Generating honeypot traffic for industrial control systems

Htein Lin, Stephen Dunlap, Mason Rice, Barry Mullins

Chapter 11

GENERATING HONEYPOT TRAFFIC FOR INDUSTRIAL CONTROL SYSTEMS

Htein Lin, Stephen Dunlap, Mason Rice and Barry Mullins

Abstract Defending critical infrastructure assets is an important, but extremely difficult and expensive task. Historically, decoys have been used very effectively to distract attackers and, in some cases, convince attackers to reveal their attack strategies. Several researchers have proposed the use of honeypots to protect programmable logic controllers, specifically those used in the critical infrastructure. However, most of these honeypots are static systems that wait for would-be attackers. To be effective, honeypot decoys need to be as realistic as possible. This chapter introduces a proof-of-concept honeypot network traffic generator that mimics a genuine control system in operation. Experiments conducted using a Siemens APOGEE building automation system for single and dual subnet instantiations indicate that the proposed traffic generator supports honeypot integration, traffic matching and routing in a decoy building automation network.

Keywords: Honeypots, network traffic generation, industrial control systems

1. Introduction

The United States Ghost Army conducted deception operations in France, Belgium, Luxembourg and Germany during World War II [1]. The mission was to deceive the enemy and lure German units away from Allied combat units. Engineers set up inflatable armored tanks, aircraft, airfields, tents and motor pools. Other tactics such as looping convoy traffic, deploying military police and posting general and staff officers in public places were used to draw Axis resources (e.g., intelligence assets and combat power) away from the real targets. The Ghost Army also played recordings of actual armor and infantry units over loudspeakers. Deceptive radio transmissions were broadcast on fabricated networks called "Spoof Radio." The Ghost Army leveraged the comprehensive deployment of decoys and deception techniques to overload enemy sensors and intelligence gathering capabilities.

Deception techniques and decoy technologies are employed in cyberspace in the form of honeypots. These systems may be simple (e.g., virtual machines) or complex (e.g., full-scale replicas of industrial control systems). Limited-scale industrial control system honeypots do not generate traffic that truly represent control systems. Thus, attackers who passively monitor the honeypots may be able to differentiate them from operational systems. Network traffic generators (NTGs) could increase realism and help deceive potential attackers, but they are geared for traditional information technology network performance testing as opposed to creating industrial control system decoys. This chapter introduces a proof-of-concept honeypot network traffic generator that can mimic genuine industrial control systems.

2. Background

Critical infrastructure systems have long been considered immune to network attacks that have plagued traditional information technology systems. Historically, process control and supervisory control and data acquisition (SCADA) systems have relied on proprietary hardware, software and isolation for security. However, the convergence of information technology and operational technology is pushing towards open standards based on Ethernet, TCP/IP and web-based technologies and protocols. According to Gartner [5], operational technology systems are becoming more like information technology systems, including in terms of their vulnerabilities.

The information technology/operational technology convergence introduces several security challenges. Operational technology systems often run software without updates for 15 to 20 years compared with the three to five year lifecycles of information technology systems [23]. Since the 1960s, SCADA system architecture trends show a drastic decline in the use of proprietary hardware from 60% to 2% and software from 100% to 30% [17]. As a result, security practices (e.g., security through obscurity) used in older-generation operational technology systems are no longer applicable to the newer systems [5].

Information technology systems are capable of handling multiple functions and support the addition of third-party security applications. In contrast, operational technology systems are designed to support specific industrial processes and have resource constraints. Adding resources or features to these systems may not be possible because they often lack the memory and/or computing resources to support the enhancements. Implementing traditional information technology security solutions in industrial control systems may also cause timing disruptions and may negatively impact performance and availability [23].

2.1 Control System Threats

Trend Micro [25] has published a study covering attacks on external-facing industrial control devices and honeypot technologies developed to identify threat actors and their motivations behind attacks. The study highlights five activities conducted by attackers: (i) reconnaissance using free and open-source tools

(e.g., ShodanHQ); (ii) port scanning of an intended target with an IP address and surrounding subnets; (iii) fingerprinting of devices for operating system and other identifiable information; (iv) persistence and lateral movement; and (v) data exfiltration.

A report by Idaho National Laboratory [8] highlights the security challenges associated with information technology networks that are applicable to operational technology systems as a result of convergence. Critical cyber security issues that need to be addressed in operational technology systems include: (i) backdoors and holes in network perimeters; (ii) protocol vulnerabilities; (iii) attacks on field devices; (iv) database attacks; and (v) communications hijacking and man-in-the-middle attacks. The report provides guidance for developing defense-in-depth strategies as best practices for control systems in a multi-tier information architecture. In building defense-in-depth for operational technology systems, time-sensitive requirements (e.g., clock cycles of programmable logic controllers) may make proven information technology security technologies inappropriate for control systems. Another recommendation is to use intrusion detection systems that passively analyze traffic and network activity without impacting operations. The systems compare the observed data against predefined rule sets and attack signatures. While contemporary intrusion signatures work for a wide range of attacks, they are inadequate for control networks because they render intrusion detection systems blind to attacks on industrial control systems.

2.2 Honeypots

Honeypots can help mitigate the control system threats discussed above. A honeypot is a decoy-based intrusion detection technology that attracts attackers and supports the analysis of adversary actions and behaviors [4, 11]. Honeypots can be: (i) low-interaction systems; or (ii) high-interaction systems. Low-interaction honeypots emulate services and operating systems, provide limited activity and are generally easy to deploy. High-interaction honeypots use real operating systems, applications and hardware to provide more realistic environments, but are far more difficult to deploy [7].

Hybrid honeypots can be effective at protecting industrial control systems. Winn et al. [26] have shown that honeypots can leverage proxy technology with a programmable logic controller to provide multiple instances of control devices. They also describe an approach for creating low-cost control system honeypots that are authentic and targetable by using data from a programmable logic controller while maintaining authenticity via unique network identities (e.g., IP and media access control (MAC) addresses) that match the corresponding honeypot devices. Warner [24] has proposed a framework that automatically configures emulation behavior by building protocol trees from networked programmable logic controller traces (i.e., captured network data). Girtz et al. [6] have extended Warner's work by forwarding unknown requests to a programmable logic controller proxy to provide responses and updates to a protocol tree. Their research has bolstered the targetability and authentic-

ity of industrial control system honeypots and has demonstrated the ability to emulate the characteristics and interactions of programmable logic controllers.

A successful implementation of honeypot technology can enhance cyber security by incorporating defense-in-depth measures to mitigate vulnerabilities. Complementing conventional security technologies (e.g., firewalls and intrusion detection systems) with honeypots can support early intrusion detection and threat intelligence collection against unknown vectors while providing defenders with valuable knowledge and time to address security concerns [21]. However, by design, honeypots do not manifest authorized activities and are not meant for operational use. As such, any activity in these systems can be considered suspicious [11]. A honeypot functions as a litmus test to detect unauthorized access. The downside to the approach is that honeypots do not actively engage in autonomous network communications. Instead, they rely on interactions with an attacker to generate network activity. This poses a problem because real operational technology networks have non-stop, recurring traffic flows. If an attacker were to target a honeypot, the absence of operational technology network traffic would indicate that it is not part of a real industrial control system. As a result, the honeypot would no longer entice the attacker, who would instead seek a real attack target. This reduces the effectiveness of a honeypot as a security mechanism.

2.3 Network Traffic Generation

An understanding of network architecture is necessary because network traffic visibility depends on the level of compromise. In a switched network, an attacker may only be able to map the network using broadcast messages. The segregation of traffic in a switched network would normally prohibit an attacker from observing control system traffic. Traffic collected would be restricted to the packets originating from or destined to a compromised host.

In the case of a skilled attacker, network device exploitation can offer different vantage points that enable more traffic to be visible. Using Figure 1 as an example, compromising the switch in Subnet 2 could reveal all the traffic in the subnet. Layer 3 traffic can be seen when compromising the router. Exploiting the Subnet 1 switch or the human-machine interface (HMI) would reveal traffic from all the control devices that communicate with the human-machine interface. An attacker may be able to isolate active systems from non-active systems (e.g., honeypots) by passively monitoring the traffic after compromising key nodes in the network.

It is important for an attacker to study network activity because it can increase the odds of a successful attack. If the attacker were to focus resources on the wrong target (e.g., honeypot), the cost of revealing attack information would be detrimental. As such, vigorous network discovery, reconnaissance and network enumeration actions would most likely occur prior to an attack, especially when a zero-day exploit is used. At the system level, an attacker could collect traffic data going to and from a compromised host. This would reveal the identity and function of the machine. At the network level, a compromised

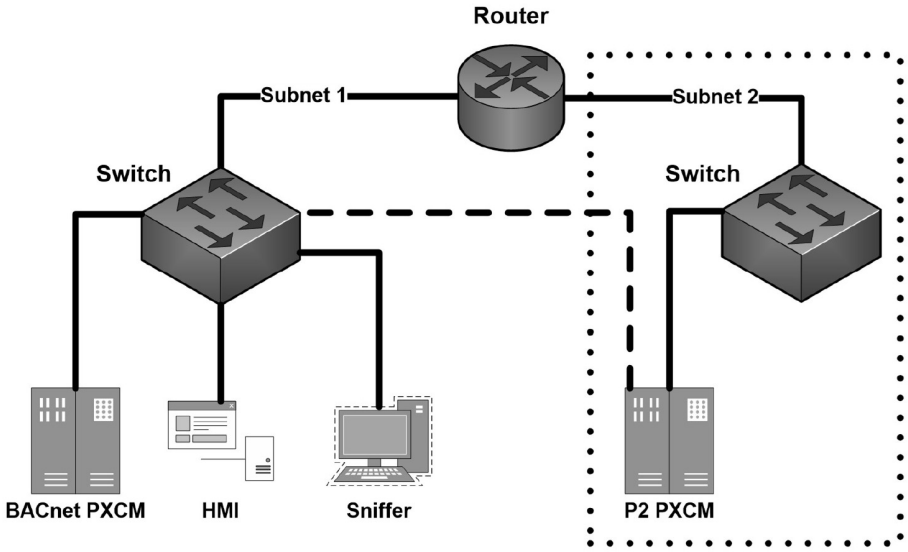


Figure 1. APOGEE platform dual subnet network design.

device provides traffic data from connected systems. With this data, an attacker could identify control devices by analyzing traffic patterns and packet content. This would also identify false targets (i.e., honeypots) due to the absence of traffic originating from the devices.

Honeypots designed for industrial control systems should have the same data traffic and patterns as the systems they are designed to emulate. Full network traffic generation would render the honeypots more effective as decoys. This chapter discusses the use of network traffic generators in conjunction with low-interaction and hybrid honeypots.

The criteria in Table 1 – comprising traffic matching, honeypot integration, network routing and scalability – are specified for assessing viable network traffic generators for industrial control system honetpots.

2.4 Network Traffic Generators

Several commercial off-the-shelf and open-source network testing products were selected as candidates for honeypot network traffic generation. They were evaluated by analyzing product literature reviews against the criteria listed in Table 1.

Commercial Network Traffic Generators. The commercial products considered included: (i) SolarWinds WAN Killer; (ii) NetLoad Stateful Traffic Mix Tester Solution; and (iii) Ixia IxLoad.

Table 1. Honeypot network traffic generator evaluation criteria.

Traffic Matching	**Content Matching:** Generated traffic should match the packet data of the control device that the honeypot is emulating. **Extraneous Packets:** Packets that do not match control system packets must be avoided during generation (e.g., generator control and traffic synchronization). **Packet Ordering:** Generated traffic should not have packets that are out of sequence. **Timing Consistency:** Generated traffic should replicate trace timing as accurately as possible.
Honeypot Integration	**Honeypot Pairing:** Generated traffic must be sent to and from the corresponding honeypot. **Honeypot Header Matching:** IP and MAC addresses in the generated packet headers should reflect the corresponding honeypot systems.
Network Routing	**Distributed Operation:** Generated traffic should originate from multiple points in a network. **Layer 2 Addressing:** Generated traffic that has the same characteristics as the associated honeypots must not cause network addressing problems (e.g., MAC table conflicts). **Layer 3 Routing:** Generated network traffic must be routable in multi-subnet environments.
Scalability	**Cost:** Industrial control devices are distributed physically and logically in a network, which requires a large number of honeypot instances. The high cost of each network traffic generator instance makes it challenging to replicate a complete control system. **Flexibility:** Industrial control systems have diverse components that are distributed across geographic locations. In order to accurately generate control system traffic, network traffic generator instances may have to be installed in multiple locations. The network traffic generators must be configurable to accommodate a large variety of industrial control system applications.

The SolarWinds WAN Killer is one of more than 60 network management tools included in the Engineer's Toolset [22]. WAN Killer enables network administrators to generate random traffic in a wide-area network. The tool can manipulate packet size, circuit bandwidth and bandwidth utilization with randomly-generated data. The tool simulates network traffic primarily for load testing and does not generate traffic corresponding to industrial control system protocols. Due to its inability to generate control network traffic, WAN Killer does not meet the traffic matching criteria.

The NetLoad Stateful Traffic Mix Tester Solution provides off-the-shelf network processing hardware and software as a single package [12, 14]. It supports

network testing by generating TCP and user datagram protocol (UDP) network traffic and providing a packet capture (PCAP) file replay feature. The PCAP replay feature generates packets using PCAP data. It can use packet timestamps to mimic the timing of the original PCAP packets. NetLoad reports an inter-packet timing (IPT) of less than one microsecond [13]. The software identifies bidirectional traffic when using PCAP replay and allows directed traffic from two ports. Other features enable the PCAP replay load to be distributed among the four ports. While the appliance can replay captured industrial control network traffic, it was designed for network testing and, therefore, does not implement honeypot integration features. The documentation does not indicate a way to load custom honeypot software on the appliance nor does it feature a way to modify PCAP data to match honeypot characteristics.

Ixia IxLoad is a suite of software and hardware that supports network performance testing [10]. The software can be used in a virtual environment loaded on a server or used in conjunction with proprietary Ixia products. IxLoad delivers a variety of stateful information technology protocols for emulating web, video, voice, storage, virtual private network, wireless, infrastructure and encapsulation/security protocols. For unsupported and propriety protocols, IxLoad provides TCP/UDP replay traffic options through its Application Replay feature [9]. This feature replays network packet captures and can create bidirectional traffic flows through unique ports. Of the three commercial solutions evaluated, IxLoad best meets the network traffic generation requirements based on product literature and vendor contacts. However, IxLoad was not designed for honeypot network traffic generation. Multiple licenses may be required for implementations in large networks, making the solutions cost prohibitive. In any case, further evaluation is needed to determine if IxLoad meets all the criteria for honeypot network traffic generation.

Open-Source Network Traffic Generators. Three open-source traffic generators were considered in this research: (i) Ostinato; (ii) Distributed Internet Traffic Generator; and (iii) Tcpreplay.

Ostinato is a network packet crafter, traffic generator and analyzer with a graphical user interface (GUI) and a Python application programming interface (API). The software operates in a controller-agent architecture, where the controller performs command and control (C2) operations and the agents generate network traffic [15]. The software generates traffic in the network using crafted packets or by replaying data from PCAP files. For the controller-agent architecture to function, Ostinato transmits trace data and device configurations from the controller to the agents during initialization. Timing limitations (i.e., packet transmission based on packets per second in burst modes) were observed during the testing. As a result, the generated traffic did not maintain the original trace timing. This limitation was identified using WireShark analysis tools. A final implementation of a honeypot network traffic generator using Ostinato would require modifications to how timing is handled by the software. Another limitation was found in the handling of command and control operations. These

operations (e.g., synchronization and state status) are continuously performed by the controller and agents. The presence of these identifiable packets in a network would likely alert an attacker to the presence of an Ostinato network traffic generator. Ostinato did not meet the traffic matching criteria as a result of these limitations.

The Distributed Internet Traffic Generator (D-ITG) produces IP traffic by replicating the workload of Internet applications [3]. D-ITG generates traffic using stochastic models for packet size and inter-packet timing that mimic supported application-level protocol behavior. The packet size and inter-packet timing data can also be loaded from capture files. Network traffic generation is performed between ITGSend (sender component of the platform) and ITGRecv (receiver component). A command and control connection exists between the two components that controls the traffic generation process for each traffic flow (e.g., port assignments). In the case of large-scale distributed environments, multiple ITGSend instances can be remotely controlled by the D-ITG API. However, the platform does not offer a method for modifying header data to match honeypot characteristics. Traffic is generated one-way from the ITGSend to ITGRecv instances. Two-way traffic generation is implemented using multiple instances of ITGSend and ITGRecv on each pair of honeypots to perform conversational traffic flow. This can be an overwhelming task in a large-scale implementation. Additionally, application layer payload data is ignored and only packet size and inter-packet timing data from traces are used. As such, D-ITG does not meet the honeypot integration and traffic matching criteria.

The Tcpreplay suite of tools enables previously-captured traffic to be replayed through various network devices. Tcpreplay can generate trace data using recorded timestamps. It also supports bidirectional traffic flow through the use of two interfaces. The suite also includes a tool for modifying packet header information. Tcpreplay was selected as a candidate for the pilot study.

3. Test Environment

The test environment incorporated the Siemens APOGEE building automation system (BAS). The environment comprised the following components:

- Siemens Insight software that provides a human-machine interface and engineering workstation functionality.
- Ubuntu VM with a Honeyd honeypot and network traffic generator software (Tcpreplay and a custom network traffic generator).
- Ubiquiti EdgeRouter X router.
- Two Netgear ProSafe Plus switches.
- Two Siemens APOGEE PXC100 programmable controller modular systems with input/output modules.
- Field level network devices (e.g., sensors, lights and fans).

Figure 2. APOGEE platform.

3.1 Design Considerations

The environment was designed to replicate an APOGEE platform that provides building automation system functionality on a single field panel as shown in Figure 2. The two PXC100 programmable controller modular (PXCM) systems were mounted side-by-side with wiring from the input/output module expansions to the field level network (FLN) devices. Serial connections were made to a Siemens Simatic S7-200 PLC and Siemens 550-833 unit conditioner circuit board with the programmable controller modular systems. User feedback and interaction were provided through sensor liquid crystal display panels, physical lights, endpoint devices and the human-machine interface.

3.2 Network Topology

The APOGEE platform incorporates three networks: (i) management level network; (ii) automation level network; and (iii) field level network.

The management level network has servers and client workstations that provide management controls for the APOGEE automation system [19, 20]. The hardware systems at this level include servers and workstations running Siemens Insight or InfoCenter software suites, web accessible terminals, mobile devices and programmable controller modular systems with management level network functionality. Communications integration is provided by proprietary and standard protocols for building automation systems.

The automation level network provides field-panel-to-field-panel communications as well as communications between the management level network and

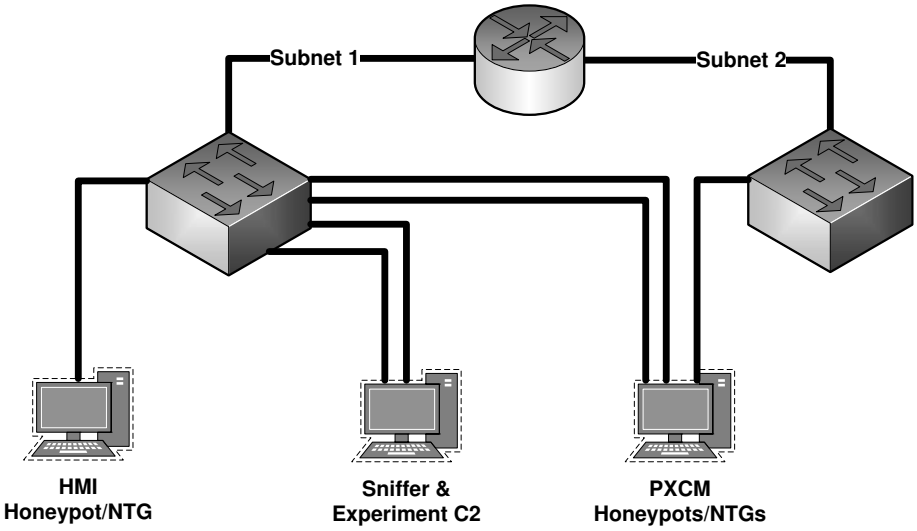


Figure 3. Test platform single and dual subnet network designs.

field level network. The hardware components comprise programmable controller modular system supervisory field panels. These panels can operate in the networked or standalone configurations and provide control, monitoring and energy management functions to field level network devices.

The field level network is the lowest level of the APOGEE building automation network. All the endpoint devices reside at this level and vary depending on the application (e.g., terminal equipment controllers and sensor units).

The network implementation for the test environment platform was based on the recommended Ethernet single subnet and multi-subnet configurations in the Siemens APOGEE technical specifications [18]. The two network topologies considered were: (i) single subnet; and (ii) dual subnet. Due to equipment availability, a dual subnet configuration was used to test the multi-subnet environment. One P2 programmable controller modular system (P2 is a Siemens proprietary protocol) was used in the single subnet and dual subnet configurations with minimal wiring and configuration changes (e.g., IP address reassignment).

Figure 3 shows the APOGEE platform network design. The design incorporates a dual subnet environment with two Siemens programmable controller modular systems: (i) P2 protocol over TCP/IP; and (ii) BACnet protocol over UDP. A network sniffer was added to a mirrored port in Subnet 1 for network capture and analysis functionality. The single subnet design represents a building automation infrastructure for a single building.

The multi-subnet design represents a multi-building infrastructure. The control networks were distributed with network connectivity provided by individual

switches. In this setup, a central router connected the multiple buildings using switches to form the building automation network infrastructure.

Honeypots and network traffic generators were employed to replicate the APOGEE platform (see Figure 3). A honeypot and network traffic generator were used to replicate the human-machine interface in Subnet 1. A network sniffer/command and control workstation were incorporated in Subnet 1 to provide: (i) network capture and analysis functionality on one network interface card connected to a mirrored port on the switch; and (ii) experimental trial automation on a second network interface card.

A third workstation, containing the programmable controller modular system honeypots and network traffic generators, was connected to both subnets. Three network interface cards were used to provide: (i) a BACnet programmable controller modular honeypot connection to Subnet 1; (ii) a P2 programmable controller modular system honeypot connection to Subnet 1; and (iii) an additional P2 programmable controller modular system honeypot connection to Subnet 2. Connections to both subnets for the P2 programmable controller modular system honeypot and network traffic generator enabled multiple experimental trials to be conducted without significant changes between runs.

4. Pilot Studies

This section highlights the results of the pilot studies.

4.1 APOGEE Network Traffic Analysis

Network communications between the programmable controller modular systems and the human-machine interface workstation were analyzed to discern what an attacker might see in the network. Traffic collection was performed on a Windows-based client using Wireshark. Multiple collections were made with different settings on each switch (e.g., switched and mirrored port configurations). In addition, network traffic was collected with the control system devices in normal operation and failed states.

The captures represent traffic that can be observed by an attacker at various levels of network compromise. The switched ports represent what an attacker who compromises a network node would see on the host. While the attacker may be able to see traffic traversing the compromised node, most traffic destined for other nodes would not be observed. The mirrored port configuration replicates the access that a skilled attacker may gain through various exploits (e.g., MAC address flooding, administrative credential compromise, switch vulnerability exploitation and switch misconfiguration). A successful network compromise could reveal all the traffic traversing the switch to an attacker.

When the sniffer was connected to a switched port, no control traffic between the programmable controller modular systems and the Insight human-machine interface workstation was collected. This is expected because unicast traffic would normally be restricted between the intended source and destination by

the switch. When alternating the programmable controller modular system and the human-machine interface workstation into failed states, address resolution protocol (ARP) and BACnet/IP broadcast management device (BBMD) requests were seen. The analysis confirmed that the system uses unicast traffic for normal operations and broadcast messages to discover nodes. Since the broadcast messages are visible without compromising the network, they provide an attacker with information for mapping the control network.

The next set of traffic collection was conducted using a mirrored port. This resulted in the sniffer capturing all unicast traffic between the programmable controller modular systems and human-machine interface workstation. The capture included routine network traffic for the APOGEE platform. By alternating the programmable controller modular system and human-machine interface workstations in failed states, unicast TCP handshakes and BBMD messages were observed in addition to the broadcast requests that were observed previously.

4.2 Identifying Honeypots

The honeypots were constructed using Honeyd [16]. The Honeyd daemon helps create virtual hosts that appear to run specific operating systems and services. It can simulate multiple addresses from a single host. Building the honeypot configuration profile involved retrieving identity data (e.g., operating system fingerprints, MAC addresses and services) from the control system devices using NMAP. The information gathered was transferred to the honeypot configuration profile for Honeyd. Note that the efficiency of honeypot software was not tested, it was only tasked with providing targetable nodes in the network. The final implementation of the network traffic generator should work with any honeypot platform.

A honeypot system with unique IP and MAC addresses was created for each control system device. Figure 4 shows the network topology obtained using an active NMAP scan. For experimentation purposes, the honeypots were configured to drop packets from the network traffic generator. In the final implementation, non-replay and replay traffic could be segregated and handled by the appropriate honeypot network traffic generator platform.

A pilot study was performed to demonstrate that the lack of control network traffic can give away the identities of the honeypots. NMAP was used to perform an active scan of the test network, which verified that the honeypot systems and control devices, were active and detectable in the network. It was also used to validate that the honeypot characteristics matched those of the corresponding APOGEE system. The network topology in Figure 4 shows three real control system devices (i.e., `10.1.3.2`, `10.1.3.3` and `10.1.3.5`) and three honeypot systems (i.e., `10.1.3.111`, `10.1.3.112` and `10.1.3.104`) for the single subnet configuration. Note that a separate honeypot (i.e., `10.1.4.104`) with the dual subnet configuration was used in Subnet 2. From the standpoint of an active scan, the introduction of the three honeypots potentially decreases an attacker's target selection success by 50% (three real control system devices

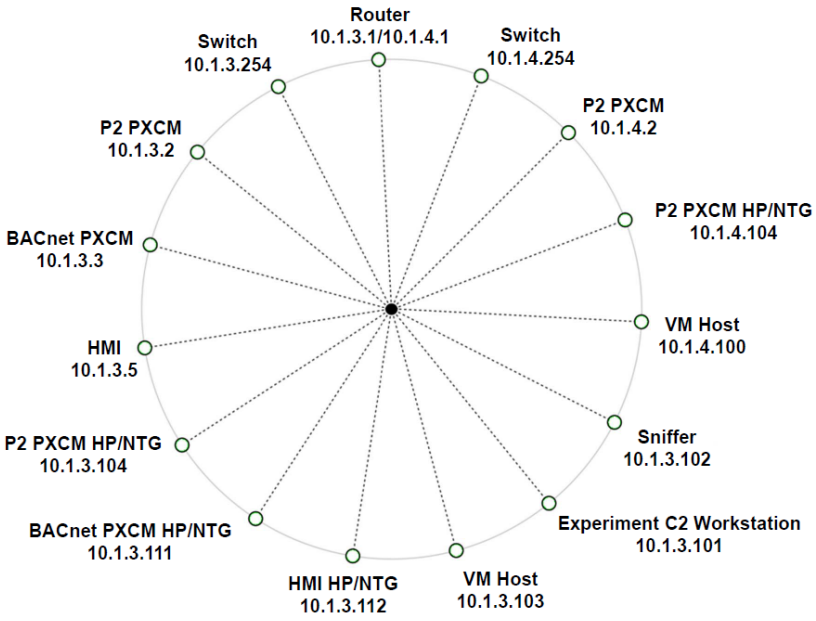


Figure 4. Test network implementation.

out of six control system devices). This target selection success rate can be further decreased by introducing additional honeypots.

An assessment of the traffic data is required because a skilled attack may involve passive network monitoring. To simulate the highest level of compromise to the network, a Wireshark sniffer was placed at a mirrored port of the switch that serviced the human-machine interface. Upon examining only a few minutes of traffic data, the three real control system devices were identified using Wireshark's endpoints statistics tool (see Figure 5). An attacker who can view network traffic would conclude that the targets of interest have the IP addresses `10.1.3.3`, `10.1.3.5` and `10.1.4.2` because only these systems actively transmitted on the network. With this additional reconnaissance, an attacker's target selection success returns to 100%. While the initial implementation of honeypot systems in the control network appeared promising, the pilot study demonstrates that a skilled attacker can easily detect a honeypot via passive monitoring.

4.3 Tcpreplay Network Traffic Generation

In this pilot study, network captures (single and dual subnet configurations) were taken from the APOGEE platform. These PCAP files are referred to as production traces. The Tcpreplay suite contains a variety of tools including: (i) `tcpprep`; (ii) `tcpreplay`; and (iii) `tcprewrite`.

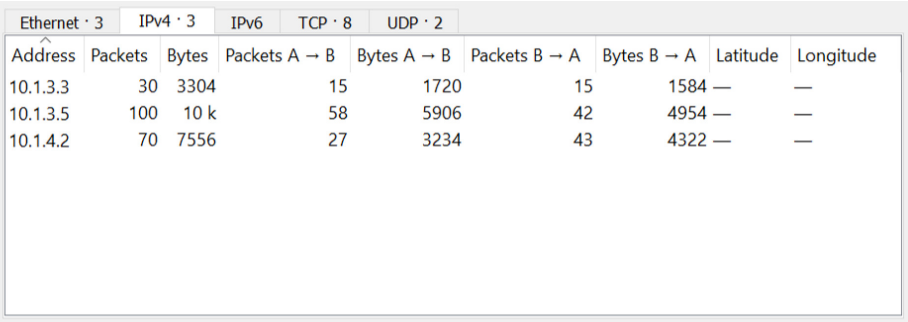


Figure 5. Pre-network traffic generation passive monitoring.

The `tcpprep` tool is a PCAP pre-processor for `tcpreplay` and `tcprewrite`. It identifies and "splits" traffic into two sides of a conversation and assigns a network interface to each packet. It enables the `tcpreplay` tool to generate network traffic through two network interface cards (primary/secondary and client/server). While it can emulate two-way traffic through two unique ports, network traffic generation is still limited to a single workstation and two interfaces. This is a limitation when simulating an industrial control environment that typically incorporates a number of networked devices that are physically and logically distributed.

The `tcprewrite` tool is a PCAP file editor that rewrites TCP/IP and Layer 2 packet headers. It replaces the Layer 2 source and destination addresses of packets so that they can be processed by the correct devices. Operational tests revealed that, while the tool could be used to rewrite packets, it was dependent on `tcpprep` being able to identify conversations properly. In fact, the tool failed to modify the source and destination addresses of all the packets in a sample capture from the APOGEE platform.

The `tcpreplay` tool was used to generate traffic in the network using the original timings recorded in the production trace. A sample capture consisting of one minute of network traffic data was recorded by the sniffer.

Figure 6 presents the results of a capture obtained by the Wireshark endpoints statistics tool. The figure shows that an attacker who can monitor the network would see traffic between all six systems, maintaining 50% target selection success probability corresponding to three real control system devices out of six control system devices. Because the real and honeypot systems generate traffic, an attacker would have to perform a deeper analysis to determine the targets of interest. This requires more steps, which provides defenders with more time to detect and mitigate the attack. This demonstrates that adding decoys with the corresponding network traffic makes it more difficult for an attacker to select a legitimate target.

After the initial test showed that `tcpreplay` was capable of deception in passive scans, a full hour of traffic was generated in Subnet 1. Wireshark was

Ethernet · 5 | IPv4 · 6 | IPv6 | TCP · 17 | UDP · 4

Address	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Latitude	Longitude
10.1.3.3	32	3566	16	1853	16	1713	—	—
10.1.3.5	104	11 k	60	6171	44	5219	—	—
10.1.3.111	30	3304	15	1720	15	1584	—	—
10.1.3.112	103	11 k	42	4992	61	6202	—	—
10.1.4.2	72	7824	28	3366	44	4458	—	—
10.1.4.104	73	7890	46	4482	27	3408	—	—

Figure 6. Post-network traffic generation passive monitoring.

used to capture the generated traffic for data collection and analysis in Subnet 1 and Subnet 2. Statistical tools provided by Wireshark were used to compare the production trace packets against the generated packets (referred to as the generated trace).

Ethernet · 4 | IPv4 · 4 | IPv6 | TCP · 58 | UDP · 3

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.1.3.111	10.1.3.112	1,803	199 k	902	103 k	901	95 k	0.000000000	3592.027962	230	212
10.1.3.111	10.1.3.255	2	120	2	120	0	0	700.733082000	1799.853148	0	0
10.1.3.112	10.1.4.104	4,308	468 k	1590	201 k	2718	266 k	0.289063000	3598.016725	448	591
10.1.3.112	10.1.3.255	6	360	6	360	0	0	700.735001000	2444.526332	1	0

System PCAP

Conversation	Rel Start (s)	Duration (s)
10.1.3.111 <> 10.1.3.112	0.000000000	3592.027962
10.1.3.111 <> 10.1.3.255	700.733082000	1799.853148
10.1.3.112 <> 10.1.4.104	0.289063000	3598.016725
10.1.3.112 <> 10.1.3.255	700.735001000	2444.526332

Figure 7. Conversation statistics from `tcpreplay` (production trace).

The traffic capture from Subnet 1 was analyzed first. Wireshark conversation statistics that list conversations (traffic between two endpoints) revealed that there were four pairs of conversations. Conversation statistics for the production trace and generated trace are shown in Figures 7 and 8, respectively. The numbers of packets transmitted and the total sizes of the conversations in the two traces matched. However, there was an average delay of 11.12 ms before the start of conversations and an average increase of 113.65 ms in the durations of conversations. The `tcpreplay` timing measurements (using methods described in Section 4.6) revealed that the difference in the inter-packet timings between the production trace and generated trace had a mean of 0.07 ms. Overall, an increase of 132 ms was observed in the one-hour duration of the replay.

Ethernet · 4 | IPv4 · 4 | IPv6 | TCP · 58 | UDP · 3

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.1.3.111	10.1.3.112	1,803	199 k	902	103 k	901	95 k	0.000000000	3592.159244	230	212
10.1.3.111	10.1.3.255	2	120	2	120	0	0	700.750636000	1799.940574	0	0
10.1.3.112	10.1.4.104	4,308	468 k	1590	201 k	2718	266 k	0.289150000	3598.148088	448	591
10.1.3.112	10.1.3.255	6	360	6	360	0	0	700.750709000	2444.630851	1	0

Generated PCAP

Conversation	Rel Start (s)	Duration (s)
10.1.3.111 <> 10.1.3.112	0.000000000	3592.159244
10.1.3.111 <> 10.1.3.255	700.750636000	1799.940574
10.1.3.112 <> 10.1.4.104	0.289150000	3598.148088
10.1.3.112 <> 10.1.3.255	700.750709000	2444.630851

Figure 8. Conversation statistics from `tcpreplay` (generated trace).

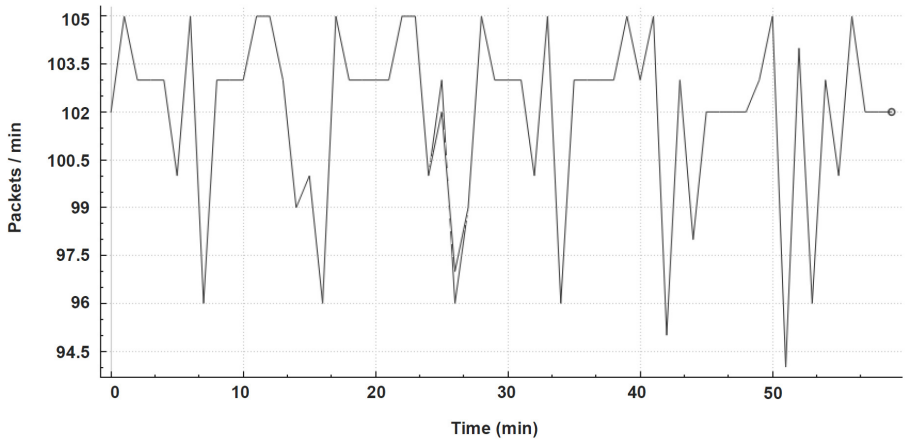


Figure 9. Packet transmission rates for sample production and generated traces.

Some minor variations in the numbers of packets sent was observed at the 22 to 28 minute mark due to processing or networking delays. The Wireshark input/output graph statistics show the numbers of packets transmitted per unit of time over the course of the capture. Figure 9 shows superimposed graphs of the packet transmission rates from a sample production trace and a sample generated trace. The superimposition reveals that the two traces have similar network traffic patterns. However, it would be difficult for an attacker to use this visual representation to identify the traffic pattern that is associated with the real system.

The pilot study revealed some limitations of the `tcpreplay` tool. Sniffing in Subnet 2 revealed that only a limited number of packets destined for the P2 programmable controller modular system honeypot were received. This is attributed to the port assignments made by the switch in its MAC address

table and the way in which `tcpreplay` operates. Since `tcpreplay` transmitted the generated traffic via an interface in Subnet 1, all the MAC addresses from the production trace were entered into the address table of the switch and assigned to a particular port in Subnet 1. All network communications (including those outside the generated traffic) destined for the P2 programmable controller modular system honeypot were then sent to this port. Occasionally, the MAC tables refreshed with the actual locations of the honeypot (via the router MAC address). During these periods, packets were forwarded correctly to the router and the correct subnet. This is a limitation in using `tcpreplay` for distributed systems in multi-subnet environments.

If the network topology had comprised a single subnet, then an implementation using the `tcpreplay` tool would be a viable solution. It would also require that the honeypots be collocated on the same workstation as `tcpreplay`; otherwise, multiple associations of non-unique MAC addresses would occur. If any honeypot were to be placed separately from the network traffic generator, an attacker would not see any replayed traffic on the host. This imposes a hardware limitation for designs that require multiple honeypots in the same workstation. Indeed, the pilot study reveals that the `tcpreplay` suite does not meet the honeypot integration, network routing and scalability criteria specified in Table 1.

5. Implementation

While authentic network traffic can be generated by a full-scale system, the cost of such a decoy can be prohibitive [26]. Low-interaction and hybrid honeypot systems can be used to emulate industrial control systems at a much lower cost. However, protocol-specific traffic generation requires significant knowledge, time and resources to replicate the operations of a genuine system with high fidelity.

An alternate solution is to use packet captures for network traffic generation. This method maintains authenticity because the data and patterns are derived from real systems. However, the confidentiality of the traffic is not maintained because a capable attacker would most likely be able to view the traffic in a real system in the absence of honeypots. One way to mitigate this and maintain some form of confidentiality is to use multiple sets of false captures. The false captures would be generated by real systems that run in a fake operational environment. The traffic would look real, but it would have no operational use. Thus, the real system and the honeypots would have distinct traffic, but the honeypots would still function as decoys and present themselves as attractive targets to an attacker.

The experimental network incoporated five components: (i) production network; (ii) honeypot platform; (iii) honeypot integration; (iv) distributed network traffic generation; and (v) decoy network. Figure 10 shows a block diagram of the design. The production network provided device characteristics (e.g., operating system fingerprints, network addresses and trace data). The honeypot platform provided targetable honeypots in a decoy network with

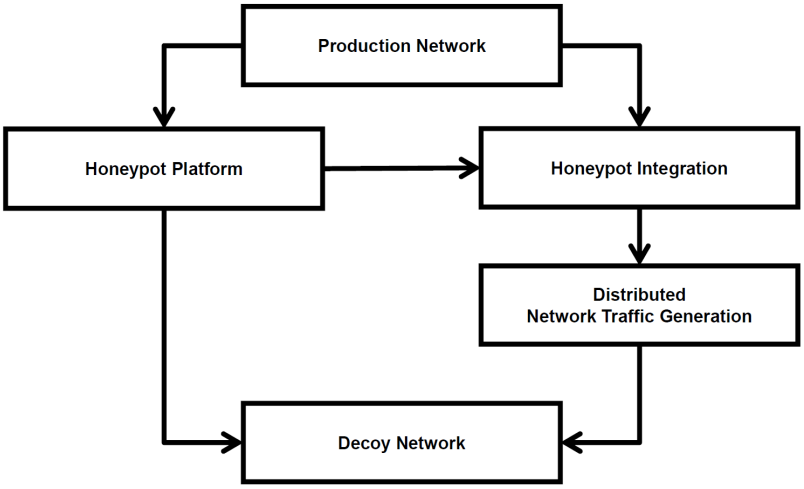


Figure 10. Honeypot-network traffic generation design.

matching characteristics and emulated services associated with the production network. Honeypot integration modified the headers of trace data captured from a production network to match the corresponding honeypot systems. After the honeypot integration was completed, the trace data was used by the distributed network traffic generators to produce and replay industrial control system traffic in the decoy network.

A custom network traffic generator solution was designed due to various limitations in common off-the-shelf and open source products. The distributed network traffic generator (DNTG) was created using Python and Scapy following the design criteria in Table 1 and drawing on the design characteristics of D-ITG [2]. DNTG has two main components: (i) DNTG Prep; and (ii) DNTG Replay. DNTG Prep is a PCAP tool that modifies packet header information to match the honeypot characteristics (i.e., IP and MAC addresses). These characteristics are passed as parameters to DNTG Prep, which modifies the control system packets with the corresponding honeypot information.

DNTG Replay was created to run on each corresponding honeypot device and to operate in a distributed environment. Each DNTG Replay instance functions as a listener and sender. As a listener, DNTG Replay continuously waits and listens for incoming packets. When a packet is received, DNTG Replay searches the production trace to verify if the packet corresponds to generated traffic. If a match is found, DNTG Replay searches for the corresponding responses. The appropriate responses are then placed in a queue for network traffic generation. As a sender, DNTG transmits the queued packets based on the inter-packet timing determined from the production trace. Figure 11 shows an example DNTG architecture containing three DNTG Replay instances in a decoy network.

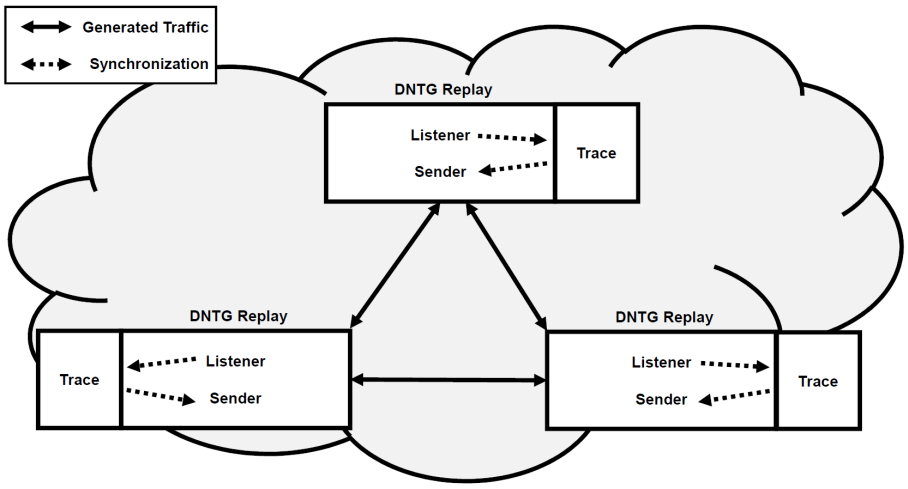


Figure 11. Example DNTG architecture.

5.1 Traffic Matching

Synchronization is important for multiple DNTG instances to operate in a distributed environment. However, no extraneous packets are allowed during network traffic generation per the criteria in Table 1. Without any command and control traffic, synchronization is dependent on the generated packets. If a network traffic generator is not ready to receive, it may inadvertently miss incoming packets. It is crucial during initialization that each DNTG is ready before traffic generation begins. To perform the initial synchronization, a master DNTG is selected based on the first packet in the production trace. The DNTG instance with the originating IP source address of the packet becomes the master and all other instances listen for network traffic upon initialization. The master DNTG sends a periodic UDP request to the other DNTG instances. When a request is received, a response is sent back to the master. After the responses from all the DNTG instances are received, the master DNTG sends a UDP start packet to each DNTG instance to begin traffic generation. No additional synchronization packets are required after this initial period.

Each DNTG uses an identical production trace and operates asynchronously. By removing command and control traffic during traffic generation, each DNTG is responsible for keeping track of the current location in the production trace and resynchronizing based on the received packets. As each DNTG receives and sends replay packets, it resynchronizes with the production trace. These command and control operations are transparent to an attacker because they are performed locally and not over the network. Only production trace data is transmitted by each DNTG instance during network traffic generation.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.4.104	10.1.3.112	TCP	60	52810→5033 [ACK] Seq=1 Ac
2	0.019935	10.1.4.104	10.1.3.112	TCP	60	39482→5402 [ACK] Seq=1 Ac
3	1.708573	10.1.3.112	10.1.3.111	BACnet-APDU	67	Confirmed-REQ readPrope
4	1.792864	10.1.3.111	10.1.3.112	BACnet-APDU	91	Complex-ACK readPrope
5	1.797696	10.1.3.112	10.1.3.111	BACnet-APDU	132	Confirmed-REQ confirmed
6	1.802435	10.1.4.104	10.1.3.112	TCP	139	19394→5400 [PSH, ACK] Seq
7	1.807349	10.1.3.111	10.1.3.112	BACnet-APDU	130	Complex-ACK confirmed
8	1.831848	10.1.3.112	10.1.4.104	TCP	144	5400→19394 [PSH, ACK] Seq

Figure 12. Production trace sample.

Conversation: 10.1.4.104 <> 10.1.3.112

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.4.104	10.1.3.112	TCP	60	52810→5033 [ACK] Seq=1 Ac
2	0.019935	10.1.4.104	10.1.3.112	TCP	60	39482→5402 [ACK] Seq=1 Ac
3	1.802435	10.1.4.104	10.1.3.112	TCP	139	19394→5400 [PSH, ACK] Seq
4	1.831848	10.1.3.112	10.1.4.104	TCP	144	5400→19394 [PSH, ACK] Seq

Conversation: 10.1.4.111 <> 10.1.3.112

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.3.112	10.1.3.111	BACnet-APDU	67	Confirmed-REQ readPrope
2	0.084291	10.1.3.111	10.1.3.112	BACnet-APDU	91	Complex-ACK readPrope
3	0.089123	10.1.3.112	10.1.3.111	BACnet-APDU	132	Confirmed-REQ confirmed
4	0.098776	10.1.3.111	10.1.3.112	BACnet-APDU	130	Complex-ACK confirmed

Figure 13. Sample conversations.

To meet the packet ordering criteria, the DNTG was designed to handle multiple conversations simultaneously. The DNTG parses the production trace and separates the packets into conversations. All traffic to and from an IP address pair are identified as a single conversation to limit the amount of data processed by each DNTG. As such, a conversation is defined as a traffic flow between two unique IP addresses. Using multi-threading, each conversation is processed independently to enable unique conversations to intermingle. This adds variability to the overall capture while maintaining packet order within each conversation. Figure 12 shows packets from a production trace and Figure 13 shows how the packets are segmented into conversations.

Timing consistency is required to replicate the original system as accurately as possible. Autonomous industrial control network traffic consists of routine/polling messages that are sent and received at timed intervals. The DNTG should replicate these intervals (inter-packet timing from the production trace) and not use transmission timing methods such as packets per second or burst modes.

Two methods were considered to handle the timing: (i) compute the time between the current queued packet and the first packet of the conversation; and (ii) use the inter-packet timing. The first method implements a catch up

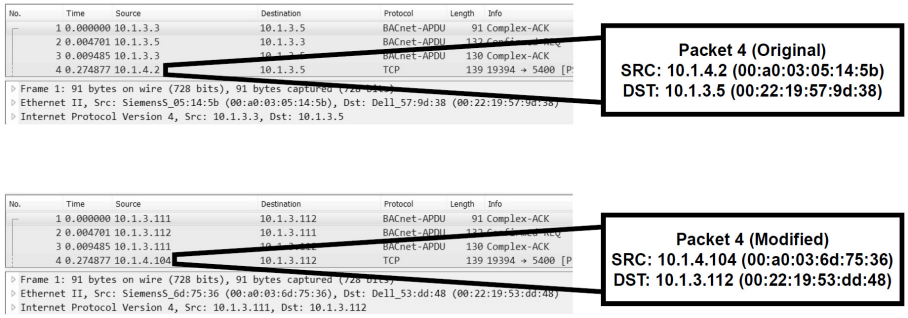


Figure 14. Production trace modification.

algorithm to ensure that processing and network delays do not add to the overall length of traffic generation (start to finish); however, significant changes in inter-packet timing were observed. The second method uses inter-packet timing to generate packets without a catch up algorithm. This method was chosen for implementation; although, the overall length of a generated trace is longer than that of a production trace, the inter-packet timing is more consistent with the original intervals.

5.2 Honeypot Integration

As shown in Figure 1, the main network traffic collection occurs in Subnet 1. Since control system devices communicate with the human-machine interface located on this subnet, this is the best traffic collection point for an attacker. To obtain the production trace, network traffic was collected from a mirrored port for ten minutes. IP address filters were used to capture network traffic only from the APOGEE platform. In the single subnet, the IP addresses `10.1.3.2`, `10.1.3.3` and `10.1.3.5` were captured. In the dual subnet setup, the IP addresses `10.1.4.2`, `10.1.3.3` and `10.1.3.5` were captured. A separate PCAP file was created for each subnet configuration.

The production trace used for traffic generation contains characteristics of the real control system devices. Since the generated traffic must match the characteristics of the honeypot systems (i.e., IP and MAC addresses), the trace requires modification. Preparation of the production trace using DNTG Prep was chosen instead of altering the values during traffic generation. This reduces the impact on DNTG Replay performance during runtime. Each packet in the production trace was thus overwritten with the desired replacement IP and MAC addresses of the corresponding honeypot system. Checksum values were also corrected to maintain packet validity. The top portion of Figure 14 shows the original production trace with the addresses reflecting the real control system devices. The bottom portion of the figure shows the modified production trace with the replacement addresses of the honeypot systems.

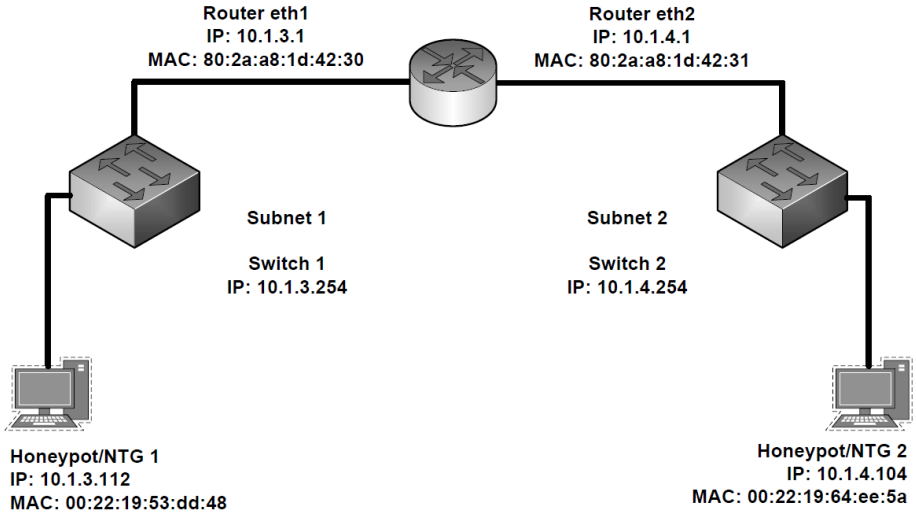


Figure 15. Example dual subnet environment.

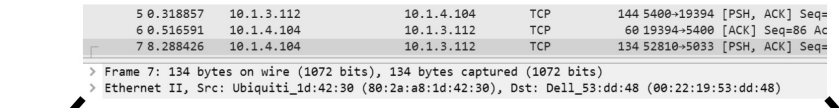
5.3 Network Routing

The network routing criteria focus on the ability of DNTG instances to operate in a distributed environment. To meet this requirement, the generated traffic must have proper Layer 2 and Layer 3 addresses.

Layer 2 addressing requires that generated traffic that shares MAC addresses with the corresponding honeypots must not cause network addressing problems. To eliminate Layer 2 networking problems, network traffic generators that share IP and MAC addresses with a honeypot must be collocated on the same workstation.

Proper Layer 3 routing ensures that DNTG instances can operate in a multi-subnet environment. Figure 15 shows an example dual subnet topology.

Recall that the production trace was captured in Subnet 1 and all traffic from outside subnets would have recorded the MAC address of router `eth1` as the source (shown in Figures 15 and 16). Problems arise when generating traffic from devices in Subnet 2 with trace data captured in Subnet 1. Using packet 7 in Figure 16 as an example, the header contains incorrect address values: (i) the source is the MAC address of router `eth1` (`80:2a:a8:1d:42:30`); and (ii) the destination is the MAC address of NTG 1 (`00:22:19:53:dd:48`). DNTG identifies these instances and modifies the header with the correct values: (i) the new source is the MAC address of NTG 2 (`00:22:19:64:ee:5a`); and (ii) the new destination is the MAC address of router `eth2` (`80:2a:a8:1d:42:31`). These corrections are performed during runtime to avoid customizing the production trace for each DNTG instance.



Source	Destination
10.1.4.104	10.1.3.112
Ubiquiti_1d:42:30 (80:2a:a8:1d:42:30)	Dell_53:dd:48 (00:22:19:53:dd:48)

Figure 16. Example MAC addressing problem.

6. Experiments

Multiple experimental trials were conducted to evaluate if the DNTG implementation satisfies the network traffic generation criteria listed in Table 1. Each experimental trial involved generating network traffic from a ten-minute production trace. The trials were conducted using an automated script that alternated the operating environments (single and dual subnets). A total of 177 iterations were conducted for each environment, corresponding to a grand total of 354 experimental trials. The trials were conducted over a 65-hour time period and the outputs provided more than 375,000 sample packets for analysis.

6.1 Metrics

This section outlines the metrics used to validate the DNTG implementation against the criteria listed in Table 1.

Table 2. Traffic matching metrics.

Content Matching	**Packet Bytes:** Generated packets bytes match the production trace bytes.
Extraneous Packets	**Quantity of Packets:** Number of generated packets match the number of production trace packets.
Packet Ordering	**Packet Order:** Generated conversations match the production trace conversations.
Timing Consistency	**Δ Inter-Packet Time:** Generated traffic timing patterns match the production trace timing patterns.

Traffic Matching. Table 2 describes the metrics used to evaluate the DNTG implementation against the traffic matching criteria. Content matching

is evaluated by directly comparing corresponding packets between the generated trace and production trace. Two packets match if both packets contain the same bytes. A trial is considered to be successful if every packet in the generated trace matches the corresponding packet in the production trace.

The generated trace is determined to have no extraneous packets if it contains the same number of packets as the production trace. A trial is considered to be successful if the quantity and content of the packets in the generated trace match those in the production trace.

Packet ordering is determined to match if the packet orders of conversations in the generated trace match the packet orders of the corresponding conversations in the production trace. A trial is considered to be successful if every packet in the generated trace is in the correct order.

Timing consistency is measured by comparing the inter-packet timing of packets in the generated trace with the inter-packet timing of packets in the production trace.

The inter-packet time of packet n in the generated trace $GIPT_n$ is computed as:

$$GIPT_n = GT_n - GT_{n-1} \tag{1}$$

The inter-packet time of packet n in the production trace $PIPT_n$ is computed as:

$$PIPT_n = PT_n - PT_{n-1} \tag{2}$$

Finally, the difference in the inter-packet timing in the two traces ΔIPT is computed as:

$$\Delta IPT_n = ABS(GIPT_n - PIPT_n) \tag{3}$$

The Wilcoxon rank-sum non-parametric statistical test was used to compare the distribution of $GIPT$ values to the distribution of $PIPT$ values for each generated trace. A significance level of 0.05 was used to determine if the two distributions are statistically similar. A trial is considered to be a success if the Wilcoxon rank-sum test returned a p-value greater than 0.05.

Honeypot Integration. Honeypot pairing was evaluated based on direct comparisons of corresponding packets from the generated and production traces. The DNTG was designed to only generate traffic if it started a conversation or was responding to a received packet. In addition, the DNTG was hosted on the same honeypot workstation and used matching IP and MAC addresses. Honeypot pairing was determined to be a match if packets were sent and received from each honeypot workstation. To validate the honeypot header matching, an NMAP scan was used to obtain the IP and MAC addresses of each honeypot. This information was then compared against the production trace and generated traces to validate that the generated traffic matched the honeypots. A trial is considered to be successful if every packet header matched the intended honeypot information and every packet satisfied the content matching and extraneous packets criteria.

Table 3. Traffic matching criteria success rates.

	Trials	**Single Subnet**	**Dual Subnet**
Content Matching	354	100%	100%
Extraneous Packets	354	100%	100%
Packet Ordering	354	100%	100%
Timing Consistency	354	0%	0%

Network Routing. The DNTG was designed to run with multiple instances at various locations in a network. Network traffic generation involves each DNTG instance receiving and generating network traffic. A trial is considered to be successful if every packet in the generated trace and the corresponding production trace satisfies the traffic matching criteria. A successful trial implies that the distributed operation, Layer 2 addressing and Layer 3 routing evaluation criteria are satisfied.

Scalability. The scalability of a network traffic generator is based primarily on the design, implementation and pricing of the final implementation. This research did not consider the actual costs (in terms of dollars), because most instantiations would require engineering efforts to determine the proper placement of the traffic generators. While the DNTG is designed to be as flexible as possible, the experiments were limited to the Siemens APOGEE system. Experiments with other implementations are left for future work.

6.2 Experimental Results

A total of 354 experimental trials were conducted and evaluated against the criteria specified in Table 1.

Traffic Matching. All 354 trials achieved 100% matching of the production and generated traces for the metrics: (i) packet bytes; (ii) quantity of packets; and (iii) packet ordering. Table 3 shows the traffic matching success rates for the various criteria.

Timing consistency was evaluated by using a Wilcoxon rank-sum test. This test compared the $GIPT$ distribution in each of the 354 generated traces with the $PIPT$ distribution in the production trace. The Wilcoxon rank-sum test returned a p-value less than 0.05 for all 354 generated trace distributions. Therefore, the generated traffic timing does not match the production traffic timing.

Table 4 shows the ΔIPT summary for all 354 trials. Mean differences of 2.07 ms (single subnet) and 2.26 ms (dual subnet) were observed between the production and generated traces. Further examination using a Wilcoxon rank-sum test revealed that that the generated traces from multiple trials are consistent with each other.

Table 4. ΔIPT (ms) summary.

	Mean	Min	Max	SD
Single Subnet	2.07	0.00	95.78	1.59
Dual Subnet	2.26	0.00	98.93	1.68

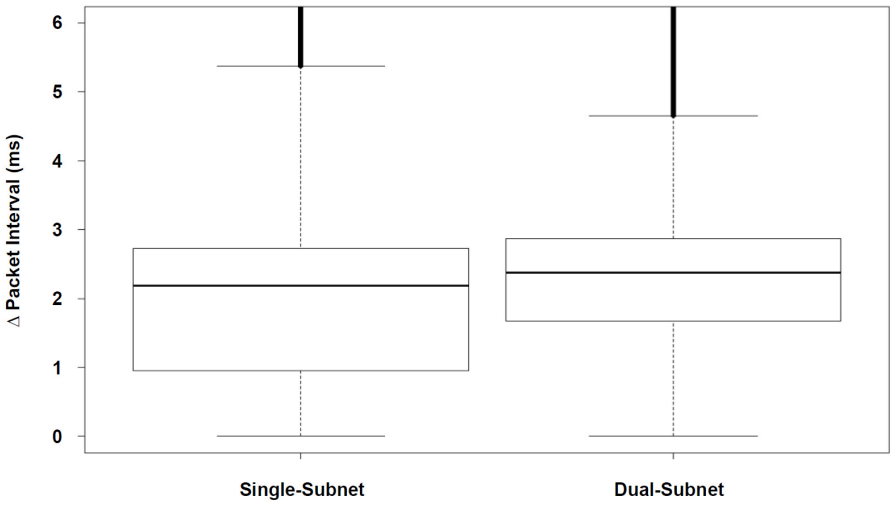


Figure 17. Difference in timing between system and generated intervals.

Figure 17 shows a boxplot of two subnet configurations with similar timing. Figure 18 shows an overlay of the production trace (line) and a sample generated trace (points). Figure 19 shows detailed views of the intervals between packets 800 and 820. It is difficult to visually distinguish the real system from the honeypot system.

The DNTG implementation successfully replicates control system traffic in a distributed environment. While the results indicate that the production and generated traces do not have the same timing, multiple runs demonstrated consistency in the DNTG outputs. This suggests that optimizing the software could reduce the timing differences.

Honeypot Integration. All 354 trials resulted in 100% matches between the production and generated traces for the criteria: (i) content matching; (ii) extraneous packets; and (iii) packet ordering. Satisfaction of these three criteria demonstrates that traffic was generated to (and from) the honeypot network traffic generator pair. In addition, because the generated traffic matches the production trace (validated against the honeypots using NMAP), the honeypot

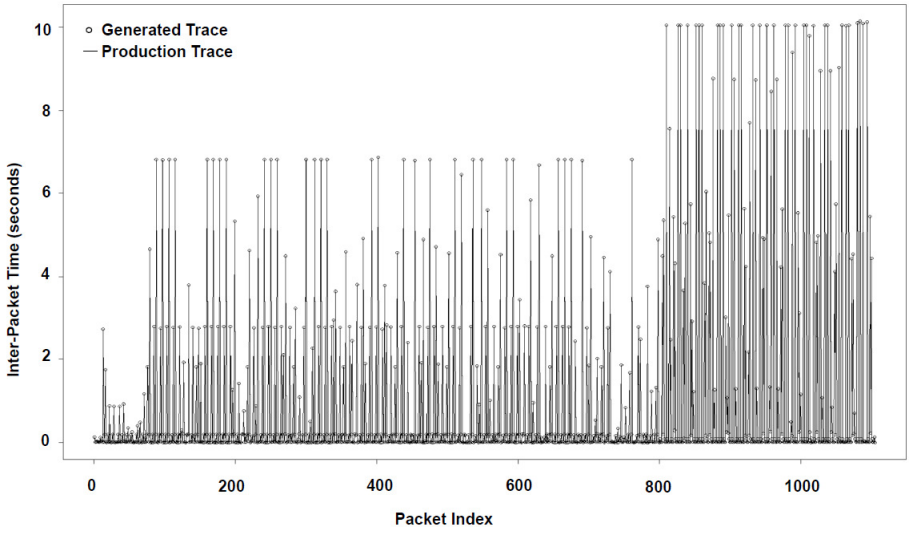


Figure 18. Traffic timing pattern.

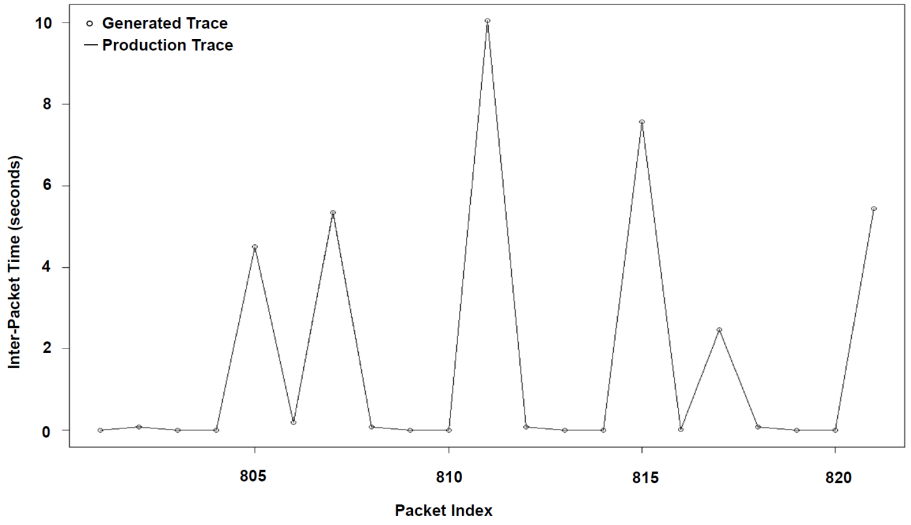


Figure 19. Detailed traffic timing pattern.

header matching criterion is also satisfied. Table 5 shows the honeypot integration criteria pass rates. Figure 20 shows an NMAP active scan and a Wireshark passive mapping of the decoy network used during the experiment. The figure shows that three honeypot systems are detected during an active network scan.

Table 5. Honeypot integration criteria pass rates.

	Trials	Single Subnet	Dual Subnet
Honeypot Pairing	354	100%	100%
Honeypot Header Matching	354	100%	100%

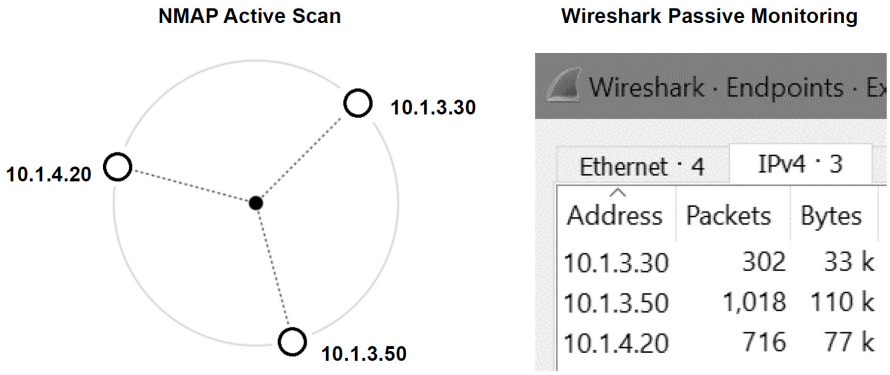


Figure 20. Active and passive network mapping.

The network traffic generated by the DNTG is observed during passive network monitoring. The combination of honeypot systems and network traffic mimic an operating industrial control system.

Table 6. Network routing criteria pass rates.

	Trials	Single Subnet	Dual Subnet
Distributed Operation	354	100%	100%
Layer 2 Addressing	354	100%	100%
Layer 3 Routing	354	100%	100%

Network Routing. All 354 trials achieved 100% matches between the production trace and generated traces for the criteria: (i) content matching; (ii) extraneous packets; and (iii) packet ordering. Satisfaction of these three criteria demonstrates that traffic was generated to and from multiple instances of the DNTG located in two different network configurations. This shows that Layer 2 addressing and Layer 3 routing were successfully accomplished and validates that the DNTG satisfies the distributed operation criteria. Table 6 shows the network routing criteria pass rates.

Scalability. The DNTG is designed to be cost effective and flexible, but the research did not evaluate these characteristics. The evaluation of cost effectiveness and flexibility is left for future work.

7. Conclusions

Honeypots can be used to protect industrial control systems. The effectiveness of a honeypot is dependent on its ability to entice attackers to select it as a target. Using network traffic generators to create traffic in honeypot networks helps build a realistic decoy industrial control system environment. Indeed, the experimental results demonstrate that network traffic generation can significantly complicate the task of honeypot identification during target selection by an attacker.

Future research will focus on alternating production traces or modifying packet data and values to maintain uniqueness during multiple iterations. This is important because replays of trace data are only as effective as the lengths of the traces. For example, if an hour-long production trace were to be used, the first hour of generated traffic would appear to be authentic. However, after this time period, the same packets would be generated again; this traffic would be automatically highlighted as a retransmission by several tools, including Wireshark.

Another topic for future research is motivated by the fact that the use of trace data does not account for real-time system changes. State changes made during operations may contradict traffic data generated by the DNTG. This would be challenging to implement because it requires detailed knowledge about industrial control system protocols. One approach is to use "live" production data to update the network traffic generator. Future research will attempt to develop and evaluate possible solutions to this problem.

Note that the views expressed in this chapter are those of the authors and do not reflect the official policy or position of the U.S. Air Force, U.S. Army, U.S. Department of Defense or U.S. Government.

Acknowledgement

This research was partially supported by the U.S. Department of Homeland Security Industrial Control Systems Cyber Emergency Response Team (ICS-CERT).

References

[1] Armed Forces History Museum, World War II's U.S. Ghost Army, Largo, Florida (`www.armedforcesmuseum.com/world-war-iis-us-ghost-army`), February 5, 2014.

[2] A. Botta, A. Dainotti and A. Pescape, A tool for the generation of realistic network workload for emerging networking scenarios, *Computer Networks*, vol. 56(15), pp. 3531–3547, 2012.

[3] A. Botta, W. de Donato, A. Dainotti, S. Avallone and A. Pescape, D-ITG 2.8.1 Manual, Computer for Interaction and Communications (COMICS) Group, Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Naples, Italy (www.grid.unina.it/software/ITG/manual), 2013.

[4] L. Even, IDFAQ: What is a Honeypot? SANS Institute, Bethesda, Maryland (www.sans.org/security-resources/idfaq/what-is-a-honeypot/1/9), July 20, 2000.

[5] Gartner, Gartner says the worlds of IT and operational technology are converging, Press Release, Stamford, Connecticut, March 16, 2011.

[6] K. Girtz, B. Mullins, M. Rice and J. Lopez, Practical application layer emulation in industrial control system honeypots, in *Critical Infrastructure Protection X*, M. Rice and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 83–98, 2016.

[7] J. Harrison, Honeypots: The sweet spot in network security, *Computerworld*, November 20, 2003.

[8] Idaho National Laboratory, Control Systems Cyber Security: Defense in Depth Strategies, External Report INL/EXT-06-11478, Idaho Falls, Idaho, 2006.

[9] Ixia, IxLoad Application Replay, Data Sheet, Document No. 915-1744-01 Rev D, Calabasas, California, 2013.

[10] Ixia, IxLoad Overview – Converged Multiplay Service Validation, Data Sheet, Document No. 915-1710-01-2161 Rev S, Calabasas, California, 2016.

[11] I. Mokube and M. Adams, Honeypots: Concepts, approaches and challenges, *Proceedings of the Forty-Fifth Annual ACM Southeast Regional Conference*, pp. 321–326, 2007.

[12] NetLoad, Test Traffic Solution, Danville, California (www.netloadinc.com/manuals/NetLoadInc_Products.pdf), 2014.

[13] NetLoad, Stateful Traffic Mix Tester Solutions, Danville, California (www.netloadinc.com/manuals/NetLoad_Inc_Brief.pdf), 2015.

[14] NetLoad, User Guide for NetLoad Product Series (Revision 8.9), Danville, California (netloadinc.com/manuals/NetLoadInc._Startup_Guide.pdf), 2016.

[15] Ostinato, Ostinato User Guide (www.gitbook.com/book/pstavirs/ostinato-user-guide/details), 2016.

[16] N. Provos, A virtual honeypot framework, *Proceedings of the Thirteenth USENIX Security Symposium*, article no. 1, 2004.

[17] W. Shaw, *Cybersecurity for SCADA Systems*, PennWell Corporation, Tulsa, Oklahoma, 2006.

[18] Siemens, APOGEE Building Level Network on TCP/IP, Technical Specification Sheet (Revision 1), Document No. 149-967, Buffalo Grove, Illinois, 2003.

[19] Siemens, Siemens 2011 APOGEE Brochure (Revision 8), Document No. 153-301 P10, Buffalo Grove, Illinois, 2011.

[20] Siemens, Siemens APOGEE Scalable Brochure, Document No. 611-056, Buffalo Grove, Illinois, 2012.

[21] S. Smith, Catching Flies: A Guide to the Various Flavors of Honeypots, InfoSec Reading Room, SANS Institute, Bethesda, Maryland, 2016.

[22] SolarWinds, WAN Killer: Network Traffic Generator with Engineer's Toolset, Austin, Texas (`www.solarwinds.com/engineers-toolset/wan-killer`), 2016.

[23] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams and A. Hahn, Guide to Industrial Control Systems (ICS) Security, NIST Special Publication 800-82, Revision 2, National Institute of Standards and Technology, Gaithersburg, Maryland, 2015.

[24] P. Warner, Automatic Configuration of Programmable Logic Controller Emulators, M.S. Thesis, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2015.

[25] K. Wilhoit, The SCADA That Didn't Cry Wolf – Who's Really Attacking Your ICS Equipment? (Part 2), Research Paper, Trend Micro, Cupertino, California, 2013.

[26] M. Winn, M. Rice, S. Dunlap, J. Lopez and B. Mullins, Constructing cost-effective and targetable industrial control system honeypots for production networks, *International Journal of Critical Infrastructure Protection*, vol. 10, pp. 47–58, 2015.