36]database. To visualize the data within the Elasticsearch database, Grafana was deployed to generate graphs summarizing network statistics along with tables showing live-event logs to provide situational awareness for security administrators.

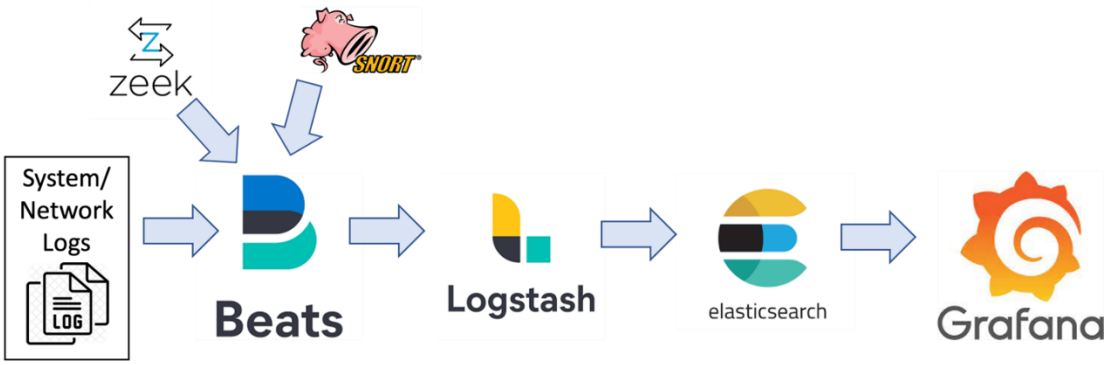

Figure 10: The process in which alerts are identified and presented to system administrators when anomalies or attacks are detected.

This framework allows for a variety of data sources to be introduced to detect cyber-physical events. For this implementation, Snort, Zeek, and system logs are used as inputs, but additional or future tools

[33] Digital Bond Quickdraw Snort IDS rules: https://github.com/digitalbond/Quickdraw-Snort
[34] Filebeat, URL: https://www.elastic.co/beats/filebeat
[35] Logstash, URL: https://www.elastic.co/logstash/
[36] Elasticsearch, URL: https://www.elastic.co

can also be introduced into the framework. The data stored within the elasticsearch database provides a centralized location to consolidate the logs files for an administrator or for additional processing/analytics, i.e. machine learning analysis. Grafana was selected for this implementation, although Kibana or other visualization tools could have been selected. Grafana was chosen only because it was readily available for our implementation which was built on a low-power, low-cost Raspberry Pi. Figure 11 shows the Grafana dashboard implemented on a bump-in-the-wire device. The top graphic displays network data over a configurable interval (the last 5 minutes is shown in the diagram). The spike in network activity near the middle of the graph shows the increase in volume of traffic that results from a nmap Christmas tree scan – a network scan with all flags turned on. The bottom table of the dashboard shows the raw alerts and the source where those alerts were generated. In Figure 11, the "NMAP scan XMAS" is detected and logged by snort as a high priority alert. Additionally, logs from Zeek and syslog are also displayed.

The installation process for each of the software tools developed for our bump-in-the-wire solution is documented in Appendix C. A Raspberry Pi model 4B with 2GB SDRAM was used for our deployment with a single ethernet interface. Although the entire ELG stack along with the IDS systems were installed on a single Raspberry Pi, the Grafana dashboard and the Elasticsearch database could be stored on another system with more memory and/or storage resources if needed.

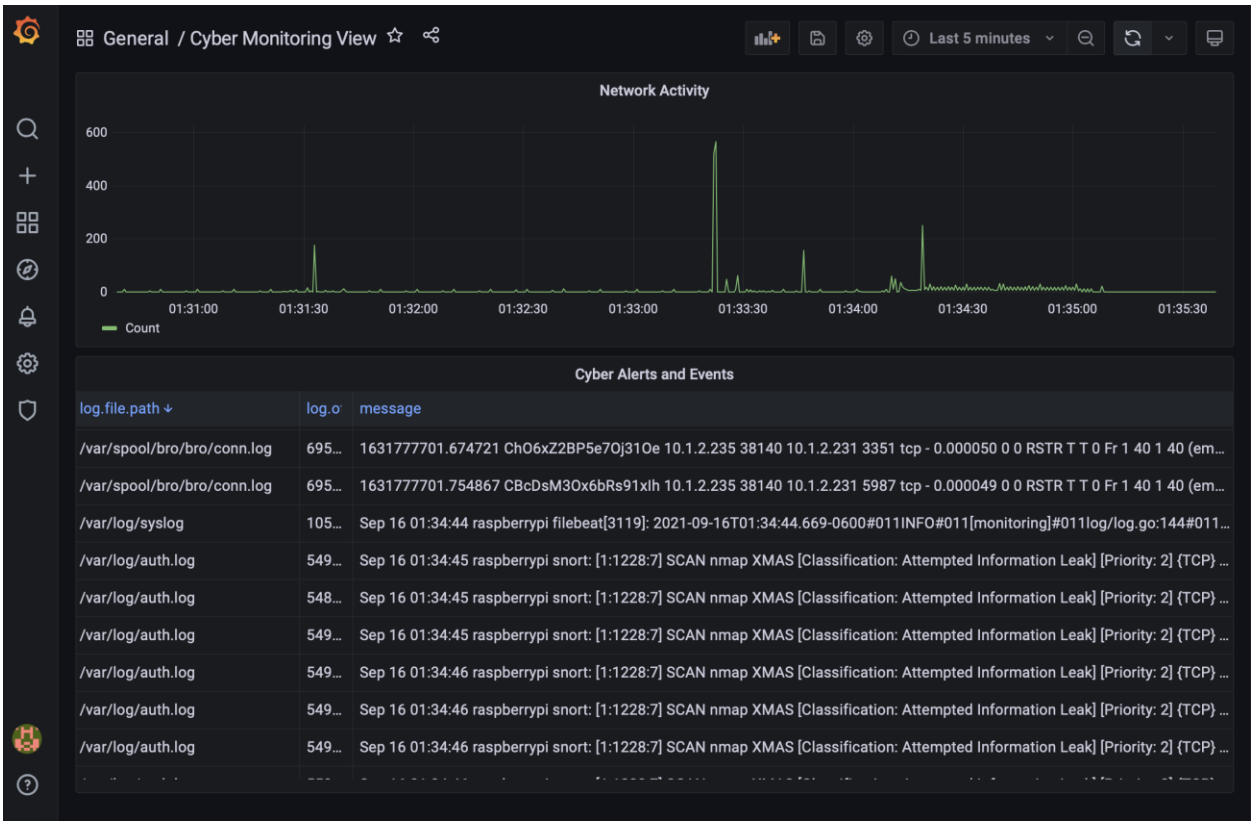

Figure 11: A Grafana dashboard displaying network activity along with system alerts stored within the Elasticsearch database.

3.2.1. Behavior-based Intrusion Detection

Unsupervised artificial neural network (ANN) algorithms are a useful tool for detecting cyber-attacks directed at PV inverters[37]. Successful implementation of ANN for intrusion detection depends on the availability of training data. Cyber network data is difficult to acquire and changes in network behavior is very common. To address these issues, the team developed and tested an online learning approach where training and testing occurs simultaneously.

This intrusion detection methodology assumes that the adversary cyber traffic travels through or is directed at the device hosting the network sensor. The behavior-based NIDS device is most likely to be installed on the DER Simulator computer although it could be a bump-in-the-wire device in the communication path to the DER honeypot/canary. The NIDS includes a Zeek network security monitor[38] that logs all the connections associated with the DER device. Data features from these logs are used as inputs into the ANN algorithm.

In this case, the Adaptive Resonance Theory (ART) ANN[39] performed the learning and anomaly detection of the network data. Implementation of the algorithm begins by normalizing the data between zero and one. Then, a complement coding of the normalized data occurs prior to the category choice calculation that finds the templates (or ART memory) that best match with the input vector. The input and best match are then evaluated to see if they pass the vigilance test. If it passes, the template is updated based on the new input.

The initialization of the online learning involved a review of one day of data. This review included the confirmation of the best input features using a confusion matrix (Figure 12) of the different correlation coefficients for response bytes, origin bytes, connection rate, and connection time. Figure 12 shows that there is a significant correlation between origin bytes and the connection time and therefore can be left out of the ART training and testing. This meant that only the three features: (1) connection rate, (2) connection time, and (3) response bytes were used for the online learning and anomaly detection.

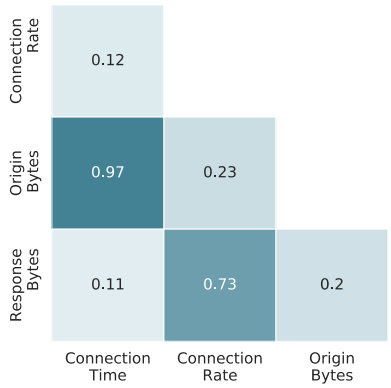

Figure 12: Data features used by the Adaptive Resonance Theory algorithm to detect network anomalies.

[37] C. B. Jones, A. Chavez, R. Darbali-Zamora, S. Hossain-McKenzie, "Implementation of Intrusion Detection Methods for Distributed Photovoltaic Inverters at the Grid-Edge", IEEE Power & Energy Society Innovative Smart Grid Technologies Conference, 2020

[38] "The Zeek Network Security Monitor", URL: https://zeek.org

[39] G.A. Carpenter, S. Grossberg, and D.B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system", Neural Networks, vol 4, no. 6, Jan. 1991

The online learning follows the process outlined in the flow chart depicted in Figure 13. It begins at time equal to zero with the first monitored data from the Zeek sensor to the ART training which creates an initial ART template. After the initial training and for all t > 0, the sensor data is passed to both the ART testing and a statistical computation. This approach is different from our prior work with a user-defined threshold limit[40]. If the new input resonates with existing templates, then it is sent to the training algorithm to update the existing templates. If it does not resonate, the data is compared with the mean and standard deviation. If it falls outside of the limits, it is flagged as a potential issue. If not, the input is sent to the ART training to start a new template in the ART memory.

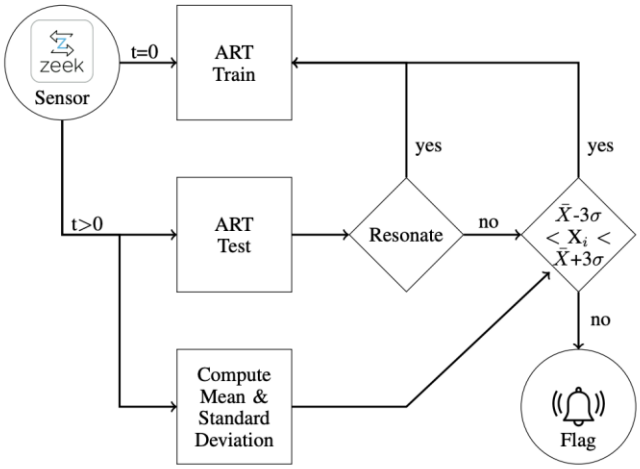

Figure 13: Flow diagram for the online learning process.

This process was tested on actual network traffic with adversary activity directed at a PV inverter. Three main scenarios were tested that included normal actions (i.e. Modbus, TCP/IP, SSH, etc.), adversary reconnaissance, and a denial-of-service (DOS) attack. The ART anomaly results are shown in Figure 14 for three scenarios. Figure 14(a) shows the ART hyper-boxes, which represent the multi-dimensional templates that surround the normal data. This figure shows that three templates were created. The first template was based on the initial data presented to the algorithm and the next two were formed when new data did not resonate with the original template but passed the statistical review. The algorithm then considered network traffic generated by an adversary who performed actions to learn what devices were connected to the network and attempted to deny any services by flooding the PV inverter with messages. Figure 14(b) and (c) show that reconnaissance and DOS network activity fell well outside templates and the statistical limits and were therefore considered abnormal behavior.

[40] C.B. Jones, A. Chavez, S. Hossain-McKenzie, N. Jacobs, Adam Summers, and B. Wright, "Unsupervised Online Anomaly Detection to Identify Cyber-Attacks on Internet Connected Photovoltaic System Inverters", IEEE Power & Energy Conference at Illinois (PECI), 2021

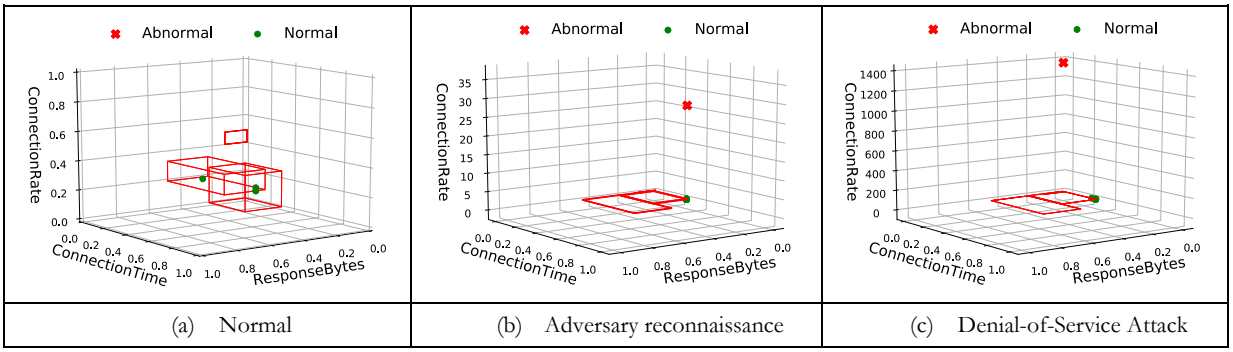

Figure 14: ART results for (a) normal, (b) adversary reconnaissance, and (c) a denial-of-service attack. No false positives occurred when subjected to the normal data and each of the cyber-attacks were detected.

3.3. Honeypot and Honeynet Networking

In the future, using Emulytics tools like minimega, representative home networks could be generated that contain several realistic components of a flat home network firewalled from the public internet. To make this environment look realistic and "lived-in," a system consisting of existing VM images that represent a windows computer, smart lightbulbs, router, and/or network storage devices could be deployed in the network with the DER. The router could be configured with a vulnerable firmware version[41] so hackers could access the equipment on the network.

3.4. Hosting Challenges

There are many legal, technical, and security issues associated with the decoy deployment locations. Placing the honeypots on hosted cloud platforms like Amazon Web Services (AWS) or Google Cloud is not ideal because their public IP spaces are public and known to sophisticated adversaries. Deploying the systems on corporate networks runs into similar problems, plus complicated issues arise if the systems are compromised and used for DoS or other attacks on national or international targets. Having traffic originating from a corporation puts the organization in a difficult space. There are also legal ramifications if compromised systems were used for other nefarious activities (hosting pornographic material, etc.) and ISPs may disconnect internet access if complaints are submitted or activity outside of the terms of use is detected. These issues were not fully resolved within the course of this project.

3.5. Threat Sharing

Malicious control of DER/Internet of Things (IoT) fleets can substantially impact the national security by affecting the electric grid, transportation sector, and other critical infrastructure. It is essential to capture adversary movements to minimize risks presented by these devices. One way to help is to establish good information sharing mechanisms as data is captured by the honeypots. Threat intelligence could be gathered for critical power system infrastructures from the honeypot analysis and directly shared with DHS, DOE IN/CESER/OE, FBI, utilities, and other government agencies based on the type of information that is gathered. Creating computationally-efficient DER honeypot fleets would allow researchers to discover new DER vulnerabilities and track adversary TTPs. It is

[41] "Unpatched vulnerability identified in 79 Netgear router models", URL: https://www.zdnet.com/article/unpatched-vulnerability-identified-in-79-netgear-router-models/

anticipated this would be an effective countermeasure for mid- and high-tier adversaries targeting DER critical infrastructure.

4. DER HONEYPOT DEMONSTRATIONS

Two use cases are explored through the course of this project. In the first, a 10-kW solar PV inverter with two DC ports was simulated as a standalone DER honeypot. In the second use case, an energy storage DER device was cloned and emulated to create a canary that matched the configuration of another device on the network. The details of each of these applications is presented in the following section.

4.1. Stand-Alone DER Honeypot Implementation

The DER honeypot initialization and operational output is shown in Appendix A. The typical startup sequence includes the following:

1. Read the JSON model of the DER SunSpec Modbus map and create the TCP and websocket server.
2. Link the AC and DC points from the server to actions in the DER simulation.
3. Start the DC simulation.
4. Start the AC simulation.
5. Enter main loop:
    a. Update simulation clock
    b. Update DC state.
    c. Update AC state.
    d. Asynchronously read requests from websockets and write responses.
    e. Asynchronously read requests from Modbus TCP sockets and write responses.
    f. Asynchronously respond to Modbus reads/writes.

Once in the main loop, the clock is updated, AC and DC measurements are read from the CSV files at a user-selected rate, and any reads or writes to the Modbus server and websockets are handled. These updates were executed using the asynchronous subprocesses.

At this time, active power curtailment functions and reactive power setpoint functions have been validated and function as expected. Autonomous functions such as volt-watt, volt-var, frequency-droop, etc. were not fully tested; although it should be noted that these functions would be more difficult for honeypot detection because DER behavior only changes based on external parameters (grid voltage, grid frequency, DER power, etc.). In the current version of the DER Simulator, timing parameters are present, but the timers were not fully implemented.

A screenshot of the SunSpec Common Model is shown in Figure 15, Capacity in Figure 16, and DER AC Measurements in Figure 17. Each of the DER AC Measurements reads produces new values based on the AC and DC simulations. Two reads of this model are shown in Figure 17. It can be seen there is around -3 kVar of reactive power from the DER device. This is because this device has been configured with a 0.9 underexcited power factor (PF) using the DER AC Controls in Figure 18. When Modbus writes to that grid support function are completed the DER Simulator is updated with the new value and the power simulation changes appropriately.

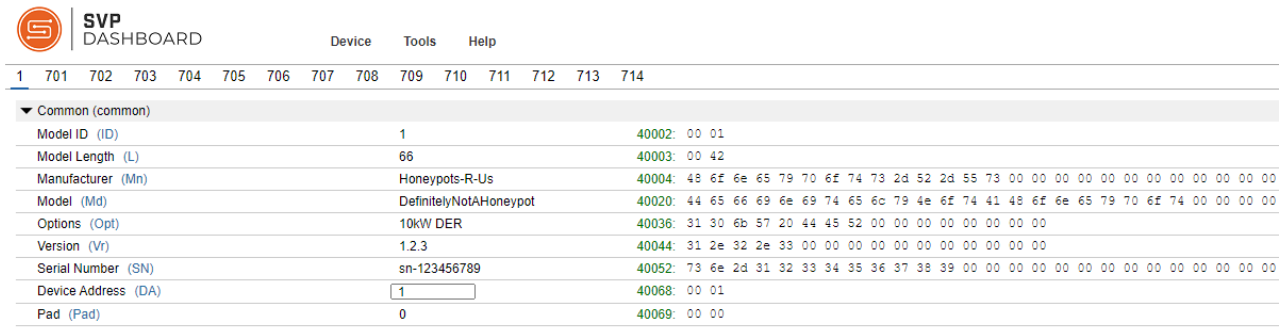

Figure 15: Honeypot Common Model data read using the SunSpec SVP Dashboard.

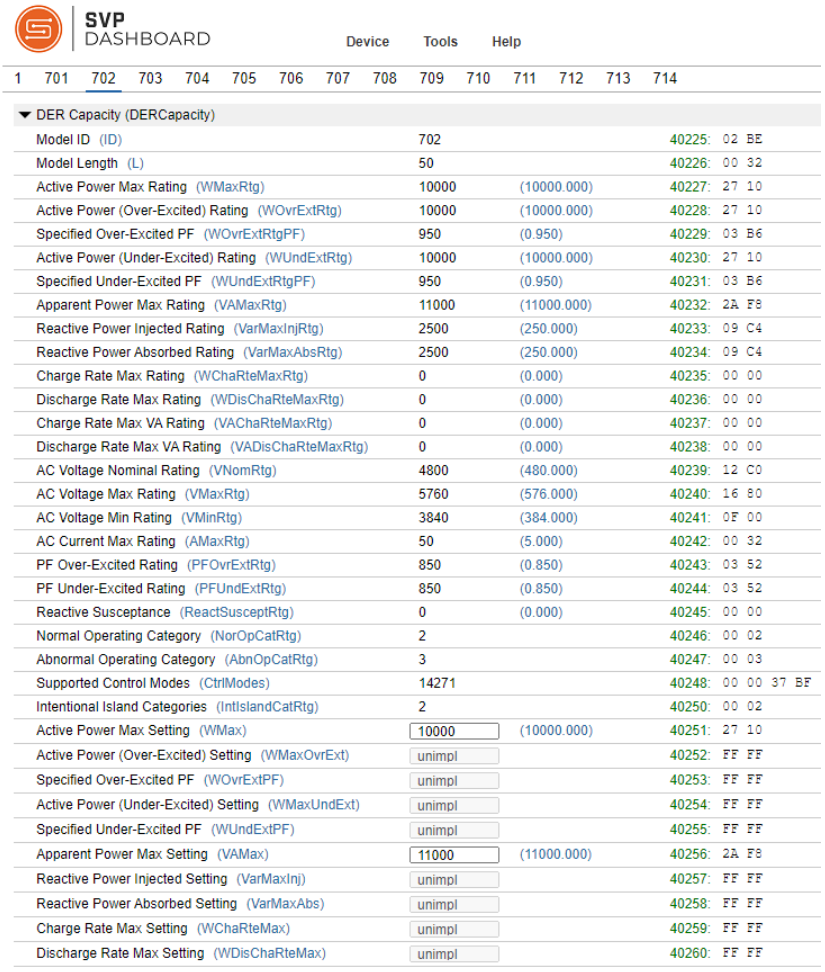

Figure 16: Honeypot DER Capacity Model data.

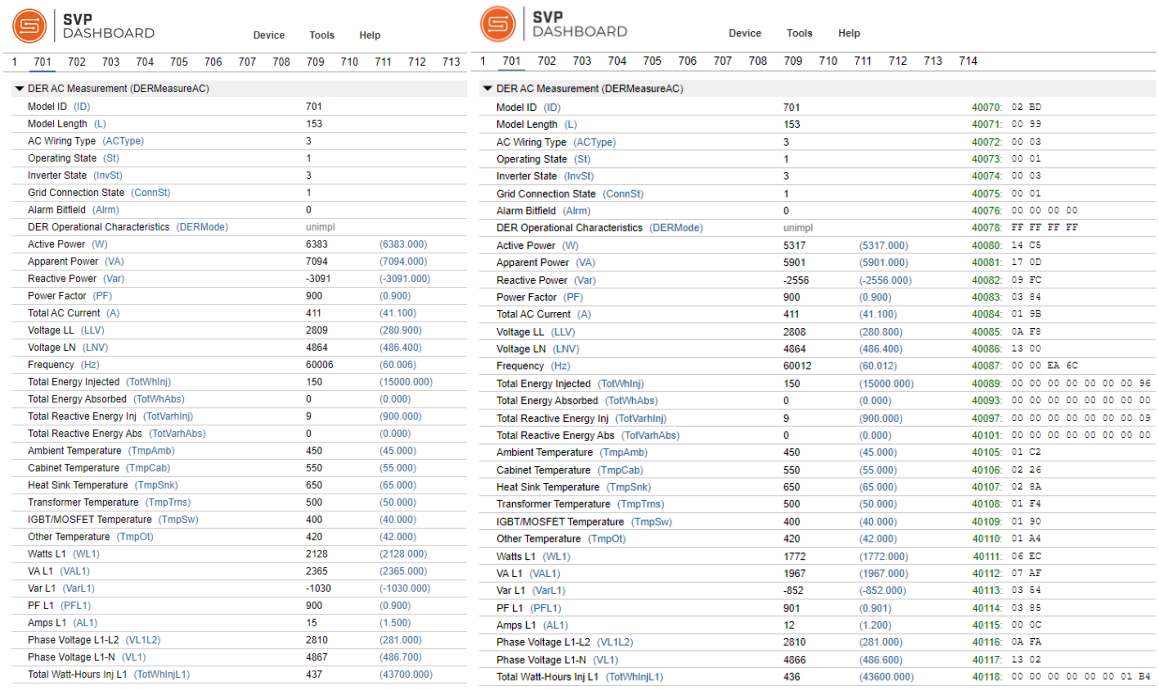

Figure 17: Multiple reads of the Honeypot DER AC Measurement Model.

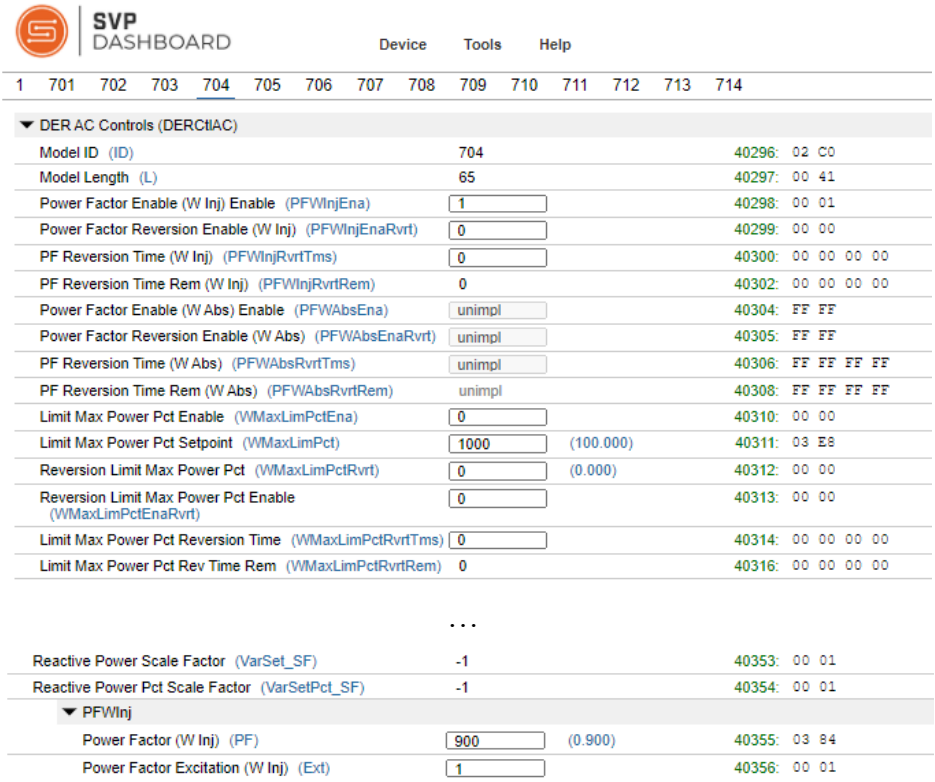

Figure 18: Honeypot DER AC Controls Model showing the PF function is set to 0.90 with an underexcited excitation.

4.2. Cloning an Energy Storage System for a DER Canary Application

The DER Simulator was configured with SunSpec Modbus configuration representing a 7.7 kW prototype energy storage DER device at DETL. This was done by generating a JSON file of the SunSpec Modbus map with the SunSpec SVP Dashboard and then loading this file into the DER Simulator. The canary was deployed on one of the Raspberry Pi 4B computers with 32 GB of drive space shown in Figure 19 and then run in parallel with the physical device to compare the two devices.

Figure 19: Raspberry Pi 3B+ and 4B computers configured with DER Simulator software.

The SVP Dashboard was connected to the physical DER device and the honeypot clone on the Raspberry Pi. A comparison of the DER Capacity Data is shown in Figure 20. As shown in the Figure, there were no changes to the writable settings in the physical ESS since the Modbus snapshot was taken, so there are no differences between the physical and honeypot/canary systems. Over time, if settings are changed in the physical device or the honeypot, this data will not match. That is expected of two physical devices, so long as changes to those settings are mirrored in the measurements and behavior of the equipment.

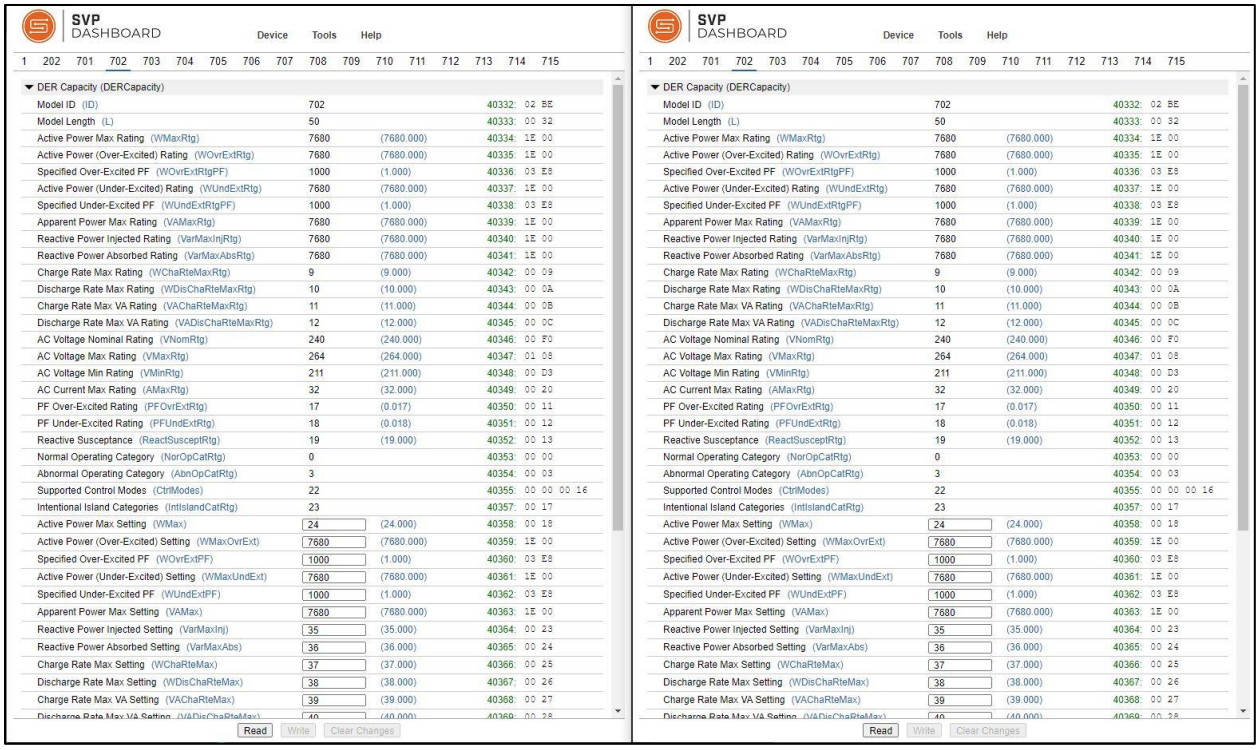

Figure 20: DER ESS (left) vs DER Honeypot (right) for SunSpec Model 702 *DER Capacity*.

After a period of time, the AC and DC measurements of the DER ESS and DER honeypot were not identical, as shown in Figure 21 and Figure 22. For instance, the Operating State and Inverter State changed for the physical DER device changed. The power system simulation using the pre-recorded grid voltage and frequency time profiles changed the grid measurements on the DER simulator and they no longer matched the ESS device. Deviations in frequency from other grid-connected equipment would be suspicious, although the frequency normally operates within such a tight envelope it might go unnoticed. Differences in voltages between DER equipment at the same site would be a big red flag. For instance, if DER canary device(s) were added to an OT network with multiple physical DER and the voltages were significantly different, this would be a clear sign that some equipment was not reading grid voltage correctly or the devices were not authentic and the adversary may not interact with the canaries. An alternative approach would be to read the voltage and frequency from the real devices every 1-2 seconds. This would be a good way to mirror the grid state on the DER canary so long as the adversary was unable to detect these communications. Similar approaches could be used to align DER operating temperatures—which would similarly expose the DER simulation.

There are other small corrections needed to improve the DER emulations. For instance, the DC Energy Injected and DC Energy Absorbed are coded to start at a large, random value but, in this case, the DER canary should start at the value provided in the JSON Modbus map and count based on the DC operations.

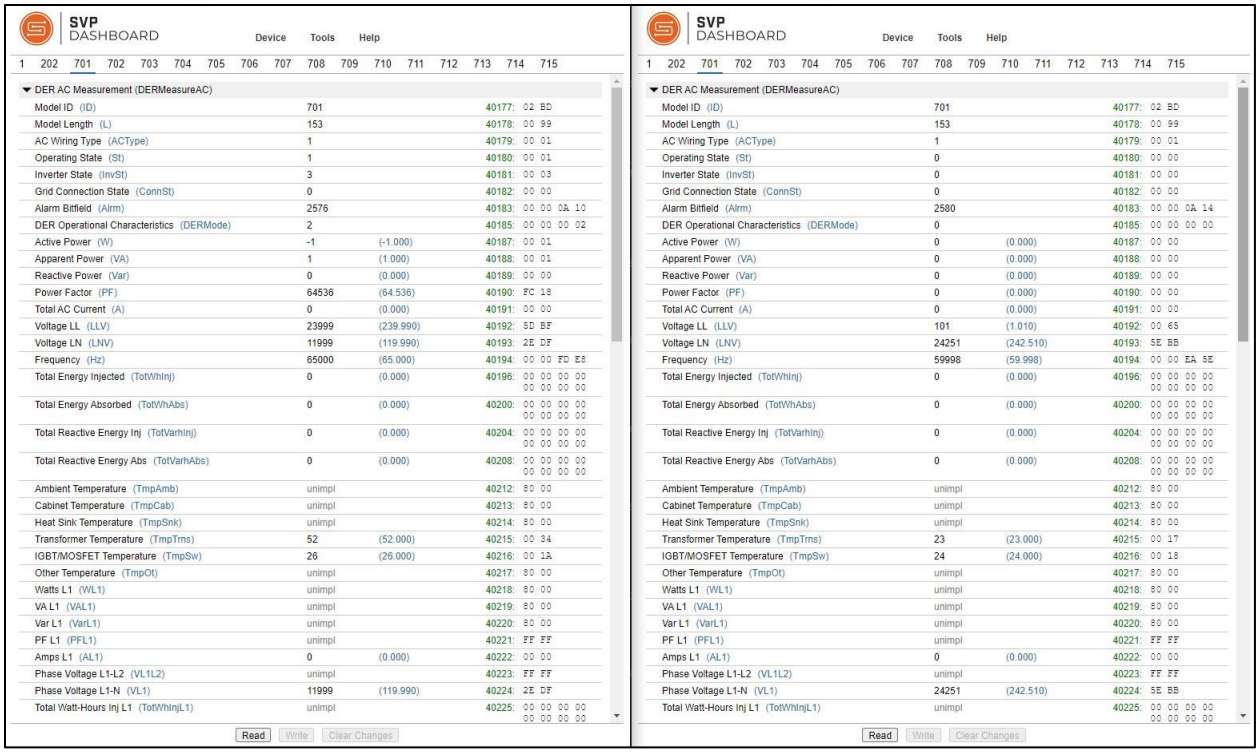

Figure 21: DER ESS (left) vs DER Honeypot (right) for SunSpec Model 701 *DER AC Measurement*.

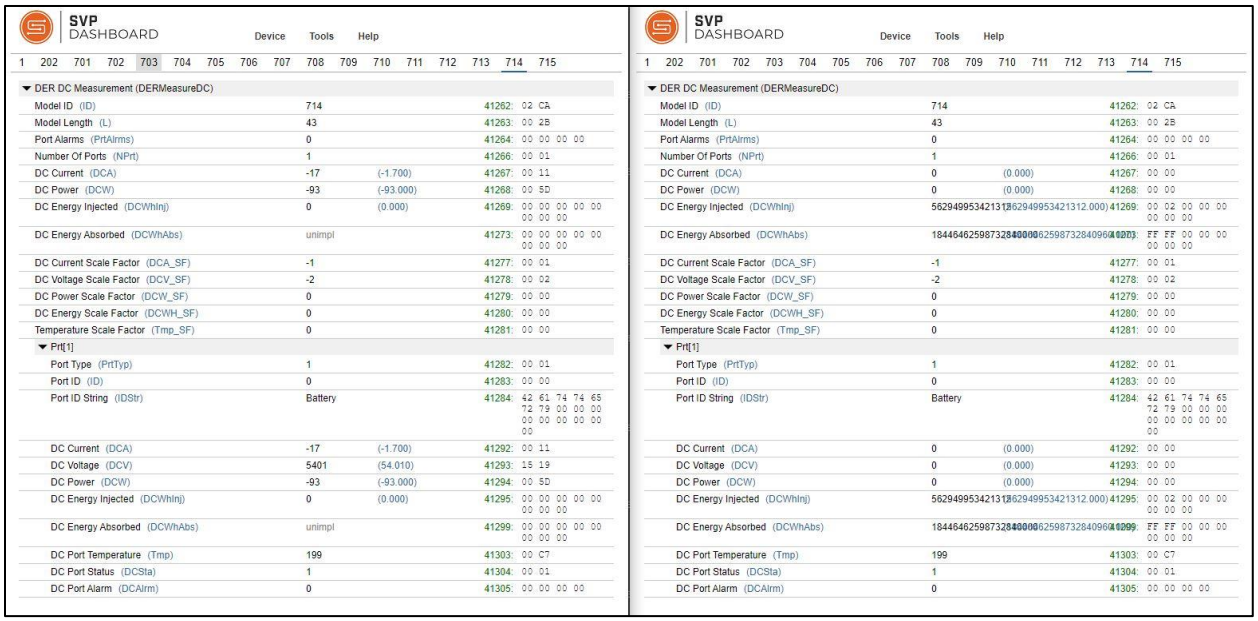

Figure 22: DER ESS (left) vs DER Honeypot (right) for SunSpec Model 703 *DER DC Measurement*.

It is not a good approach to clone DER device maps in their entirety at regular intervals because the DER devices are independent and contain their own states, parameters, and control modes. The power factor, power curtailment, active power, reactive power parameters from SunSpec Model 704 *DER AC Controls* is shown in Figure 23. The parameters for the voltage-reactive power (volt-var) function are shown in Figure 24. As grid operators or adversaries interact with the DER equipment or canaries, these devices should behave as though they are operational equipment to provide network

defenders and incident responders more time to analyze adversary actions. As soon as the adversary knows they have been detected, they may remove critical evidence of their activities, accelerate their attack schedule, deploy more powerful tools at their disposal, or exfiltrate as much data as they can. A convincing honeypot will gain time for the incident response team, so that they may quickly and strategically remove their access to the OT network and wipe adversarial tools and backdoors into the network.

Differences in volt-var or other settings between the physical devices and canaries may appear strange to an adversary, but this would be representative of the behavior of real devices. However, grid operators are likely to have uniform settings for the autonomous grid-support functions for a given region, circuit, or facility. If there were scheduled or commanded control settings issued to all the DER equipment on a site, it would be possible to also make this change to the DER canary, though this traffic would need to be filtered appropriately from the intrusion detection algorithms. This is one of the operational considerations that would need to be decided by the grid operator team.

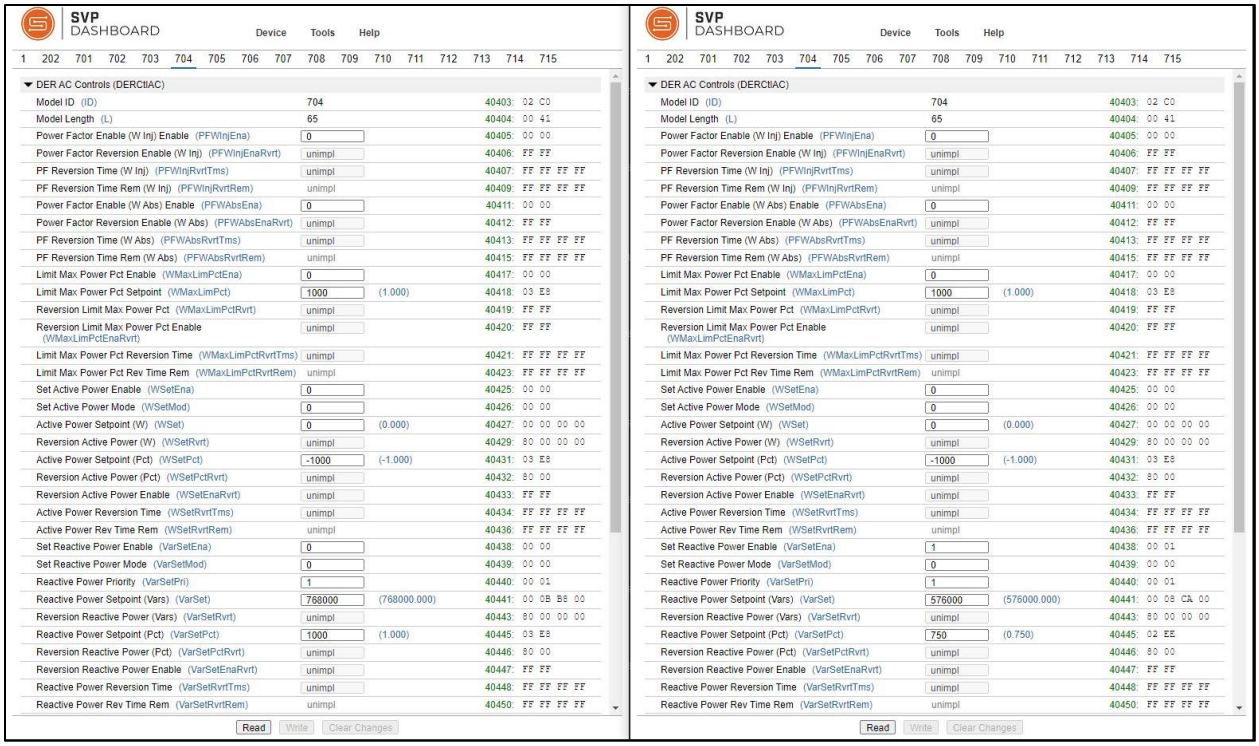

Figure 23: DER ESS (left) vs DER Honeypot (right) for SunSpec Model 704 *DER AC Controls*.

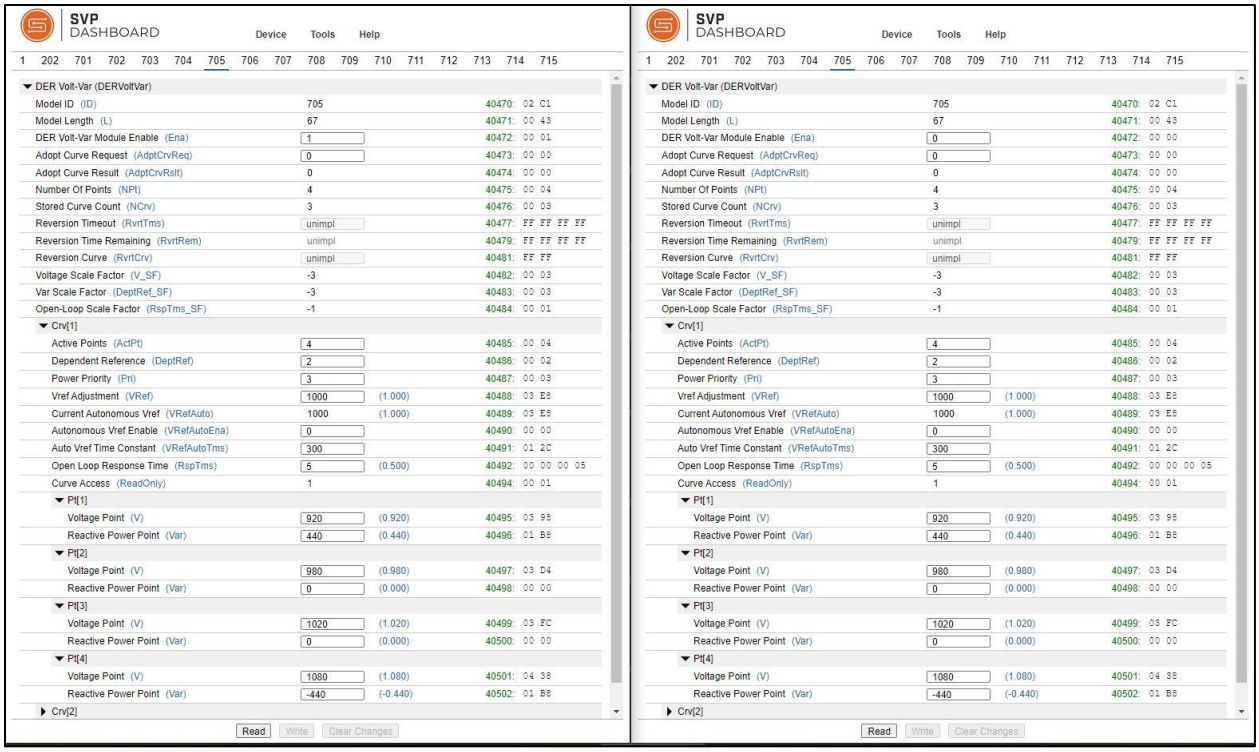

Figure 24: DER ESS (left) vs DER Honeypot (right) for SunSpec Model 705 *DER Volt-Var*.

5. ALTERNATIVE DER SIMULATOR APPLICATIONS

The DER Simulator is useful for applications in addition to being deployed as DER canaries or honeypots. Two potential applications are briefly described below.

5.1. Training Tools for Cyber Defenders

High-fidelity training environments known as cyber ranges could be quickly spun up for DER network operators, owners, utilities, and incident responders to train in red team/blue team scenarios using DER Simulators as the backbone for cyberwar games. Training defense challenge courses for electricity subsector owners and operators would enhance their preparedness against a cyber incident impacting DER systems with a hands-on, simulated, demonstration of a cyberattack. This would train participants to respond to DER cyber-attacks in the future, clean up internal communication lines, and establish better communication channels to external agencies to support overall grid reliability and resiliency. For DER vendors, operators, and owners, scenarios could be designed to highlight potential cyberattack vectors and what operational technology defenses they should consider in the future. These elements could be deployed in well-developed training programs such as the annual DOE CyberForce Competition[42], annual DoD Cyber Flag cyber training exercise[43], INL's CyberStrike workshops[44], and other real-time cyber defense scenarios.

5.2. Real-Time Grid Integration Studies

DER devices include multiple adjustable power control functions, so grid operators have a difficult decision of selecting the best operating modes and settings for the DER. While using physical DER hardware to explore adjustable power control functions has been explored previously using a power hardware-in-the-loop environment and on distribution systems, this requires working with live power equipment[45,46]. Alternatively, using the DER Simulator with a real-time, distribution simulation could incorporate real communication protocols and networking behaviors at low voltages for DER grid-integration co-simulation or hardware-in-the loop (HIL) studies. This would provide valuable insight into new DER software deployments without the need for integrating power equipment into the simulations. Typically, the way to close the simulation loop is to regularly feed DER devices grid voltage and frequency from the power simulation and then inject the DER active and reactive power back into the power simulation. The current DER Simulator has these input and outputs through the Modbus interface. Future work is recommended to investigate if the simulator provides sufficient fidelity to perform real-time grid integration studies.

---

[42] Department of Energy's CyberForce Program, URL: https://cyberforcecompetition.com/, accessed 9/15/21.
[43] USCYBERCOM Public Affairs, "Media Advisory: Cyber Flag 21-2 winner announcement" URL: https://www.cybercom.mil/Media/News/Article/2671401/media-advisory-cyber-flag-21-2-winner-announcement/, accessed 9/15/21.
[44] INL, "CyberStrike Training: Practical Training for Energy Sector Owners and Operators," White Paper, https://inl.gov/wp-content/uploads/2021/07/21-50064_CyberstrikeFlyer.pdf, accessed 9/15/21.
[45] Summers, A.; Johnson, J.; Darbali-Zamora, R.; Hansen, C.; Anandan, J.; Showalter, C. A Comparison of DER Voltage Regulation Technologies Using Real-Time Simulations. Energies 2020, 13, 3562. https://doi.org/10.3390/en13143562
[46] Darbali-Zamora, R.; Johnson, J.; Summers, A.; Jones, C.B.; Hansen, C.; Showalter, C. State Estimation-Based Distributed Energy Resource Optimization for Distribution Voltage Regulation in Telemetry-Sparse Environments Using a Real-Time Digital Twin. Energies 2021, 14, 774. https://doi.org/10.3390/en14030774

6. CONCLUSION

This LDRD project investigated the technical challenges associated with creating DER honeypots and canaries. The objective of the work was to minimize the number of artifacts that exist in the virtualized environment in order to increase the likelihood that sophisticated adversaries would interact with the devices and expose their objectives, tactics, techniques, and procedures. The project created multiple DER emulators with local intrusion detection systems. The virtualized devices were deployed in the Distributed Energy Technologies Laboratory (DETL) alongside a cloned physical DER energy storage system and directly compared. While the initial SunSpec Modbus point maps were a perfect duplicate, the physical device and simulated DER drifted apart as the AC and DC DER simulations executed. In the future, these outstanding emulation artifacts will need to be removed to create realistic canaries and DER honeypots. This project also deployed Snort and Zeek IDSs along with an ELG stack for alerting as a network monitoring technology on the honeypot to store network data and warn stakeholders of anomalous or malicious traffic.

APPENDIX A. EXAMPLE DER HONEYPOT OUTPUT

2021-09-15 13:17:50.187216 D Running with arguments: Filename “” port 502
2021-09-15 13:17:50.187216 I Using the native der_config python dict, not a JSON file...
2021-09-15 13:17:50.188215 I Adding DER - sid = 1 addr = 40000
2021-09-15 13:17:50.214201 I Adding model 1
2021-09-15 13:17:50.214201 I Adding model 701
2021-09-15 13:17:50.214201 I Adding model 702
2021-09-15 13:17:50.214201 I Adding model 703
2021-09-15 13:17:50.214201 I Adding model 704
2021-09-15 13:17:50.214201 I Adding model 705
2021-09-15 13:17:50.215201 I Adding model 706
2021-09-15 13:17:50.215201 I Adding model 707
2021-09-15 13:17:50.215201 I Adding model 708
2021-09-15 13:17:50.215201 I Adding model 709
2021-09-15 13:17:50.215201 I Adding model 710
2021-09-15 13:17:50.216200 I Adding model 711
2021-09-15 13:17:50.216200 I Adding model 712
2021-09-15 13:17:50.216200 I Adding model 713
2021-09-15 13:17:50.216200 I Adding model 714
2021-09-15 13:17:50.216200 I Adding model 30003
2021-09-15 13:17:50.218199 I Adding model 30004
2021-09-15 13:17:50.338134 I DC Side: Time = 35400, Irradiance = 416.00, DCA = 10.53, DCW = 4928.89, DCWhInj = 34612.26
2021-09-15 13:17:50.811913 I AC Side: Frequency-Droop Controls configured with [{'DbOf': 0.03, 'DbUf': 0.03, 'KOf': 0.4, 'KUf': 0.4, 'RspTms': 6.0}, {'DbOf': 0.031, 'DbUf': 0.031, 'KOf': 0.41000000000000003, 'KUf': 0.41000000000000003, 'RspTms': 0.2}]
2021-09-15 13:17:50.812912 I AC Side: Volt-Var model configured with the following curves: [{'ActPt': 4, 'DeptRef': 1, 'Pri': 1, 'VRef': 100.0, 'VRefAuto': 100.0, 'VRefAutoEna': 0, 'VRefAutoTms': 500, 'RspTms': 0.6000000000000001, 'V_Pts': [92.0, 96.7, 103.0, 107.0], 'Var_Pts': [30.0, 0.0, 0.0, -30.0]}, {'ActPt': 4, 'DeptRef': 1, 'Pri': 1, 'VRef': 100.0, 'VRefAuto': 100.0, 'VRefAutoEna': 1, 'VRefAutoTms': 1000, 'RspTms': 0.2, 'V_Pts': [93.0, 95.7, 102.0, 106.0], 'Var_Pts': [30.0, 0.0, 0.0, -40.0]}, {'ActPt': 4, 'DeptRef': 1, 'Pri': 2, 'VRef': 100.0, 'VRefAuto': 100.0, 'VRefAutoEna': 0, 'VRefAutoTms': 500, 'RspTms': 0.4, 'V_Pts': [94.0, 95.7, 105.0, 108.0], 'Var_Pts': [20.0, 0.0, 0.0, -20.0]}]
2021-09-15 13:17:50.816910 I DC Side: Time = 35401, Irradiance = 414.00, DCA = 10.48, DCW = 4904.22, DCWhInj = 34612.91
2021-09-15 13:17:50.817910 I AC Side: PFs = 0.9998, 0.9998, 0.9998. PF = 0.98
2021-09-15 13:17:50.817910 I AC Side: Modbus Update: 4830.7 W, 20.0 var, 4831.5 VA, 41.10 A, PF = 0.980
2021-09-15 13:17:50.819909 D Creating DC Sim with <dersimx.der_dc.DERSimDC object at 0x0000024AF56772E8>
2021-09-15 13:17:50.820908 I Starting DC simulation
2021-09-15 13:17:50.820908 D Creating AC Sim with <dersimx.der_ac.DERSimAC object at 0x0000024AF3BDE7F0>
2021-09-15 13:17:50.824906 I DC Side: Time = 35401, Irradiance = 414.00, DCA = 10.48, DCW = 4904.22, DCWhInj = 34612.92
2021-09-15 13:17:50.824906 I Starting AC simulation
2021-09-15 13:17:50.824906 I AC Side: Var priority target (-2105.98 Var) moving power to 9775.73 W.
2021-09-15 13:17:50.824906 I AC Side: curtailment via VV, WV, VarSet or PF.
2021-09-15 13:17:50.829903 I AC Side: curtailment may push DC side off MPP. Power set to 9776.488345913534 W.
2021-09-15 13:17:50.829903 I AC Side: var target set to -2105.97.
2021-09-15 13:17:50.830903 I AC Side: PFs = 0.9161, 0.9161, 0.9161. PF = 0.9998337114748067
2021-09-15 13:17:50.830903 I AC Side: Modbus Update: 4830.7 W, -2106.0 var, 5272.9 VA, 41.10 A, PF = 1.000
2021-09-15 13:17:50.831902 I Starting websocket on localhost:8503
2021-09-15 13:17:50.832902 I Simulator starting on 0.0.0.0:502
2021-09-15 13:17:51.829980 I DC Side: Time = 35402, Irradiance = 413.00, DCA = 10.46, DCW = 4891.88, DCWhInj = 34614.29
2021-09-15 13:17:52.334219 I AC Side: Var priority target (-2298.38 Var) moving power to 9732.29 W.
2021-09-15 13:17:52.334219 I AC Side: curtailment via VV, WV, VarSet or PF.
2021-09-15 13:17:52.338217 I AC Side: curtailment may push DC side off MPP. Power set to 9733.031979355026 W.
2021-09-15 13:17:52.338217 I AC Side: var target set to -2298.36.
2021-09-15 13:17:52.339216 I AC Side: PFs = 0.9025, 0.9025, 0.9025. PF = 0.9161363776453582
2021-09-15 13:17:52.340216 I AC Side: Modbus Update: 4818.5 W, -2298.4 var, 5339.2 VA, 41.10 A, PF = 0.916
2021-09-15 13:17:52.835970 I DC Side: Time = 35403, Irradiance = 412.00, DCA = 10.43, DCW = 4879.54, DCWhInj = 34615.65
2021-09-15 13:17:53.841001 I DC Side: Time = 35404, Irradiance = 411.00, DCA = 10.40, DCW = 4867.21, DCWhInj = 34617.01
2021-09-15 13:17:53.843000 I AC Side: Var priority target (-2327.31 Var) moving power to 9725.41 W.
2021-09-15 13:17:53.844000 I AC Side: curtailment via VV, WV, VarSet or PF.
2021-09-15 13:17:53.847997 I AC Side: curtailment may push DC side off MPP. Power set to 9726.342599249158 W.
2021-09-15 13:17:53.847997 I AC Side: var target set to -2327.30.
2021-09-15 13:17:53.848997 I AC Side: PFs = 0.8993, 0.8993, 0.8993. PF = 0.9024724938321005
2021-09-15 13:17:53.848997 I AC Side: Modbus Update: 4794.2 W, -2327.3 var, 5330.8 VA, 41.10 A, PF = 0.902
2021-09-15 13:17:54.849007 I DC Side: Time = 35405, Irradiance = 410.00, DCA = 10.38, DCW = 4854.87, DCWhInj = 34618.37
2021-09-15 13:17:55.352246 I AC Side: Var priority target (-2323.63 Var) moving power to 9726.29 W.
2021-09-15 13:17:55.352246 I AC Side: curtailment via VV, WV, VarSet or PF.
2021-09-15 13:17:55.357244 I AC Side: curtailment may push DC side off MPP. Power set to 9726.699735831442 W.
2021-09-15 13:17:55.357244 I AC Side: var target set to -2323.63.
2021-09-15 13:17:55.358243 I AC Side: PFs = 0.899, 0.899, 0.899. PF = 0.8993437814969738
2021-09-15 13:17:55.358243 I AC Side: Modbus Update: 4782.0 W, -2323.6 var, 5319.0 VA, 41.10 A, PF = 0.899
2021-09-15 13:17:55.854020 I DC Side: Time = 35406, Irradiance = 408.00, DCA = 10.32, DCW = 4830.19, DCWhInj = 34619.72

APPENDIX B. SUNSPEC MODEL DEFINITIONS

Custom SunSpec Modbus models were created to transfer data between the DER simulator and the websocket visualization tool. The first model, 30003, included the I-V and P-V curves for the DC PV simulation, the model and a screenshot of the SVP Dashboard of this model are shown in Appendix B.1. The second model, 30004, included writable parameters (time, grid voltage, grid frequency, etc.) to change the DER Simulator operating conditions, as shown in Appendix B.2. A screenshot of this model in the SVP Dashboard is in Figure 26.

B.1. SunSpec Model 30003


```
{
  "group": {
    "name": "DerSimIv",
    "label": "DER Simulation IV curve.",
    "desc": "Various internal data for the DER Simulation.",
    "points": [
      {
        "desc": "Model identifier",
        "label": "Model ID",
        "mandatory": "M",
        "name": "ID",
        "size": 1,
        "static": "S",
        "type": "uint16",
        "value": 30003
      },
      {
        "desc": "Model length.",
        "label": "Model Length",
        "mandatory": "M",
        "name": "L",
        "size": 1,
        "static": "S",
        "type": "uint16"
      },
      {
        "desc": "Number of points in the IV curve.",
        "label": "IV length",
        "name": "IvLen",
        "size": 1,
        "static": "D",
        "type": "count"
      }
    ],
    "type": "group",
    "groups": [
      {
        "comments": [
          "IV Curve Points"
        ],
        "count": "IvLen",
        "name": "Iv",
        "points": [
          {
            "label": "Power",
            "desc": "Power (Watts)",
            "name": "P",
            "size": 2,
            "type": "float32"
          },
          {
            "label": "Current",
            "desc": "Current (Amperes)",
            "name": "I",
            "size": 2,
            "type": "float32"
          },
```

```
                {
                  "label": "Voltage",
                  "desc": "Voltage (Volts)",
                  "name": "V",
                  "size": 2,
                  "type": "float32"
                }
              ],
              "type": "group"
            }
          ]
        },
        "id": 30003
      }
```


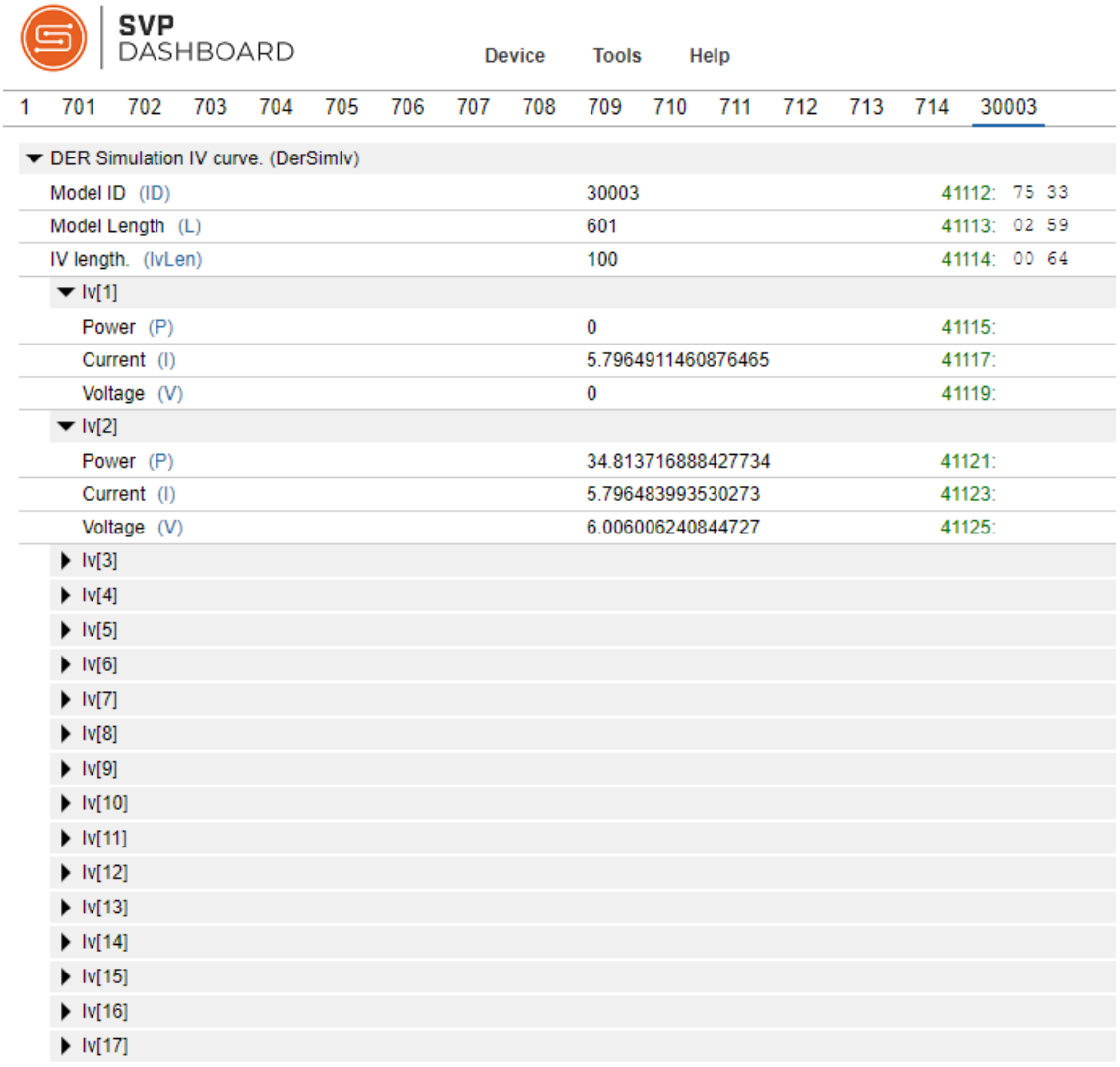

Figure 25: Model 30003 in the SunSpec SVP Dashboard.

B.2. SunSpec Model 30004


```
{
  "group": {
    "name": "DerSimControls",
    "label": "DER Simulation Controls.",
    "desc": "Configuration parameters for the DER device simulator.",
```

```
"points": [
    {
      "desc": "Model identifier",
      "label": "Model ID",
      "mandatory": "M",
      "name": "ID",
      "size": 1,
      "static": "S",
      "type": "uint16",
      "value": 30004
    },
    {
      "desc": "Model length.",
      "label": "Model Length",
      "mandatory": "M",
      "name": "L",
      "size": 1,
      "static": "S",
      "type": "uint16"
    },
    {
      "label": "Time offset",
      "desc": "Time offset into the simulation, of the format 'HH:MM:SS'",
      "name": "Time",
      "type": "string",
      "size": 10
    },
    {
      "name": "Temperature",
      "desc": "Ambient outdoor temperature (Celsius)",
      "type": "float32",
      "size": 2
    },
    {
      "name": "GridModelSource",
      "desc": "The data source for the grid model. One of: 'csv', 'const'",
      "type": "string",
      "size": 32
    },
    {
      "name": "IrradianceModelSource",
      "desc": "The data source for the irradiance model. One of: 'csv', 'const'",
      "type": "string",
      "size": 32
    },
    {
      "name": "Irradiance",
      "desc": "The irradiance on the DER device (W/m^2), for the 'const' irradiance model",
      "type": "float32",
      "size": 2
    },
    {
      "name": "GridVoltageA",
      "desc": "Voltage (W) of the first phase, for the 'const' grid model",
      "type": "float32",
      "size": 2
    },
    {
      "name": "GridVoltageB",
      "desc": "Voltage (W) of the second phase, for the 'const' grid model",
      "type": "float32",
      "size": 2
    },
    {
      "name": "GridVoltageC",
      "desc": "Voltage (W) of the third phase, for the 'const' grid model",
      "type": "float32",
      "size": 2
    },
    {
      "name": "GridFrequency",
      "desc": "Grid frequency, for the 'const' grid model",
```

```
            "type": "float32",
            "size": 2
          }
        ],
        "type": "group"
    },
    "id": 30004
}
```

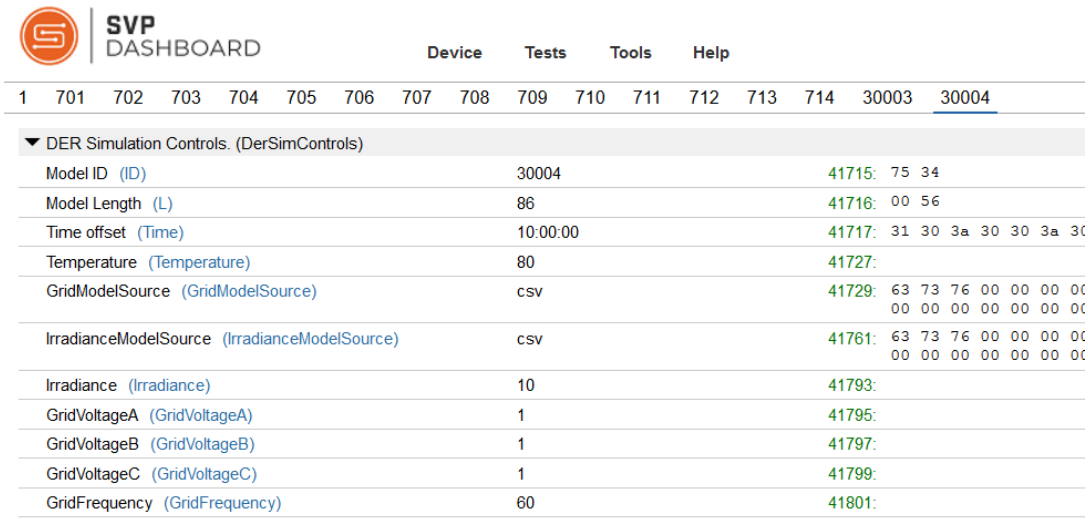

Figure 26: Model 30004 in the SunSpec SVP Dashboard.

APPENDIX C. CYBER MONITORING SOFTWARE INSTALLATION

C.1. Zeek Installation

```
$ sudo apt-get install build-essential flex bison
$ wget https://www.tcpdump.org/release/libpcap-1.9.1.tar.gz
$ tar xzvf libpcap-1.9.1.tar.gz
$ cd libpcap-1.9.1/
$ ./configure
$ make
$ sudo make install
$ cd ..
$ sudo apt-get install bro broctl
```

Edit the `/etc/bro/network.cfg` to reflect the appropriate networks to monitor and then issue the following command to start Zeek.

```
$ make
```

C.2. Snort Installation

```
$ sudo apt-get install snort
```

When prompted, enter the correct network that will be monitored by Snort. Once complete, issue the following commands to install the OT specific Snort signature rules for Modbus and DNP3.

```
$ sudo apt-get install git
$ sudo git clone https://github.com/digitalbond/Quickdraw-Snort.git
$ cp *.rules Quickdraw-Snort/
$ cp Quickdraw-Snort/* /etc/snort/rules/
```

Once complete, the next step will be to modify `/etc/snort/snort.conf` to include an entry for the `modbus.rules` and `dnp3.rules` files and then restart snort with the following command:

```
$ sudo /etc/init.d/snort restart
```

C.3. Filebeat Installation

```
$ sudo apt-get install git -y
$ git clone https://github.com/josh-thurston/easyBEATS.git
$ sudo chmod 755 easyBEATS
$ cd easyBEATS/

# FIRST NEED TO EDIT EASYBEATS TO SET HOME VARIABLE TO PARENT DIRECTORY OF easyBEATS/

$ bash easyBEATS

$ cd /etc/filebeat/modules.d
$ mv system.yml.disabled system.yml
# EDIT /etc/filebeat/filebeat.yml to point to correct IP Address for logstash as an output (localhost port 5044) and comment out all other outputs. Do the same for /etc/metricbeat/metricbeat.yml

$ sudo systemctl restart filebeat.service
```

C.4. Logstash Installation

```
$ apt-get install apt-transport-https jruby make -y
$ apt-get install texinfo build-essential ant git -y

$ update-alternative --config java
```

```
# choose java 8

# compile arm-linux jffi library
$ git clone https://github.com/jnr/jffi
$ cd jffi/
$ ant jar

# download and install logstash
$ wget https://artifacts.elastic.co/downloads/logstash/logstash-7.3.2.deb
$ dpkg -i --force-all logstash-7.3.2.deb

# replace libjffi
$ mv /usr/share/logstash/vendor/jruby/lib/jni/arm-Linux/libjffi-1.2.so
/usr/share/logstash/vendor/jruby/lib/jni/arm-Linux/libjffi-1.2.so.old
$ cp ../jffi/build/jni/libjffi-1.2.so /usr/share/logstash/vendor/jruby/lib/jni/arm-
Linux/libjffi-1.2.so

# wget
https://gist.githubusercontent.com/alexalouit/a857a6de10dfdaf7485f7c0cccadb98c/raw/06a
2409df3eba5054d7266a8227b991a87837407/fix.sh

$ wget
https://gist.githubusercontent.com/gwsales/5a27e6282063f902014d851247c5f448/raw/9a1721
8b5938b61eb70ba5806d4edac3dcc6dc80/fix.sh
sh fix.sh

$ echo "--add-opens java.base/java.io=ALL-UNNAMED\n--add-
opens=java.base/java.security=ALL-UNNAMED\n--add-opens
java.base/java.security.cert=ALL-UNNAMED\n--add-opens java.base/java.util=ALL-
UNNAMED\n--add-opens java.base/java.util.zip=ALL-UNNAMED\n--add-opens
java.base/java.util.regex=ALL-UNNAMED\n--add-opens java.base/sun.nio.ch=ALL-UNNAMED\n-
-add-opens java.base/java.io=ALL-UNNAMED\n--add-opens java.base/java.lang=ALL-
UNNAMED\n--add-opens java.base/java.lang.reflect=ALL-UNNAMED\n--add-opens
java.base/java.net=ALL-UNNAMED\n--add-opens java.base/javax.crypto=ALL-UNNAMED" >>
/etc/logstash/jvm.options

# EDIT SAMPLE LOGSTASH CONFIG WITH CORRECT IP
$ cp /etc/logstash/logstash-sample.conf /etc/logstash/conf.d/
$ systemctl enable logstash
$ systemctl start logstash
```

C.5. Logstash Installation

```
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
$ sudo apt-get install apt-transport-https
$ echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a
/etc/apt/sources.list.d/elastic-7.x.list
$ cd ~
$ wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.3.2-no-
jdk-amd64.deb

# ***Note: You may see some download errors here
$ export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-armhf
# modify /var/lib/dpkg/status elasticsearch entry and remove libc6 dependency
# edit /etc/elasticsearcg/jvm.options to include 512m of memory
$ sudo dpkg -i --force-all --ignore-depends=lib6 elasticsearch-7.3.2-no-jdk-amd64.deb
# Change the permissions for the /etc/elasticsearch folder for easier access
$ sudo chmod g+w /etc/elasticsearch
$ sudo chmod 755 -R /etc/elasticsearch
$ sudo chown -R elasticsearch:elasticsearch /etc/elasticsearch
```

open `/var/lib/dpkg/status` and remove the Elasticsearch dependency of "`libc6`" on the "`Depends:`" line of the "`Package: elasticsearch`" section while keeping all other dependencies.

Edit the /etc/default/elasticsearch file and add or uncomment the following line:

```
$ JAVA_HOME= /usr/lib/jvm/java-11-openjdk-armhf
```

Edit /etc/elasticsearch/elasticsearch.yml with the following variables set:

```
cluster.name: any_unique_name
node.name: any_unique_node_name
network.host: bump_in_the_wire_IP_Address
http.port: 9200
xpack.ml.enabled: false
node.master: true
node.data: true
node.ingest: true
discovery.type: single-node
bootstrap.system_call_filter: false
```

Finally, restart the elasticsearch, logstash, filebeat, and metricbeat services.

```
$ sudo systemctl restart elasticsearch
$ sudo systemctl restart logstash
$ sudo systemctl restart filebeat
$ sudo systemctl restart metricbeat
```

Depending on memory resources available, the /etc/logstash/jvm.options and /etc/elasticsearch/jvm.options file may need to be edited so that both elasticsearch and logstash can run at the same time. Reducing -Xms=1g and -Xmx=1g to -Xms=512m and -Xmx=512m settings should fix any issues errors that appear when restarting the services if the system has run out of memory.

C.6. Grafana Installation

```
$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
$ echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a
/etc/apt/sources.list.d/grafana.list
$ sudo apt-get update
$ sudo apt-get install -y grafana
$ sudo /bin/systemctl enable grafana-server
$ sudo /bin/systemctl start grafana-server
```

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Charles Hanley	08810	cjhanle@sandia.gov
Summer Ferreira	08812	srferre@sandia.gov
Brian Gaines	09366	bgaines@sandia.gov
Jay Johnson	08812	jjohns2@sandia.gov
Technical Library	01977	sanddocs@sandia.gov

Email—External

Name	Company Email Address	Company Name
Jeremiah Miller	jeremiah.miller@ee.doe.gov	U.S. Department of Energy
Guohui Yuan	guohui.yuan@ee.doe.gov	U.S. Department of Energy

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.