ModbusFW
Deep Packet Inspection for Industrial Ethernet

Matthew Franz (mfranz@cisco.com)

Venkat Pothamsetty (vpothams@cisco.com)

Critical Infrastructure Assurance Group (CIAG)

Cisco Systems, Inc.

http://www.cisco.com/go/ciag/

Agenda

- CIAG Research Control Systems Security Initiatives
- Industrial Network Security 101
- Review of Network Filtering Technology
- Extending Linux Netfilter/Iptables to support Modbus/TCP
- DEMO: Protecting our “SCADA System” from a “Dangerous Canadian Hacker”
- Next Steps and Follow-On Research
- Conclusions, Feedback, and Questions

CIAG Research Overview

- **Research Team Charter**

  Conduct and sponsor research to improve the security of network and computing technology used by critical infrastructures

- **Core Competencies**

  Secure Network and Protocol Design

  Vulnerability Testing & Analysis

  Security Toolkit Development

  Product Enhancement & Hardening

- **Key Deliverables**

  External Presentation/Publications

  Best Practices and Standards

  Open Source Tools and Standards

CIAG Research Activity (Overall)

- **BGP Security – Attack Tree, Secure Routing Registry, soBGP**
- **IPv6 – Analysis of Threats, Vulnerabilities, and Best Practices (especially compared to IPv4)**
- **Security of Embedded Network Devices**
- **NIAC Internet Hardening and Vulnerability Scoring Working Groups**
- **Industrial Network Security and Control Systems**

Control Systems Security Initiatives

- **Strategic commitment to understand the problem space and develop solutions—whether or not there is revenue potential**
- **Relevant research projects**
  - **AGA 12-1 Implementation**
  - **SCADA Vulnerability Testing**
  - **Factory Automation Security**
  - **Modbus/TCP Firewall Prototype**
  - **Virtual SCADA HoneyNet**
  - **Industrial Stack Testing**
  - **Building Automation Security**
  - **DNP3 Security Testing (sponsored research at BCIT)**
- **Standards participation**
  - **AGA 12-1, SP-99, PCSRF**

ModbusFW: Initial Objectives*

- Develop countermeasures to problems uncovered in NISCC-sponsored vulnerability research done by BCIT & CIAG in 2003
- Determine feasibility of adding application-layer support for Industrial Ethernet protocols to a general purpose (read “IT”) packet filter
- Identify level of interest and seed commercialization

*Although we tackled a very narrow problem, there are wider lessons to be learned here

Industrial Network Security 101

Known vulnerabilities in control system networks

Design	Implementation	Configuration
Insecure comm links	TCP/IP stack issues	Weak/default passwords
Insecure devices & protocols	Protocol flaws	802.11 Defaults (no WEP)
Less than weak authentication in devices and protocols	OS/App flaws	Inadequate filtering on router/firewall
Insecure remote access (i.e. dialin modems, partner, integrator connections)	Windows HMI Flaws	OS defaults and failure to apply patches & upgrades
Undocumented commands/backdoors	WEP/802.11 Flaws	
	DoS to Network infrastructure Device	
Ill-defined or unrealistic security requirements	*Insecure coding practices and inadequate testing*	*Default insecure features and difficult or non-scalable secure features*

Device Vulnerabilities

- Most PLCs (Communication Modules) have no ability to filter based on source IP address—let alone based on application layer message types
- Few devices have the ability to do low-level packet filtering (to mitigate network transport layer attacks)
- TCP/IP Stack Issues
  - Resource Exhaustion
  - Poor Initial Sequence Number Selection
  - Malformed Message
  - Poor TCP ISN Selection
- Use of “IT” Protocols for Industrial Applications
- See ISA ’03 Presentation for more details

Protocol Vulnerabilities

- **Most Industrial Ethernet protocols simply encapsulate serial/fieldbus protocol over TCP/UDP**

  CIP → EtherNet/IP

  MODBUS → Modbus/TCP

- **Lack of Authentication, Authorization, and Encryption in Protocol Design and Specification**
- **Implementation flaws in processing valid/invalid formatted messages**

Review of Filtering Technology

- **Intelligent switches/bridges**

  Filter on L2 (MAC) source address

- **Router Access Control Lists**

  Filter source/destination based on L3 (IP Address/Protocol) and L4 (TCP/UDP Port)

- **Stateful Firewalls**

  Filter based on TCP/ICMP/UDP "state" and limited support for some applications

- **Application Proxies**

  Complete application and protocol support, typically requires reconfiguration of client

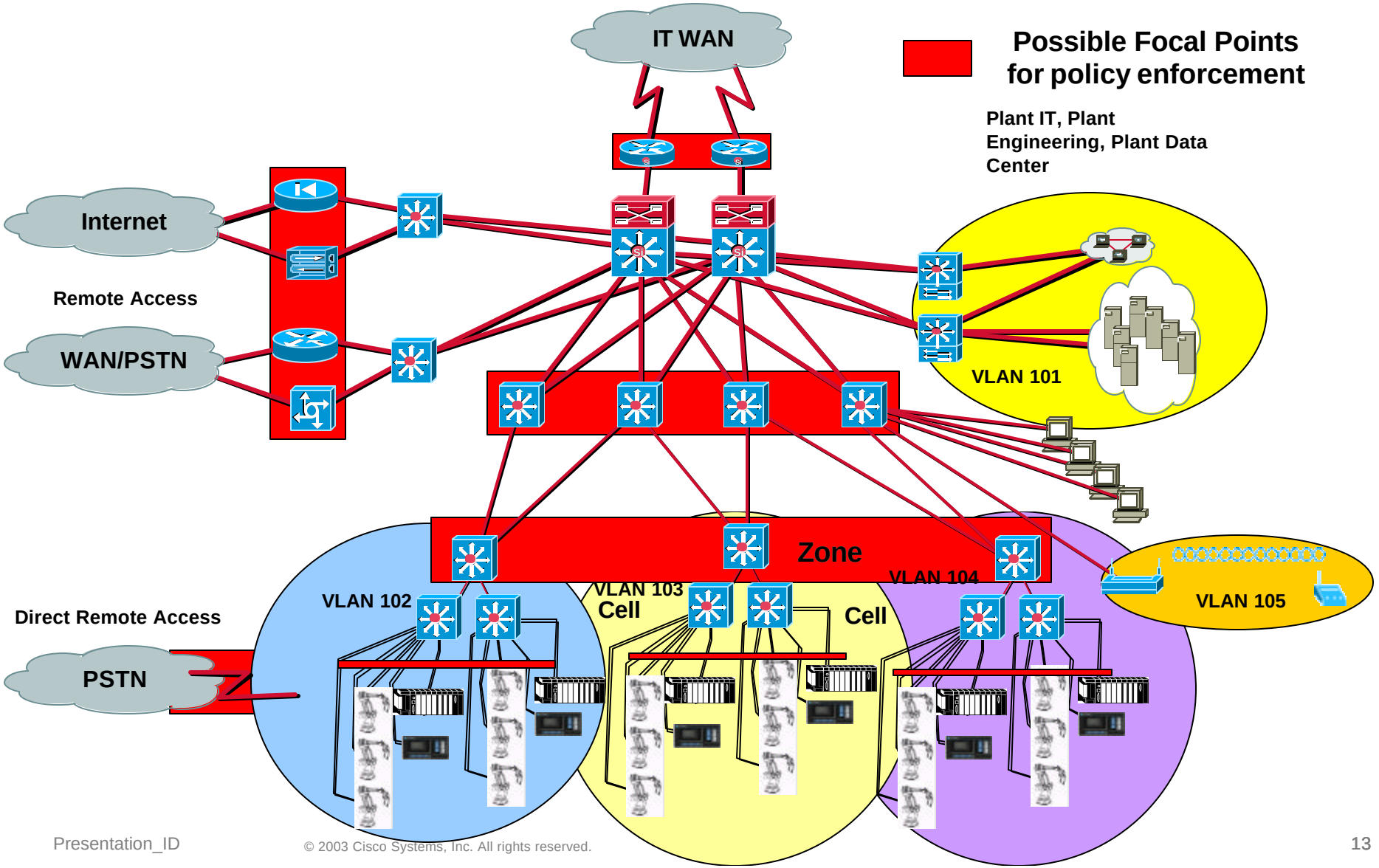
- **Deep Packet Inspection and Network IPS**

  High speed (possibly inline/transparent) filtering at all application and protocol layers

Assessment of Existing Technology

- **Assuming there is a properly configured border firewall between Plant/Enterprise networks, we need additional layers of protection**
  - **Just filtering on TCP port 502 (in the case of Modbus/TCP) may not provide granular enough access control**
  - **Differing security requirements within the Plant network may require additional layers of security—and additional security devices**
- **Operational control issues between IT and Control Systems—who owns the border security device and do they have the knowledge to adequately secure it?**

Plant-Enterprise Policy Enforcement?

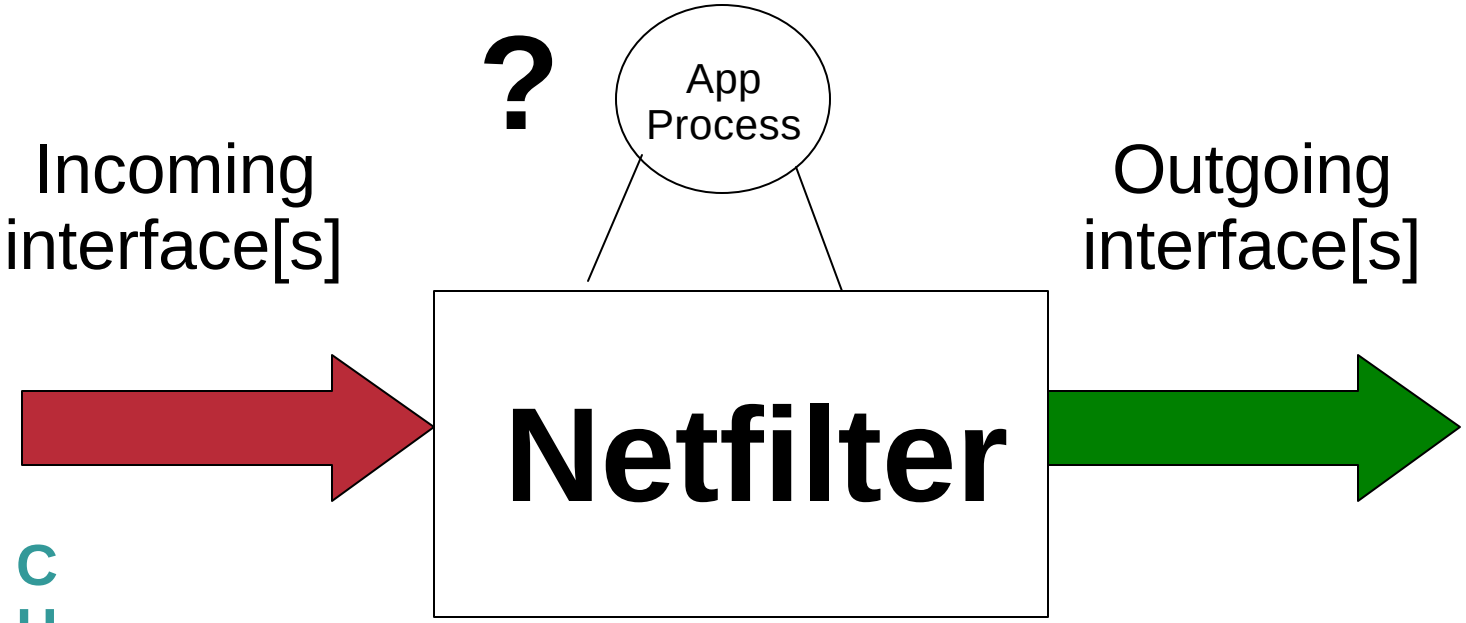


ModbusFW: Design & Implementation

Firewalls & Linux

- **Why Linux?**
  - **Most popular Open Source OS**
  - **Linux Netfilter is well-documented with a rich library of extensions**
  - **Increasing use of Linux in Embedded Systems and some interest in Industrial Computing**
- **Why not an application-layer proxy?**
  - **Concern about performance—packet filtering is done in the kernel and is fast!**
  - **Need for transparency—where is the configure proxy option in your HMI?**
- **Implementation Overview**
  - **Netfilter – provides kernel hooks to make policy decisions at critical points**
  - **Iptables – command-line tool for setting policy**
  - ***We had to modify both to implement ModbusFW***

Simplified Linux Netfilter Packet Flow



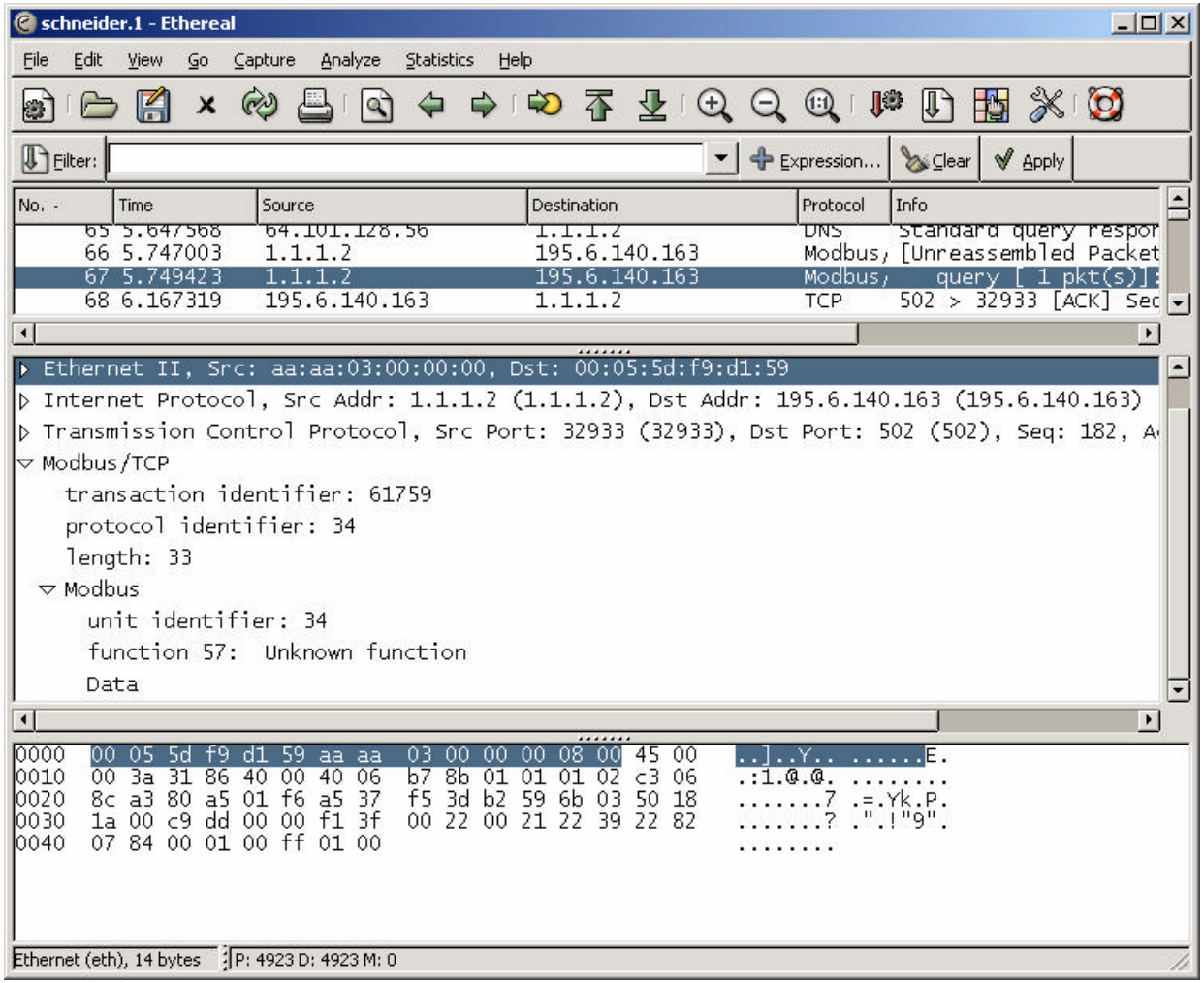
CHAINS

INPUT FORWARD OUTPUT

PREROUTING POSTROUTING

POLICY: ACCEPT, DROP, REJECT, MASQUERADE

Anatomy of A Modbus/TCP Message



Sample ModbusFW Rules


```
# iptables -A INPUT -p tcp -m modbus --funccode 8 –
  allowtcp 1 -j DROP
```

Drops whenever the function code of the received packet is 8 (Diagnostics)

```
# iptables -A INPUT -p tcp -m modbus --funccode !
  16 –allowtcp 1 -j DROP
```

Drops whenever the function code of the received packet is NOT 16 (Write Multiple Registers)

```
# iptables -A INPUT -p tcp -m modbus --funccode 16
  –allowtcp 1 --unitid !3 --refnum 5433 -j DROP
```

The packet will the dropped when either function code is 16, OR unitid is not 3 OR reference number is 5433

Analyzing Modbus/TCP Traffic to Derive Rules


```
14.455884 192.168.59.101 -> 192.168.60.171 Modbus/TCP    query [1
  pkt(s)]: trans: 42; unit: 0, func:  40: Program (ConCept).

14.462604 192.168.60.171 -> 192.168.59.101 Modbus/TCP response [
  1 pkt(s)]: trans: 42; unit: 0, func: 40: Program (ConCept).

14.580781 192.168.59.101 -> 192.168.60.171 TCP 2366 > 502 [ACK]
  Seq=617 Ack=838 Win=64019 Len=0

15.643352 192.168.59.101 -> 192.168.60.171 Modbus/TCP query [ 1
  pkt(s)]: trans: 43; unit:   0, func:  40: Program (ConCept).

15.652584 192.168.60.171 -> 192.168.59.101 Modbus/TCP response [
  1 pkt(s)]: trans: 43; unit: 0, func: 40: Program (ConCept).

15.783872 192.168.59.101 -> 192.168.60.171 TCP 2366 > 502 [ACK]
  Seq=629 Ack=850 Win=64007 Len=0

16.315610 192.168.59.101 -> 192.168.60.171 Modbus/TCP query [ 1
  pkt(s)]: trans: 44; unit: 0, func: 126: Program (584/984).

16.322522 192.168.60.171 -> 192.168.59.101 Modbus/TCP response [
  1 pkt(s)]: trans: 44; unit:   0, func: 126: Program (584/984).
```

Sample Filtering Policies (based on FC)

- **Modicon Applications**

Momentum E1 Web Interface	Concept Software
1 – Read Coil	126 – Vendor Extension
8 – Diagnostics (Program)	40 – Vendor Proprietary
126 – Vendor Extension	3 – Read Multiple Registers

- **Findings**

  Detailed protocol analysis with a sniffer (and extensive testing) is required to develop functional firewall policy based on applications

  Given extensive used of FC 126 by Modicon applications, additional filtering may be necessary to block Malicious (write/alter) messages

  Other (IT protocols, such as FTP or Telnet) may need to be filtered depending on the device

  Documentation of device/application implementation is essential!

Simple *Permissive* Ruleset


```
# Allow all ICMP and SNMP for network monitoring
iptables -A FORWARD -p icmp -j ACCEPT
iptables –A FORWARD –p udp –-dport 161 –j ACCEPT

# Allow connections to embedded webserver
iptables -A FORWARD -p tcp --sport 80 -j ACCEPT
iptables -A FORWARD -p tcp --dport 80 -j ACCEPT

# Open up function codes for Concept software
iptables -A FORWARD -p tcp -m modbus --funccode 1 --allowtcp 1 -j ACCEPT
iptables -A FORWARD -p tcp -m modbus --funccode 3 --allowtcp 1 -j ACCEPT
iptables -A FORWARD -p tcp -m modbus --funccode 8 --allowtcp 1 -j ACCEPT
iptables -A FORWARD -p tcp -m modbus --funccode 40 --allowtcp 1 -j ACCEPT
iptables -A FORWARD -p tcp -m modbus --funccode 126 --allowtcp 1 -j ACCEPT
iptables -A FORWARD -p tcp -j REJECT --reject-with tcp-reset

#Default deny
iptables -P FORWARD DROP
```

ModbusFW: Next Steps

- **Testing, Testing, Testing!**
  - **Interoperability with most common devices & applications**
  - **Test with NAT and Port Forwarding**
  - **Real Performance Testing**
  - **Penetration Testing and Source Code Audit**
- **Workarounds for the “allowtcp hack” – possible userspace handoff to spoof exception responses**
- **Modbus/TCP Function Code Library**
  - **Database of function codes used by devices and applications**
- **Deeper packet inspection**
  - **What is practical for Read/Write function codes?**
  - **Modicon FC 126 (probably requires reverse engineering)**
- **Linux Bridging Firewall for a transparent L2-L7 capability**

Demo: Protecting our “SCADA System” from a “Dangerous Canadian Hacker”

Next Steps, Follow-on Research, and Conclusions

ModbusFW: Potential Follow-on work

- Intuitive Interface for non-IT users
- Automated/Automatic Ruleset Configuration
- Support for EtherNet/IP and other protocols
  - Modbus/TCP was straightforward
  - Multicast, UDP/UDP, Connection-Oriented Protocols
- Support for Linux Bridging Firewall
- Userspace extensions to provide more graceful session termination
- Is there really any value in “bump in the wire” serial protocol filters using function codes? (probably not)
- Implement technology in commercial firewall, routers, and switches

Conclusions

- **Technology knows no organizational boundaries – "IT Security Products" *can* be altered to secure control system applications**
- **Security technologies and practices lag threats and vulnerabilities – it is now 1996?**
  - **Vulnerabilities are still known by a small but growing community – for now?!**
  - **Small target population – Industrial Ethernet (and wireless) have not reached critical mass**
  - **Threat picture is unclear – lack of automated tools (although this could change quickly)**
- **Simple "type-code filtering" is better that the status quo but is probably a partial (interim?) solution to more robust security enhancement necessary for industrial devices and protocols**

Conclusions (cont.)

- **Security enhancements for automation protocols are 3-5 years away from widespread deployment, but legacy devices will remain a problem after secure protocols become a reality**
- ***Hard problems yet to be solved:***

  Operational constraints of deploying security deep within the automation cell—especially management and monitoring

  Operation control and interaction between multiple security technologies (authentication servers, VPNs, border routers, and firewalls)

  Integration of Control System Security Software and Network Security Countermeasures?

  What device is best suited for application protocol filtering: firewall, router, intelligent switch? Where should it be deployed?

For more information

- **Copy of this presentation**

  http://www.io.com/~mdfranz/papers/

  http://www.scadasec.net/

- **Sourceforge site**

  http://modbusfw.sourceforge.net

  See Trinux ModbusFW – a small bootable CD-ROM Linux Distribution

- **MODBUS-IDA Website**

  http://www.modbus-ida.org

- **Linux Netfilter**

  http://www.netfilter.org

- **ISA SP-99 TR1**