Finding the Needle in the Hardware Haystack

•••

Identifying and Exploiting Hardware Vulnerabilities via the HRES Process

Who are we?

Timothy Wright - Penetration Testing Team Lead, American Electric Power

Currently focusing on penetration testing, threat emulation and hardware reverse engineering

19 Years of security experience with a focus on offensive security

Member of independent security research team Nullbyte

@redteam_hacker

Who are we?

Stephen Halwes - Cyber Researcher, PreTalen Ltd.

Currently focusing on embedded hardware and software reverse engineering

6 Years experience with a focus on reverse engineering and IT security

Member of independent security research team Nullbyte

@genonullfree

What is the focus of today’s talk?

Today we are talking about why reverse engineering of embedded hardware systems are an important part of a security program.

Discussion regarding current state of hardware / embedded RE security testing

We are discussing why your organization should invest time and resources into performing this work.

We are presenting a standard for review and support by the security community.

We are pushing ourselves to learn more about this area since we are both very passionate about this work and still learning.

A Little Background

Is embedded device security a new problem?

No, but now the risk is increasing with the staggering number of new embedded devices being deployed in organizations.

Proliferation of new powerful microcontrollers are allowing for expanded functionality with an even smaller device footprint.

“Shadow IT” or BYOD is expanding and making it easy for our users to do the things they want without thinking about or consulting with security.

Everything is now connecting to the Internet and hitting the web for content, updates, or pushing and pulling data.

Problem - No Standards for Embedded RE Testing

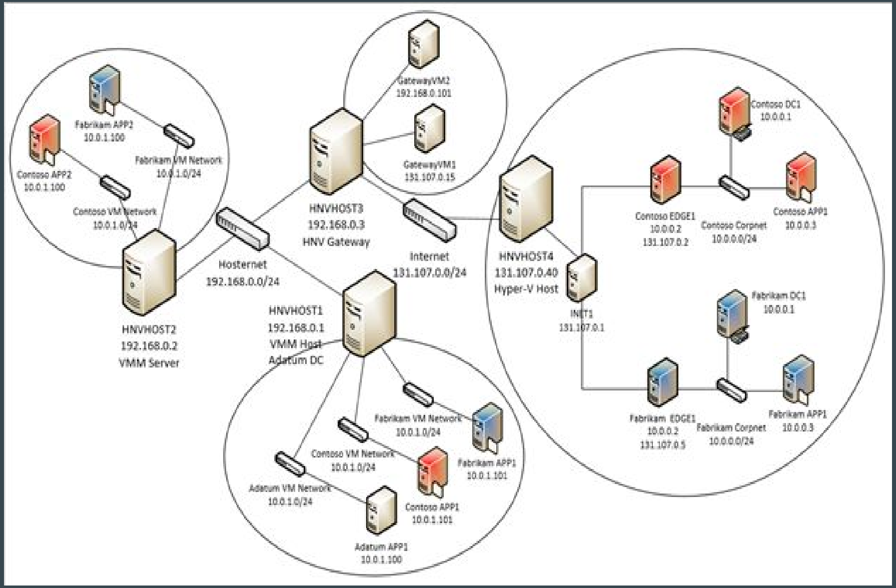
Has become a new service offering with many security consulting companies who also perform penetration testing services.

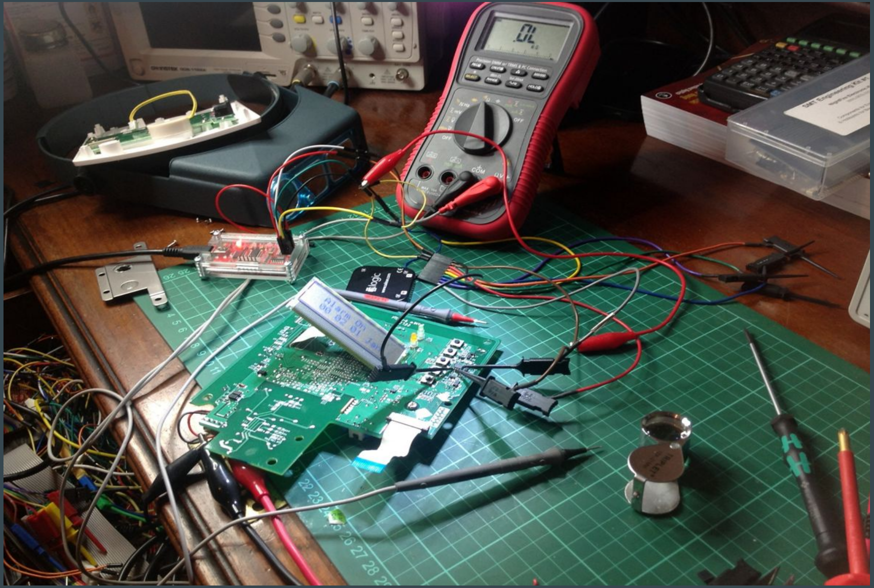
Work performed currently does not follow a defined standard for hardware reverse engineering.

This testing has become new “secret sauce” many companies are offering to clients.

Can be very difficult to scope this type of assessment if you have never requested this type of service before. You rely on the vendor to tell you what you need done.

Network Penetration Tests vs Hardware Security Testing





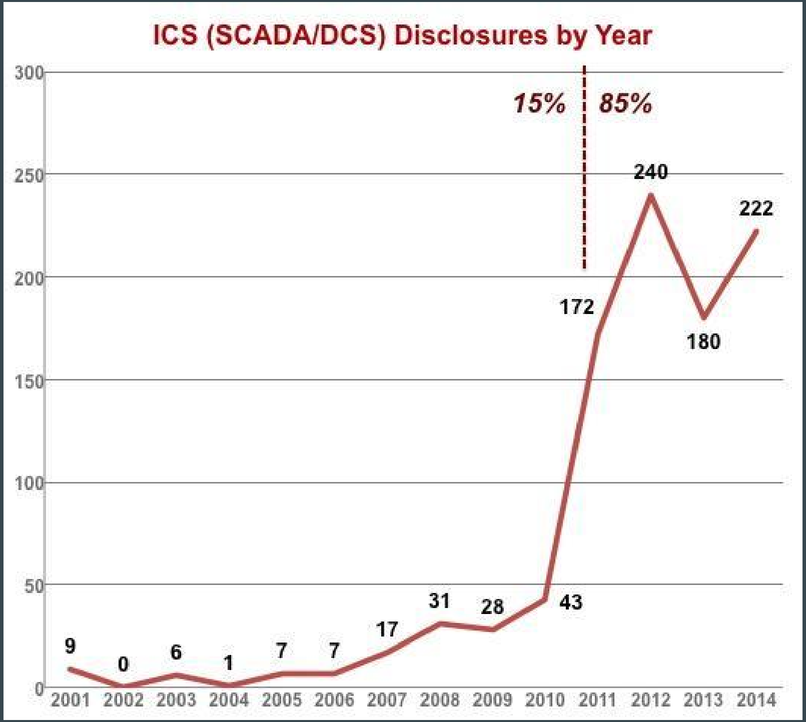
Research on Hardware Security

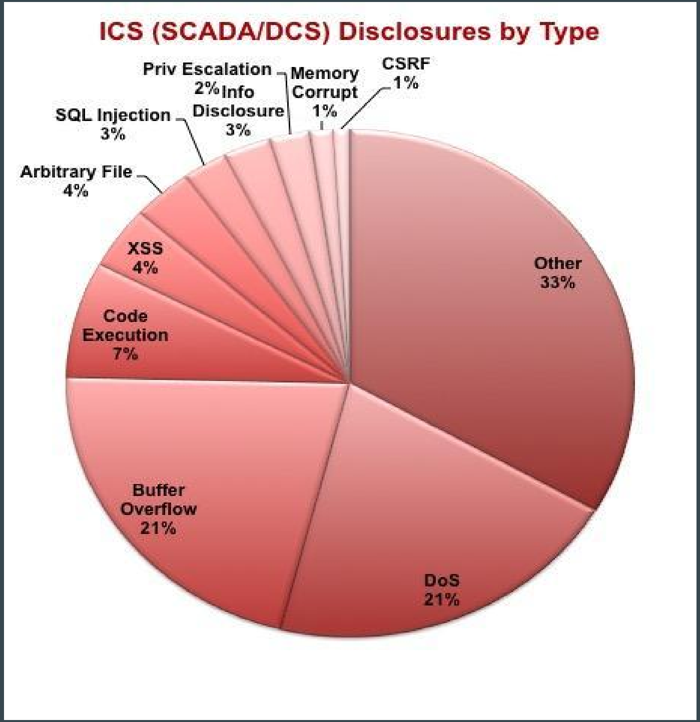
Xipiter has an excellent website and blog. Thanks for all you do guys!

We are not the first to talk about hardware and embedded system security, nor will we be the the last:

- Deral Heiland - printers and Praeda as well as SNMP
- Dave Kennedy - Home automation talk Defcon 19
- Joe Grand - Can you really trust hardware Blackhat 2005
- Travis Goodspeed - Journey to the Center of the HP28 Defcon 16
- Kevin Finisterre - Shelling out (getting root)... Derbycon 5
- Stephen A. Ridley - Hardware Attacks... 30C3

SCADA Disclosures - Trend analysis





What is the risk to your organization?

When we look at the hardware security problem we tend to associate several risks to an organization if these systems are compromised. They include:

1. Data exfiltration - Loss of IP via communication protocols enabled on device.

2. Access to the network - Bridgehead on the network; network perimeter compromised.

3. Physical damage - Legacy systems that are more sensitive damaged either intentionally or unintentionally by attack.

4. Modification - System or data tampered with and system altered

5. Theft of Service - Stealing services or unlocking functionality that would normally require a charge.

Common embedded vulnerabilities

When we look at hardware security we see a common theme in many of these risks that we feel hardware reverse engineering can help you identify. These include:

1. Stack overflow, code execution and/or DoS vulnerabilities
2. Undocumented backdoors in the system
3. Physical compromise via console or debug interface (JTAG, UART)
4. Wireless protocol compromise
5. Weak cryptography
6. Weak firmware management processes (firmware updates)
7. Web interface attacks

Stack Overflow, Code Execution, DoS Vulnerabilities

Risk:

Many embedded devices are running on a UNIX or Linux based operating system and have applications running code that was developed in C and in some cases C++.

These applications can be susceptible to buffer overflows or DoS attacks if not properly developed.

If exploited by an attacker they could execute code remotely on the device and possibly gain some type of remote connectivity or control of the system. Or crash the system and cause damage to the device.

Undocumented Backdoors in System

Risk:

The core OS can have user accounts and default passwords still enabled, undocumented commands and debug functionality that have been left active and can be used to login or for remote command execution.

If compromised the attacker could gain remote access to the device and/or network, execute system commands and possibly pivot onto other machines.

Physical Compromise via Console or Debug Port

Risk:

- Debug test points often leave root or full hardware access available to anyone with physical access to the device, allowing anyone to compromise the device's integrity.
- Supply chain attacks can happen if the right people are motivated or compromised.

Wireless Protocol Compromise

Risk:

- Security flaw in improperly designed or implemented wireless protocols could lead to a total compromise of device.
- Wireless compromise could be worse than wired compromise, because it can be more difficult to locate or disconnect the perpetrator or device.

Weak Cryptography

Risk:

- Default SSH/SSL keys embedded into firmware or each device could allow someone to remotely connect or sniff the network traffic.
- Advertized device cryptography is not always accurate, or implemented properly.
- Improper security implementation could allow automatic connection negotiation to fall back to an insecure cipher suite.

Weak Firmware Management Process

Risk:

- Unprotected interfaces (web admin, USB, console etc.) can allow attacker to gain access to update functions of device.
- Updating the OS or firmware seems to never be done in organizations.
- Attacker can upload patched firmware of the device.
- Added functionality could allow for remote access, communications monitoring, data tampering or even full system destruction/DoS as was seen in the BlackEnergy malware.

Web Interface Attacks

Risk:

An attacker can target the web management interface of the device to gain access to functionality within the device without authenticating.

The attacker could also target any embedded databases or files in order to extract sensitive data from the device.

Since many of these embedded devices are connected to the Internet directly, finding the web management interface is not difficult using Google or Shodan.

Risk Examples

Seagate NAS hardcoded root password

The Seagate NAS had an undocumented root account with the password set to “root”. This account was discovered when a security researcher reverse engineered the device firmware.

An attacker could enable the telnet service remotely on the device and then connect with the root credentials and have full control of the device.

The attacker could also use the device as a pivot onto the target network or access all data stored on the device.

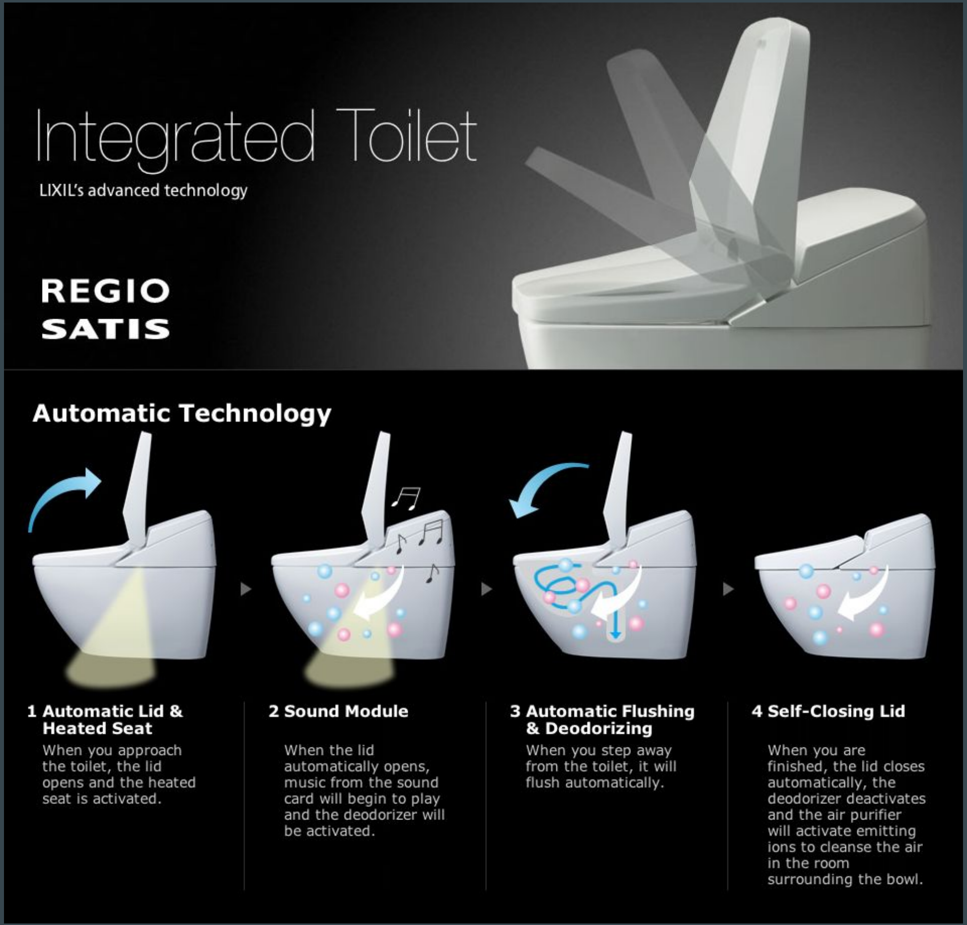
More info: CVE-2015-2874

TheMoon worm

Bypassed admin authentication on Linksys routers without actually knowing the credentials by targeting several .cgi files directly.

Once authenticated, the malware started to flood the network with TCP 80 and 8080 outbound traffic. This results in degraded Internet performance.

An RCE Exploit has been released to ExploitDB which uploads a binary to the device and binds a shell to a TCP port.



Integrated Toilet

LIXIL's advanced technology

REGIO SATIS

Automatic Technology

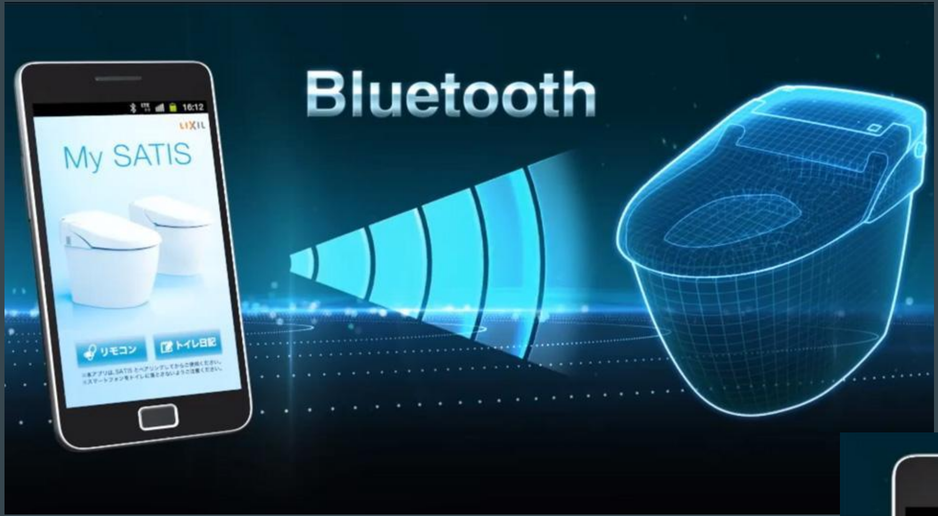
1 Automatic Lid & Heated Seat
When you approach the toilet, the lid opens and the heated seat is activated.

2 Sound Module
When the lid automatically opens, music from the sound card will begin to play and the deodorizer will be activated.

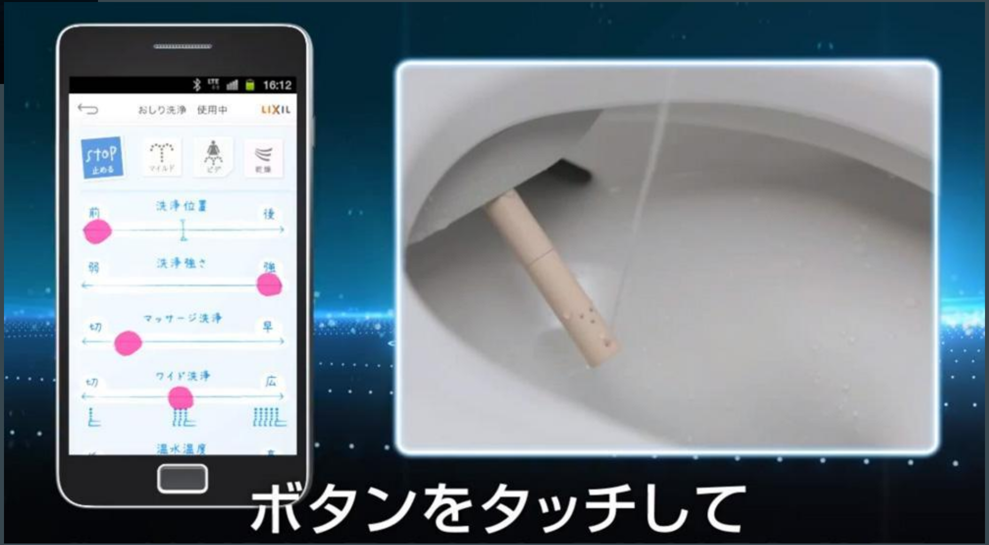
3 Automatic Flushing & Deodorizing
When you step away from the toilet, it will flush automatically.

4 Self-Closing Lid
When you are finished, the lid closes automatically, the deodorizer deactivates and the air purifier will activate emitting ions to cleanse the air in the room surrounding the bowl.

Bluetooth controlled Toilet. What could go wrong?



Bluetooth exploit allowed hackers to flush, activate the built-in bidet function.



Trustwave found that the toilet had a hard coded “0000” pin for bluetooth. So as long as you can talk to the device you can connect.

No worries. Bluetooth is short range!

Not so short range...

BlackEnergy

This malware left 225,000 people in the dark in the Ukraine.

Utilized a KillDisk component that would render targeted computers unbootable.

Targeted two processes: Komut.exe (“command” in Turkish) and sec_services.exe. The second process may belong to software called ELTIMA Serial to Ethernet Connector or to ASEM Ubiquity, a platform commonly used in Industrial Control Systems.

This resulted in the utility having to move to manual operations in order to restore power to customers.

The attackers used the control systems and the software via direct interaction to cause the outage.

The risk justifies the work

These are just a few examples of systems that have been found to be lacking in security or have been targeted in attacks.

We believe that the risk justify the work.

Based on recent trends these risks will only continue to increase unless we work with vendors to help improve product security.

Threat actors are performing this research. We (security) need to be more proactive in our security testing.

We need more than a simple risk assessment or trusting the vendor’s that the device was designed with security in-mind and tested properly.

How do we do the testing?

Defining the testing Process

We feel that the community needs to define a standard on how to properly RE hardware and embedded devices.

Having properly defined testing processes will allow for measurable and repeatable testing to be performed.

Providing more than just a vulnerability scan of a target device.

So based on all of these reasons we are proposing....

Hardware Reverse Engineering Standard (HRES)

The HRES is based on the Penetration Testing Execution Standard (PTES) format, but modified and augmented to concentrate on hardware testing.

The HRES is Open Sourced, licensed GFDL 1.2, as is the PTES.

We are calling for collaboration with the hardware hacking community to help create a standard for the general hardware reverse engineering and testing process.

Our aim is to make this a complementary fork of the PTES.

The HRES Process

HRES maps to the 7 PTES main sections

- We felt the HRES can map to the 7 main sections of the PTES and yet it deserves a stand alone process.
- Not all hardware security assessments are considered penetration tests. However the findings from these assessment could enable your penetration testing team during an engagement to gain further access.
- The PTES does an outstanding job in providing us with the framework we need to perform testing. We just need to define each testing section.

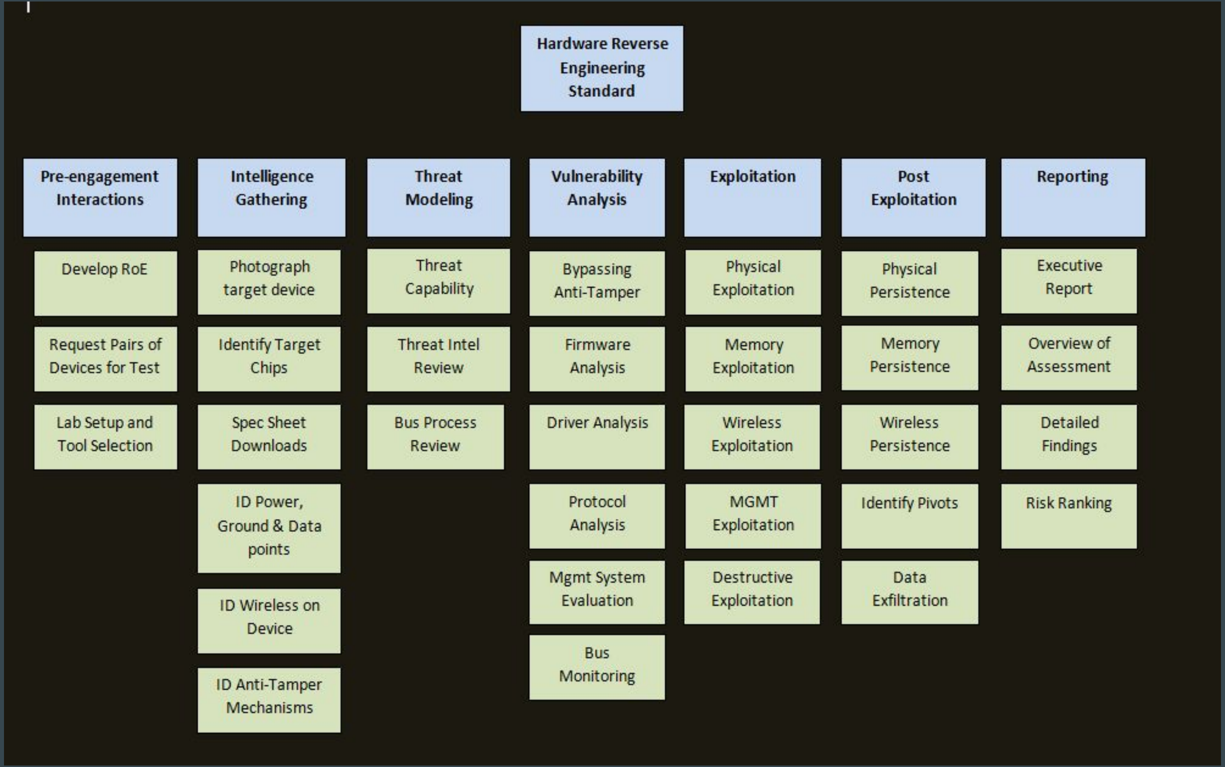
<u>PTES - Main Section Outline</u>

1. Pre-engagement Interactions
2. Intelligence Gathering
3. Threat Modeling
4. Vulnerability Analysis
5. Exploitation
6. Post Exploitation
7. Reporting

<u>HRES - Main Section Outline</u>

1. Pre-engagement Interactions
2. Intelligence Gathering
3. Threat Modeling
4. Vulnerability Analysis
5. Exploitation
6. Post Exploitation
7. Reporting

HRES process outline



Phase 0: Safety First!

Please keep safety in mind when doing hardware work!

You can get seriously hurt if you mess with powered systems.

Always unplug your devices when disassembling.

Always make sure your device is grounded properly.

When in doubt ask for help.

Phase 1: Pre-Engagement Interactions

First ensure that your legal department has reviewed your testing plan and given approval for testing of devices.

Establish Rules of Engagement and scope:

1. Require a minimum of 2 sets of devices: One for constructive testing and the other for destructive testing (soldering and desoldering on boards etc.).
2. Develop testing goals: What are you looking to prove in the engagement? Is it just DoS or do you want to see if the attacker can gain remote access to a substation from the PLC?
3. Setup testing lab / bench: Time to get your work space setup and ready for testing.
4. Clear lines of communication: Who do you call if you brick a device and need to have it reset or have another shipped to your testing location?
5. Request debugging or programming tools: Some devices have specialized programming cables or devices. Requesting a set for testing if you do not have a set available.

United States Code, Title 17, Section 906

US law does allow for hardware RE as it relates to a security analysis of a device.

According to US law :

“It is not an infringement... for a person to reproduce the mask work solely for the purposes of teaching, analyzing, or evaluating the concepts or techniques... or the circuitry, logic flow, or organization of components used in the mask work.”

We recommend that you have your internal legal team review this but that is should be included in the HRES so that it can be referenced by corporations as their right to test devices and systems for risks and vulnerabilities.

Phase 2: Intelligence Gathering

During this phase we will be gathering data regarding our device, the chips and any firmware on device. We will also need to document how the device looks prior to disassembly.

- Spec sheet research and schematic download.
- Photograph device prior to, and during, disassembly
- Identify chips on boards (QFP, BGA,TSOP,DIP,SOIC,PDIP)
- Utilize a multimeter to identify power, ground and data pins.
- Identify test interfaces on boards (I2C, JTAG, UART, etc.)
- Identify wireless chips on board (802.11x, bluetooth, zigbee, etc.)
- Identify any anti-tamper mechanisms on the device (lock bits, glitch detection, self destruction etc.),

Phase 3: Threat Modeling Process

This phase will help you narrow your testing focus by identifying potential targets. This should include business process reviews, threat intel analysis, and threat capability analysis.

Which components pose greatest risks of being compromised?

- Customer billing data sent unencrypted over mesh network
- Gas meter at customer site with RF interface
- Remote access into support system via substation wireless AP

Using tools like a CARVER We can now start to prioritize our testing efforts based on our risks

CARVER Risk Evaluation Method

CARVER is a system used to assess targets to see which one needs to be addressed first based on a risk score.

Rank	**1 (low risk) to 10 (high risk)**
Criticality	- Target value
Accessibility	- How easy to get in and out
Recuperability	- Time to recover
Vulnerability	- Chance of successful attack
Effect	- Positive or negative effects
Recognizability	- How easy to recognize target

http://fas.org/irp/doddir/army/fm34-36/appd.htm

Phase 4: Vulnerability Assessment

During this phase we will be testing both the hardware and software for potential vulnerabilities. This can include:

- Solder jumpers on board (as needed)
- Extract data from flash chips (SPI, EEPROM, etc.)
- Test access level / availability (I2C, JTAG, UART, etc.)
- Sniff data from buses with logic analyzer
- Record and process wireless data as needed
- Scan for network services to review
- Software reverse engineer binaries, data from flash chips, configuration and backup files

Shikra - Connecting into your target

Shikra is a USB device that supports many protocols out of the box (UART, JTAG, SPI, etc.)

It’s based on the FTDI FT232H chip

Using tools like minicom you can use Shikra to connect into various interfaces found in hardware.

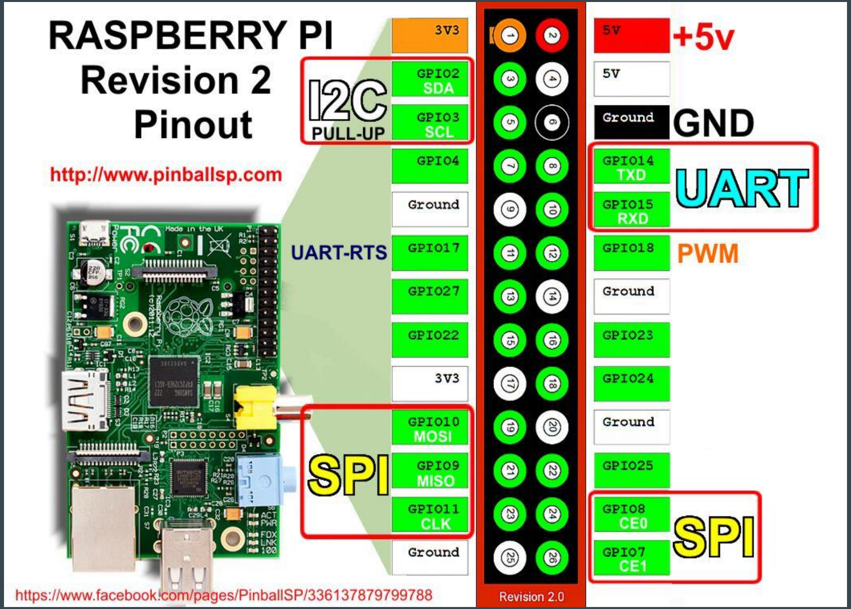
Supports OS X and Linux natively.

http://www.xipiter.com/musings/using-the-shikra-to-attack-embedded-systems-getting-started

Building your own RE system on a Raspberry Pi

By leveraging a Raspberry Pi we can build a very nice testing space on a very small footprint.

Supports UART, I2C and SPI on the board. With Linux on the device it makes doing firmware analysis pretty easy once you have it downloaded.



Phase 5: Exploitation

Physical Exploitation, Memory Exploitation, Wireless Exploitation, Management System Exploitation, and Destructive Exploitation will all be avenues of attack.

Develop Proof-of-Concept exploits against discovered vulnerabilities to demonstrate code execution and process redirection.

Bypass restrictions (firewall [Data Diodes] or IDS, access permissions, etc.) to show that network controls can be bypassed.

If push comes to shove, assess creative exploit methods (social engineering) to demonstrate the insider threat.

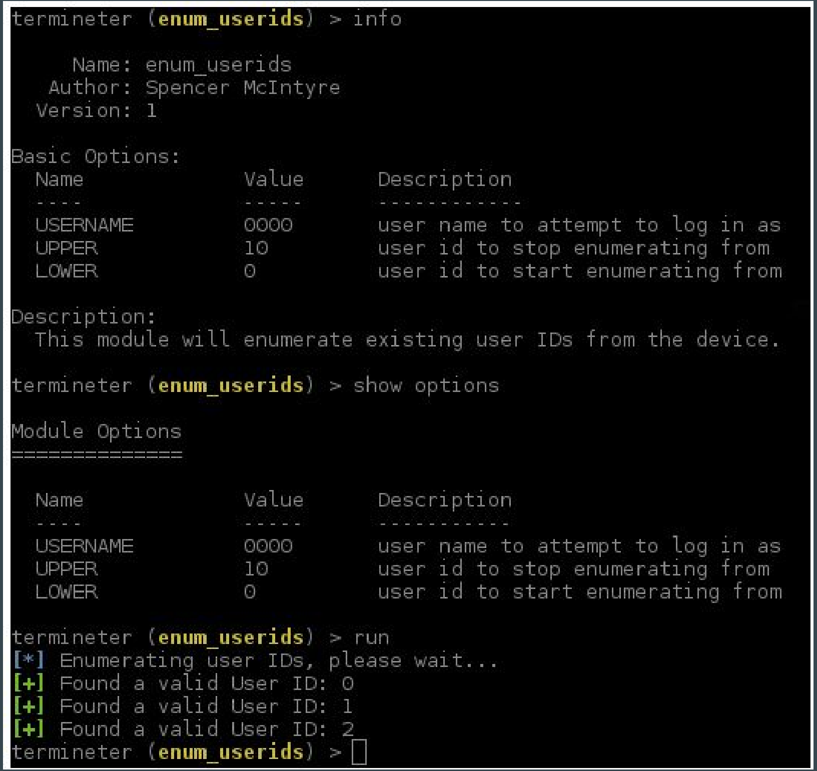
Example: Termineter Framework

Python framework which provides security platform to test smart meters.

Similar to metasploit but just focuses on meter testing and exploitation.

Implements the C1218 and C1219 protocols for communication over optical interface.

Communicates with meters via a connection using ANSI type-2 optical probe with a serial interface.



```
termineter (enum_userids) > info

     Name: enum_userids
   Author: Spencer McIntyre
  Version: 1

Basic Options:
  Name                 Value      Description
  ----                 -----      ------------
  USERNAME             0000       user name to attempt to log in as
  UPPER                10         user id to stop enumerating from
  LOWER                0          user id to start enumerating from

Description:
  This module will enumerate existing user IDs from the device.

termineter (enum_userids) > show options

Module Options
==============

  Name                 Value      Description
  ----                 -----      -----------
  USERNAME             0000       user name to attempt to log in as
  UPPER                10         user id to stop enumerating from
  LOWER                0          user id to start enumerating from

termineter (enum_userids) > run
[*] Enumerating user IDs, please wait...
[+] Found a valid User ID: 0
[+] Found a valid User ID: 1
[+] Found a valid User ID: 2
termineter (enum_userids) >
```

Phase 6: Post-Exploitation

During this phase we will now be showing how exploiting the device could lead to further system compromise. Other areas of interest include data exfiltration, network pivoting, destruction of device, DoS.

Some things to think about for this phase of testing:

- Code developed should enable persistent access to device
- Code developed should enable privilege escalation on device
- Code developed should enable data exfiltration from target network
- If allowed, develop code that could damage or destroy device

Phase 7: Testing Report

"Responsible Disclosure" should be performed for any 0-day discovered in vendor systems.

Lanes of communication should already be open so this should not be difficult at this stage of the game.

Two reports should be developed:

- Executive Report: for higher-level summary for senior management. This should be concise to the point. Keep it under a page if possible and no hex or assembly.
- Technical Report: full reproduction of RE and exploitation scenario. Should have details on each finding and the risk associated with the results.

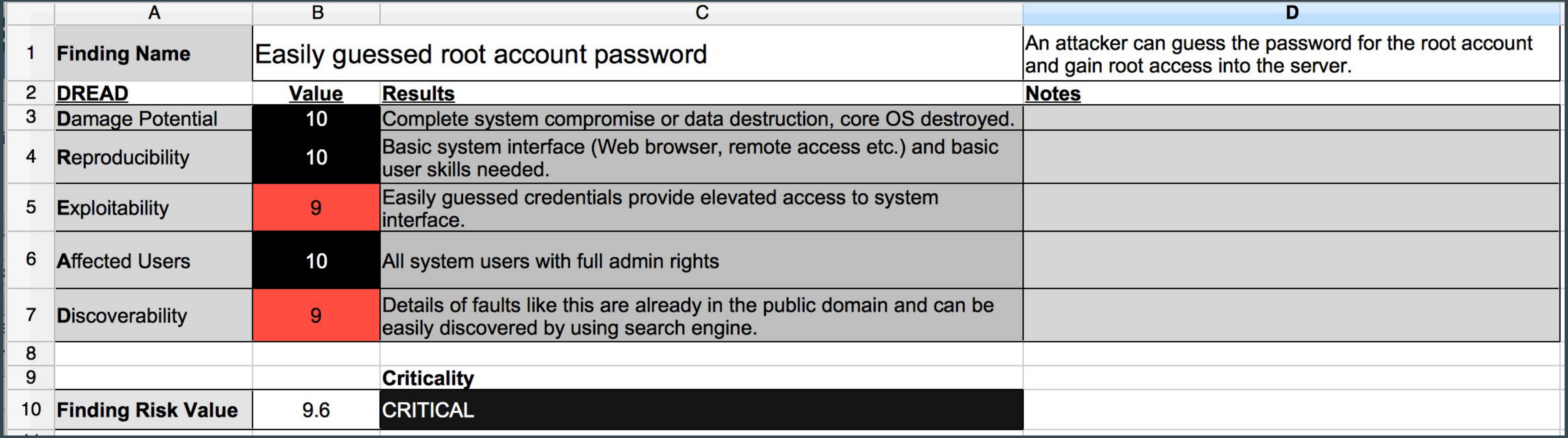
DREAD Vulnerability Modeling

As we develop the findings from the testing we will need to risk rank what we have discovered. We can use risk modeling tools to help facilitate this process.

There are many risk ranking frameworks online but for this talk we are discussing DREAD and will walk through an example.

Generally uses 5 tiers for risk (Critical, High, Medium, Low, Informational). These are just recommendations. If you have already defined risk tiers for your organization you can just apply them to the model.

DREAD risk model of default root account password



	A	B	C	D
1	**Finding Name**	Easily guessed root account password		An attacker can guess the password for the root account and gain root access into the server.
2	**DREAD**	**Value**	**Results**	**Notes**
3	**D**amage Potential	10	Complete system compromise or data destruction, core OS destroyed.	
4	**R**eproducibility	10	Basic system interface (Web browser, remote access etc.) and basic user skills needed.	
5	**E**xploitability	9	Easily guessed credentials provide elevated access to system interface.	
6	**A**ffected Users	10	All system users with full admin rights	
7	**D**iscoverability	9	Details of faults like this are already in the public domain and can be easily discovered by using search engine.	
8				
9			**Criticality**	
10	**Finding Risk Value**	9.6	CRITICAL	

HRES - A Repeatable Measurable Process

Based on what we have outlined and tested we feel this process is repeatable and measurable.

Can provide our organizations a way to test hardware and embedded systems and ensure each assessment is following a documented standard.

Based on the testing performed we felt that the HRES should be a stand alone process and not just extra steps included in the PTES. But it maps well to the 7 main sections of PTES.

We are hoping that the community can come together on this and help us further define a hardware and embedded security testing strategy.

Closing Thoughts and Next Steps

Where do we go from here?

1. Risks are increasing as we add more embedded devices and systems into our organizations.
2. Our security teams should be testing this hardware to identify the vulnerabilities within these systems and to help manage or reduce the risk.
3. We propose the development of the HRES standard to help define proper hardware reverse engineering testing so that it is a repeatable, measurable testing process.
4. We are at present setting up the website, a wiki, and getting each section of the HRES process outlined. We are also working on expanding the mind map.
5. If you would like to help please go to http://hre-standard.com and start supporting and contributing.

Thank you for your time!

Questions?

Contact us:

Timothy Wright - @redteam_hacker

Stephen Halwes - @genonullfree

http://www.nu11byte.com

Build up your testing lab

Cheap Hardware to Reverse

You can pick up all kinds of old hardware at your local thrift store for next to nothing.

I have found Linksys routers, DSL modems and various devices with ethernet interfaces very cheap.

It is also nice to reverse a target and not feel bad when you let the smoke out of a device that only cost you 2 dollars.

You can also find really cheap shoes and socks.

Tool time

So we have spent some time discussing the risks, the testing processes involved so now it’s time to talk software and hardware tools.

In order to perform this testing you will need to invest into a lab. This could be a low end portable solution all the way up to a full hardware RE lab space. If you have a awesome manager and a corporate card then charge it up!

But if you just want to get started on the cheaper end then we have a few solutions for you.

Physical Tools

When we talk “Physical Tools” we are referring to the tools your grand dad used to build radios or repair the VCR. These tools include:

- Soldering Iron with fine tip
- Multimeter
- Screwdriver set with security bits
- Hot Air Gun / Hair Dryer
- Needle nose pliers and wire cutters
- Dental picks

Electronic Tools

When we say “Electric tools” we are referring to the devices we use to measure or connect to the hardware device. These include:

Shikra / Bus Pirate

JTAGULATOR

Saleae

Picoscope / DSO Nano v3

USB RS232 breakout board

Raspberry Pi 2 / 3

DualComm DCGS-2005L

RTL-SDR 2832 / HackRF / BladeRF

Pomona 5250

Software Tools

This includes all of the software applications we could use as part of the assessment. These include:

IDA Pro / Ollydbg / Immunity

Hexdump / Objdump

Minicom

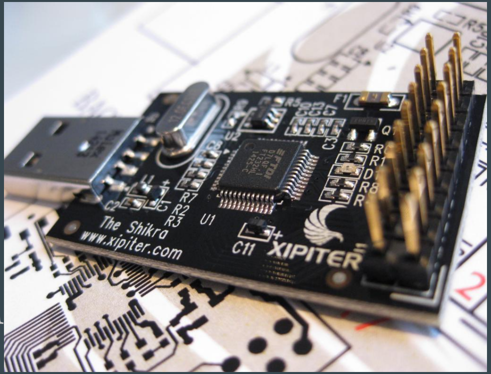
HT Editor / radare2

Flashrom

GNURadio

Promotion of stuff we love

Shikra

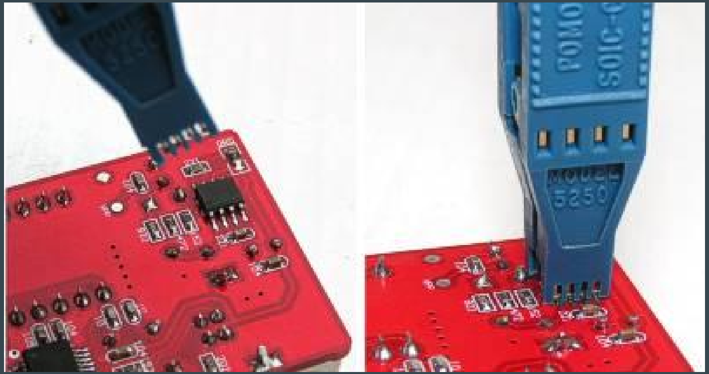


Communicates over UART, JTAG, I2C, SPI, GPIO

Fast, very stable, works great with flashrom for dumping flash chips

Supported on Linux and Mac OS X

Pomona 5250



Clips directly onto an 8-pin SOIC

Quick, solderless testing

Raspberry Pi

Extremely versatile

Can communicate over I2C, SPI, UART via GPIO pins

RTL-SDR

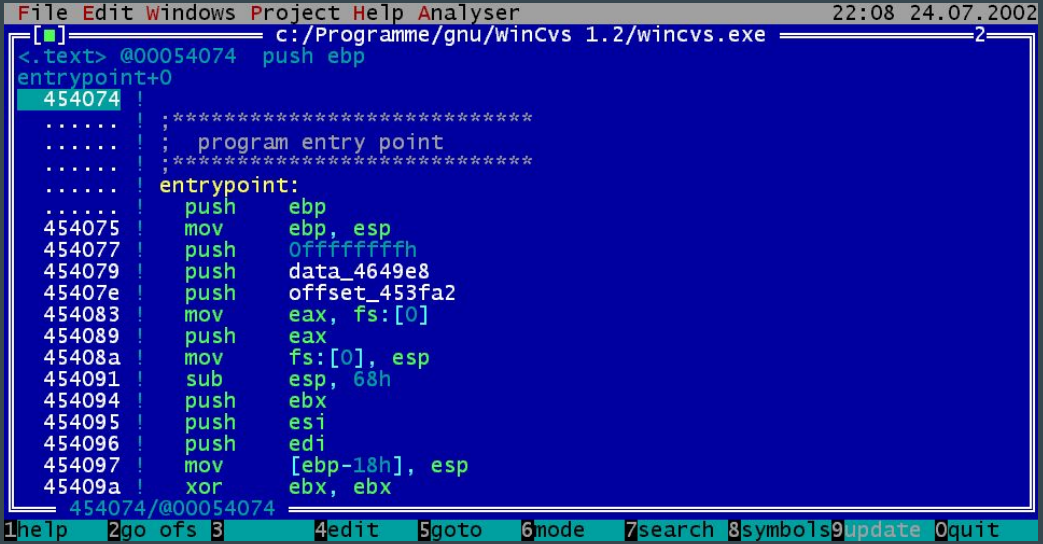
Amazing value for entry level SDR

With GNURadio you can decode many protocols right out of the box

HT Editor

HT is a file editor, viewer and analyzer for executables. The goal of this tool was to combine the low-level functionality of a debugger and the usability of IDE's.

It is distributed under GPL on http://hte.sourceforge.net



```
 File Edit Windows Project Help Analyser                              22:08 24.07.2002
=[■]============== c:/Programme/gnu/WinCvs 1.2/wincvs.exe ==============2=
<.text> @00054074  push ebp
entrypoint+0
  454074 !
  ...... ! ;******************************
  ...... ! ;  program entry point
  ...... ! ;******************************
  ...... ! entrypoint:
  ...... !   push    ebp
  454075 !   mov     ebp, esp
  454077 !   push    0ffffffffh
  454079 !   push    data_4649e8
  45407e !   push    offset_453fa2
  454083 !   mov     eax, fs:[0]
  454089 !   push    eax
  45408a !   mov     fs:[0], esp
  454091 !   sub     esp, 68h
  454094 !   push    ebx
  454095 !   push    esi
  454096 !   push    edi
  454097 !   mov     [ebp-18h], esp
  45409a !   xor     ebx, ebx
=== 454074/@00054074 =====================================================
1help  2go ofs 3       4edit   5goto   6mode   7search 8symbols9update 0quit
```

Radare2

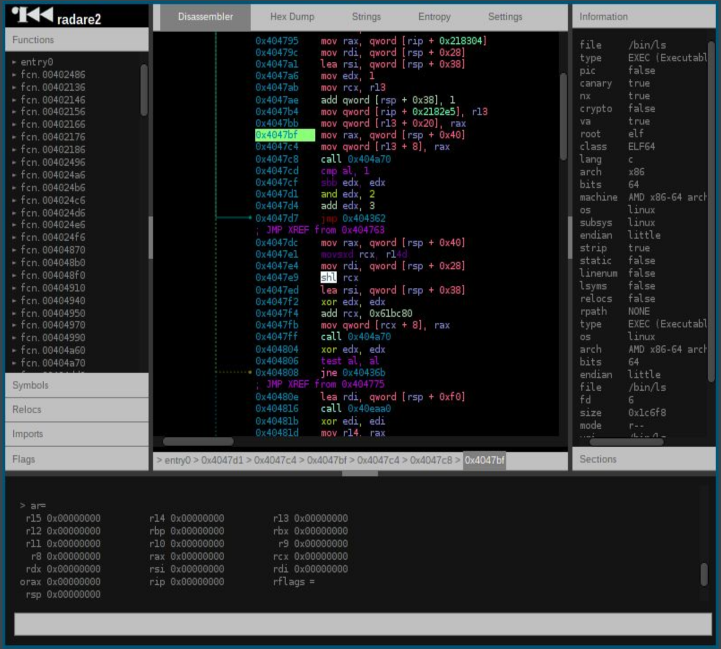
Disassemble and assemble for many different architectures (ARM, x86 & x64, CSR).

Debug with local native and remote debuggers (gdb, rap, windbg, webui, r2pipe)

Supports scripting in Python, Javascript, Go and more.

Incredibly powerful command line decompiler

Sharp TUI and data visualizer



```
entry0
fcn.00402486
fcn.00402136
fcn.00402146
fcn.00402156
fcn.00402166
fcn.00402176
fcn.00402186
fcn.00402496
fcn.004024a6
fcn.004024b6
fcn.004024c6
fcn.004024d6
fcn.004024e6
fcn.004024f6
fcn.00404870
fcn.004048b0
fcn.004048f0
fcn.00404910
fcn.00404940
fcn.00404950
fcn.00404970
fcn.00404990
fcn.00404a60
fcn.00404a70
```

```
0x404795    mov rax, qword [rip + 0x218304]
0x40479c    mov rdi, qword [rsp + 0x28]
0x4047a1    lea rsi, qword [rsp + 0x38]
0x4047a6    mov edx, 1
0x4047ab    mov rcx, r13
0x4047ae    add qword [rsp + 0x38], 1
0x4047b4    mov qword [rip + 0x2182e5], r13
0x4047bb    mov qword [r13 + 0x20], rax
0x4047bf    mov rax, qword [rsp + 0x40]
0x4047c4    mov qword [r13 + 8], rax
0x4047c8    call 0x404a70
0x4047cd    cmp al, 1
0x4047cf    sbb edx, edx
0x4047d1    and edx, 2
0x4047d4    add edx, 3
0x4047d7    jmp 0x404362
; JMP XREF from 0x404763
0x4047dc    mov rax, qword [rsp + 0x40]
0x4047e1    movsxd rcx, r14d
0x4047e4    mov rdi, qword [rsp + 0x28]
0x4047e9    shl rcx
0x4047ed    lea rsi, qword [rsp + 0x38]
0x4047f2    xor edx, edx
0x4047f4    add rcx, 0x61bc80
0x4047fb    mov qword [rcx + 8], rax
0x4047ff    call 0x404a70
0x404804    xor edx, edx
0x404806    test al, al
0x404808    jne 0x40436b
; JMP XREF from 0x404775
0x40480e    lea rdi, qword [rsp + 0xf0]
0x404816    call 0x40eaa0
0x40481b    xor edi, edi
0x40481d    mov r14, rax
```

> entry0 > 0x4047d1 > 0x4047c4 > 0x4047bf > 0x4047c4 > 0x4047c8 > 0x4047bf

```
file     /bin/ls
type     EXEC (Executabl
pic      false
canary   true
nx       true
crypto   false
va       true
root     elf
class    ELF64
lang     c
arch     x86
bits     64
machine  AMD x86-64 arch
os       linux
subsys   linux
endian   little
strip    true
static   false
linenum  false
lsyms    false
relocs   false
rpath    NONE
type     EXEC (Executabl
os       linux
arch     AMD x86-64 arch
bits     64
endian   little
file     /bin/ls
fd       6
size     0x1c6f8
mode     r--
```

```
> ar=
 r15 0x00000000        r14 0x00000000        r13 0x00000000
 r12 0x00000000        rbp 0x00000000        rbx 0x00000000
 r11 0x00000000        r10 0x00000000         r9 0x00000000
  r8 0x00000000        rax 0x00000000        rcx 0x00000000
 rdx 0x00000000        rsi 0x00000000        rdi 0x00000000
orax 0x00000000        rip 0x00000000        rflags =
 rsp 0x00000000
```

Portable laboratories, anyone?

Hardware testing kits

So the following tool kits were designed to help you get started doing this work and do so without breaking the bank.

The kits have also been designed to allow a team to take the kit on the road or move around an organization to test systems that may be only found in one location or department.

However, you can build a workbench of tools to allow you to do even more in-depth testing. The workbench style-testing setups will be outlined in the HRES wiki online.

Small Testing Kit

1. Raspberry Pi 2 / 3
    a. Flashrom
    b. Radare2
    c. HT Editor
    d. Minicom
2. Shikra
3. Pomona 5250
4. Small screwdriver toolkit

Large Testing Kit

1. Laptop (spare, running Linux)
   a. Flashrom
   b. Radare2
   c. HT Editor
   d. Minicom
   e. Saleae Logic
   f. GNURadio
2. Shikra
3. Saleae

5. RTL-SDR
6. DSO Nano v3
7. Small screwdriver toolkit
8. Multimeter

The End. For real this time...