SANDIA REPORT

SAND2013-8274
Unlimited Release
Printed October, 2013

WeaselBoard: Zero-Day Exploit Detection for Programmable Logic Controllers

John Mulder, Moses Schwartz, Michael Berg, Jonathan Roger Van Houten, Jorge Mario Urrea, Michael Aaron King, Abraham Anthony Clements, Joshua Jacob

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2013-8274
Unlimited Release
Printed October 2013

WeaselBoard: Zero-Day Exploit Detection for Programmable Logic Controllers

John Mulder, Moses Schwartz, Michael Berg,
Jonathan Roger Van Houten, Jorge Mario Urrea, Michael Aaron King,
Abraham Anthony Clements, Joshua Jacob

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-MS0671

Abstract

Critical infrastructures, such as electrical power plants and oil refineries, rely on programmable logic controllers (PLCs) to control essential processes. State of the art security cannot detect attacks on PLCs at the hardware or firmware level. This renders critical infrastructure control systems vulnerable to costly and dangerous attacks.

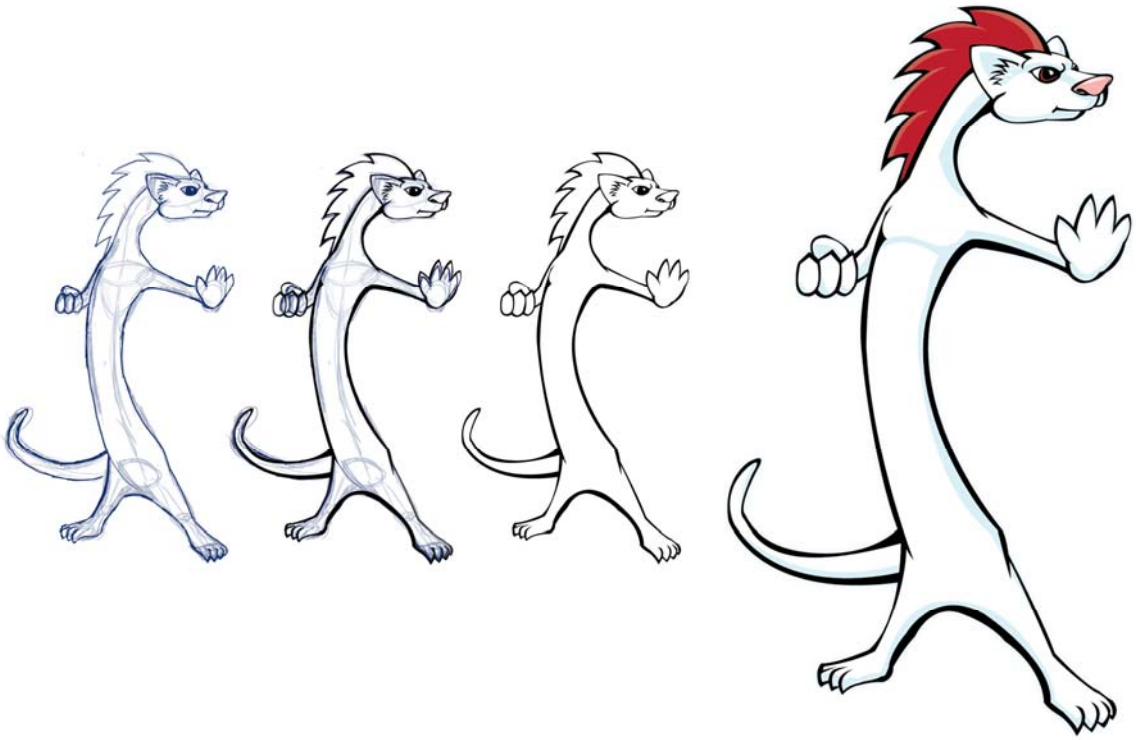
WeaselBoard is a PLC backplane analysis system that connects directly to the PLC backplane to capture backplane communications between modules. WeaselBoard forwards inter-module traffic to an external analysis system that detects changes to process control settings, sensor values, module configuration information, firmware updates, and process control program (logic) updates.

WeaselBoard provides zero-day exploit detection for PLCs by detecting changes in the PLC and the process. This approach to PLC monitoring is protected under U.S. Patent Application 13/947,887.

ACKNOWLEDGMENTS

WeaselBoard is a product of years of research, building on decades of experience and expertise at Sandia National Laboratories, and we cannot hope to individually thank everyone who contributed along the way. We would, however, like to specifically acknowledge Jason Trent, Regis Cassidy, Cynthia Veitch, and Jennifer Trasti for their parts in this research.

This work was funded under Sandia National Laboratories LDRD Project Number 158752, "PLC Backplane Analyzer for Field Forensics and Intrusion Detection".



CONTENTS

FIGURES

NOMENCLATURE

DOE	Department of Energy
PLC	Programmable Logic Controller
ICS	Industrial Control System
SCADA	Supervisory Control and Data Acquisition
SNL	Sandia National Laboratories
GUI	Graphical User Interface
UDP	User Datagram Protocol
FPGA	Field Programmable Gate Array
I/O	Input/Output
IP	Internet Protocol
RAM	Random-Access Memory
CPU	Central Processing Unit
FIFO	First-In-First-Out
CIP	Common Industrial Protocol
LDRD	Laboratory Directed Research and Development
SOC	System on a Chip
TRL	Technology Readiness Level

1 INTRODUCTION

Critical infrastructures, such as electrical power plants and oil refineries, rely on PLCs to control essential processes. State of the art security cannot detect attacks on PLCs at the hardware or firmware level. This renders critical infrastructure control systems vulnerable to costly and dangerous attacks.

Most attacks on control systems focus on network communications, Windows PCs, and PLC logic, but not on PLCs at the hardware or firmware level. PLCs are currently not monitored for security compromise.

These industrial control system (ICS) components receive little attention as an asset requiring security monitoring. Recent high profile events like the Stuxnet attack (2010) and Digital Bond's Basecamp (2012) have highlighted this critical vulnerability.

There is a critical need to inspect and monitor PLC hardware and firmware, and create an assurance platform for responding to attacks as these systems scale up in the future. Millions of dollars in equipment damage, lost uptime, and ultimately, casualties among operating personnel can be prevented by early detection.

1.1 Technical Approach

WeaselBoard captures and analyzes backplane communications between PLC modules. WeaselBoard connects directly to the PLC backplane either in a chassis or an ICS and forwards inter-module traffic to an external analysis system.

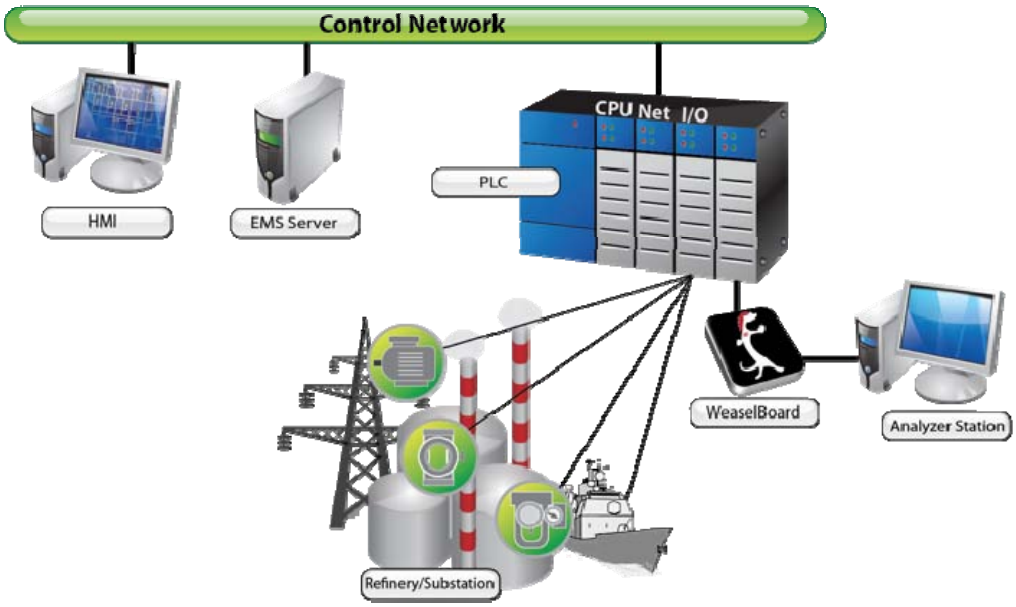


Figure 1. WeaselBoard in an industrial control system. WeaselBoard is connected directly to the PLC backplane and sends captured backplane traffic to a separate analysis workstation.

WeaselBoard takes the signals from the backplane and sends them to the analysis workstation using a custom protocol called WeaselTalk. Analysis software displays the backplane traffic, which is similar to network traffic, but is based on proprietary physical layer protocols. The analysis workstation then extracts fields at each protocol layer. These fields have been tested using mechanisms to identify malicious behavior: a rule set and a machine-learning algorithm. The rules-based mechanism causes an alert when predetermined behavior is seen, and can be customized to process-specific limits. The machine-learning algorithm is a Bayesian classifier trained to alert on traffic classified into known bad states.

Using this system, operators can detect any compromise that affects the process because WeaselBoard alerts on the effects of the attack in progress, not on signatures of previously catalogued attacks. This allows zero-day exploits to be detected, unlike systems using signature-based detection methods.

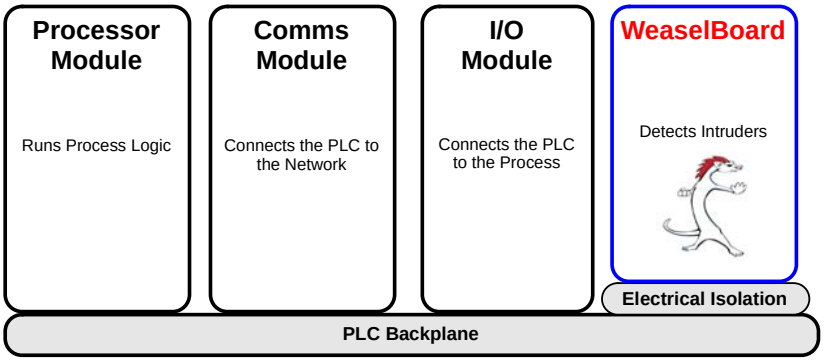


Figure 2. WeaselBoard in a PLC chassis.

1.2 Applications

WeaselBoard detects zero-day exploits against PLCs as soon as the state of the PLC changes instead of after serious damage has occurred.

WeaselBoard addresses the problem of low-frequency, high-impact attacks from sophisticated adversaries that use zero-day attacks against PLCs. Backplane analysis provides defenders with low-level PLC behavior in real time, enabling early detection. By detecting attacks in the early stages, asset owners can mitigate or stop malicious attacks before damage occurs.

PLC devices control billions of dollars worth of production, manufacturing and utility equipment in the United States. These processes require high availability and any cyber attack could result in casualties among operating personnel, lost uptime and costly equipment damage. Interoperability with existing network monitoring will facilitate integration and minimize the training needed for WeaselBoard users.

1.3 State of the Art

Many security systems monitor Windows PC activity and network communications. No other security system monitors and protects PLCs. The benefit of looking at PLCs directly is that they are simpler and more consistent, so malicious activity is easier to detect.

Control system security products provide network firewalls, network intrusion detection, and assessment scanning. These tools can detect known attacks on PCs and networks, but leave the systems vulnerable to zero-day exploits that are aimed at the PLCs. There is no tool that provides direct, real-time monitoring of PLC integrity.

Current industry practice forces critical infrastructure owners to react to malicious attacks after the damage has occurred, without the ability to detect PLC exploits at the firmware or hardware level. WeaselBoard detects changes in the PLC and the process. This capability in PLC monitoring is a unique approach, protected under U.S. Patent Application 13/947,887. WeaselBoard fills the gap that currently exists for protection of industrial control systems.

2 WEASELBOARD IMPLEMENTATION

A WeaselBoard hardware module has a main CPU board and a PLC-specific adapter board. This modular design reduces the cost of supporting additional PLC's, as the adapter boards are much simpler to design and produce compared to the CPU boards. The common design of the CPU board also reduces the software and FPGA logic development costs as it enables significant reuse of both.

We have developed adapter boards for the Allen Bradley Control Logix 5000 PLC and the Siemens S7-300.

2.1 WeaselBoard CPU Board

The CPU board uses a MicroSemi SmartFusion SOC as the primary chip. This SmartFusion SOC has an ARM Cortex M3 microcontroller and an FPGA. The SmartFusion enables logic to be implemented in the FPGA to capture the backplane traffic and then easily transfer the data to the microcontroller. The microcontroller then sends it as UDP packets over Ethernet. In addition to the SmartFusion SOC the CPU board also provides connectors for the adapter board, an Ethernet port, a USB to serial adapter for debugging, external RAM, external flash, a programming port, and test points for debugging.

2.2 Adapter Boards

Two adapter boards were created that enable the WeaselBoard to work with the Allen Bradley Control Logix 5000 and Siemens S7-300 PLCs. The Allen Bradley adapter board converts the logic levels of the data pins on the backplane to those used by the SmartFusion, and connects to the 24V power pins provided by the backplane. The Allen Bradley backplane has 32 data lines, 16 of these lines appear to contain framing data and the other 16 are a parallel data bus. The Siemens S7-300 backplane has two buses: an RS-485 serial bus, and a proprietary synchronous serial bus. The Siemens S7-300 adapter board provides logic level conversion of the synchronous serial bus and an RS-485 to RS-232 converter and connections to the 5V power lines in the backplane.

2.3 WeaselBoard Firmware

The WeaselBoard firmware main consists of setup and a main control loop. First, the CPU board is initialized by enabling global interrupts, setting up memory regions, initializing timers, setting up callback functions, and initializing the UDP/WeaselTalk protocol stack. Since the WeaselBoard hardware consists of the CPU board paired with a backplane adapter board, there is also some hardware initialization that depends on what backplane adapter board is plugged in. After the hardware is initialized and the board is in a state to begin capturing data, the main control loop takes over.

The WeaselBoard main control loop checks the hardware buffers for capture data and handles the data if there is any. Data handling consists of removing data from the hardware buffers, freeing that buffer to again be used by the hardware, and then processing the data using the relevant internal protocol stack dissector. The result is then packed into a WeaselTalk capture packet and buffered to be sent when the WeaselTalk buffer is full. The last thing the control loop does is to check to see if there have been any Ethernet packets buffered that need to be

processed. If there are packets to be processed, a call to the Lightweight IP (lwip) library is invoked. lwip handles responding (or not) to all network traffic that is not WeaselTalk. However, when there is a WeaselTalk payload, that packet propagates all the through the lwip stack, which eventually calls the WeaselTalk protocol stack.

The only WeaselTalk packets the WeaselBoard expects to see are command/control packets. If it sees other WeaselTalk packet types, its behavior is simply to immediately drop the packet. Otherwise, the WeaselTalk protocol stack will call one of its application function callbacks that correspond to the command that was sent. That callback is implemented in terms of the specific WeaselBoard firmware target to perform the action requested by the command/control message.

2.4 Siemens S7-300 Backplane

The Siemens S7-300 backplane has two data buses, the K-bus and P-bus, where K is for communication (in German) and P is for peripheral. Modules that transfer large amounts of data, but at infrequently intervals primarily use the K-bus. The K-bus uses the Profibus protocol over RS-485. The K-bus is also where traffic from remote I/O modules would primarily be seen. Configuration and programming information is also carried across this bus. Modules that transfer small amounts of data frequently, primarily use the P-bus. These modules include the I/O modules and some remote I/O modules. This is primarily where I/O information will be seen. Little is publicly available about the P-bus. It uses form of synchronous serial communication with two meta-lines. The two lines appear to indicate whether the data on the data line is metadata (a command, an address, etc.) or data.

To capture the P-bus, the microcontroller provides the FPGA with an address of a buffer and the buffer size. The FPGA logic then samples eight pins from the backplane on the rising edge of the P-bus clock and writes them to the buffer provided by the microcontroller. Eight lines are sampled, which includes everything other than the power, ground, and K-bus. This is done because the K-Bus was not fully understood the FPGA logic was initially designed. When the buffer gets full, the FPGA triggers an interrupt with very high priority in the microcontroller and continues capturing to an internal buffer in the FPGA. The microcontroller then provides a new buffer to the FPGA,which the FPGA then transfers the data in it internal buffer to and continues capturing until the new buffer is full. When the buffer is full it again interrupts the microcontroller which then gives the FPGA a new buffer again. This repeats indefinitely. When the microcontroller receives an interrupt indicating that the buffer for the P-bus is full gives a new buffer to the FPGA and sets a flag indicating that the P-bus has data to send and then returns from the interrupt. The main loop of the program then takes the buffer, wraps it in WeaselTalk, and sends it over Ethernet to the configured destination IP using UDP. The analysis workstation receives the P-bus data and decodes the raw capture into bytes for analysis.

Capturing of the K-bus is a little different. First, the adapter board converts the RS-485 physical layer to RS-232. Then, a transceiver with auto-baud rate detection in the FPGA captures the data to an internal FIFO. When the FIFO has data it triggers an interrupt in the microcontroller. The microcontroller then reads the bytes out of the FPGA to an internal buffer and returns from the interrupt. The main loop then takes the bytes out of the buffer and provides them to a Profibus stack running on the microcontroller. The Profibus stack provides packet boundary detection and some error checking. When a packet is detected, it is put into a WeaselTalk buffer along with a

few bytes of header information, indicating both error status and packet length. When the WeaselTalk buffer gets full it is sent to the analysis workstation over Ethernet. Both the P-bus and K-bus capturing happen concurrently.

2.5 Allen Bradley Control Logix 5000 Backplane

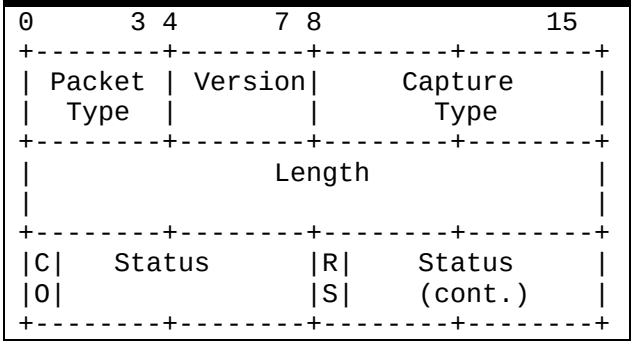
Capturing of the Allen Bradley backplane traffic is identical to the way the P-Bus is captured on the Siemens S7-300, except that 32 bits of data are captured with each sample instead of eight bits, and data is sampled on the falling edge of the clock instead of the rising. Nearly identical logic in the FPGA and firmware in microcontroller are used. Raw captures are sent to the analysis workstation over Ethernet, where the data is packetized and parsed. WeaselTalk is used to wrap the raw captures and indicate what that they are raw Control Logix 5000 captures. The Control Logix 5000 backplane protocol is similar to the Common Industrial Protocol (CIP), but the differences are not currently fully understood. Of the 32 bits captured, it appears that 16 are framing information and 8 or 16 bits contain data depending on the framing bits.

3 WEASELTALK PROTOCOL

This WeaselTalk protocol is defined to make available a means to communicate between the WeaselBoard and an analysis workstation. The protocol assumes that UDP is the underlying protocol and the physical media is Ethernet. WeaselTalk provides ways to stream captured data from the WeaselBoard to another computer on the network as well as accept commands to configure and control the WeaselBoard.

3.1 Header Format

WeaselTalk Header Format



3.1.1 Packet Type

This field defines the format for the body of the WeaselTalk packet. It can be any of the following values:

Packet Type	Value
ERROR	0x00
CC	0x10
CCRESP	0x20
ANALYSIS	0x40

3.1.2 Version

This field specifies the version of the WeaselTalk protocol. Currently it is set to `0x00`, to indicate that it is version 0.

3.1.3 Capture Type

This field defines what type of captured data is being transported in a capture packet. This field is only valid for the CAPTURE packets and is set to NA for other packet types. It can be any of the following values:

Capture Type	Value
NA	0x00
CIP	0x01
RAW_CIP	0x02
PROFIBUS	0x03
ALL	0xFF

3.1.4 Length

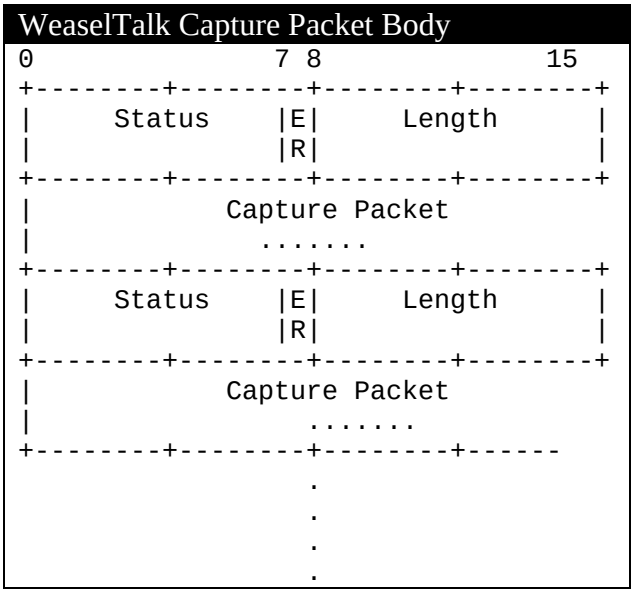
This field describes the length of the rest of the packet. This includes the header status bytes (2) plus the body of the packet.

3.1.5 Status

This field contains status about the state of WeaselBoard and its application. The bits of the byte are:

Bit	Description
0	Capture Overflow Flag
1-7	Reserved
8	Response Success Flag (0) failure (1) success
9-15	Reserved

3.2 Capture Packet Format



This is the packet format that the WeaselBoard streams out containing the data that was captured. The body of a WeaselTalk Capture packet is composed of 'x' number of individual packets of the captured protocol. Each captured packet is lead by a status field and a length of the captured packet.

3.2.1 Status

This field contains a status about the captured packet. The bits of the byte are:

Bit	Description
0-6	Reserved
7	Parsing Error

3.2.2 Length

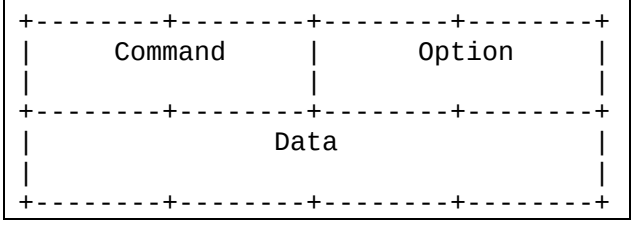
This is the length of the captured packet

3.2.3 Capture Packet

This field is the captured packet read by the WeaselBoard.

3.3 Command and Control Packet Format

Command and Control Packet Body



3.3.1 Command

This field specifies which command the WeaselBoard is to carry out. The options are:

Command	Value
RESET	0x00
SETSRCIP	0x10
SETDESTIP	0x20
SETMAC	0x30
STOPCAP	0x40
STARTCAP	0x50
GETBOARDCONFIG	0x60

3.3.2 Option

This field is obsolete and will be removed in the next version of the protocol.

3.3.3 Data

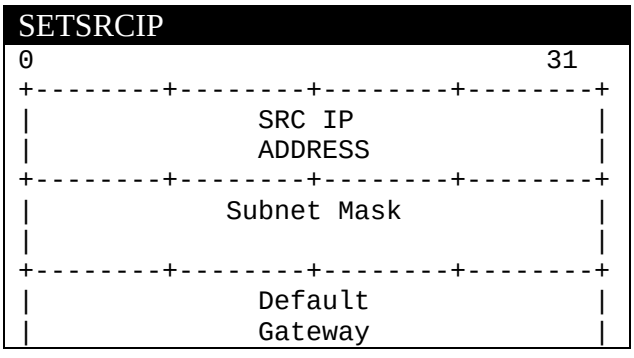
Each command carries different data. For each command the data fields are as follows:

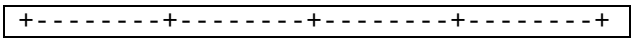
3.3.3.1 RESET

No Data

3.3.3.2 SETSRCIP

SETSRCIP

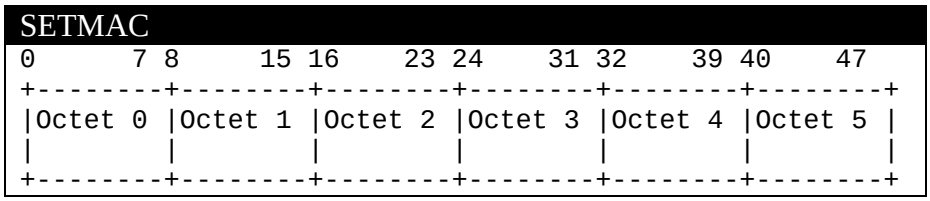




3.3.3.3 SETDESTIP

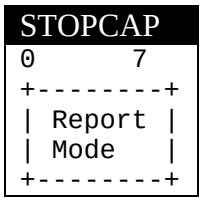
3.3.3.4 SETMAC

Each of the six octets of the MAC address are transmitted in network byte order.



3.3.3.5 STOPCAP

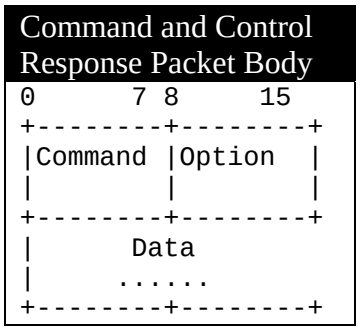
The Report mode field values are the same as the capture type. They encode a bus on a device to stop capture on.



3.3.3.6 GETBOARDCONFIG

No Data

3.4 Command and Control Response Packet Format



3.4.1 Command

This command is one of the command codes from the command and control packet. In the response packet it specifies what command is being responded to. The status byte in the header specifies if the command was a success or a failure.

3.4.2 Option

This field is obsolete and will be removed in the next version of the protocol.

3.4.3 Data

The only command response carrying data is the `GETCONFIG` response. This field is currently undefined.

4 WEASELWARE SOFTWARE

The WeaselWare software parses, analyzes, and displays PLC backplane traffic, and provides an interface to configure and control WeaselBoard. Backplane traffic is similar to network traffic, but tends to be based on proprietary physical layers. The WeaselBoard system takes the signals from the backplane and pulls out fields at each protocol layer.

The analysis software uses two different approaches to identify malicious behavior: rules and machine learning. The rules alert on predetermined behavior and can be customized based on process specific limits. The machine learning is done using a Bayesian Classifier trained on known system states.

WeaselWare GUI (weasel_gui) is the primary graphical user interface (GUI) for interacting with the WeaselBoard from the analysis workstation. This GUI has tabs for each of the command/control/analysis/display functions.

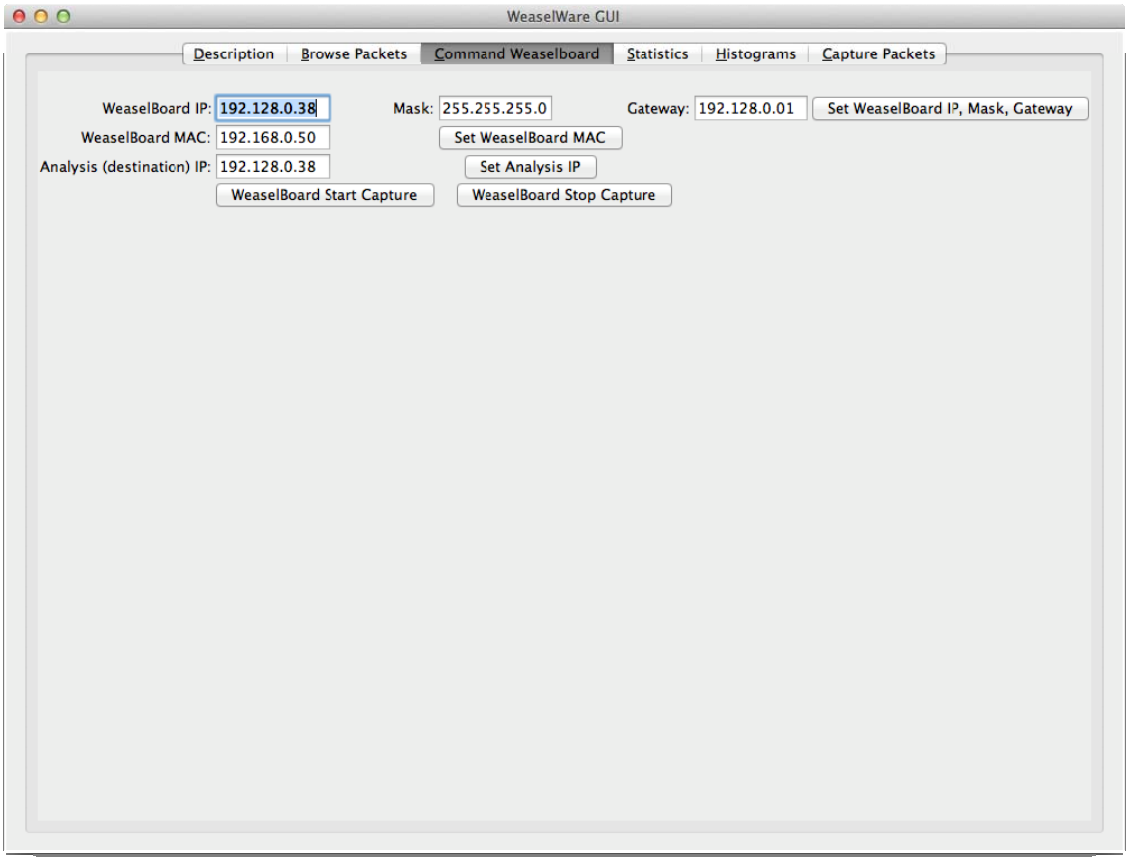


Figure 3. WeaselBoard configuration.

The Command WeaselBoard tab (weaselcommand_gui) sends WeaselTalk command packets on the network to a WeaselBoard.

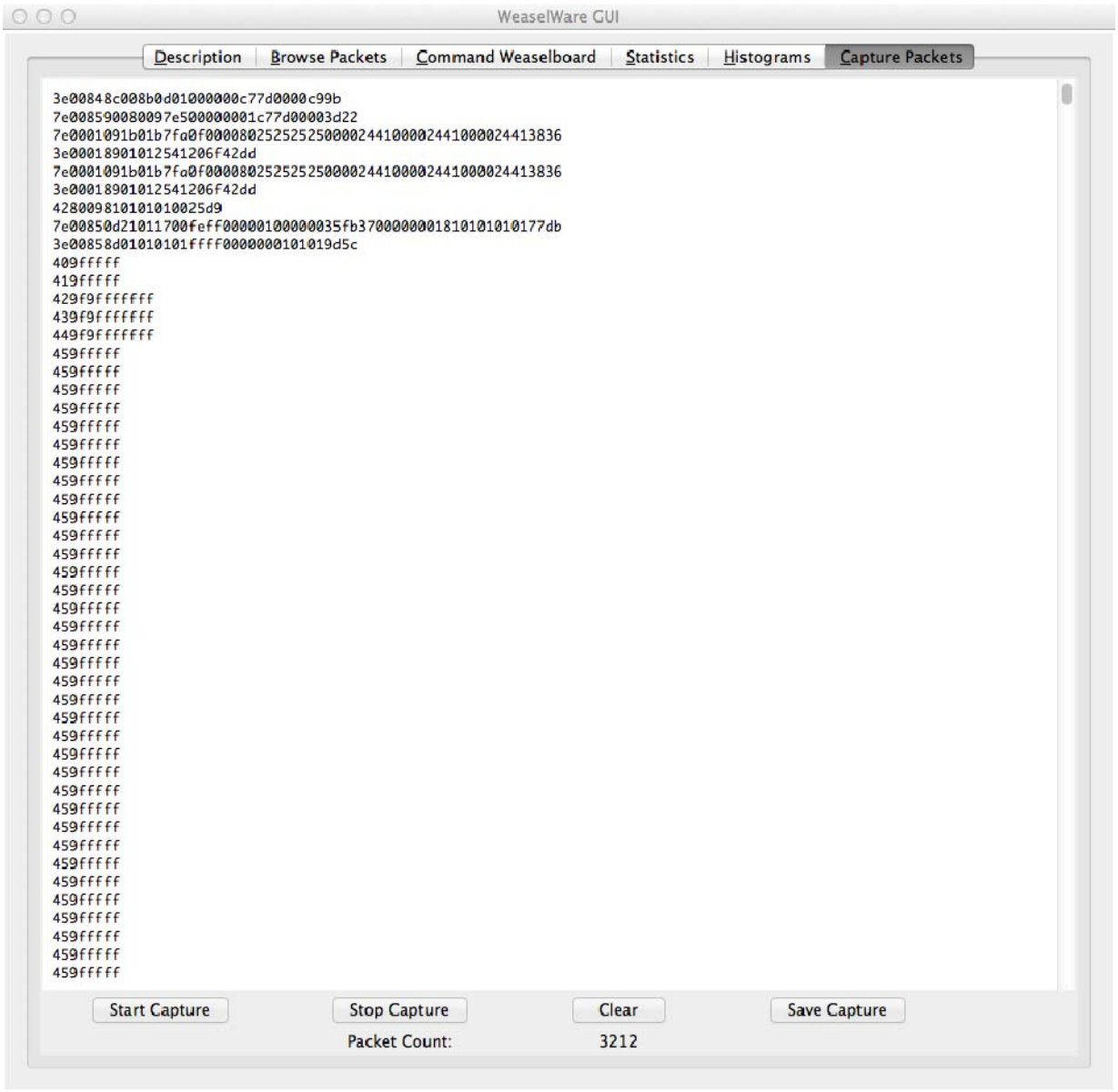


Figure 4. Control and view packet captures.

The Capture Packets tab (weaselcapture_gui) allows the user to start and stop captures, and displays received packets.

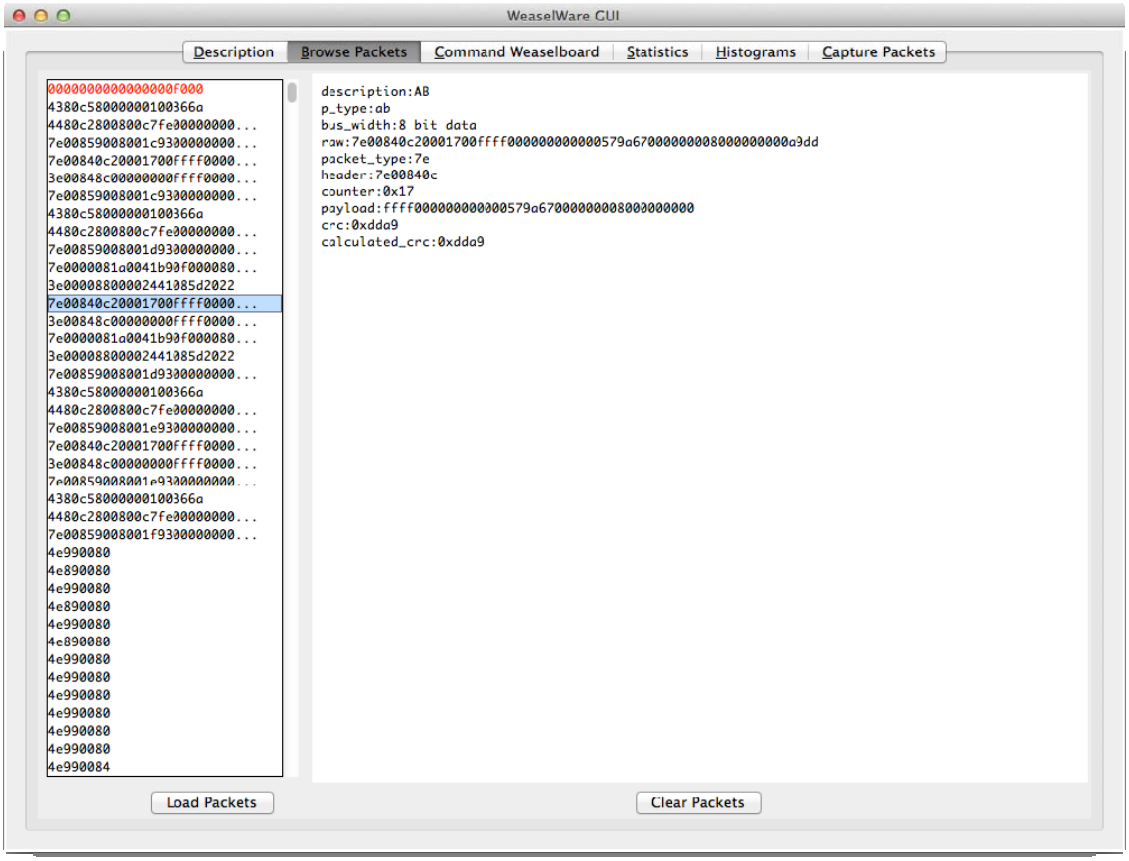


Figure 5. Browse captured packets.

The Browse Packets tab (weaselbrowse_gui) provides a GUI that displays WeaselTalk packets, with known fields parsed out.

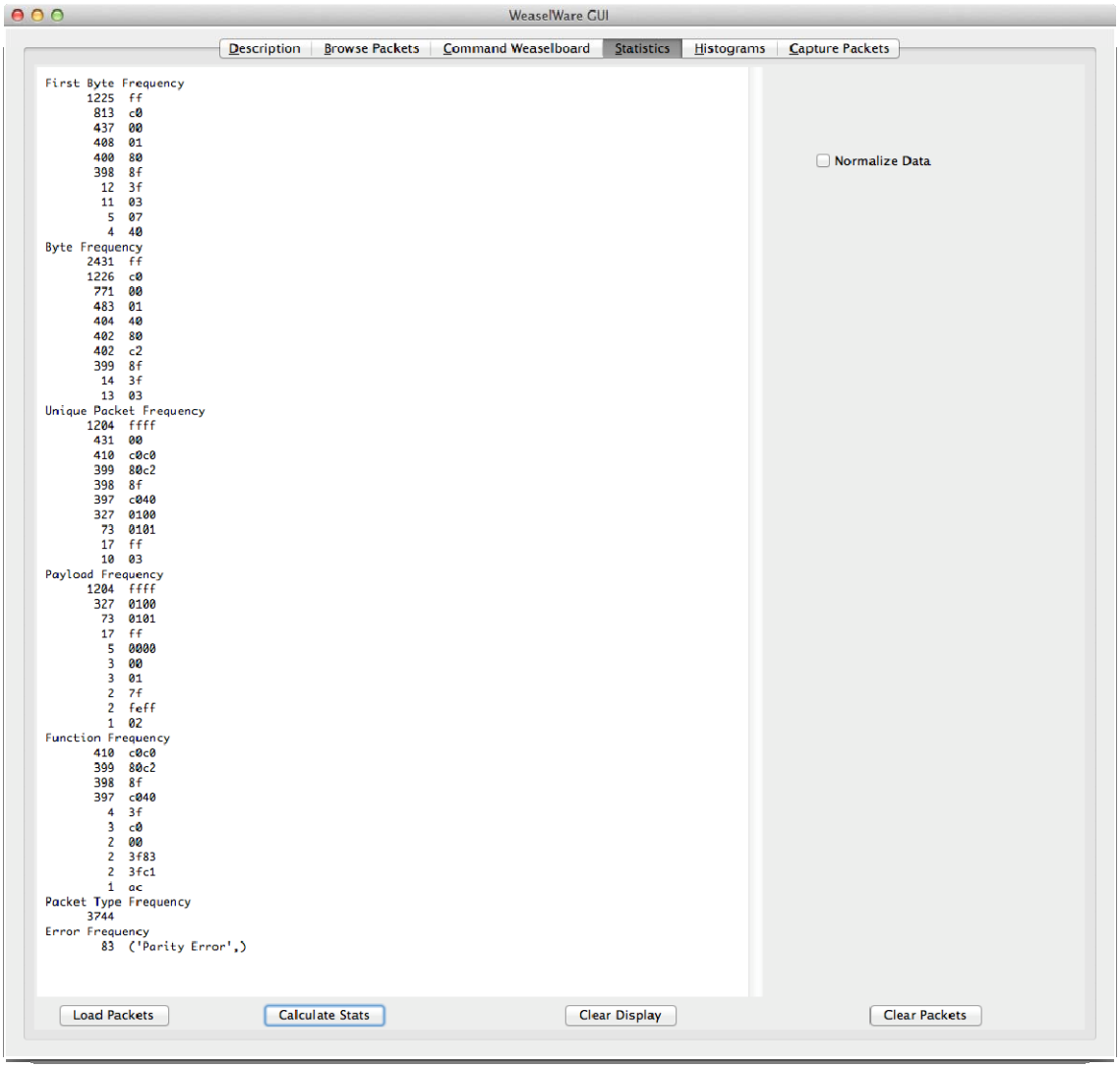


Figure 6. Display captured packets statistics

The Statistics tab (weaselstats_gui) provides a GUI for getting basic packet statistics (including byte frequency, unique packet frequency, etc.) from a capture.

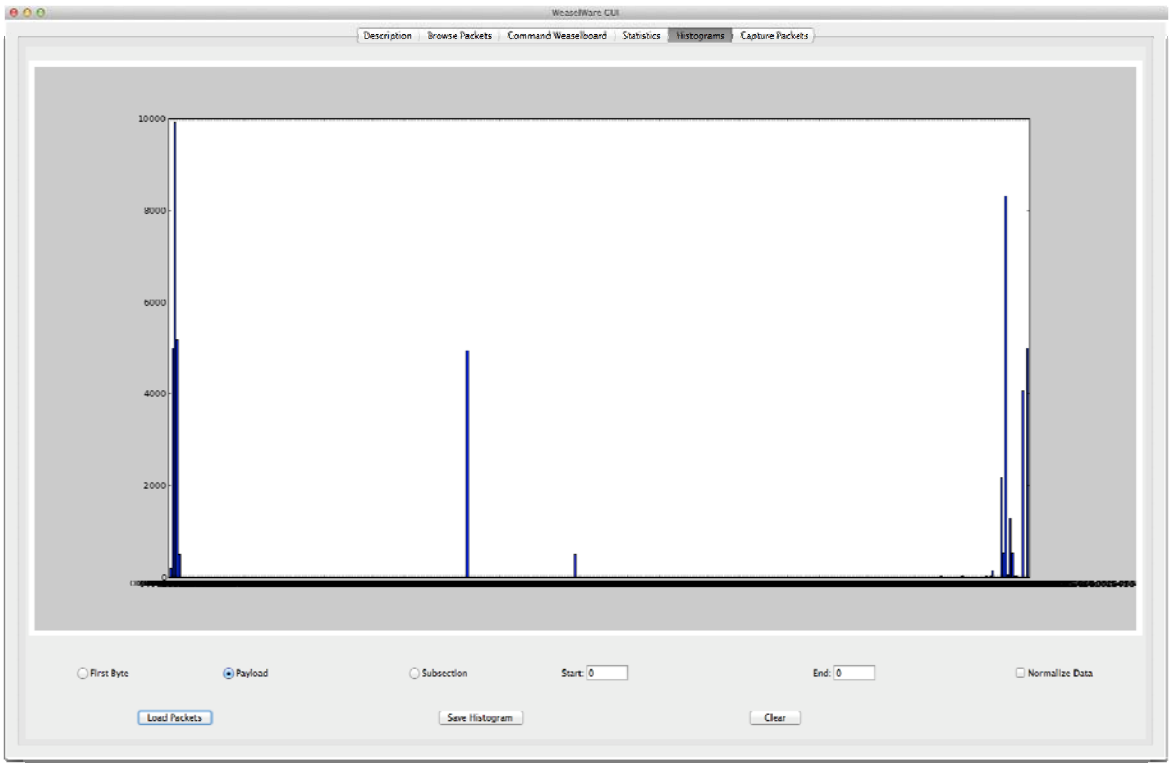


Figure 7. Histograms of packet properties.

The Histograms tab (weaselgraph_gui.py) generates histograms of the frequencies of first-bytes and payloads for all of the backplane packets. Histograms can be written to file or displayed.

4.1 Bayesian Classifier

The analysis workstation implements a simple packet classifier (weaselclassify) using Bayesian classification. A Bayesian classifier calculates the probability for each feature to exist for each category based on experience through training. Such a classifier is "naive" if it assumes that the features are independent of each other. The system trains by taking previous packet captures and categorizes of that document, extracts the features, and updates the internal feature and category data structures. The system classifies traffic by calculating the probabilities that the captured set is in each of the known categories.

4.2 Rules Based Detection

Some packets are only used for particular behavior, such as uploading new logic or firmware. By detecting those specific packets, we can detect this behavior.

5 DISCUSSION

We built the WeaselBoard, a device that connects to modular PLCs, to address the threat of low-frequency, high-impact attacks from sophisticated adversaries that use zero-day attacks against PLCs. By connecting directly to the PLC backplane, WeaselBoard has access to all traffic at a low (hardware) level, and can detect the effect of exploits against PLCs as soon as the state of the PLC changes, instead of after serious damage has occurred.

WeaselBoard introduces a new capability for PLC monitoring, and has applications for real-time monitoring of high-assurance process control systems, forensics as part of an incident response investigation, and periodic system audits and maintenance.

WeaselBoard has been tested in in a variety of systems at Sandia and government laboratories. WeaselBoard has been validated using control system physical processes to provide realistic environments, resulting in a Technical Readiness Level (TRL) of 6. Sandia National Laboratories is continuing to develop this breakthrough technology.

Next steps include exploring alternative hardware platforms and embedded operating systems that could support traffic analysis on-device instead of processing on an analysis workstation and transitioning the design to a practical commercial product that could be used in industrial settings by operators without special training. We are also investigating: correlation between network and backplane traffic; prevention of some backplane traffic (for use as an intrusion prevention system), possibly by taking an active role on the backplane; and additional statistical classification methods for traffic anomaly detection.

6 DISTRIBUTION

1	MS0359	D. Chavez, LDRD Office	1911
1	MS0620	David White	5620
1	MS0671	Jennifer Depoy	5628
1	MS1205	James Peery	5600
1	MS0899	Technical Library	9536 (electronic copy)


Sandia National Laboratories