Using a Specification-based Intrusion Detection System to Extend the DNP3 Protocol with Security Functionalities

Hui Lin[1], Adam Slagell[2], Zbigniew Kalbarczyk[1], Ravishankar K. Iyer[1]
[1]Coordinated Science Laboratory, University of Illinois at Urbana-Champaign,
1308 W. Main Street, Urbana, IL, 61801
[2]National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign,
1205 W. Clark Street, Urbana, IL, 61801
[1]{hlin33, kalbarcz, rkiyer}@illinois.edu, [2]slagell@illinois.edu

ABSTRACT
Modern SCADA systems are increasingly adopting Internet technologies to control distributed industrial assets. As proprietary communication protocols are increasingly being used over public networks without efficient protection mechanisms, it is increasingly easier for attackers to penetrate into the communication networks of companies that operate electrical power grids, water plants, and other critical infrastructure systems. To provide protection against such attacks without changing legacy configurations, SCADA systems require an intrusion detection technique that can understand information carried by network traffic based on proprietary SCADA protocols.

To achieve that goal, we adapted Bro, a specification-based intrusion detection system, for SCADA protocols in our previous work [13]. In that work, we built into Bro a new parser to support DNP3, a complex proprietary network protocol that is widely used in SCADA systems for electrical power grids [4]. The built-in parser provides clear visibility of network events related to SCADA systems. The semantics associated with the events provide us with a fine-grained operational context of the SCADA system, including types of operations and their parameters. Based on such information, we propose in this work two security policies to perform authentication and integrity checking on observed SCADA network traffic. To evaluate the proposed security policies, we simulated SCADA-specific attack scenarios in a test-bed, including real proprietary devices used in an electrical power grid. Experiments showed that the proposed intrusion detection system with the security policies can work efficiently in a large industry control environment that can include approximately 4000 devices.

Categories and Subject Descriptors
K.6.5 [**Security and Protection**]

General Terms
Security

Keywords
SCADA, DNP3, Bro, specification-based intrusion detection system, authentication

1. INTRODUCTION
SCADA (Supervisory Control And Data Acquisition) systems monitor and control geographically distributed assets found in power grids, water plants, and other critical infrastructures. Exposing such control systems to public networks increases the risk of attacks and failures inherited from the commercial off-the-shelf (COTS) network infrastructure. However, many companies operating critical infrastructures still use proprietary communication protocols that have been integrated directly into the TCP/IP stack without addition of appropriate protection mechanisms. Consequently, the cyber threat to SCADA operations is "one of the most serious economic and national security challenges we face" [17]. This threat no longer exists in theory only. For example, in 2011, an attacker penetrated the control system of a water plant in Texas [12]; in a similar incident in 2012, gas pipelines were attacked by cyber intruders [10].

Some efforts to provide secure communications have been made, such as design of more secure protocols [5][16]. However, deployment of those approaches on existing legacy hardware and software cannot be accomplished overnight.

To provide secure communications without changing the current communication structure in control environments, we propose use of a specification-based intrusion detection system (IDS) to extend proprietary protocols with security functionalities. Specifically, we adapted Bro [20][22], a real-time network traffic analyzer, to integrate parsers for proprietary network protocols, such as a DNP3 protocol used in the cyber infrastructures of the electrical power grid. The built-in parsers generate network events related to SCADA commands. In previous work, we described how the IDS extracts semantics from DNP3 network packets, so we could validate conformance of the communication pattern to the protocol definitions [13].

In this paper, we focus on details of how the SCADA semantics can be used to enhance proprietary SCADA protocols to provide secure communications. We propose the introduction of security policies into the IDS to provide the SCADA protocols with basic but also critical security functionalities: (1). verify that the observed operations are from the authenticated site; and (2) verify that the operation is free from corruption during communication. With the help of the parser, the proposed IDS can distinguish observed control operations based on their physical locations, access types, command parameters, and other attributes. Based on that information, we can adapt the policies to meet specific requirements in control environments. It is possible to implement and execute the policies in the isolated trusted IDS domain without changing current communication structures in the general industrial control environment.

We evaluated the IDS and the implementation of the security policies in a test-bed that included real proprietary devices used in the electrical power grid. To simulate attack scenarios, we successfully built a piece of Trojan malware and included it in a legacy device. In the attack scenarios, the IDS performed intensive computation while monitoring and detecting malicious network traffic.

The remainder of this report is organized as follows. In Section 2, we analyze security threats and present our assumptions. Section 3 presents the construction of the proposed IDS. Section 4 describes in detail the security policies that will extend DNP3 with authentication and integrity validation functionalities. Section 5 describes the experimental test-bed and the policy implementations. In Section 6, we evaluate the proposed IDS and the implemented security policies. An overview of related work is provided in Section 7. We conclude in the last section.

2. BACKGROUND

2.1 Threat Model

Figure 1 presents components of SCADA systems commonly used in electrical power grids. Other industry control systems, such as gas and oil pipelines and wastewater control systems, share a similar communication structure [21].

Control Center. In the control center, human machine interface (HMI) computers are included to acquire measurement data from connected remote sites. After analyzing the data, human operators issue appropriate control operations. The collected data as well as issued operations are logged in a data historian. The control center can be penetrated through dial-in modems, wireless access points, and other pathways, as shown in [25]. That causes severe problems, as attackers can masquerade as insiders. By itself, the building of more secure communications, as discussed in this paper, is not sufficient to solve those problems. As a result, in this work, we assume that the control center is trusted.

Field Devices. Located in the remote *field site*, the field devices (also referred to as *Remote Terminal Units* in some documents) act as a gateway that communicates with actuators and sensors. All field devices, actuators, and sensors tend to be connected in an Ethernet to avoid point-to-point connections. The field devices usually run COTS operating systems and can be configured remotely over communication networks. As a result, field devices can be compromised by malware to perform man-in-the-middle attacks. Notably, in this work, we inserted a piece of Trojan malware into a real proprietary field device. The malware can successfully perform false-data injection attacks by modifying measurement data in DNP3 network packets [15]. Similarly, errors can also be injected into DNP3 traffic issued to field sites.

Control Network. This long-distance communication channel connects the control center and remote field sites. The question of whether or not the information transmitted over the control network is trusted depends on what protocols are used. In this paper, we focus on deploying protections over vulnerable SCADA protocols. Consequently, the SCADA network traffic cannot be trusted during the transmission, as an attacker can connect to the control network to sniff the SCADA traffic and replay it. However, with the same physical media, we could set up trusted tunnels between our instances of IDS sensors, as shown in the section 3.3.

Actuators and Sensors. It is also possible to penetrate through actuators and sensors by generating false measurement data. Based on these data, the control center may estimate a faulty state of a field site and thus issue incorrect commands [15]. It is possible to physically protect enough actuators and sensors to prevent the generation of false data [2][11]. However, exploiting field devices to generate false data does not require physical access, and we believe that this is a more severe and realistic threat to the industry control environment.

In this work, we focus on the transmission of DNP3 packets from the trusted control center to the actuators and sensors through the malicious communication media, including the untrusted control network and field devices.

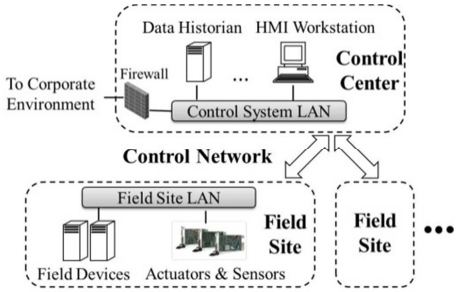


Figure 1: SCADA Systems Used in Electrical Power Grids

2.2 DNP3: Distributed Network Protocol

In this paper, we focus on extending the DNP3 protocol with security functionalities. Most previous research has focused on the Modbus protocol [23], which defines very simple structure for its network packets. The DNP3 protocol, however, is a "representative complex SCADA protocol" [7] packed into the TCP/IP stack in a complicated manner. Thus, it is difficult to manually extract information from DNP3 network packets.

DNP3 was initially used over serial lines, so it defines its own application layer, transport layer, and data link layer. That hierarchy cannot be directly mapped to the TCP/IP stack. As a result, all three DNP3 layers are packed together as a single application layer payload over the TCP layer (Figure 2).

The original application layer of the DNP3 protocol introduces complex structures as well. Each application layer fragment starts with an application header, which indicates what operations are performed. Multiple object headers can follow the application header to index target devices. Thus, it is possible to apply the same operation to multiple devices by issuing a single packet. The data object following the object header indicates the specific parameters of the operation. In this paper, we directly use DNP packets to refer to the original DNP3 application layer fragment.

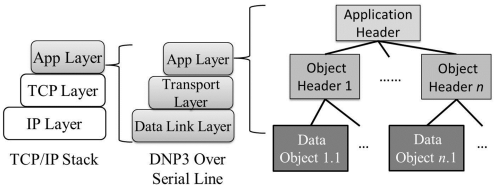


Figure 2: DNP3 over TCP

3. DNP3 ANALYZER

In this section, we present the main components of the *DNP3 analyzer*, the proposed IDS based on Bro [20][22]. Bro is a real-time network traffic analyzer widely used in forensic analysis, intrusion detection, and other network-related analysis. The modifications that we made to adapt Bro for SCADA environments are highlighted in Figure 3.

3.1 DNP3 Parser

We built a new parser for the DNP3 protocol. The main responsibility of the parser is to decode byte streams into meaningful data fields according to the protocol definition. After

parsing, the DNP3 parser generates SCADA system-specific events.

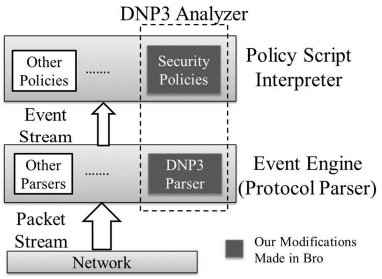


Figure 3: DNP3 Analyzer Based on Bro

The DNP3 parser exploits a compiler-assisted tool named *binpac* to shorten the development period and to ensure logical correctness [19]. At the current stage, we built and included in Bro the DNP3 parser to support complex hierarchical structure found in the DNP3 protocol. By using the same *binpac* technique, we believe that parsers to support other SCADA protocols can similarly be developed.

3.2 Event Handlers

Event handlers are used to analyze network events generated from the parsing of each DNP3 network packet. The semantic information related to each event is extracted during parsing. For example, a *dnp3_crob* (Control Relay Output Block) event is generated by the DNP3 parser if an operation to control relay outputs is found within a DNP3 request. The parameters associated with this operation, such as the type and duration of the operation, are extracted from the packet and delivered to the corresponding event handler.

A declaration of an event handler, including its name and arguments, provides an interface between the DNP3 parser and the policy script interpreter. During the parsing at run-time, the value of each argument is updated by the semantic information related to the event. We declared and associated an event handler with each type of data field defined in the DNP3 protocol; thus, the DNP3 analyzer can cover all semantic information from any type of DNP3 network packet. Although the declarations of event handlers are fixed, their definitions (what one does with an event) are left to be implemented in Bro scripts written by security experts. In specific operational contexts, such as operations in power grids, one can dynamically adjust security policies by including definitions of different event handlers.

3.3 DNP3 Analyzer Deployment

The proposed DNP3 analyzer can be connected to a network switch or a router in both the control center and the field sites (Figure 4). Most commercial switches or routers provide "span ports" or "mirror ports" that replicate all network traffic going through them. Being tapped to those ports, DNP3 analyzers are able to monitor and analyze local network events within the control center or the field sites as well as global communications between them. Furthermore, a DNP3 analyzer instance, deployed in a separate off-the-shelf workstation, is able to build a trusted communication channel, e.g. over TLS/SSL, to other DNP3 analyzer instances.

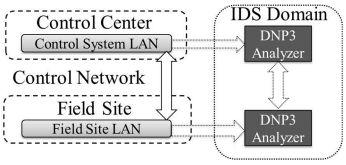


Figure 4: DNP3 Analyzer Deployment

4. SECURITY POLICIES

In our previous work, we developed a protocol validation policy to verify whether observed DNP3 network packets conform to protocol definitions [13]. In this paper, we focus in detail on how to use the extracted semantics to protect against malicious attacks. The original DNP3 protocol does not include effective authentication and encryption mechanisms[1]. In this section, we propose two security policies for detecting attacks that exploit those vulnerabilities. The proposed policies are specifically defined for the context of SCADA systems that operate electrical power grids.

The DNP3 analyzer inherits Bro's core idea of separating event generation from event analysis. Consequently, security policies presented in this section can be adjusted at run-time to meet specific operational context and attack scenarios.

4.1 Authentication Policy

4.1.1 Attack scenario

With a vulnerable network configuration, it is possible for an attacker to connect to the control network through bring-your-own-device (BYOD) mechanism. Thus, the attacker can sniff and modify DNP3 network packets to perform replay attacks. By issuing control operations, he or she can maliciously control the remote field sites.

4.1.2 Policy specifications

In the initial approach, we proposed to rely on host system activities in the control center to perform authentications. Specifically, system logs or application logs that contain the relevant information, such as who issued what operations at what time, are correlated to the run-time network packets observed by the DNP3 analyzer.

This technique, which feeds host semantics back to the network IDS, has been used in general computing environments [6]. However, we found that it must be adapted to meet specific requirements in the control environment. In the industry control environment, network traffic can be divided into two types:

- **Automatic operations.** Controlled by hardwired machines, these operations are frequently issued to retrieve measurement values from remote sites, e.g., DNP3 operations that read binary outputs.
- **Manual operations.** Controlled by human operators, these operations are used to configure field devices or operate actuators and sensors, e.g., DNP3 operations that edit a configuration file or open/close a replay.

Based on that classification, an authentication policy includes two rules:

[1] DNP3 provides only authentication, based on a plain-text password, for opening or deleting files in remote devices.

(1) A host activity (in terms of system logs or application logs) must be found to match each manual DNP3 operation observed in the network.

(2) It is recommended, but not required, that automatic DNP3 operations be authenticated.

With the help of the DNP3 parser, we can divide the observed DNP3 network packets into those two categories. Consequently, the DNP3 analyzer can selectively authenticate the manual operations.

4.2 Integrity Policy

4.2.1 Attack scenario

If an attacker penetrates into field devices, he or she may use the mediator that connects the control center and the field site (as shown in Figure 1) to perform a man-in-the-middle attack. For example, the attacker can corrupt the measurement data sent to the control center and also change the operations issued to remote field sites. In this scenario, we assume that the field devices, actuator, and sensors are connected through network switches or routers.

4.2.2 Policy specifications

The integrity policy provides site-awareness detections on corrupted network packets. A single DNP3 packet can be related to multiple physical devices, which are indexed by the object headers (Figure 2). As a result, when a DNP3 packet passes a field device, the packet is usually divided into several packets issued to different physical devices. The original network header may be replaced with the new ones, but the payload of each DNP3 packet, such as issued operations or the measurement data, should not be changed.

Based on that understanding, the following integrity policy is defined and validated against each field device:

(1) Payload values contained in the ingress and egress DNP3 packets of the same field device must not be modified.

The comparison requires only logic operations, so detection and location of the corrupted field device can be done very efficiently. With the help of the DNP3 parser, we can also easily locate which values are corrupted in order to better understand the attacker's intentions. For example, if we can inform the control center that an attacker has changed a DNP3 request that was supposed to open a relay into the one that closes it, the system operator will have more information to use in performing remedial activities.

5. IMPLEMENTATION

5.1 Experimental Environment

We implemented the proposed security policies, i.e., the authentication policy and the integrity policy, in an experimental test-bed with real-world hardware devices and software to mimic realistic configuration and operation of power grid substations, as shown in Figure 5[2]. Furthermore, we added various different components to the test-bed to simulate the attack scenarios mentioned in Section 4.

[2] The name and the model of the hardware devices are hidden to preserve the manufacturers' privacy

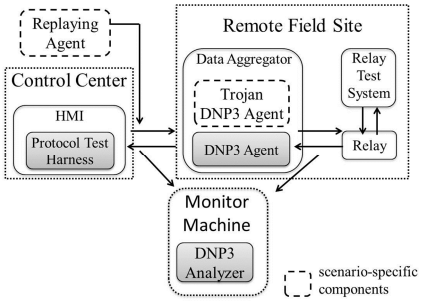


Figure 5: SCADA Test-bed

The test-bed included the following basic components:

HMI. This workstation can run COTS systems, such as Windows XP or Ubuntu, to simulate the control center. In the HMI, we ran Protocol Test Harness, software from Triangle MicroWork Inc. [24], to control simulated field sites though DNP3 network packets.

Relay. The protection replay was connected to a Relay Test System and periodically monitored its health status.

The Relay Test System. This proprietary hardware device simulated the configurations of power systems. For example, the IEEE 14-bus system or IEEE 30-bus system can be configured and run in the relay test system.

Data Aggregator. The data aggregator included a DNP3 agent that forwarded an operation from the control center to the relay and aggregated measurement data from it. The data aggregator ran a custom Ubuntu operating system on the PowerPC architecture.

Monitor Machine. The monitor machine was a separate COTS workstation in which the proposed DNP3 analyzer ran independently without affecting the operations of the simulated control center and field site. All the components were connected to a network switch. The switch was configured such that all network traffic was "mirrored" to the monitor machine.

The test-bed included the following attack-scenario-specific components:

Replaying Agent. This module was written in C and relied on the TCP/IP socket API. It sniffed the DNP3 network packets on the network and replayed them with errors injected into random locations. Corresponding error detection codes (CRC values) were recalculated to simulate malicious modifications.

Trojan DNP3 Agent. Similar to the replaying agent, this malware was written in C and relied on the TCP/IP socket API. In addition to performing the same packet forwarding done by the original DNP3 agent, it modified the forwarded DNP3 packets at random locations with random values. Corresponding error detection codes (CRC values) were recalculated to simulate malicious modifications. The module was cross-compiled and installed in the proprietary data aggregator, and the error-injected packets were successfully processed without generating any errors.

5.2 Authentication Policy Implementation

In general enterprise systems, an SSH client or FTP client usually delivers application logs to operating systems. In a real power grid control center, the data historian plays a similar role by logging local activities, such as what operations are issued. In the HMI included in the test-bed, we logged operations issued by the

Protocol Test Harness and delivered the logs to the monitor machine. We implemented that approach by configuring a remote syslog server in the monitor machine. The syslog messages were transmitted over encrypted tunnels, so they could not be replayed. We believe that this implementation is also possible in a real power grid environment.

Table 1: Pseudo Code of the Authentication Policy

```
event syslog_message(facility: count, userID: count, UsrID: count, logFc: count, obj_type: count, ……){
    if(facility >= 16 && facility <= 18){
        globalUsrID = UsrID;
        globalFc = logFc;
        globalObj = obj_type;
    …….
    }
}
event dnp3_obj_header(c: connection, is_orig: bool, app_control: count, fc : count, obj_type : count, ….){
    if( globalSyslog – current_time() < INTERVAL
        && globalFc == fc && globalObj == obj_type
        && …….){
        MATCH;
    }
    else{  ALERT;  }
}
```

Table 1 shows pseudo-code of the authentication policy written by Bro scripts. In the implementation, we defined the *syslog_message* event handler that was already declared by Bro's syslog analyzer. The event handler records in global variables the attributes of an operation logged by the HMI, such as timestamp, operator ID, and operation type and parameters. The definitions of three event handlers, i.e., *dnp3_request_header*, *dnp3_response_header*, and *dnp3_object*, extract values of the function code, the object type, and other semantic information from the observed DNP3 network packet. The DNP3 analyzer relies on the information recorded by the *syslog_message* event handler within a predefined time interval to verify whether the network packet was issued from the control center.

Even though the control center can issue operations in parallel, the legacy devices, such as the relay machine, can only handle operations in serial, for compatibility reasons. Consequently, for each hardware device that can be identified by IP addresses, we matched a single DNP3 packet with a single syslog message. If an attacker injects different SCADA operations through the replaying agent, this activity can always be detected.

5.3 Integrity Policies Implementation

As SCADA systems tend to rely on Internet technology for communication, field devices can be identified through IP addresses. In our test-bed configuration, the HMI, the data aggregator, and the relay machine were assigned different IP addresses.

Table 2: Pseudo Code of the Integrity Policy

```
……
event dnp3_analog_input_32(c: connection, value: count) {
    if ( c $ id $ orig_h == Relay_IP
        &&  c $ id $ dest_h == Data_Aggregator_IP ){
        globalValue = value;
    }
    If (c $ id $ orig_h == Data_Aggregator_IP
        && c $ id $ dest_h == Control_Center _IP){
        If (gloabValue != value)  ALERT;
    }
}
……
```

The implementation included the definitions of four groups of event handlers: *dnp3_analog_input_xx*, *dnp3_analog_output_xx*, *dnp3_binary_input_xx*, and *dnp3_binary_output_xx* ("xx" denotes data formats, e.g., 8-bit, 16-bit, etc.). Because of space limitations, we present only the implementation on a *dnp3_analog_input_32* event handler (see Table 2). This event handler is executed whenever a 32-bit analog input value is found in a DNP3 response.

Based on the source and destination IP addresses, the DNP3 analyzer distinguishes between network packets with different directions. If a packet is an ingress packet delivered from the relay (represented by the first "if" statement), we store this value in a global variable. When an egress packet is observed from the data aggregator to the control center (represented by the second "if" statement), a comparison is made to determine whether or not the measurement was corrupted during processing.

6. DNP3 ANALYZER EVALUATION

6.1 Evaluation Traces

In order to intensify the computations of the authentication policy, we used the replaying agent to randomly inject DNP3 operations among legal DNP3 operations issued and logged by the HMI. Based on the logs from the HMI, we could also verify whether or not the proposed DNP3 analyzer generated correct alerts.

Similarly, we replaced the original DNP3 agent in the data aggregator with the Trojan DNP3 agent. The Trojan DNP3 agent injected errors into the measurement data carried by responses to the HMI. Injection of errors into the control operations carried by requests can be performed similarly. However, as we randomly injected errors, we observed that corrupted requests triggered warnings from the relay machine and made it respond inappropriately. Therefore, in the experiment, we injected errors only into the measurement data, to perform stealthy attacks. The Trojan DNP3 agent was further instrumented to log all the injected errors, to help us determine whether the DNP3 analyzer made the right detections.

In both attack scenarios, we planned to further reduce the latency between issued operations, to analyze the processing capabilities of the DNP3 analyzer. However, if the interval between DNP3 operations is less than or equal to 0.5 seconds, the data aggregator and the relay will miss network packets from time to time. In the real power grid environment, DNP3 packets are usually issued at a frequency of one or two packets per second [9]. Consequently, instead of evaluating the DNP3 analyzer online, we collected network traces in two attack scenarios and evaluated the DNP3 analyzer offline.

We refer to the two network traces as the *A-trace* (extracted from the attack scenario evaluating the authentication policy) and the *I-trace* (extracted from the attack scenario evaluating the integrity policy). Detailed descriptions of both traces are presented in Table 3.

Table 3: DNP3 Network Traces

Trace	Description	Size (MB)	Size (DNP3 packets)
A-trace	Contained both syslog messages and DNP3 network packets.	1100	2,200,100
I-trace	Contained DNP3 packets carrying measurement data	995	2,040,000

6.2 Performance Overhead

As the DNP3 analyzer is used to analyze industry control environments passively, it must process network packets in real time to provide useful detection results. In this section, we evaluate the throughput of the DNP3 analyzer. We adopt two throughput metrics for evaluation: the number of bits processed per second (bps), and the number of packets processed per second (pps). Performance overheads generated by the DNP3 parsers and the policies are analyzed separately.

The DNP3 analyzer processed all packet traces, i.e., both the A-trace and I-trace, off-line on the monitor machine. At run-time, the DNP3 analyzer ran independently without affecting operations of the simulated SCADA components. The purpose of the additional off-line analysis performed in this section was to evaluate analyzers' ultimate processing capabilities on DNP3 network packets. The resulting analysis can give us an idea of how the proposed DNP3 analyzer fits the real SCADA systems in addition to the simulated test-bed.

The monitor machine was a VMware virtual machine with a single logical processor of two 3.07GHz cores and a 1GB RAM. During the processing, the monitor VM ran exclusively on the host machine in order to avoid interference from other virtual machines.

6.2.1 Performance overhead by the parser

The DNP3 parser (the DNP3 analyzer without any policy loaded) will always be used to parse DNP3 network packets. As a result, it should be efficiently implemented. To demonstrate its efficiency, we compared it to the FTP parser (parsing FTP control commands only) that was already integrated in Bro. The reason we chose the FTP parser was that application layer functionality and payload size of FTP network packets are similar to those of the DNP3 packets.

For the work described in this paper, we used FTP traces from [18] as the workload of the FTP parser. However, the traces could not be directly used in this experiment, as all FTP requests/responses were included in a single TCP session. In the collected DNP3 packets, however, each pair of DNP3 requests/responses were included in an individual TCP session. To solve that problem, we extracted the application payload from the FTP traces. Then we replayed over contained internal networks each pair of FTP request and response in separate TCP sessions. The replay resulted in a traffic trace with a large size. We collected 989M of the resulting trace, which included 2,392,000 FTP packets.

We ran Bro's FTP analyzer against that FTP trace and the DNP3 analyzer against the A-trace (with all syslog messages removed). We did not load any policies for either analyzer. As a result, only the parser processed the corresponding network trace. We performed each experiment for 10 runs to measure the average execution time.

The evaluation results are presented in Table 4. Even though a DNP3 packet can have complex structures, the DNP3 parser had a better throughput (about 40% better) than the FTP parser in terms of both evaluation metrics. The reason was probably due to the fact that the DNP3 parser was implemented by the *binpac* scripts. The *binpac* scripts were automatically optimized and translated into the resulting C++ codes. The FTP parser, however, was directly written in C++ in Bro with few manual optimizations. As most of Bro's prebuilt parsers, including the FTP parser, have been evaluated in an intense computing environment, we can expect that the DNP3 parser will have a similar performance in a large-scale industry environment.

Table 4: Comparison of the DNP3 Parser and the FTP Parser

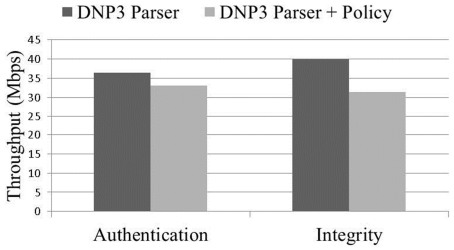
Evaluation Target	Throughput (Mbps)	Throughput (pps)
DNP3 Parser	35.34	7123.5
FTP Parser	23.56	13950.4

6.2.2 Performance overhead by the policies

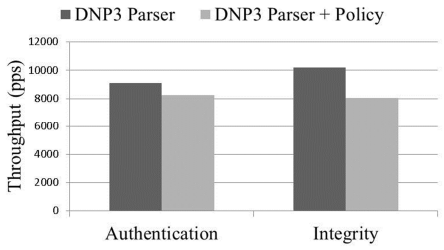
To evaluate the additional performance overhead generated by policies, we loaded the DNP3 parser with two policies and ran the analyzer against the corresponding traces. We performed each experiment for 10 runs to measure the average execution time.

The evaluation results in terms of two throughput metrics are presented in the two sub-figures of Figure 6. Since we evaluated each policy on two different network traffic traces, i.e., the A-trace and I-trace (see Table 3), we separated the results into two separate groups. In each group, we used "DNP3 Parser" to represent the scenario in which the DNP3 analyzer processed the network trace without any policy loaded. In other words, only the DNP3 parser was working. In the other scenario, represented by "DNP3 Parser + Policy," both the DNP3 parser and the policy script interpreter worked to process the network trace.

The throughput degradation (in both metrics) ranged from 9% to 15%. Notably, the implementations of the security policies were not optimized; for example, redundant alerts were not removed. During the processing of both the A-trace and I-trace, the DNP3 analyzer performed intense analysis and I/O operations (writing alerts to local files).



(a)



(b)

Figure 6: The Throughput of the DNP3 Analyzer with Policies: (a) in Mbps; (b) in pps

Even under those conditions, more than 8000 DNP3 network packets were processed every second (when the I-trace was processed). In an industrial control environment such as the power grid, legacy devices usually issue one or two DNP3 network packets every second [9]. Based on those figures, we anticipate that the proposed DNP3 analyzer can monitor a field site consisting of 4000 to 8000 devices. When more DNP3 analyzers are distributed into different host machines to form a monitor cluster, a larger-scale control environment can be monitored.

7. RELATED WORK

Traditional signature-based intrusion detection techniques are not widely used in control environments, because little analysis of real attacks is publicly available. Instead, anomaly-based intrusion detection techniques were initially used in the area. These techniques detect intrusions based on deviation from profiled baseline behavior. In [14][26], normal network communication patterns are formed based on destination host addresses, port numbers, and other information extracted from the network packet header through profiling or self-adaptive learning. Suspicious events caused by attackers, such as a random scan in field sites, usually generate network behavior that deviates from normal patterns and thus can be detected. However, anomaly-based detection techniques fail to detect malicious network packets following normal communication patterns, as in the attack scenarios discussed in this paper.

Specification-based techniques rely on a specific model or policy that is constructed based on the internal logic of the monitored system. At run-time, any system behavior that deviates from this policy or model triggers alerts. The work in [3] assumes that the attackers' activities usually result in malformed network packets. A policy is defined to verify whether the structure of each network packet conforms to the definitions of the Modbus protocol, another proprietary protocol used in SCADA systems [23]. Compared to Modbus, many other proprietary protocols, such as DNP3, are much more complex and contain more diverse semantics. Without fully understanding the SCADA-specific semantics, it is hard to design security policies to analyze well-formatted network traffic with malicious intentions. Work presented in [1] applies a specification-based technique to the advanced metering infrastructure (AMI), which is a very different wireless communication environment. [1] emphasizes the design of system models or specifications and their formal verification. The difference between [1] and our work is that we focus on the design of a SCADA-specific IDS that can be used in real SCADA systems to provide various run-time semantic analyses, such as extension of proprietary protocols with the security functionalities proposed in this paper.

The research described in [8] performed forensic analysis of semantic information collected from real critical infrastructure. The method used in the paper is similar to the specification-based detection techniques. Our work is different in that we propose an online IDS that can fit into a real SCADA operational environment. In SCADA systems, the latency of intrusion detection and response should be small so system operators can perform necessary remedial processes instantly. Thus, the proposed DNP3 analyzer was evaluated for its ultimate processing capabilities.

8. CONCLUSION

In this paper, we propose a DNP3 analyzer, an IDS that performs in-depth analysis on semantics from network traffic in control environments. With the help of our built-in DNP3 parser, the analyzer is able to generate real-time events related to SCADA systems. A sufficient number of event handlers are declared and associated with packet data fields to cover all semantic information carried by DNP3 network traffic.

We further propose two security policies to extend the DNP3 protocol with capabilities to authenticate and validate the integrity of each network packet. Knowledge of the SCADA system's operational context, such as the type of operations and their parameters, enables efficient and accurate policy design and implementation. To evaluate the proposed security policies, we simulated SCADA-specific attack scenarios in a test-bed that included real proprietary devices. The attack scenarios forced the DNP3 analyzer perform intensive computation to monitor and detect malicious activities. Based on the experimental results, the proposed DNP3 analyzer presented good processing capabilities, which shows a potential to work in a large-scale environment.

In future work, we plan to correlate SCADA-specifics semantics observed from the network with domain-specific security analysis, such as contingency analysis, in the power grid environment. As a result, the proposed DNP3 analyzer can better understand the effect of the semantics observed from the network traffic. Based on this analysis, the DNP3 analyzer can detect DNP3 network packets with legal communication patterns but carrying malicious control operations in payload.

9. ACKNOWLEDGMENTS

This material is based upon work supported in part by the Department of Energy under Award Number DE-OE0000097, by the National Science Foundation under Grant No. OCI-1032889, by Infosys Limited, and by the Boeing Company. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other sponsors.

10. REFERENCES

[1] Berthier, R. and Sanders, W. H. Specification-based intrusion detection for advanced metering infrastructure. In *Proceedings of 2011 IEEE 17th Pacific Rim International Symposium on Dependable Computing* (Pasadena, CA, USA, Dec. 12–14, 2011), 184–193.

[2] Bobba, R., Rogers, K., Wang, Q., Khurana, H., Nahrstedt, K., and Overbye, T. Detecting false data injection attacks on dc state estimation. In *Preprints of the First Workshop on Secure Control Systems (SCS 2010)* (Stockholm, Switzerland, April 12, 2010).

[3] Cheung, S., Dutertre, B., Fong, M., Lindqvist, U., Skinner, K., and Valdes, A. Using model-based intrusion detection for SCADA networks. In *Proceedings of the SCADA Security Scientific Symposium 2007* (Miami Beach, FL, USA, Jan. 24–25, 2007), 127–134.

[4] Curtis, K. *A DNP3 protocol primer*. Technical report. DNP User's Group, 2000.

[5] DNP Users Group. DNP3 Secure Authentication Version 5 (SAv5). 2011.

[6] Dreger, H., Kreibich, C., Paxson, V., and Sommer, R. Enhancing the accuracy of network-based intrusion detection with host-based context. In P*roceedings of the Second International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2005)* (Vienna, Austria, July 7–8, 2005), 206–221.

[7] Fovino, I. N., Coletta, A., and Masera, M. *Taxonomy of security solutions for the SCADA sector*. Technical Report, ESCoRTS, Security of Control and Realtime Systems, Mar. 2010.

[8] Hadziosmanovic, D., Bolzoni, D., Hartel, P., and Etalle, S. Melissa: Towards automated detection of undesirable user actions in critical infrastructures. In *Proceedings of 2011 European Conference on Computer Network Defense (EC2ND 2011)* (Gothenburg, Sweden, Sept. 6–7, 2011), 41–48.

[9] Heine, E., Khurana, H., and Yardley, T. Exploring convergence for SCADA networks. In *Proceedings of 2011 IEEE PES Innovative Smart Grid Technologies* (Anaheim, CA, USA, Jan. 17–19, 2011), 1–8.

[10] ICS-CERT. *ICS-CERT Monthly Monitor*. http://www.us-cert.gov/control_systems/pdf/ICS-CERT_Monthly_Monitor_June-July2012.pdf, June/July 2012.

[11] Kosut, O., Jia, L., Thomas, R., and Tong, L. Malicious data attacks on smart grid state estimation: Attack strategies and countermeasures. In *Proceedings of First IEEE International Conference on Smart Grid Communications (SmartGridComm 2010)* (Gaithersburg, Maryland, USA, Oct. 4–6, 2010), 220–225.

[12] Liebowitz, M. Hacker says he breached Texas water plant's control system. *Security Daily News* (Nov. 21, 2011): http://www.securitynewsdaily.com/1249-hacker-texas-water-plant.html

[13] Lin, H., Slagell, A., Di Martino, C., Kalbarczyk, Z., and Iyer, R. Adapting Bro into SCADA: Building a specification-based intrusion detection system for the DNP3 protocol. In *Proceedings of 8th Annual Cyber Security and Information Intelligence Research Workshop (CSIIRW 2012)* (Oak Ridge, TN, USA, Oct. 30–Nov. 1, 2012), to appear.

[14] Linda, O., Vollmer, T., and Manic, M. Neural network based intrusion detection system for critical infrastructures. In *Proceedings of 2009 International Joint Conference on Neural Networks (IJCNN 2009)* (Atlanta, GA, USA, June 14–19, 2009), 1827–1834.

[15] Liu, Y., Ning, P., and Reiter, M. False data injection attacks against state estimation in electric power grids. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)* (Chicago, IL, USA, Nov. 9–13, 2009), 21–32.

[16] Majdalawieh, M., Parisi-Presicce, F., and Wijesekera, D. DNPSec: Distributed network protocol version 3 (DNP3) security framework. *Advances in Computer, Information, and Systems Sciences, and Engineering* (K. Elleithy, T. Sobh, A. Mahmood, M. Iskander, and M. Karim, Eds.), Springer Netherlands, 2006, 227–234.

[17] Obama, B. Taking the Cyberattack Threat Seriously. *Wall Street Journal Online* (July 19, 2012): http://online.wsj.com/article/SB10000872396390444330904577535492693044650.html

[18] Pang, R. and Paxson, V. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2003)* (Karlsruhe, Germany, Aug. 25–29, 2003), 339–351.

[19] Pang, R., Paxson, V., Sommer, R., and Peterson, L. Binpac: A yacc for writing application protocol parsers. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC '06)* (Rio de Janeiro, Brazil, Oct. 25–27, 2006), 289–300.

[20] Paxson, V. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31(23–24), Dec. 14, 1999, 2435–2463.

[21] Stouffer, K., Falco, J., and Kent, K. Guide to supervisory control and data acquisition (SCADA) and industrial control systems security. *NIST Special Publication*, May, 2006.

[22] The Bro Project. Bro Network Security Monitor. http://bro-ids.org, 2012.

[23] The Modbus Organization. Modbus messaging on TCP/IP implementation guide v1.0b, 2006. http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf.

[24] Triangle MicroWorks, Inc. Communication Protocol Test Harness. http://www.trianglemicroworks.com/documents/Protocol_Test_Harness_Fact_Sheet.pdf, 2012.

[25] Turk, R. *Cyber incidents involving control systems.* Technical Report, Idaho National Engineering and Environmental Laboratory, 2005.

[26] Valdes, A. and Cheung, S. Communication pattern anomaly detection in process control systems. In *Proceedings of 2009 IEEE Conference on Technologies for Homeland Security (HST'09)* (Waltham, MA, USA, May 11–12, 2009), 22–29.