HELOT–Hunting Evil Life in Operational Technology

Syed Akailvi, Uddhav Gautam, Praveshika Bhandari, Hadi Rashid, Philip D. Huff, *Member, IEEE*, and Jan P. Springer, *Member, IEEE*

***Abstract*—Operational technology (OT) refers to the industrial counterpart of information technology (IT). OT encompasses technology, systems, and protocols used in industrial operations for controlling, monitoring, and operating of industrial systems. Unlike standard IT systems, OT systems, such as those used in the industry, usually cannot be taken off-line in the event of postmortem forensics investigations. To remedy this situation we present a software architecture and prototype realization that allows the continuous capture of events within OT systems, IT systems, and the interconnected network(s). Our architecture can be realized with existing technologies while also allowing for extension and customization in functionality as well as application to diverse domains. We outline two application cases: capturing of forensics artifacts from a live OT system and possible paths for automation to reduce the cognitive load for cybersecurity operators in combined IT/OT environments.**

***Index Terms*—Computer forensics, cybersecurity operations, operational technology, power-delivery systems.**

I. Introduction

OPERATIONAL technology (OT) encompasses any digital computing system that is used for controlling, monitoring, and operating of industrial systems such as manufacturing plants, automated assembly lines, and service infrastructures like the electricity power grid. Industrial Control Systems (ICS) form a major segment of OT systems and are often used in real-time applications like the routing and switching of power from electrical substations to various parts of the electrical grid. These critical systems must be extremely reliable and must also constantly operate without interruptions or shutdowns.

One characteristic of current OT systems is the lack of comprehensive and extensible functionality compared to systems in information technology (IT). OT systems are application-domain specific unlike IT systems where general purpose applicability is more important. This lack of general purpose capabilities allows the design of OT systems with limited capabilities in computational power, storage, and networking, to name just a few. Such a limited architecture also means that OT systems are more robust and reliable than general IT systems. However, the downside of this approach is that OT systems are often implemented using highly proprietary technologies and communication protocols, making it difficult to access and interface OT systems with general purpose IT systems.

These characteristics of OT systems present particular challenges in the area of cybersecurity operations in the context of operational technology. Due to the limited functionality, OT systems are particularly susceptible to cybersecurity threats by remote and on-site actors. Live forensics is a valuable tool in combating cybersecurity threats, especially in OT systems where post-mortem forensics is often too little too late.

We present HELOT, which stands for *Hunting Evil Life in Operational Technology*, a software architecture and prototype realization that *allows capturing information for live forensics from interconnected OT and IT systems* as well as network traffic between these systems. Our architecture avoids the requirement of a shutdown in the event of necessary forensics operations and is based on Google Rapid Response (GRR) [13], a highly scalable and extensible live forensics tool. Unfortunately, GRR, in its currently available form as open-source software, does not support live forensics on OT systems. We remedied this shortcoming by extending GRR to allow access to and retrieval from OT systems. Additionally, we added the ability to continuously capture events from OT systems, connected IT systems, and the interconnected network(s). The results are stored in a database-like setup using search-engine based database technologies such as ElasticSearch/Logstash/Kibana (ELK) [18]. This allows, in addition to storing large volumes of raw data for a multitude of system events, the immediate analysis and transformation into additional formats as well as custom indexing, including storing previous searches and their results.

We also outline two immediate application cases. First, we show the results of our prototype experiments in capturing of forensics artifacts from a combined live OT and IT system based on deployment configurations used in the power-delivery industry. Second, we discuss possible paths for automation to reduce the cognitive load for cybersecurity operators in combined OT/IT environments.

II. Background

Historically, *Supervisory Control and Data Acquisition* (SCADA) systems have not been connected to wider networks

Manuscript received 9 January 2022; revised 24 May 2022 and 18 September 2022; accepted 5 November 2022. Date of publication 15 November 2022; date of current version 21 June 2023. This work was supported by the U.S. Department of Energy under Award DEOE0000779. Paper no. TSG-00035-2022. *(Corresponding author: Jan P. Springer.)*

The authors are with the Department of Computer Science and the Emerging Analytics Center, UA Little Rock, Little Rock, AR 72224 USA (e-mail: jpspringer@ualr.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSG.2022.3222261.

Digital Object Identifier 10.1109/TSG.2022.3222261

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

and used domain-specific communication protocols, hence resulting in some measure of security through obscurity and isolation. As technology has advanced to greater efficiency, sophistication, and capabilities, the aforementioned isolation steadily decreased between OT systems and IT systems. Factors like new and open standards, increased connectivity to conventional enterprise information systems, and the encapsulation of legacy control protocols within conventional networking protocols (e.g., DNP3 over TCP/IP) tend to increase attack exposure. While these trends have led to general improvements in functionality and productivity, they also raised concerns that industrial control systems (ICS) are now vulnerable to remote attacks with an increasing potential for catastrophic consequences if critical infrastructure systems are compromised.

Early work by Cheung et al. [12] attempted to construct models that characterize the expected behavior of a system and detect attacks that cause the system to behave outside a model's prediction. By specifying the expected communication patterns among network components it is possible to detect attacks that violate these patterns. Porras and Neumann [41] deployed an implementation of a multi-algorithm detection appliance based on EMERALD. This appliance was integrated into a control-system test bed and experimental results showed that a model-based intrusion-detection system (IDS) is a promising approach for monitoring process-control networks.

Yang et al. [54] present two basic intrusion detection approaches, signature detection and anomaly detection, and applied these techniques to monitor critical-process systems for anomalous intrusion detection. Yang et al. [54] used an auto-associative kernel regression model coupled with a statistical test and applied this to a simulated SCADA system. Their results showed that the method can be generally used to detect a variety of common attacks and allows generalization of existing techniques to identify inconsistent as well as consistent states of a given system. Subsequent research refined on these *flow-based IDS* approaches by specifically targeting SCADA systems [5] with the goal of creating flow-level signatures of attacks and to provide more information about the anomalous traffic. Modeling anomalous traffic is made challenging by attackers who are motivated to conceal their activities. Our approach with HELOT is to extend live monitoring and intrusion detection with additional GRR components to also collect and analyze data retrieved from OT systems.

Network monitoring and analysis techniques were further extended to include monitoring of low-level OT instrumentation devices when Goldenberg and Wool [23] designed a *model-based IDS* for SCADA systems. Goldenberg and Wool [23]'s proposed approach functioned by flagging anomalies in the Modbus system [35]. This IDS successfully detected real anomalies during the troubleshooting of a human-machine interface (HMI) system and helped to identify a programmable logic controller (PLC) that was incorrectly configured. Yang et al. [55] designed a multi-layer cybersecurity framework for protecting SCADA systems against intrusions. Their solution was able to mitigate different cyberattacks. A SCADA-IDS with a white list and a behavior-based protocol was utilized to detect both known and unknown cyberattacks in the network. Sayegh et al. [48] introduced a SCADA specific IDS to detect attacks based on traffic behavior and frequent network-traffic patterns. Time correlation between the packets was estimated to identify whether the activity is normal or anomalous.

Maglaras et al. [33] developed a one-class support-vector machine (OCSVM) technique to detect intrusions in SCADA systems using a statistical algorithm to improve the performance of the OCSVM module. Additionally, a *k*-means clustering algorithm allows classifying of alarms into categories. The main drawback of this method were mistakes with respect to the origin of the intrusion, which may introduce severe costs for the intrusion-detection system. Almalawi et al. [3] presented a new IDS approach to detect tailored attacks in SCADA networks by identifying normal and critical states of a given system using a data-driven clustering technique that includes automatic state identification, automatic detection-rule extraction, reduction of high false positive rate, and measures to evaluate criticality. However, the suggested system did not address the frequency of changes in the SCADA system specification. Ponomarev and Atkison [40] developed an approach to detect intrusions into networks based on telemetry analysis where internal and external traffic were differentiated in the server-client separation stages. The telemetry based IDS monitored all packets in the ICS network thereby detecting anomalies in the network traffic.

Li et al. [32] introduced a Dirichlet distribution based scheme for detecting opportunistic attacks in a smart grid cyberphysical system. Data was collected from an IEEE 39 bus [4] power system in a power world simulator and a three-tier hierarchical control framework was employed. Li et al. [32] demonstrated that a potential class of opportunistic attackers in smart grids can adapt their attack probabilities according to the dynamic system's noise and the proposed scheme could effectively detect and identify the opportunistic attackers. In addition, the scheme has been shown to accommodate occasional system faults, achieving a high detection rate over long observation windows.

McElwee et al. [34] provide a machine-learning case study for the initial triage of security alerts to help reduce manual burden placed on cybersecurity analysts. Results demonstrated that the accuracy of a deep neural-network classifier was very high and was able to determine the heuristics that cybersecurity analysts used in their reviews. Demonstrations and interviews with analysts showed that the prototype was able to quickly categorize security alerts into meaningful categories, provided fast queries of the alerts, and saved time in generating reports. The system allows to pull necessary information to generate indications of compromise reports in a STIX/TAXII format [37] to be shared across government agencies.

HaddadPajouh et al. [28] utilized a recurrent neural network to analyze ARM-based IoT applications' execution operation codes (OpCodes). Santos et al. [46] proposed a method to classify variants of known malware families based on OpCodes' frequency. Runwal et al. [45] built a similarity graph based on an application's OpCodes to detect metamorphic malware. O'Kane et al. [36] utilized a support-vector machine (SVM) classifier, a type of deep learning algorithm that performs

supervised learning for classification or regression of data groups, and *n*-gram techniques, an approach that investigates the structure of a program using bytes, characters, or text strings, to evaluate OpCodes and identify optimum features for malware detection. Santos et al. [47] proposed a method based on frequency of appearance of OpCode sequences under different machine-learning classifiers and reportedly obtained an accuracy rate of over 96 %. Ding et al. [16] utilized the *n*-gram technique as well to extract an application's OpCode sequence and the findings indicated an accuracy rate of 99.88 %. An accuracy of 99 % was reported in their 70 % (training) - 30 % (testing) data-set split evaluation.

Most industrial plants today use networked servers as process historians for storing process data as well as other business and process relevant information. The adoption of Ethernet-based networks and TCP/IP for process control networks as well as wireless technologies such as IEEE 802.x and Bluetooth has further reduced the isolation of SCADA networks [57]. In recent years, SCADA systems have undergone a series of changes that may increase the amount and variety of risks to which they are exposed. For example, increased connectivity permits remote control over the public Internet and incorporation of general-purpose tools may introduce already known vulnerabilities. Additionally, SCADA systems also perform vital functions in critical national infrastructures such as electric power, oil and gas distribution, water distribution and waste-water treatment, or transportation and logistics systems. These systems are also at the core of many health-care devices, military systems, and transportation-management systems. The disruption of these critical infrastructure systems would have a significant impact on public health and safety and may lead to large-scale economic losses [8]. As a consequence there is an increasing interest in the security and forensic research community on SCADA systems. This is mostly due to the heightened focus of governments worldwide on protecting their critical infrastructures, including SCADA systems [2].

Ensuring the security of SCADA systems is a very important part of smart-grid security [53]. Because attacks are continuously targeting industrial systems, focus is shifting to planning and developing new techniques that will adapt to SCADA environments and their protocols [44]. Post-mortem analysis tools still require to shut down such system to inspect the contents of storage devices and to identify proofs of interest regarding attacks. In that process, there is a possibility for breaking of network connections and disconnection of encrypted disks, which cause major loss of potential evidence and may also cause disruption of important system tasks [10]. Computer forensics also rely on log events for identification that any security incident happened and ICS often lack the components required to conduct forensic analysis of relevant logging or storage formats. The huge amounts of recorded events in a SCADA system, in combination with a lack of standard logging formats, complicates analysts' tasks [30]. There is an observable rise in both the number and the complexity of cases that require analysts' attention. Most recent forensic tools are insufficient because they are designed to run on a single computing device where the investigator can issue queries over the accumulated evidence. For smaller forensic targets these devices work best but when larger forensic targets are under investigation these tools are too slow to provide acceptable turnaround times [43]. Existing commercial forensics support systems such as Encase [38], FTK [22], UFED [9], [26], or Diffy [15] are not capable of performing ongoing live forensics for OT environments. The innovations proposed in our work would be equally applicable to these commercial tools. However, proprietary systems are by their very nature not open for (easy) extension or closer inspection necessary for research.

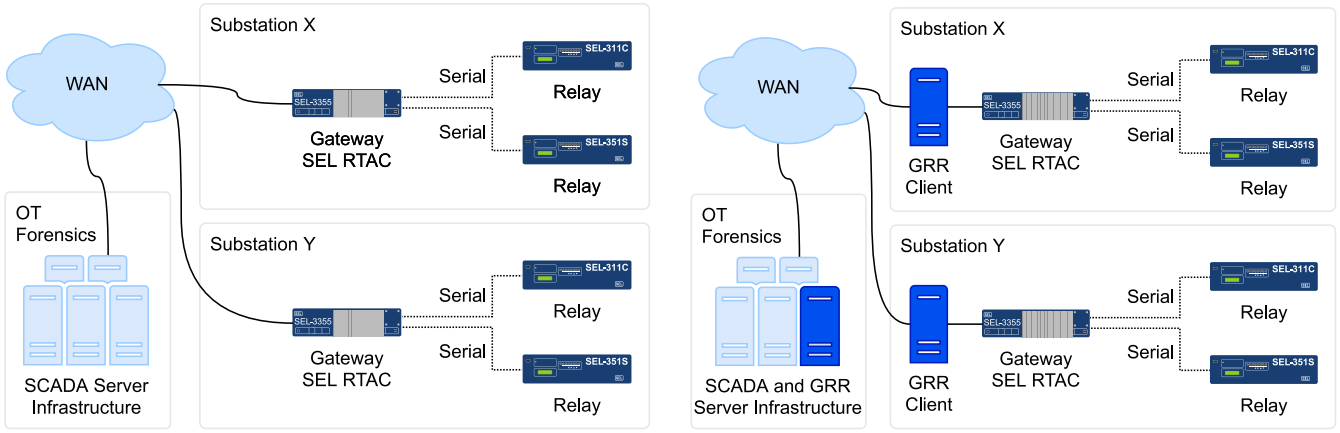
HELOT has been developed to allow combining the strengths of various approaches previously described. We have been working on HELOT in the context of live forensics for industrial OT systems combining OT devices with connected IT system. We were able to integrate GRR with an ELK based storage component using additional custom scripts and plugins for data gathering with GRR on OT devices. In addition, the visualization of data collected by GRR has been successfully used in Kibana, the visualization component of ELK. We are utilizing various machine-learning algorithms to process heterogeneous data to classify the behavior of a system, including (possible) attacks. The visualization part also allows to detect potential threats in related network traffic and trigger control signals to stop or counteract these attacks.

III. Architecture

A. Overview

HELOT operates in the context of modern computer systems that incorporate elements of IT and OT at multiple levels of interaction. At the outermost level are wide-area networks (WAN) controlled and managed by Internet service providers. At the next level are internal or enterprise networks, which are established and administered by companies, institutions, or individuals in their homes. Communication between these two levels is usually restricted in the sense that only certain endpoint systems in the internal networks are reachable from the WAN. Additionally, internal networks may also feature designated hosts for handling any traffic from the WAN that is not explicitly allowed for distribution in internal networks. These designated hosts are typically referred to as a *demilitarized zone* (DMZ). Similarly, OT systems are connected to internal networks by dedicated hosts as well through dedicated *human-machine interface* (HMI) systems. However, OT systems themselves exhibit a network structure as well, connecting the various parts for controlling and actuating physical devices. Network traffic between an OT system's internal network and the enterprise network is severely restricted. This general setup in mixed OT/IT system networks guaranties a certain amount of safety against malicious or accidental actions on an OT system. Unfortunately, it also provides for challenges with respect to information gathering from the combined OT/IT system.

Analyzing various workflows described by previous work in this area (cf. Section II *Background*) shows that these workflows generally consists of three main steps. First, the collection of data from various systems located in the internal networks as well as from OT related networks. Second, storage



(a) Schematic diagram of existing OT system setups.

(b) Schematic diagram of an *ideal* OT system setup (using GRR).

Fig. 1. Schematic diagrams of (a) existing and (b) *ideal* OT system setups. While a standard OT system setup directly connects to the larger network using a gateway, our *ideal* OT system setup introduces a GRR client between the OT system gateway and the larger network as well as GRR server infrastructure for scalable communication between the centralized information store and the large amount of distributed OT systems.

of collected data and possible transformation to formats advantageous for specific use cases (e.g., frequency analysis, signal analysis). Third, actual data analysis based on domain-specific requirements (e.g., real-time observation of systems behavior, reports for management).

Figure 1 shows schematic diagrams of existing OT system setups (cf. Figure 1a) and our *ideal* (cf. Figure 1b) OT system setup. While a standard OT system setup directly connects to the larger network using a gateway, our *ideal* OT system setup introduces a GRR client on the Local Area Network (LAN) with a mirrored network for raw packet visibility of all internal network traffic. In addition, our *ideal* architecture introduces GRR server in the control-system data center for scalable communication and storage to accommodate large amounts of aggregate data from distributed OT systems. Our architecture deploys two distinct pipelines for data ingestion. One data-ingestion pipeline is configured to collect data from distributed OT system devices over low-priority communication channels whereby forensic capture communication cannot disrupt SCADA processes. The other data-ingestion pipeline collects data from selected IT devices in the control-center systems. The two ingestion pipelines together enable effective data collection from all levels of the combined OT/IT systems including traffic in the distributed system networks. Additionally, control-center systems are monitored at a much higher frequency and can be used to prompt additional procedures or data collection by operational and maintenance personnel.

Our architecture allows a complete timeline of events leading to a power-system cyberattack. Existing architectures may only enable the reconstruction of events occurring on IT systems or segmented network segments in the control center. However, adversaries have no such restrictions and with credentials to OT devices the adversary may complete the attack without IT system interaction. For example, access to a SCADA system may allow tampering with control signals, manipulation of reliability thresholds, or modification of metering data, all of which can occur without IT system visibility. Our architecture overlays the GRR system on the existing SCADA system to address the gap in forensic purview but does not disrupt existing reliability network traffic and processes. First, power system control centers often have traditional data-center server pools, in which the central GRR server can scale to meet the forensic capture needs of the SCADA system. Second, the distributed GRR components passively collect forensic data and forward it to the GRR server over low-priority connections. Finally, GRR can be installed on control center OT components with minimal storage or CPU usage impact.

Deploying our architecture presents challenges in adding a system component to the distributed LAN setup. For example, changes to the network configuration at a substation require significant effort and time in planning, testing, and deployment. However, many distributed system networks exist in electric substations with uniform designs, making wide scale changes easier to test and deploy. Also, GRR components installed on control-center OT devices provide improved forensic visibility by allowing SCADA event-data collection from the control center to the distributed system.

B. Data Collection

All events of interest fall into three main categories: OT system events, IT system events, and internal network traffic. These events are tracked in their respective systems by identifying and collecting important data generated by the system. The data is recorded in multiple log files generated by specialized data collection and logging devices, distributed throughout the OT/IT system(s) at various critical points.

There are many different types of data that can be collected from various points in a system depending on the level of access that is available and the authentication system in place. The lowest level of system devices accessible for data extraction are intelligent electronic devices (IEDs), which can be a PLC or other automated controller programmed to send and receive various control and monitoring signals directly to the instrumentation devices that control a process. Additionally, access to data historians, i.e., specialized devices with the

purpose of continuously collecting operational data and log files, allows the collection of historical log data.

The collection of data at various levels of abstraction throughout the system requires accessing and querying assets using proprietary protocols. Safety, security, and availability requirements are a high priority for OT devices, which in turn means that they are less available and accessible for convenient or continuous data collection than devices in IT system. On the other hand, IT system environments have relatively less stringent constraints when it comes to accessibility and availability for data collection. Furthermore, IT systems also exhibit far more robust and effective solutions and techniques for data collection. There is a noticeable disparity when compared to the performance and capabilities available in the more unique OT environment.

Finally, network traffic is relatively simple to capture from within IT systems. Within OT systems this may require additional capabilities in the vendor specific implementations for access and control by HMIs. However, being able to correlate events in IT and OT systems with additional information about network traffic will be very useful for real-time as well post-event forensics analysis.

C. Data Storage

Collected data needs to be stored for later analysis. Traditionally, this would be achieved with a database setup, possibly centralized in nature. However, because it is necessary to collect a multitude of different types of data from a multitude of separate data sources at different frequencies, standard database technology will potentially become a bottleneck. Alternatively, newer database technology can be used that will not only change priorities from centralized service towards distributed high-performance access but also emphasizes indexing and search capability over preconceived structural layouts. Our decision to base our infrastructure approach on GRR was also encouraged by its ability to support different data-store technologies. In our prototype realization, we use ElasticSearch [17] for storing collected data instead of a more standard setup based on SQL.

Standard database technologies based on SQL are very capable in storing data as well as providing different views on it. Modern implementations are also able to provide scalability for read and write access by using distributed implementations. However, the collected data from as disparate sources as OT and IT systems as well as network-traffic data requires already analysis when inserting and indexing the data into the data store. Later use of the indexed data then conforms more to patterns similar to using search engines because indexed log-file input does not require additional updates or deletion after processing. This means not only will it be important to store *raw data* and identify correlation but it will also be equally important to store *searches and their results* into the data store. This support will allow additional analysis and automation in a scalable way.

D. Data Analysis

The final stage in our architecture is concerned with supporting the analysis and further use of collected data. Our focus here is on two main aspects. The first aspect is concerned with support for investigation of current and historical data by human operators. The second focus is related to support for automation (e.g., pattern identification, anomaly detection, reporting). Both aspects are actually interdependent and cannot be completely separated. This stems from the fact that problem-domain specific tasks by operators may lend themselves to automation after phases of exploration. On the other hand, any automation requires some specification as well as the definition of goals worth achieving. This usually starts with supporting exploration by human operators, which brings us back to the initial aspect.

Current trends in analyzing complex and frequently updated data also require support through machine-analysis tools. These allow for various specific tasks in detecting anomalies in single systems or sets of systems, including network traffic between these systems. However, as we will describe later (cf. Sections IV-A *Implementation* and IV-B4 *Machine-Learning Based Adaptive Filtering*), machine-analysis tools also provide the potential of *learning* baseline behavior of arbitrary systems, provided sufficient data is available to describe valid state sets of baseline behavior. In addition, such an approach, in combination with the ability of storing *searches and their results*, will provide opportunities to provide adaptive interfaces for presenting and interacting with data-analysis results.

IV. Discussion

A. Implementation

Google Rapid Response (GRR) [27] is an incident response framework focused on remote live forensics. It is based on a server-client architecture allowing remote monitoring of a wide variety of devices over multiple network topologies. GRR is optimized for mass data collection and provides a readily usable system for efficiently collecting data from large amounts of systems distributed across geographically widespread regions and running on a variety of operating systems and architectures. GRR servers and clients can be fully deployed on most modern versions of Windows, Linux as well as selected OSX versions and mobile platforms. A general overview of GRR's infrastructure components is provided in Figure 2.

GRR leverages modern communication frameworks and protocols including Fleetspeak [21], a framework for communicating with a large fleet of machines with a focus on security monitoring and basic administrative use cases, Protocol Buffers [42], a language-neutral and platform-neutral extensible mechanism for serializing structured data, and additional advanced features such as bandwidth throttling to improve communication and load balancing for a wide variety of network-deployment scenarios. GRR's overall distributed communication processing is optimized for the collection of forensic artifacts in a variety of IT systems, environments, and use cases. GRR is mainly geared towards IT systems with ready to use features including local action on a client (*flows*), distributed actions on multiple clients (*hunts*), scheduled flows or hunts (usually, using cron jobs or similar techniques), and a variety of output plugins such as ElasticSearch, BigQuery [7],

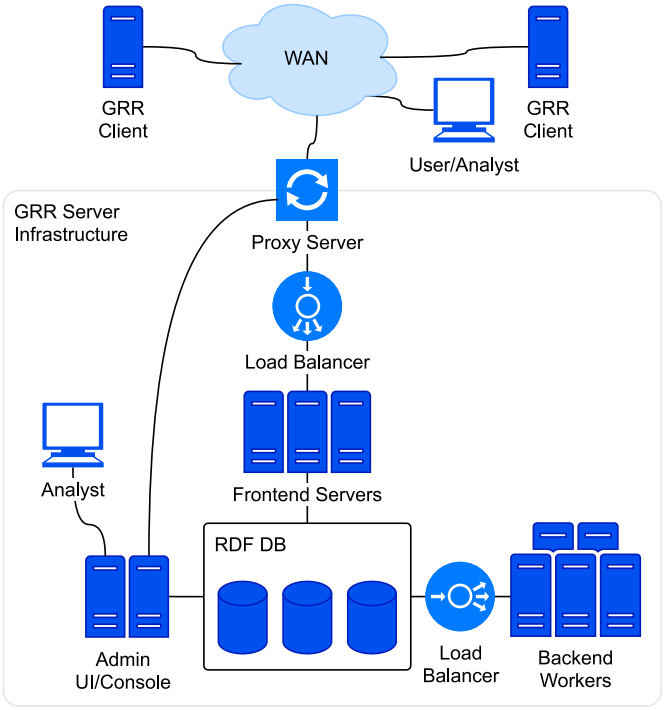


Fig. 2. General overview of the client-server infrastructure in a Google Rapid Response (GRR) setup.

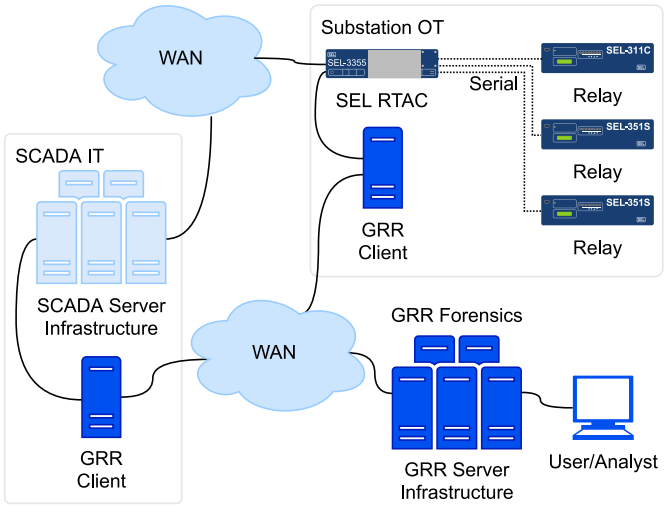


Fig. 3. Overview of an OT system integration into a Google Rapid Response (GRR) setup. Note that both the IT and OT components contain a local GRR client while the GRR server infrastructure is actually location independent and maybe even deployed on a cloud infrastructure.

Splunk format [51], or CSV format. Standard plugins exist to collect forensic artifacts, which include various types of logs, files, registry objects, and relevant metadata. These forensic artifacts are provided as artifact definitions using YAML descriptions and allow to easily define and configure new artifact types. Together with the ability to filter, quickly access, and view collected data and artifacts, this provides for a convenient way to display results on a Web-based user interface. The user interface also allows for structured selection and interaction with the entire fleet of machines or smaller subsets. The Web-based interface together with the built-in YAML artifacts specifications helps forensics IT experts with easy selection and configuration of several industry standard forensics and cybersecurity workflows.

All communication between a GRR server and its clients are encrypted, currently using AES256 encryption [31]. Clients periodically check for a valid server certificate to protect against a compromised server. A GRR server communicates with a client through a requests-response scheme. Each request by the server executes a client action on the remote client's system. This may be a primitive action, which generates results possibly combined into other complex operations by the server. Protection against *replay attacks*, a particularly problematic feature in remote-forensics systems, is established by using sequence-numbered time stamps as well as encrypted and signed payloads for each request-response interaction. GRR provides users with fine grained controls over operations that can then be automated to complete all types of acquisitions while also avoiding exposure of protected personal user data. This can be achieved with predefined tasks and by specifying assets as YAML artifacts that isolate the desired resources of interest. These interests are then categorized for further review by forensics operators or other authorized personnel depending on whether they are likely to contain protected or private information. A simple example of a custom protection or detection scheme that can be implemented using GRR is that of periodically hashing specific file-system directories that are critical to system operation and are not expected to change under normal operation. A client can be configured to automatically run at a preset time interval on a deployed remote system and send the resulting hash to the GRR server to be compared with a ground truth value and take necessary actions, if required. An overview of our development setup incorporating a GRR infrastructure is provided in Figure 3.

ELK stack is the collective name of three open-source projects: ElasticSearch, Logstash, and Kibana; developed, managed, and maintained by Elastic [17]. In our prototype implementation we were using version 7.9 of ELK. ElasticSearch is a full-text search and analysis engine, based on the Apache Lucene [24] search engine. Logstash is a data aggregator and processing pipeline for a multitude of source formats, providing transformations and enhancements as well as the ability of shipping processed data to various output destinations. Kibana is a visualization layer on top of ElasticSearch providing users with the ability to visually analyze and interact with their data. The latest addition to ELK stack, but not reflected in the acronym, is Beats, which provides lightweight agents on edge hosts to collect a multitude of data to be forwarded to Logstash for further processing or to ElasticSearch for direct indexing and storage. Together, these components are the most commonly used infrastructure elements for monitoring, troubleshooting, and securing IT environments, though there are many more use cases for deploying ELK stack such as business intelligence and Web analytics. Beats and Logstash are the main tools for data collection and processing, ElasticSearch provides indexing and storage of data, while Kibana adds a user interface for querying, visualizing, and interacting with data by human operators.

All of the components in the ELK stack are provided under open-source licenses.

In our prototype implementation, we collect log data from IT and OT systems with GRR clients as well as from network traffic using Filebeat [20], a lightweight log collector for ELK stack deployments. While log file collection on IT systems is based on pre-existing tools, modules, and techniques, our log file collection on OT systems consists of combinations of custom software plugins based on GRR plugin interfaces and is described in detail in Section IV-B *Results*. To allow the correlation of events from IT and OT systems, we also collect network traffic data between these systems as well as incoming and outgoing communication in the larger IT system networks. This network traffic is collected by similar custom software plugins using Filebeat and a ring-buffer technique plus configurable update intervals for sending the collected data to the ELK infrastructure in our prototype setup. Logstash is used for standard analysis and transformation (e.g., labeling of packet types, recognition of certain payloads such as binary dumps or plain text) for indexing and storage into an ElasticSearch instance.

As mentioned before, GRR by default does not have OT-system specific plugins. We therefore developed a set of plugins for GRR that allow data retrieval from OT systems. Through our access to the NCREPT facility at UA Fayetteville, we were able to access SEL RTAC and SEL 351S/311C systems through the vendor provided AcSELerator software. Unfortunately, this software only provides an interactive GUI mode as well as an interactive command-line mode but not a query API for automation in a scripting environment. We therefore developed our OT-system plugin as a thin wrapper around the interactive command line using the most general commands to minimize query frequency. The OT-system responses are initially processed in the GRR plugin instance and send for further event processing to its GRR server and finally to the ELK stack infrastructure. The OT-system plugin, while currently viable for SEL hardware, has been developed as a set of configurable plugin scripts for a GRR client that can be adapted for other industrial OT-system vendor hardware. Additional details of the plugin integration and the custom filter processing are provided in Section IV-B *Results*.

Our prototype development and implementation is heavily based on virtualization technology. While the goal of our development always was to be deployed on existing hardware setups, such as a local energy provider's network of sub-stations, we needed a *safe playground* for experimentation and initial testing. Originally, our intention was to use the NCREPT facility at UA Fayetteville but even this proved not to be safe enough for other users to not be disrupted by our development testing. We therefore developed a set of configurations using virtualization technology to encapsulate specific hosts and their behavior as well as complete networks of such hosts and their network-traffic patterns. All of these configurations are semi-automated in the sense that scripted base configurations are used to create variations of network and host deployments based on configuration input. Only certain host types needed to be manually setup and provided as virtual machines (or, sometimes, as containers). Once these initial elements exist, they can be easily reused or reconfigured. For smaller experimental deployments such setups can be run on a single modern medium-sized workstation with configurations of virtual-machine hosts ranging in the hundreds but less than a thousand with a negligible impact on performance, mostly because almost all communication will be constrained by memory transfer rates and not by actual network transfer rates. Larger setups, i.e., configurations with more than a thousand virtual-machine hosts, have been tested on a dedicated research cluster maintained by the Emerging Analytics Center at UA Little Rock. Here, while the bandwidth for network interconnects was more constrained, actual network sizes and communication patterns approached scenarios similar to real-world configurations and were spread out across multiple nodes in the cluster. Using this *simulation approach*, we were able to make certain that our implementation would not be disruptive to potential users in actual system deployments. We were also able to confirm this at the NCREPT facility at UA Fayetteville (cf. Figure 4)

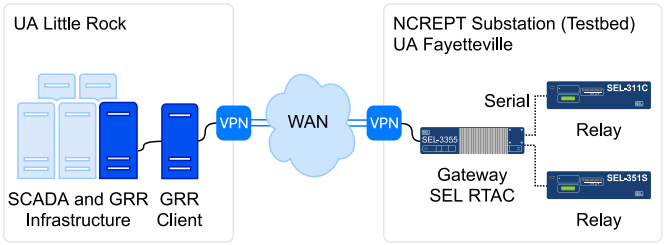


Fig. 4. Deployment setup of the HELOT GRR server and client integration into the OT system test bed provided by NCREPT's substation at UA Fayetteville. The actual GRR client connection to the OT system at NCREPT was established through a VPN tunnel to allow control access without the necessity of actual physical access to NCREPT's facilities.

B. Results

1) OT System Plugin for GRR: We implemented custom configurations in GRR for HELOT as Python scripts, which, uploaded to a GRR server, can be distributed to GRR clients and executed on deployed clients in the engineering or technician system connected to the internal network shared with the OT devices. This provides the ability to deploy scripted instructions during system operation, which is especially useful during development and custom configuration. The deployed GRR clients can initiate execution of locally stored scripts or other proprietary communication processes to collect and store data from OT devices. These collected artifacts from OT devices include device generated logs and reports, which are stored locally on the GRR client prior to their ingestion into the analytic pipeline of ELK stack. This type of implementation in which the actual communication and interaction with the OT device is handled by a secondary script or proprietary solution allows for added protection of sensitive login credentials and to adapt to virtually any proprietary OT device endpoint.

The exact OT device specific communication and log generation commands differ between devices and are therefore configured on the system that connects to the actual OT device

```
name: Client_data_to_server
doc: Collect client collected data to server.
sources:
- type: FILE
  attributes:
  paths: ['C:\rtac\RTAC_Logs.log']
supported_os: [Windows]
```

Listing 1. File `Client_Data_to_Server.yaml` is a GRR artifact describing a log file and its location on the client.

```
name: OT_data_to_client
doc: Export RTAC logs to client-machine directory.
sources:
- type: COMMAND
  attributes:
    args: ['C:\rtac\RTAC_Logs.sh']
    cmd: 'C:\cygwin\bin\bash'
labels: [System]
supported_os: [Windows]
```

Listing 2. File `OT_Data_to_Client.yaml` is a GRR artifact describing the action of executing a command for gathering log-file data from an OT system.

type. Depending on the OT device and the connection type, the interaction with the OT device will rely on the execution of locally deployed software and scripts, some of which may have been provided by the manufacturer for standard communication with the device. The specific plugin script or process to initiate execution on the client side for each OT device connected to the GRR client system can also be defined and uploaded to the GRR server as a custom YAML artifact plugin that points to a script or program on the GRR client system. This functionality enables the specification of device specific plugins for routine operations that can be quickly adapted or reused in the future for similar or identical device types or even for restoring a prior plugin configuration.

Listings 1 and 2 exemplary show artifact definitions for an output file and triggering of a command execution, respectively, for gathering log data from an OT system. Listing 3 shows an example `expect` [19] script actually logging into the OT system, gathering data, and saving results on the local GRR client. In addition to the low-level OT system data, a GRR client may simultaneously run additional IT system specific monitoring, reporting, and collection flows on any of the deployed GRR client system. The results of these IT system specific flows are not stored locally following their transmission to the GRR server. The GRR server then sends these results directly to the ElasticSearch node of the ELK stack for processing and further analysis.

2) Logstash Configurator: To validate and test the functionality provided by HELOT, we used data sets from a multitude of sources. The first source we used for data is a hardware-in-the-loop virtual power-system test bed consisting of a signal generator and two SEL power-system IED devices connected to a SEL RTAC controller (cf. Figures 4 and 5). The two devices are a feeder protection system, a SEL-351S and a SEL-311C. These and similar devices are used at scale in the operation and monitoring of live power systems via synchrophasor readings. We were able to acquire synchrophasor data [6] that details the exact state of the power characteristics such as frequency, amperage, and voltage of the electrical

```
#!/usr/bin/expect
set timeout -1

spawn telnet $IPADDR $PORT
expect "*"

# Send username, wait for password prompt
send "ACC\r"
expect "Password: "

# Send password, wait for shell prompt
send "$PASSWD\r"
expect ">"

# Send prebuilt command, wait for shell prompt
send "por 1\r"
expect "="

send "ACC\r"
expect "Password: "

send "$PASSWD\r"
expect "=>"

send "SER\r"
expect "913"

# Log output to file, wait for EOF, close log
log_file "[timestamp -format {%Y%m%dT%H%M%S}].log"
expect "=>"
log_file;

send "\x1d\r"
expect "telnet>"

send "close\r"
expect eof
```

Listing 3. File `RTAC_Process.sh` will, upon execution, connect to an SEL RTAC system via telnet, input the required password supplied via command-line invocation, and store the results in a time-stamped log file before closing the connection to the OT system.

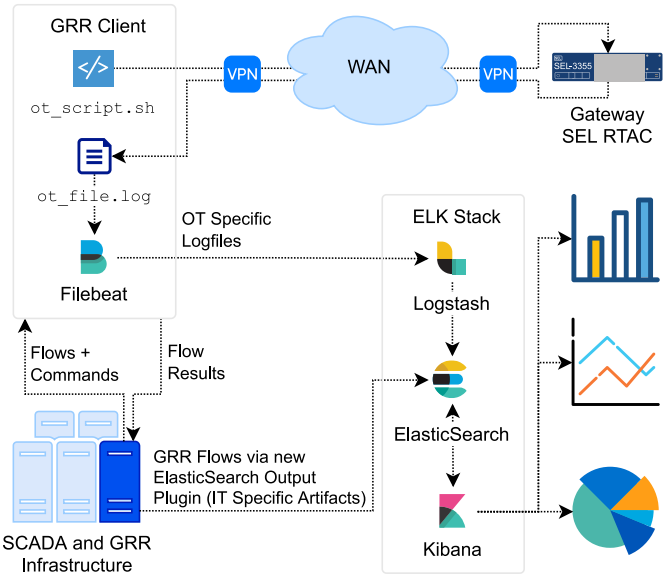
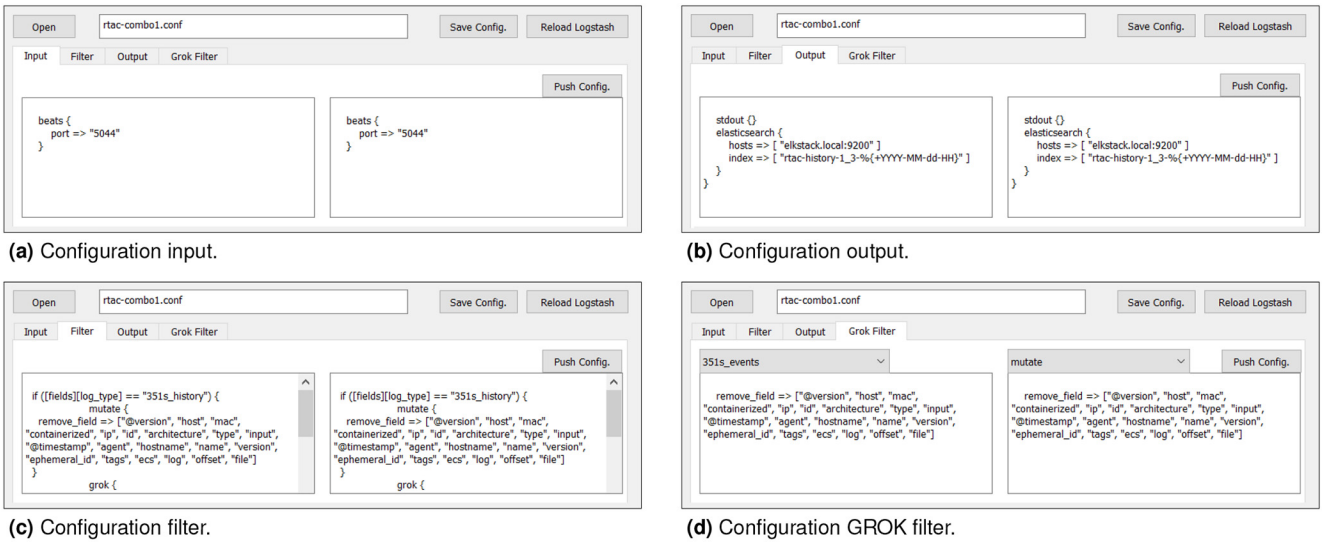


Fig. 5. Overview of data gathering in our deployment setup including an OT system and its connected GRR client, GRR server infrastructure, and ELK stack deployment for ingestion, post-processing and transformation as well as visualization approaches.

power supply for all three phases. The synchrophasor data was then used to train a machine-learning model to predict the likelihood of intrusion based on the state of the power signal. Following this, we also acquired similar data to that of our test bed scaled to a larger system consisting of twice the number of feeder lines. This data set was also for a 3-phase



(a) Configuration input.

(b) Configuration output.

(c) Configuration filter.

(d) Configuration GROK filter.

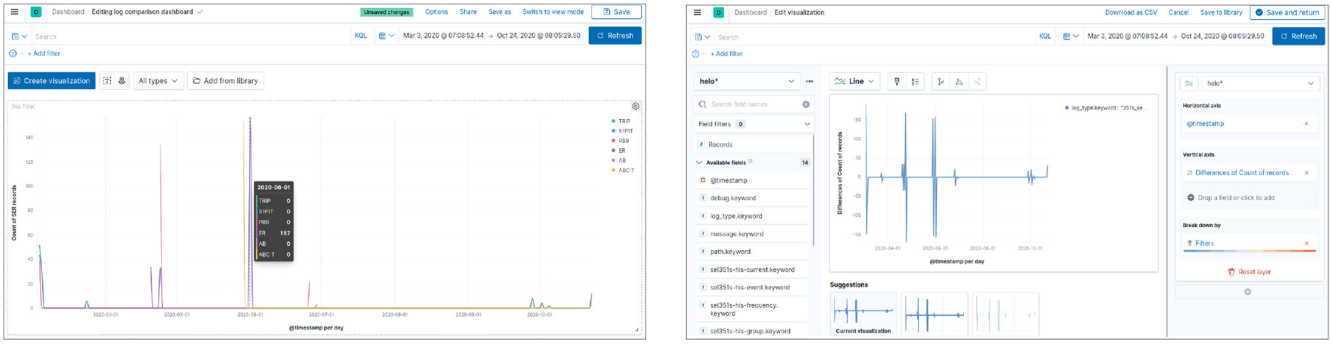
Fig. 6. Screenshots of the *Logstash Configurator* show a sample configuration using the application's tabs. (a) and (b) show the input and output of the configuration, respectively. (c) shows a filter transforming input from (a) to be processed by a GROK filter in (d) for additional transformation into index classification before result delivery to the output stage in (b). The *Logstash Configurator* allows loading or saving of configurations from file as well as application of the current setup to its connected Logstash instance.

virtual power grid and contained readings for 128 data points that are significantly scaled up from our NCREPT test bed data set.

Efficient and effective indexing of the OT device logs into ElasticSearch depends on how they are parsed by Logstash. This is the second part of the ELK stack and allows the ingestion of plain text logs from OT devices of interest. The variation in the output formatting of different types of logs from different OT devices necessitates the use of a regular expression based *GROK filter module*. This consists of a regular expression pattern essentially separating an ingested line of text into named fields and values. Logstash incorporates a large number of built-in GROK filters that can parse standard data elements such as IP addresses, geolocation coordinates, or timestamps from unstructured text. Additional GROK filters can be created for parsing OT device specific logs in individual pipelines. The creation, monitoring, and deployment of Logstash filters for different OT devices can become cumbersome and unwieldy when scaling to a multitude of devices from multiple vendors. To mitigate this and to enable efficient scaling of this particular feature, we developed a *Logstash Configurator* with a clean, user friendly GUI that allows users to quickly inspect the currently active filter in Logstash, make edits, and deploy these changes with a single click. This application also provides a starting point for scripting and batch handling of Logstash GROK filters at scale with further enhancements to allow the quick editing of regular expression patterns in selected GROK filters. Screenshots of an example configuration of the *Logstash Configurator* can be found in Figure 6.

3) Kibana Visualization: HELOT provides the ability to retrieve data from OT and IT systems at the same time with the help of GRR, Packetbeat, and Filebeat. This means that the forensic data available for analysis consists of more dimensions than system data from a singular OT or IT system. This in turn opens up the possibility of identifying, correlating, and linking events of interest occurring between separate system components rather than having only a local or single device perspective. This type of multi-domain collection and evaluation of artifacts also raises the issue of requiring to analyze extremely large data volumes to find subtle hints of anomalous behavior, such as an attack, that may well be still in progress. This is especially true in scenarios where investigators have to balance the use of automated and manual techniques for the preparation, inspection, and analysis of forensic data artifacts and multitudes of log files, which may contain anywhere from hundreds of thousands to millions of event records. Furthermore, unless a forensic investigator is directly familiar with the system being investigated, all of these log entries are somewhat abstract and difficult to correlate with other similarly abstract forensic data artifacts from other sources in the system. This is a problem of both scale and speed, which ends up slowing down the overall forensics process. We experimented with using Kibana to approach and, possibly, provide a solution for this problem domain by distilling these large data sets into various interactive visualizations that can be examined and inferred from at a glance, taking an investigator mere seconds instead of hours or days.

HELOT's data artifact collection and visualization capabilities open up the opportunity for analytical techniques that would be impossible when exclusively using OT or IT system artifacts. For example, we are able to analyze multiple log files within the same system to detect any inconsistencies between devices and their subsystems. Such inconsistencies potentially can be the result of an as yet undetected man-in-the-middle attack or denial-of-service attack on the innermost OT network layer. These attacks disrupt individual devices on the OT network and may even hide the disruption from being discovered by the system, manifesting as an unexpected system malfunction rather than a deliberate cyberattack. In such a scenario, comparing log entries generated by devices at various points in the OT system allows to validate and confirm



(a) Timestamp vs. count of SER records (view only).

(b) Timestamp vs. event-count differences (editable).

Fig. 7. Examples of Kibana visualization types based on OT artifact data gathered with our prototype. (a) shows an automatically generated plot representing the top three events in the SER and the HIS log, clearly making discrepancies visible. (b) displays the differences of raw event counts for two separate logs, again using SER and HIS logs, against time, which allows to visually represent when logs deviate and may indicate intrusion or a compromised system. (b) also shows the on-the-fly editing interface of Kibana with additional user-interaction elements.

that all of the devices and subsystems in the OT system are operating as signaled by the control system. While an attacker may be able to obscure the visibility of a device from a central monitoring system, it will be significantly more difficult to obscure or manipulate all internal logs on the instrumentation device itself due to design features like safety interlocks that are hard coded in non-volatile memory (e.g., ROM) and cannot be manipulated at run time. In such an event, the internal log file of a device will show a clear discrepancy with respect to the rest of the system.

To test these ideas, we collected data from a test-bed OT system (qv. Section IV-B2 *Logstash Configurator*) containing a SEL RTAC controller connected to a SEL-351S feeder protection relay and an additional SEL-311C transmission protection relay. The system generates multiple logs at the level of the RTAC and also at the individual relay devices. The SEL-351S device generates two distinct event logs that are created by two sub-routines running on the protection relay and retrieved by separate commands on the GRR client. These log files exhibit two different types of logs, detailing event histories over varying time spans. One type of log file is known as the sequential-events recorder (SER) log and lists 500 of the most recent sequential events. The other type of log file is event-history (HIS) log and contains only the most recent 22 events. The individual events themselves are identically represented in both logs (e.g., a line trip is referred to as a TRIP in both HIS and SER logs), which allows to directly compare these logs. After collection of both logs a visual log consistency check can be performed by comparing the events and their corresponding timestamps. Kibana is used to visually represent the individual logs and their event counts over a given time interval as well as allows a side-by-side layout for visual comparison or even overlaid on top of each other to emphasize any differences (cf. Figure 7). This allows for rapid inspection and to spot any inconsistencies over the entire time period of interest without inspection of individual log events, thereby saving time and effort on the part of forensic investigators. Another way different logs can aid in finding suspicious events early is to compare timestamp offsets. This is a slightly more subtle method but also highly sensitive to any deviations of OT devices' normal operations. If multiple logs are generated in a system and these share any common event entries, i.e., common events that appear in all of these logs, there will be a time offset between the appearance of such an event from one log entry to the next. This time offset is a potent tool that can be used to define and monitor the quality of system operations and therefore detect any disruptions.

Synchrophasor data is another class of OT data that can be acquired from power systems. While the history, events, and other device logs contain information about the system operation as well as its status and can help in tracking system activities, synchrophasor data consists of readings of phase-vector values like voltage, current, and phase angle for individual power phases. The synchrophasor values are utilized for training machine-learning algorithms, which are used to process incoming synchrophasor values at run time for classification as the result of an attack or normal system operation. We used the XGBoost algorithm [11] for this analysis to help monitor the actual values and operational parameters of the system, which is described in more detail in Section IV-B4 *Machine-Learning Based Adaptive Filtering*. Monitoring and analyzing only the synchrophasor values at different relays also allows to compare timestamps within the ranges of synchrophasor values and notify a user via Kibana dashboard capabilities in case of any anomalous values.

Kibana visualization also allows to *interactively drill down* into any particular log entries. For example, while displaying a visualization of the counts of each of the different types of events in a given log during a period of interest, a double click on a log item will display detailed information regarding that particular log (e.g., network information), which may help to identify whether or not the source was authorized or if there was any external intervention. Kibana dashboards also allow to display network characteristics of OT device log data as well as the interaction between specific servers and clients. Additionally, Kibana's *discover dashboard* feature allows to visually overlay IT and OT data logs to match the timestamps of each and to monitor all activities that are ongoing in all parts of the system, i.e., both the IT and the OT sub-systems. The visualization of different log sources can be combined into a more robust analytical feature set allowing for further analysis and to make accurate predictions by taking large amounts of

data points from the fleet of clients at once and convert into sensibly usable visual representation.

4) Machine-Learning Based Adaptive Filtering: For our initial experiments with machine-learning based evaluation of our collected data, we utilized a power-system data set from [6]. This data set consists of a set of telemetry streams, which includes examples of command injection attacks, data injection attacks as well as events where no attacks occurred. The data set also includes measurements related to an electric transmission system's normal, disturbance, control, and cyberattack behaviors. Measurements are from synchrophasor output and Snort data logs [50] as well as from a simulated control panel and additional relays. We preprocessed the data with the Scikit-Learn library [39], using LabelEncoder to normalize labels and transform non-numerical labels to enumerations. The synchrophasor measurement values were processed with MinMaxScaler, which transforms features by scaling each to a unified range while preserving the shape of the original data distribution, i.e., the operation does not change the information embedded in the original data. Note that MinMaxScaler does not reduce the importance of outliers. A default range of $[0 \ldots 1]$ was used as recommended by [29]. For the machine actual learning part, we utilized two data sets from [6]. One for the learning process, which was divided into 70 % for training and 30 % for validation. Actual testing used a separate data set from the same database. After preprocessing the labeled data was used for feature learning with two separate machine-learning algorithms: Decision Tree [49, ch. 18] and XGBoost [11]. Preliminary results indicate that the Decision Tree technique achieved an accuracy of 73 % on average with testing data set while XGBoost achieved an accuracy of 98 % on average with a 2 % false positive rate for the same testing data set. This means that XGBoost was able to detect all of the anomalies, i.e., any possible attacks, in the testing data set from [6]. We successfully deployed the XGBoost based machine-learning model trained on the data from [6] on data gathered by our infrastructure. Integrated into the Kibana dashboard development, this technique allows for more adaptive automation approaches while also likely reducing attention fatigue by human operators.

C. Cyberforensics Artifacts

Computer forensics analysis and acquisition can generally be categorized as static data or volatile or live data [52]. Static data is generally stored within a file system that is not active (e.g., unpowered hard disk) while volatile or live data is in use at the time of investigation by active hardware (e.g., RAM). Live data is aptly referred to as volatile data as it is ever-changing, however, if captured and analyzed, it will offer analysts helpful information. For forensics related to industrial control systems, live data acquisition is a priority, as pointed out before in Section I *Introduction*, because taking such systems offline for cyberforensics investigations is nearly impossible without harming these or its dependent systems.

During a cyberforensics investigation, common pieces of information need to be quickly retrieved. These include items such as logs, configured services, job schedules, system-patch state, user accounts, and more. The retrieved data are known as forensic artifacts and their location and format drastically vary across systems [27]. We have utilized a framework to describe forensic artifacts that allows them to be collected and customized quickly based on GRR. Artifacts in GRR are defined using YAML descriptions and collected using the artifact collector flow [14], which allows to collect multiple artifacts at the same time (qv. Section IV-B *Results* as well as listings 1 and 2). Collection of artifacts from multiple machines at the same time is particularly powerful because the collection can be scheduled in a single command across all systems and the artifacts themselves determine the paths to download from for each system.

GRR is designed with a fine-grained *separation of concern* principle in mind. It provides a multilayered architecture, including Fleetspeak, reverse proxies, HTTP servers, RDF data pools, client workers, schedulers, forwarders, and remote operator machines. Reverse proxies are used to distribute workload among servers. For instance, a reverse proxy for admin_ui distributes request/response control signals among different admin_ui nodes. Similarly, a reverse proxy for frontend distributes actual data among different frontend nodes. A scalable RDF data store is the central component of the server infrastructure used to store cyberforensics data. The cyberforensics data are further wrapped into Advanced Forensics Format version 4 (AFF4) [28], [40]. AFF4 [1] is an open and extensible file format to store disk images and their associated metadata with a high rate of data read/write operations and supports large payloads. Further, GRR's scheduler components allows scheduling several parallel tasks to client workers. Operators from any platform-agnostic remote machine are able to launch a flow to collect data from GRR clients, while the GRR server infrastructure is used to process and store that data.

D. Cyberoperations Automation

Advances in situational awareness technology have led to the creation of increasingly sophisticated tools across different application domains, often including data thatis non-textual, high dimensional, and multimedia in nature. Automated tools aim to address a number of situational awareness challenges such as complex system topology, rapidly changing technologies, high noise to signal ratio, and multi-faceted threats. These factors make real-time situational awareness of cybersecurity operations very difficult to achieve [25]. Cybersecurity vendors seem to employ operational automation for added efficiency as well as a way to save effort in man power or head count. However, operational automation is also a tool to be used to better predict process behaviors and achieve faster protection of critical systems. Appropriately implemented and using the right set of tools, operational automation allows aiding in the successful prevention of cyberattacks.

Secure (cyber)operations centers (SOC) are designed to provide comprehensive cyberprotection for on-premises, cloud, or hybrid environments across both IT and OT assets. SOCs are an integration point for all cybersecurity activities across a department or an institution's enterprise architecture synchronizing the protection of information, systems, functions,

and networks. SOCs usually employ multiple automated security measures, such as traffic monitors, firewalls, vulnerability scanners, and intrusion-detection or intrusion-prevention system [56]. SOCs heavily rely on cybersecurity analysts to investigate data from security measures to identify the true *signals* and to *connect the dots* to answer higher-level questions about attack activities (e.g., whether the network is under attack, what did the attackers do, what might be their next steps). However, with incoming network data collected and accumulated at a rapid speed, analysts are usually overwhelmed by tedious and repeated data-triage tasks, impeding them to concentrate on in-depth analysis to create timely incident reports.

Our architecture and prototype realization utilizes GRR for accessing and retrieving information that can be used for forensic purposes in a continuous or time relevant manner. In case of any suspicious activity, we have also developed a mechanism to trigger the conditional execution of processing pipelines utilizing machine-learning based filtering to allow the creation of visualizations and alerts for human operators involved in the forensic analysis. Not only can this to be used for forensic purposes but also for live hunting of malicious events. We have also developed a configurator application (qv. Section IV-B2 *Logstash Configurator*) that allows to retrieve and modify the configurations of Logstash configurations, which in turn aids in the retrieval of relevant artifacts.

IT cyber operations are effectively automated in a scalable manner by a GRR server using GRR clients deployed on relevant target systems. The GRR server and associated clients greatly streamline the process of navigating and routing data, metadata, and other forensic assets over complex network topologies extending over multiple public and corporate network zones. This allows to automate the collection and transmission operations from highly sensitive OT network zones as well as IT network zones into a single modular component. Once collection is completed, identification of threats that behave similarly within the massive amounts of data is used to predict adversaries' next step(s). Machine learning and automation will allow faster and more accurate data sequencing to be accomplished. Protective measures need to be created and distributed faster than an attack can spread throughout an organization's networks, endpoints, or cloud deployment. The time penalty added by any analysis suggests that the best place to stop a newly discovered attack is not at the location where it was discovered but rather at the attack's predicted next step. Automation can speed up the process of creating countermeasures without straining resources while keeping pace with an attack. To stop an active attack before data leaves the attacked network, analysts need to react as fast or faster than the attack itself. In order to identify an infected host or suspicious behaviors, data from the environment must be analyzed backward and forward in time, searching for combinations of behaviors indicating if and where a host in that environment has been infected. Similar to analyzing unknown threats attempting to enter a network, manual correlation and analysis of data across networks, network endpoints, and cloud instances will be difficult to scale. Automation of these processes does allow for faster analysis as well as faster detection and therefore the ability of intervention.

V. Conclusion

We presented a software architecture and prototype realization for capturing live-forensics information from interconnected OT and IT systems as well as the network traffic between these systems. Our architecture is based on Google Rapid Response (GRR), which we extended to allow access to and retrieval of relevant information from OT systems using GRR's extensible plugin system. Further, we added the ability to continuously capture events from OT systems, connected IT systems, and the interconnected network(s). The results are stored in a storage setup using the ElasticSearch/Logstash/Kibana (ELK) stack, though other search-engine based database technologies could be used instead. In addition to storing raw data from collected events, this also allows for immediate analysis and transformation into additional formats as well as custom indexing. We presented aspects of our prototype implementation, such as our *Logstash Configurator*, discussed virtualization approaches, and provided preliminary results of integrating machine-learning technology for adaptive signal classification and detection.

Based on our results, we outlined and discussed two application cases. First, we showed the results of our prototype experiments in capturing of forensics artifacts from a combined live OT and IT system based on exemplary configurations used in the power-delivery industry. Second, we discussed possible paths for automation to reduce the cognitive load for cybersecurity operators in combined OT/IT environments.

A. Future Work

Future work includes extending the GRR framework to include threat hunting templates for commonly used ICS devices and to provide these to the open-source community. This will include instructions on expanding GRR for additional devices where each device template includes mechanisms such as a connector script for connecting to the device, a feature selector YAML configuration file specifying cyberobservable locations, and a feature extractor script for transforming collected features into advanced forensics file format (AFF4) [1] or any successor standard.

Our visualization experiments need to be extended to prove useful for actual users. We currently think in two directions. One direction is the setup of existing and newly developed visualization templates to allow users to choose from. The other direction is to explore the inclusion of non-standard display and presentation modes. The first approach would allow expert users to customize and to experiment with how information is provided depending on their subject-matter expertise. While there is long-standing research in visualization techniques, the results are (usually) far from simple to allow automatic application or to be useful for software-design pattern driven approaches. Part of the problem here is the non-formalized knowledge body in visualization. However, in our opinion, the more important contributing factor is that

human operators have their very subjective view(s) on what works for them or what does not work. Sometimes a pie chart is all that will be necessary but increasingly interaction, instantaneous feedback, and exploration are the key to successful information comprehension. This is also enforced by our second direction, which would first make available and later integrate non-standard display and presentation modes. This may include current technology developments such as augmented-reality and virtual-reality displays as well as tiled, large-screen, and multi-view display installations. In both cases additional modes of perception and interaction are provided in exchange for an increased development and, sometimes, maintenance effort. However, these non-standard display and presentation modes also provide, on top of more natural interfaces for interaction and perception, the ability for collaboration between participating subject-matter experts independently of their location.

The discussion in Section IV-D *Cyberoperations Automation* already hinted at our believe that automation of many parts of cyberoperations is an important aspect in the future. While there are obvious first steps, such as automating signal classification and detection, we are of the opinion that automation requires more and more far reaching efforts for the simple reason that many threat vectors are based on automated tools as well. To successfully engage in such scenarios will require the help from automated and adaptive tool chains to counter, control, and dominate these threats.

We are planning to make available our prototype developments for the GRR plugin(s) as well as our *Logstash Configurator* as open-source software. In addition to negotiations with the project's funding agency this will also have to include a review of software source code to ensure a reliable baseline as well as discussions with industry entities about possible approaches for multi-vendor support for gathering OT system data.

While our work is at a mature stage for potential deployment, we are conscious it is also a first step in defining inter-connectivity and intercommunication between OT and IT systems and their various connecting networks. Our experiences and prototype results should be a valuable resource for future standardization and consolidation processes.

ACKNOWLEDGMENT

The authors thank Chris Farnell, Managing Director & Test Engineer, NCREPT, UA Fayetteville for providing access to the SEL RTAC device at NCREPT substation. The authors also acknowledge Hamdi Albunashee, Ph.D. student at UA Fayetteville, for sharing a sample SEL RTAC project.

REFERENCES

[1] "AFF4—The advanced forensics file format." May 2022. [Online]. Available: http://www2.aff4.org

[2] I. Ahmed, S. Obermeier, M. Naedele, and G. G. Richard, III, "SCADA systems: Challenges for forensic investigators," *IEEE Comput.*, vol. 45, no. 12, pp. 44–51, Dec. 2012, doi: 10.1109/MC.2012.325.

[3] A. Almalawi, X. Yu, Z. Tari, A. Fahad, and I. Khalil, "An unsupervised anomaly-based detection approach for integrity attacks on SCADA systems," *Comput. Security*, vol. 46, pp. 94–110, Oct. 2014, doi: 10.1016/j.cose.2014.07.005.

[4] T. Athay, R. Podmore, and S. Virmani, "A practical method for the direct analysis of transient stability," *IEEE Trans. Power App. Syst.*, vol. PAS-98, no. 2, pp. 573–584, Mar. 1979, doi: 10.1109/TPAS.1979.319407.

[5] R. R. R. Barbosa and A. Pras, "Intrusion detection in SCADA networks," in *Proc. IFIP Int. Conf. Auton. Infrastruct., Manage. Secur.*, 2010, pp. 163–166.

[6] J. Beaver, R. Borges, M. Buckner, T. Morris, U. Adhikari, and S. Pan. "Industrial control system (ICS) cyber attack datasets." 2014. Accessed: May 5, 2022. [Online]. Available: http://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets

[7] "BigQuery." Accessed: May 5, 2022. [Online]. Available: http://cloud.google.com/bigquery

[8] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: Risk assessment, detection, and response," in *Proc. 6th ACM Symp. Inf. Comput. Commun. Security*, 2011, pp. 355–366, doi: 10.1145/1966913.1966959.

[9] "Cellebrite." Accessed: Sep. 11, 2022. [Online]. Available: http://cellebrite.com

[10] E. Chan, S. Venkataraman, F. David, A. Chaugule, and R. Campbell, "Forenscope: A framework for live forensics," in *Proc. 26th Annu. Comput. Security Appl. Conf.*, 2010, pp. 307–316, doi: 10.1145/1920261.1920307.

[11] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[12] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for SCADA networks," in *Proc. SCADA Security Sci. Symp.*, vol. 12, 2007, pp. 1–12.

[13] M. I. Cohen, D. Bilby, and G. Caronni, "Distributed forensics and incident response in the enterprise," *Digit. Investig.*, vol. 8, pp. 5101–5110, Aug. 2011, doi: 10.1016/j.diin.2011.05.012.

[14] "Collecting GRR artifacts." Accessed: May 5, 2022. [Online]. Available: http://grr-doc.readthedocs.io/en/latest/investigating-with-grr/artifacts/collecting.html

[15] "Diffy: Visual regression testing made easy." Accessed: Sep. 11, 2022. [Online]. Available: http://diffy.website

[16] Y. Ding, W. Dai, S. Yan, and Y. Zhang, "Control flow-based opcode behavior analysis for malware detection," *Comput. Security*, vol. 44, pp. 65–74, Jul. 2014, doi: 10.1016/j.cose.2014.04.003.

[17] "Elastic." Accessed: May 5, 2022. [Online]. Available: http://www.elastic.co

[18] "ELK stack." Accessed: May 5, 2022. [Online]. Available: http://www.elastic.co/what-is/elk-stack

[19] "Expect." Accessed: May 5, 2022. [Online]. Available: http://core.tcl-lang.org/expect

[20] "Filebeat." Accessed: May 5, 2022. [Online]. Available: http://www.elastic.co/filebeat

[21] "FleeatSpeak." Accessed: May 5, 2022. [Online]. Available: http://github.com/google/fleetspeak

[22] "FTK forensic toolkit." Accessed: Sep. 11, 2022. [Online]. Available: http://www.exterro.com/forensictoolkit

[23] N. Goldenberg and A. Wool, "Accurate modeling of modbus/TCP for intrusion detection in SCADA systems," *Int. J. Crit. Infrastruct. Prot.*, vol. 6, no. 2, pp. 63–75, 2013, doi: 10.1016/j.ijcip.2013.05.001.

[24] O. Gospodnetic and E. Hatcher. *Lucene in Action*. Stamford, CT, USA: Manning Publ., 2004.

[25] R. Graf, F. Skopik, and K. Whitebloom, "A decision support model for situational awareness in national cyber operations centers," in *Proc. Int. Conf. Cyber Situational Awareness Data Anal. Assess.*, 2016, pp. 1–6, doi: 10.1109/CyberSA.2016.7503281.

[26] A. Greenberg. "Cellebrite says it can unlock any iPhone for cops." 2019. Accessed: Sep. 11, 2022. [Online]. Available: http://www.wired.com/story/cellebriteufed-ios-12-iphone-hack-android

[27] "GRR documentation." Accessed: May 5, 2022. [Online]. Available: http://grr-doc.readthedocs.io/en/v3.2.0/investigating-with-grr/artifacts/overview.html

[28] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep recurrent neural network based approach for Internet of Things malware threat hunting," *Future Gener. Comput. Syst.*, vol. 85, pp. 88–96, Aug. 2018, doi: 10.1016/j.future.2018.03.007.

[29] J. Hale. "Scale, standardize, or normalize with scikit-learn." 2019. Accessed: May 5, 2022. [Online]. Available: http://towardsdatascience.com/scale-standardizeor-normalize-with-scikit-learn-6ccc7d176a02

[30] J. Herrerias and R. Gomez, "A log correlation model to support the evidence search process in a forensic investigation," in *Proc. 2nd Int. Workshop Syst. Approaches Digit. Forensic Eng.*, 2007, pp. 31–42, doi: 10.1109/SADFE.2007.1.

[31] *Information Technology—Security Techniques—Encryption Algorithms—Part 3: Block Ciphers*. International Standard ISO/IEC 18033-3, 2010.
[32] B. Li, R. Lu, W. Wang, and K.-K. R. Choo, "DDOA: A Dirichlet-based detection scheme for opportunistic attacks in smart grid cyber-physical system," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 2415–2425, 2016, doi: 10.1109/TIFS.2016.2576898.
[33] L. A. Maglaras, J. Jiang, and T. Cruz, "Integrated OCSVM mechanism for intrusion detection in SCADA systems," *Electron. Lett.*, vol. 50, no. 25, pp. 1935–1936, 2014, doi: 10.1049/el.2014.2897.
[34] S. McElwee, J. Heaton, J. Fraley, and J. Cannady, "Deep learning for prioritizing and responding to intrusion detection alerts," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, 2017, pp. 1–5, doi: 10.1109/MILCOM.2017.8170757.
[35] "Modbus." Accessed: May 5, 2022. [Online]. Available: http://www.modbus.org
[36] P. O'Kane, S. Sezer, K. McLaughlin, and E. G. Im, "SVM training phase reduction using dataset feature filtering for malware detection," *IEEE Trans. Inf. Forensics Security*, vol. 8, pp. 500–509, 2013, doi: 10.1109/TIFS.2013.2242890.
[37] "OASIS cyber threat intelligence (CTI) TC." Accessed: May 5, 2022. [Online]. Available: http://www.oasis-open.org/committees/tchome.php?wgabbrev=cti
[38] "OpenText EnCase forensic." Accessed: Sep. 11, 2022. [Online]. Available: http://security.opentext.com/encase-forensic
[39] F. Pedregosa et al., "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html
[40] S. Ponomarev and T. Atkison, "Industrial control system network intrusion detection by telemetry analysis," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 2, pp. 252–260, Mar./Apr. 2016, doi: 10.1109/TDSC.2015.2443793.
[41] P. A. Porras and P. G. Neumann, "EMERALD: Event monitoring enabling response to anomalous live disturbances," in *Proc. 20th Nat. Inf. Syst. Security Conf.*, vol. 3, 1997, pp. 353–365.
[42] "Protocol buffers." Accessed: May 5, 2022. [Online]. Available: http://developers.google.com/protocolbuffers
[43] G. G. Richard and V. Roussev, "Next-generation digital forensics," *Commun. ACM*, vol. 49, no. 2, pp. 76–80, 2006, doi: 10.1145/1113034.1113074.
[44] A. Rodrigues, T. Best, and R. Pendse, "SCADA security device: Design and implementation," in *Proc. 7th Annu. Workshop Cyber Security Inf. Intell. Res.*, 2011, p. 25, doi: 10.1145/2179298.2179325.
[45] N. Runwal, R. M. Low, and M. Stamp, "Opcode graph similarity and metamorphic detection," *J. Comput. Virol.*, vol. 8, pp. 37–52, Apr. 2012, doi: 10.1007/s11416-012-0160-5.
[46] I. Santos et al., "Idea: Opcode-sequence-based malware detection," in *Proc. Int. Symp. Eng. Secure Softw. Syst.*, 2010, pp. 35–43, doi: 10.1007/978-3-642-11747-3_3.
[47] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Inf. Sci.*, vol. 231, pp. 64–82, May 2013, doi: 10.1016/j.ins.2011.08.020.
[48] N. Sayegh, I. H. Elhajj, A. Kayssi, and A. Chehab, "SCADA intrusion detection system based on temporal behavior of frequent patterns," in *Proc. 17th IEEE Mediterr. Electrotechn. Conf. (MELECON)*, 2014, pp. 432–438, doi: 10.1109/MELCON.2014.6820573.
[49] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[50] "Snort." Accessed: May 5, 2022. [Online]. Available: http://www.snort.org
[51] "Splunk documentation." Accessed: May 5, 2022. [Online]. Available: http://docs.splunk.com/Documentation
[52] J. Stirland, K. Jones, H. Janicke, and T. Wu, "Developing cyber forensics for SCADA industrial control systems," in *Proc. Int. Conf. Inf. Security Cyber Forensics Soc. Digit. Inf. Wireless Commun.*, 2014, pp. 98–111.
[53] N. P. Taveras, "SCADA live forensics: Real time data acquisition process to detect, prevent or evaluate critical situations," *Eur. Sci. J.*, vol. 9, no. 21, pp. 253–262, 2013.
[54] D. Yang, A. Usynin, and J. W. Hines, "Anomaly-based intrusion detection for SCADA systems," in *Proc. 5th Int. Topical Meeting Nucl. Plant Instrum. Controls Human Mach. Interface Technol.*, 2006, pp. 12–16. [Online]. Available: http://www.osti.gov/biblio/22030013
[55] Y. Yang et al., "Multiattribute SCADA-specific intrusion detection system for power networks," *IEEE Trans. Power Del.*, vol. 29, no. 3, pp. 1092–1102, Jun. 2014, doi: 10.1109/TPWRD.2014.2300099.
[56] C. Zhong, J. Yen, P. Liu, and R. F. Erbacher, "Learning from experts' experience: Toward automated cyber security data triage," *IEEE Syst. J.*, vol. 13, no. 1, pp. 603–614, Mar. 2019, doi: 10.1109/JSYST.2018.2828832.
[57] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on SCADA systems," in *Proc. Int. Conf. Internet Things 4th Int. Conf. Cyber Phys. Soc. Comput.*, 2011, pp. 380–388, doi: 10.1109/iThings/CPSCom.2011.34.

Syed Akailvi received the B.Tech. degree in mechanical engineering from Jawaharlal Nehru Technological University, India, in 2011, and the M.Sc. degree in systems engineering from UA Little Rock in 2016, where he is currently pursuing the Ph.D. degree in the Computer and Information Science Graduate Program.

Uddhav Gautam received the B.Sc. degree in computer science from Tribhuvan University, Nepal, in 2013, and the M.Sc. degree in computer science from UA Little Rock in 2017, where he is currently pursuing the Ph.D. degree in the Computer and Information Science Graduate Program.

Praveshika Bhandari received the B.Sc. degree in computer science from Leeds Beckett University, U.K., in 2018. She is currently pursuing the Ph.D. degree in the Computer and Information Science Graduate Program with UA Little Rock.

Hadi Rashid received the B.Sc. degree in computer science from the University of Mosul, Iraq, in 2003, and the M.Sc. degree in computer science from the University of Duhok in 2009. He is currently pursuing the Ph.D. degree in the Computer and Information Science Graduate Program with UA Little Rock and a Systems Administrator with Emerging Analytics Center.

Philip D. Huff (Member, IEEE) received the M.Sc. degree in computer science with Infosec Specialization from James Madison University in 2008, and the Ph.D. degree in computer science from UA Fayetteville in 2021. He is an Assistant Professor with the Department of Computer Science and the Director of Cybersecurity Research with the Emerging Analytics Center, UA Little Rock. He is a Certified Information Systems Security Professional.

Jan P. Springer (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from Bauhaus-Universität Weimar, Germany, in 1999 and 2008, respectively. He is the Director of the Emerging Analytics Center and an Associate Professor with the Department of Computer Science, UA Little Rock.