TCP veto: A novel network attack and its application to SCADA protocols

John T. Hagen, *Member, IEEE*, and Barry E. Mullins, PhD, *Senior Member, IEEE*

***Abstract*—TCP veto is a detection-resistant variation of the TCP connection hijacking attack. While not limited to SCADA protocols, Modbus TCP, the Ethernet Industrial Protocol (EtherNet/IP), and the Distributed Network Protocol (DNP3) each meet the necessary assumptions of the attack. Experimental results reveal that the integrity of messages transmitted using each of the three SCADA protocols are vulnerable to TCP veto. Additionally, TCP veto produces up to 600 times less network traffic during its attack than connection hijacking. This work underscores the vulnerability of current SCADA protocols that communicate over TCP/IP to network attack. A method to definitively identify TCP veto requires a detection system to perform deep packet inspection on every TCP packet of a monitored connection. Methods for mitigating the attack through message authentication include implementing DNP3 with Secure Authentication, tcpcrypt, or Internet Protocol Security (IPsec).**

***Index Terms*—Cyberspace, Intrusion detection, IP networks, Message authentication, Network security, SCADA systems, TCPIP**

I. Introduction

CRITICAL infrastructure such as electric, oil, and gas intersect with cyberspace in the form of technologies that perform Supervisory Control And Data Acquisition (SCADA). Specialized legacy devices originally designed to communicate over serial networks are adapted for the Transmission Control Protocol and the Internet Protocol (TCP/IP) to increase scalability, bandwidth, and communication distance. Inherent with this transition is a dependence on protocols that were never designed to support end-to-end authentication or encryption. This property means endpoint devices on SCADA networks are vulnerable to a variety of classic TCP network attacks.

The goals of this research are threefold:

1. Describe TCP veto, a variation of the connection hijacking attack that is more resistant to detection.
2. Experimentally demonstrate that TCP veto compromises the integrity of several common SCADA protocols.
3. Provide methods to detect and mitigate TCP veto.

This work was supported in part by the Center for Cyberspace Research, Air Force Institute of Technology, and the Department of Homeland Security.

J. T. Hagen is with the Electrical and Computer Engineering Department, Air Force Institute of Technology, WPAFB, OH 45433 USA (email: johnthagen@ieee.org)

B. E. Mullins is with the Electrical and Computer Engineering Department, Air Force Institute of Technology, WPAFB, OH 45433 USA (email: barry.mullins@afit.edu)

II. Background

The following topic areas are necessary for understanding the novel attack described in this research and the experimentation conducted.

A. Network Security

Security is often defined with the Confidentiality, Integrity, and Availability (CIA) model. While each of these principles have potentially critical implications to SCADA networks, this research focuses on attacks that compromise integrity, specifically origin integrity. Origin integrity, or authentication, is the principle that a receiver can verify the identity of the sender of a message.

This paper uses the following actors to describe TCP veto and perform experimentation:

- Client – The client initiates communication with the server. In SCADA networks this is typically the Human Machine Interface (HMI).
- Server – The server listens for communication requests from and provides services to the client. In SCADA networks this is typically the Programmable Logic Controller (PLC).
- Attacker – The malicious attacker's goal is to compromise the integrity of the communication between the client and server.

B. Connection Hijacking

Connection hijacking is an active attack against TCP in which the attacker creates a desynchronized state between the client and server so that they can no longer exchange data [1]. The attacker then creates synchronized packets that mimic those being transmitted by the client and server, thereby allowing the attacker to compromise the integrity of the communication between the client and server by modifying their messages or injecting additional messages.

Joncheray proposes two methods for creating a desynchronized TCP connection, early desynchronization and null data desynchronization [1]. Early desynchronization consists of resetting the TCP connection between the client and server in its early stage and creating new connections with each side using different sequence numbers. To perform null data desynchronization, the attacker injects null data (data that does not affect the state of the client or server application) into the connection to increment the sequence numbers into a desynchronized state.

A weakness of connection hijacking is that desynchronizing a TCP connection creates what Joncheray termed as an "ACK storm." After the client and server are desynchronized, each transmitted, desynchronized packet is acknowledged by an ACK packet (a TCP packet with no payload) containing the sequence numbers that the receiver believes to be correct. This packet in turn is acknowledged by another ACK packet, creating a seemingly endless loop of ACK packets. As a result, the ACK storm generates a large amount of extraneous network traffic, making connection hijacking detectable through several methods. Joncheray reports that on a local Ethernet connection, a single desynchronized packet generates between 10 and 300 ACK packets [1].

C. SCADA Protocols

Modbus TCP is a member of the Modbus protocol family that operates over TCP/IP [2]. The Modbus protocol was originally developed by Modicon in 1979 and was later transferred to the Modbus Organization to become an open standard. Modbus TCP features a simple design that utilizes function codes to specify request and response format. Modbus TCP communication is identified by its use of TCP port 502.

The Ethernet Industrial Protocol (EtherNet/IP) is another SCADA protocol that operates over TCP/IP. EtherNet/IP was originally developed by Rockwell Automation in the late 1990s and was turned over to be managed by the Open DeviceNet Vendors Association (OVDA) in 2001 [3]. Specifications for the protocol are controlled and can only be obtained by subscribing as an authorized vendor through OVDA. EtherNet/IP provides extensible, object-oriented communication through the Common Industrial Protocol (CIP). The presence of EtherNet/IP communication is identified by its use of TCP port 44818.

The Distributed Network Protocol (DNP3) is a SCADA protocol designed to be transported across various physical media, including TCP/IP networks. Westronic Incorporated originally began development of DNP3 in the early 1990s, and the protocol is currently maintained by the DNP Users Group [4]. DNP3 was adopted as a standard by the IEEE in 2010 [5]. Notable characteristics of DNP3 include the use of a three-layer architecture (data-link, transport, application) and an optional Secure Authentication specification first proposed in [6] and updated to the latest version in [7]. Secure Authentication uses hash-based message authentication codes (HMAC) to authenticate users to SCADA endpoint devices. DNP3 communication on IP networks is identified by its use of TCP port 20000.

III. Attack Anatomy

TCP veto, described in this section, is designed to be an enhancement of connection hijacking that avoids generating an ACK storm. TCP veto targets specific application-layer payloads in a TCP connection that have predictable lengths and replaces them with a malicious payload. The attack is crafted in such a way that the client and server's original communication appears to continue uninterrupted with little extraneous network traffic generated. Because of this, the attacker can begin and end TCP veto at any time during a connection. The assumptions for this work are selected to match those used in Joncheray's original connection hijacking paper [1]. To facilitate a concise description of the attack, only the essential details of the TCP connection are included. A detailed description of the behavior of the TCP protocol can be found in [8].

A. Assumptions

The following assumptions are taken from the connection hijacking attack.

1. The attacker is positioned somewhere along the TCP communication path between the client and server and can sniff traffic, and inject packets with arbitrary IP, TCP and application layers.
2. The attacker can sniff and inject packets faster than the targeted system, permitting the attacker to exploit a race condition.

The following assumption is an additional constraint of TCP veto not required to perform connection hijacking.

3. The targeted application-layer payload uses a predictable, fixed-length format.

B. TCP Veto

This paper adopts the naming conventions used in Joncheray's description of connection hijacking to describe the flow of TCP packets [1]. For simplicity, only sequence and acknowledgement numbers and segment lengths are discussed. All packets are assumed to fall within the client and server's respective TCP windows and packet loss and retransmission are not considered. The following terms are used to describe the attack:

- CLT_SEQ – The sequence number of the next byte to be sent by the client
- CLT_ACK – The sequence number of the next byte to be received by the client
- SVR_SEQ – The sequence number of the next byte to be sent by the server
- SVR_ACK – The sequence number of the next byte to be received by the server
- SEG_SEQ – The sequence number in the TCP header of a packet
- SEG_ACK – The acknowledgement number in the TCP header of a packet
- SEG_LEN – The length of the TCP payload of a packet

Fig. 1 depicts the exchange of messages between the client, server, and attacker during TCP veto. Attacker packets are marked with dashed lines. First, the client connects to the server by performing the TCP three-way handshake. This portion of the connection is of no concern to the TCP veto attack. The attacker need not affect the connection at this time.

The attacker may begin the attack at any time. Let w, x, and y represent the following TCP header fields of the packet transmitted by the client immediately preceding the attack:

- SEG_ACK = w

- SEG_SEQ = x
- SEG_LEN = y

The attacker sniffs and dissects the TCP header of the client packet to create the following TCP header fields that appears to have originated from the server:

- SEG_ACK = x+y; properly acknowledge the client request packet
- SEG_SEQ = w; the next byte that the client is expecting from the server

After creating an acceptable TCP header, let z be the length of the TCP payload the attacker predicts that the server will transmit in response to the client's packet. This requires that the attacker have knowledge of how the application-layer protocol operates. The attacker appends exactly z bytes of malicious payload. This is the essential step in TCP veto that allows payload data to be injected without desynchronizing the targeted TCP connection.

- SEG_LEN = z

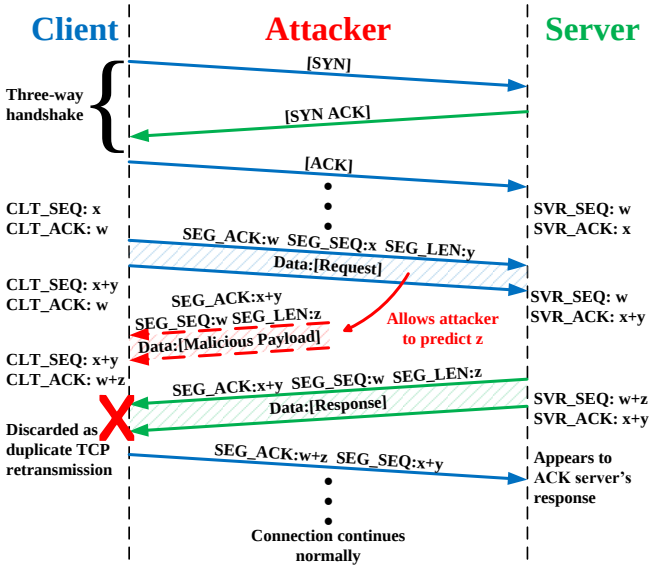

Fig. 1. TCP connection diagram of TCP veto.

The client's TCP stack accepts the attacker's packet and passes the malicious payload to the application. When the server's response is eventually transmitted and received by the client, it is discarded because it appears to the client as an old duplicate TCP retransmission [9]. This occurs because on receipt of the server's response: SEG_ACK = CLT_SEQ and SEG_SEQ + SEG_LEN = CLT_ACK.

TCP veto, in contrast to connection hijacking, does not desynchronize both the client and server. Because of this, when the client sends an ACK for the attacker's malicious packet, it appears to the server to be properly acknowledging its own response. The attacker must ensure that the malicious packet is not sent too soon such that the client's ACK arrives at the server before the server transmits its response, which can desynchronize the connection. For the default configuration of the hardware studied in this research, no such additional delay is required to perform the attack.

Avoiding two-sided desynchronization results in the following benefits:

- No ACK storm is generated, resulting in substantially increased resistance to detection.
- The attacker is not required to continue forwarding packets between the client and server to maintain their connection. As a result, the attacker may begin and end the attack at any time.

IV. Methodology

To test the effectiveness of TCP veto, a simple SCADA network is created to serve as a means to quantitatively compare its behavior to connection hijacking.

A. System Description

The following parameters define the components of the experimental network depicted in Fig. 2. The attacker is connected to an Ethernet hub with the HMI to satisfy the attack assumption of being along the communication path between the client and server.

- HMI (Client)
  - Dell Precision M4600
  - Windows 7 x64 SP1
  - Modbus TCP Client – Automated Solutions ASComm 3.2.0.11 Modbus Master Simple Read & Write
  - EtherNet/IP Client – Automated Solutions ASComm 3.2.0.11 A-B Legacy Simple Read & Write
  - DNP3 Client – Triangle Microworks Protocol Test Harness 3.13.4.0
- Attacker
  - Dell Precision M4600
  - Windows 7 x64 SP1
  - Packet Sniffer/Injector – WinPcap 4.1.2
  - Packet Capture – Wireshark 1.8.0 x64
- Modbus TCP PLC (Server)
  - OMRON CP1L with CP1W-MODTCP61 Modbus TCP protocol adapter
- EtherNet/IP PLC (Server)
  - Allen Bradley MicroLogix 1100 Ser. B
- DNP3 PLC (Server)
  - Allen Bradley MicroLogix 1400 Ser. B

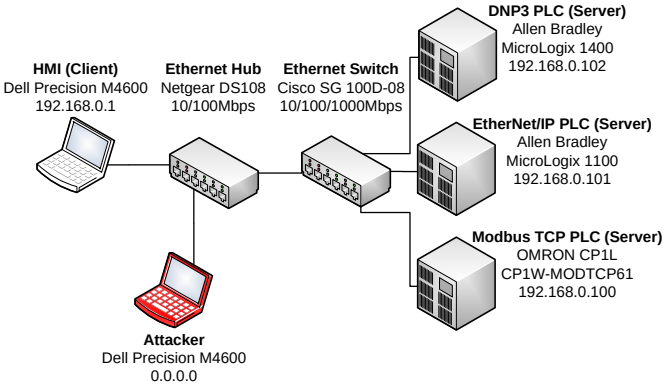

Fig. 2. Experimental network diagram.

B. Experimental Design

During experimentation, a SCADA HMI client polls the corresponding PLC at a fixed interval using a specific protocol. The attacker then executes an attack, which is successful if the HMI displays incorrect data. To determine the effect of the network attacks, the factors varied to test the experimental network are SCADA protocol and attack. Levels selected for each factor are summarized in Table I. The connection hijacking implementation used in this research employs null data desynchronization. The client and server communicating via DNP3 do not employ the optional Secure Authentication features of DNP3.

TABLE I
FACTORS AND LEVELS

Factor	Levels
Protocol	Modbus TCP, EtherNet/IP, DNP3
Attack	None, TCP veto, Connection hijacking

A full factorial design is selected to explore the interaction between each attack and protocol. For each factor level combination, 30 replications are conducted to establish a standard error (SE) for the sample mean.

V. ANALYSIS OF DETECTION RESISTANCE

Results from experimentation described in the preceding section are provided here. The metrics are selected to specifically demonstrate the effectiveness of TCP veto and compare TCP veto's detection resistance to connection hijacking.

A. Attack Success

TCP veto is 100% successful in compromising the integrity of each SCADA protocol in all 30 replications. This reliability is a result of several factors. First, the simple, low-traffic nature of the experimental network allows for the reliable injection of packets by the attacker. In the conducted experiments, the attacker's packets are never lost. For this experimental configuration, the attacker also possesses the ability to process and respond to packets on the order of 10 times quicker than the targeted PLCs, meaning that, throughout experimentation, the attacker always succeeds in exploiting the race condition between the client and server.

These results confirm the practicality of TCP veto, demonstrating that the SCADA protocols studied in this paper are vulnerable to the attack. The hardware and software selected for this research consists of real-world components used in SCADA networks and the attacker's machine possesses mainstream hardware.

B. ACK Storm Detection

A key design goal of TCP veto is to avoid the ACK storm experienced by connection hijacking to produce an attack that is more resistant to detection. Table II shows a summary of the network traffic collected for each protocol and attack combination. Connection hijacking data for Modbus TCP is not available (N/A) because no mechanism could be determined to inject null data that did not modify the state of the client, causing it to close the connection. For EtherNet/IP and DNP3, TCP veto generates 400 and 630 times less network traffic, respectively, than connection hijacking. A detection system would require a much higher degree of sensitivity to detect the increase in traffic resulting from TCP veto.

TABLE II
TCP TRAFFIC (PACKETS/S)

	Modbus TCP		EtherNet/IP		DNP3	
	Mean	SE	Mean	SE	Mean	SE
None	6.110	±0.0106	6.120	±0.0112	3.181	±0.0144
Veto	8.296	±0.0307	8.274	±0.0293	4.390	±0.0273
Connection Hijacking	N/A	N/A	866.0	±4.30	765.2	±6.35

Joncheray demonstrates that connection hijacking can be detected by comparing the percentage of ACK packets with and without the attack present [1]. Table III presents this percentage for each factor level combination. The connection hijacking results are validated by Joncheray's previous work in which on a local Ethernet connection "the percentage of TCP ACK is near 100%" [9]. When compared with no attack, TCP veto decreases the percentage of ACK packets by only 8-9% because for each request, response, ACK cycle the client and server exchange, the attacker injects a malicious packet with a TCP payload. The attacker's malicious packet lowers the relative weight of the ACK packet that is sent in the cycle.

TABLE III
TCP ACK / TOTAL TCP PACKETS (%)

	Modbus TCP		EtherNet/IP		DNP3	
	Mean	SE	Mean	SE	Mean	SE
None	33.4%	±0.11%	33.5%	±0.10%	33.4%	±0.23%
Veto	24.9%	±0.06%	24.8%	±0.03%	24.9%	±0.07%
Connection Hijacking	N/A	N/A	99.4%	±0.03%	99.9%	±0.00%

If it is known that the percentage of ACK packets is being used to detect the attack, the attacker could modify TCP veto to inject additional ACK packets to compensate, however this would increase the total traffic footprint of the attack.

C. Packet Retransmission

A desired side effect of TCP veto is that after an attacker injects a packet to the client, the following legitimate response from the server is discarded by the client because it appears to be a normal, duplicate retransmission. This behavior can also be used to detect the attack, however. Table IV lists the percentage of apparent TCP retransmissions observed during experimentation. When no attack is used, no TCP retransmissions are recorded for the simple experimental network. For each of the three SCADA protocols, TCP veto increases the apparent TCP retransmissions to nearly 25%. It is important to note that these are not retransmitted packets but rather the packets targeted by the attacker. So only the receiver (or a detection system that could sniff all traffic between the client, attacker, and server) could detect these anomalous packets.

The small percentage of apparent retransmissions present in connection hijacking is a result of injecting null data to desynchronize the connection between the client and server. These retransmission packets are then dwarfed by the large number of ACK packets generated by the ACK storm.

TABLE IV
TCP RETRANSMISSIONS / TOTAL TCP PACKETS (%)

	Modbus TCP		EtherNet/IP		DNP3	
	Mean	SE	Mean	SE	Mean	SE
None	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Veto	24.9%	0.06%	24.8%	0.03%	24.9%	0.07%
Connection Hijacking	N/A	N/A	0.202%	0.001%	0.014%	0.000%

It is expected that as the network is scaled to include other non-targeted TCP connections, the detectability of these apparent retransmissions diminishes.

D. Advanced Detection

Even if retransmissions are detected, it is difficult to distinguish if an attack is occurring or if the sending host is simply performing a legitimate TCP retransmission. This section presents a method to definitively detect the presence of TCP veto. This detection method is based off the principle that the TCP network stack should never correctly retransmit different data for a given sequence number.

Assume that a detector can sniff all traffic transmitted between the client, server, and attacker. The detector also possesses memory (MEM) it uses to store all TCP payload data (SEG_PAY) transmitted throughout a connection. Fig. 3 depicts the decision process the detector follows to identify a TCP veto attack. Sequence number wrap and transmission errors resulting in correct checksums are not considered in this simplified explanation.

When the detector receives a TCP segment, it first validates that the TCP checksum (SEG_CHK) is correct. If it is not, it simply discards the segment because a segment with an incorrect checksum will not be processed by the receiver and poses no danger of compromising integrity. New payload data is stored in MEM, and the detector performs a byte-wise comparison of retransmitted TCP payload (SEG_PAY[i]) against previously stored payload data (MEM[i]). Any data comparisons found to be different indicate that an attacker has performed a TCP veto attack.

This process of deep packet analysis to definitively detect TCP veto requires a great amount of computing and memory resources. Simultaneously performing byte-wise comparisons on all the TCP connections of a network is not expected to scale well. This provides additional support to TCP veto's detection resistance.

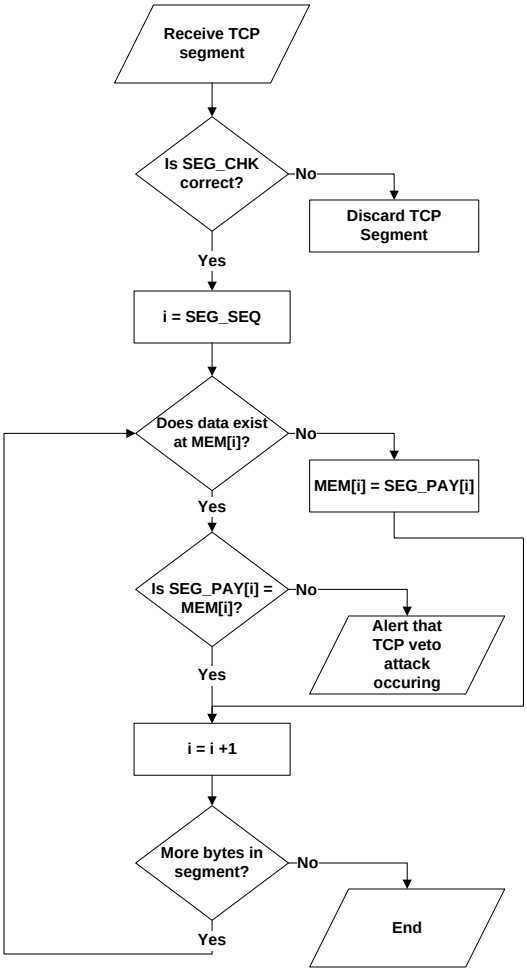

Fig. 3. Flowchart of advanced detection method.

VI. PREVENTION

The fundamental weakness that both connection hijacking and TCP veto exploit to compromise the integrity of the communication between a client and server is a lack of message authentication. Without cryptographically strong message authentication, the client and server cannot distinguish between legitimate and malicious messages. This section presents network protocols that can provide authenticated SCADA communication over TCP/IP.

A. Application Layer

Of the SCADA application-layer protocols studied in this paper, only DNP3 currently provides optional authentication. The most recent specification includes Secure Authentication version 5, which allows the use of pre-shared keys to authenticate user communication to SCADA endpoint devices [7].

Advantages of DNP3 Secure Authentication include end-to-end authentication of application data even if it traverses physical media other than IP networks and the ability to distinguish individual users or applications on a host. The disadvantage for current SCADA applications not employing DNP3 is that an entirely different application-layer protocol would have to be adopted, breaking compatibility.

B. Transport Layer

There are currently no widely-adopted security protocols for the transport layer. Tcpcrypt is a "work in progress" security protocol that integrates opportunistic encryption and optional authentication into TCP [11]. At the time of writing, tcpcrypt is a draft at the Internet Engineering Task Force (IETF) [12]. If adopted as a formal standard, tcpcrypt could be integrated into the TCP stack of SCADA client and server devices, providing a uniform security solution across many different SCADA protocols.

One advantage of using tcpcrypt is that it has the ability of authenticating much of the TCP header in addition to the application layer, which mitigates certain denial-of-service (DoS) attacks that target the TCP connection. Tcpcrypt can also protect any application-layer protocol, which preserves

backwards compatibility. The disadvantage of tcpcrypt is that SCADA applications would need to be reconfigured to interface to tcpcrypt's authentication features.

C. Network Layer

Internet Protocol Security (IPsec) is a network-layer security protocol designed to mitigate many of the security weaknesses in IP [13]. IPsec is designed to complement upper-layer protocols (e.g., TCP) such that they do not need to be re-written in order to be protected. IPsec defines two protocols – the Authentication Header (AH) described in [14] and the Encapsulating Security Payload (ESP) detailed in [15]. Both protocols can be configured to authenticate SCADA communication that travels over IP networks by integrating IPsec into the IP stack of SCADA client and server devices, providing a uniform security solution across many different SCADA protocols.

IPsec has the advantage of not requiring SCADA applications or protocols to be modified, which preserves backwards compatibility. The disadvantage of authenticating at the network layer is that IPsec cannot distinguish multiple users or applications on the same host, resulting in a loss of authentication precision.

VII. Conclusions

TCP veto is a novel detection-resistant network attack effective at compromising the integrity of messages passed between certain protocols that use predictable, fixed-length formats. All three SCADA protocols tested, Modbus TCP, EtherNet/IP, and DNP3, are vulnerable to TCP veto because of a lack of message authentication. By avoiding the disadvantage in connection hijacking that generates an ACK storm, TCP veto generates between 400 and 600 times less network traffic. The proposed method to definitively identify TCP veto requires the memory and computing power to perform deep packet inspection of every packet of a monitored connection. TCP veto is ideally suited to the broadcast nature of wireless portions of a SCADA network or to an insider threat that plants an attacking device along the path of SCADA communication. The SCADA community must prioritize the migration of critical infrastructure to authenticated network protocols such as DNP3 with Secure Authentication, tcpcrypt, or IPsec.

VIII. Acknowledgment

The authors gratefully acknowledge the contributions of David R. Hagen for his critical reading and critique of the original version of this document. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

IX. References

[1] L. Joncheray, "A simple active attack against TCP," in *Proc. of the Fifth Usenix Unix Symposium.*,Salt Lake City, UT, pp. 7-19

[2] Modbus Protocol Specification V1.1b. (2006, December). Modbus Organization. [Online]. Available: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

[3] EtherNet/IP(TM) - CIP on Ethernet Technology. (2008). Open DeviceNet Vendors Association. [Online]. Available: http://www.odva.org/Portals/0/Library/Publications_Numbered/PUB00138R3_CIP_Adv_Tech_Series_EtherNetIP.pdf

[4] About DNP3 Users Group. (2012). DNP3 Users Group. [Online]. Available: http://www.dnp.org/Pages/AboutUsersGroup.aspx

[5] *IEEE Standard for Electric Power Systems Communications – Distributed Network Protocol (DNP3),* IEEE Standard 1815-2010, July 2010.

[6] G. Gilchrist, "Secure authentication for DNP3," in *2008 IEEE Power and Energy Society General Meeting – Conversion and Delivery of Electrical Energy in the 21st Century*, Pittsburg, PA, pp. 1-3.

[7] Secure Authentication v5. (2011, November). DNP Users Group. [Online]. Available: http://www.dnp.org/Lists/Announcements/Attachments/7/Secure%20Authentication%20v5%202011-11-08.pdf

[8] J. Postel. (1981, September). Transmission Control Protocol – RFC 793. Internet Engineering Task Force. [Online]. Available: https://tools.ietf.org/html/rfc793

[9] Transmission Control Protocol – RFC 793. Internet Engineering Task Force. [Online]. Available: https://tools.ietf.org/html/rfc793 p. 69.

[10] L. Joncheray, "A simple active attack against TCP," in *Proc. of the Fifth Usenix Unix Symposium.*, p. 17

[11] A. Bittau, M. Hamburg, M. Handley, D. Mazieres, and D. Boneh, "The case for ubiquitous transport-level encryption." *USENIX Security Symposium*, Washington, DC

[12] A. Bittau, D. Boneh, and M. Hamburg. (2012, February). "Cryptographic protection of TCP Streams (tcpcrypt)." Internet Engineering Task Force. [Online]. Available: http://tools.ietf.org/html/draft-bittau-tcp-crypt-02

[13] S. Kent and K. Seo. (2005, December). Security Architecture for the Internet Protocol – RFC 4301. Internet Engineering Task Force. [Online]. Available: https://trac.tools.ietf.org/html/rfc4301

[14] S. Kent. (2005, December). IP Authentication Header – RFC 4302. Internet Engineering Task Force. [Online]. Available: https://trac.tools.ietf.org/html/rfc4302

[15] S. Kent. (2005, December). IP Encapsulating Security Payload (ESP) – RFC 4303. Internet Engineering Task Force. [Online]. Available: https://trac.tools.ietf.org/html/rfc4303

X. Biographies

John T. Hagen (M'2012) was born in the United States of America in 1988. He graduated from Cedarville University with highest honors with a B.S. in Computer Engineering. He received a M.S. in Cyber Operations from the Air Force Institute of Technology where he was awarded distinguished graduate.

His employment experience includes electrical and computer engineering design of a prototype hybrid-electric UAV for the Air Force Institute of Technology, network security analysis of mobile robotics at the Air Force Research Labs, and cryptanalysis.

His current research interests include network security, SCADA systems, embedded devices, and cryptanalysis.

Barry E. Mullins (M'1996, SM'2000) is an Associate Professor of Computer Engineering in the Department of Electrical and Computer Engineering, Air Force Institute of Technology (AFIT), Wright-Patterson AFB OH. He received a B.S. in Computer Engineering (cum laude) from the University of Evansville in 1983, an M.S. in Computer Engineering from AFIT in 1987, and a Ph.D. in Electrical Engineering from Virginia Polytechnic Institute and State University in 1997. He is a registered Professional Engineer in Colorado and a member of Tau Beta Pi, Eta Kappa Nu, Phi Beta Chi, Kappa Mu Epsilon, IEEE and ASEE.

His research interests include cyber operations, malware analysis, reverse code engineering, computer/network security, and SCADA (supervisory control and data acquisition) security.