ESE532: System-on-a-Chip Architecture

Day 26: April 24, 2017
Security

Today

• Security Issues
• Memory
• Input
• Output
• Cryptography

Message

• SoC Designers need to be concerned about security
• Hazards are real, understandable
  – Avoidable, tricky…
• Things that make it easier and harder than general-purpose, best-effort
• Consider CIS331, CIS551

Security Issues

• What security issues arise for the SoC designer and applications?
• What make easier / less of an issue?
• What make harder or more important?

Sensationalism

• Target data breach
  – Millions of credit cards; enter through HVAC
• Car Hacks
  – Take over brakes, speed, …
• DDoS
  – Through networks devices/IoT

Potential Security Concerns Arise

• Bug free program
  – with no input
  – and no output,
  – might have no security concerns
• …but could it do anything?
• …unintended outputs?
Security Concern Sources

• Bugs – may allow program subversion
  – Make it do something designer not intend
• Inputs – Allow attacker manipulate
  – Trust inputs?
  – Cause system to crash?
  – Poke bug to change data or run code
• Outputs – Give information
  – Limit to intended recipients?
  – Extract secrets (including keys)

Issues

• Confidentiality
  – Secrets remain secrets
• Integrity
  – Data and code not changed
  – Only controls as intended
• Availability
  – Continues to perform intended function

SoC Challenge

• Embedded systems interact with physical world
  – Control may cause physical (life-critical) damage
  – Sensing may make physical information available
• Networks components
  – Exposed to world at large
    • Attacks, spoofing, monitoring, crash, DoS

SoC Opportunity

• Run small, fixed set of software
  – Few Lines-of-Code (LoC)
  – Not need to run arbitrary user-supplied code
• Handle constrained input
  – Not arbitrary, unstructured user input?

Bug Rates

• Industry average is 15—50 bugs per 1000 LoC
  – Remained true for decades
  – Not all exploitable
• Google Chrome 380 in 6M LoC~0.06
  – CVSS>=7
• Firefox 395 in 8M LoC~0.05
• Cannot assume program is bug free
  – Especially if it is large

Raw Bits

• Where do we store instructions, stack, heap, integers, floating-point values, pointers?
Memory

Memory Contents

- Instructions
- Data
- Data structures
  - Pointers
- Program call graph
  - Stack

How tell them apart?

Stored Program Processor

- Instructions stored in memory
- One big, undifferentiated memory
  - Containing instructions, data, stack, heap…
- Powerful
  - Loading in new programs
  - Generating new code at runtime
  - Flexible division of memory space
- Dangerous…

C

- C allows construction of arbitrary pointers
  \[ \text{int } \ast \text{ptr} = (\text{int } \ast)0x1000; \]
  \[ \text{ptr}[3]=0x0773; \]
- …and references beyond the end of pointers
  \[ \text{int data}[100]; \]
  \[ \text{data}[2376]=0x0aa734; \]
  \[ \text{data}[-578]=0xffff348c; \]

Dangerous

- Bugs may scribble over memory
  - Violate integrity of code or data
- …or, allow attacker to access or write memory
- Pointer that points to unexpected location
- Out-of-bounds reference
Very Dangerous

int data[128];
which = read_input();
write_output(data[which]);

• What does this allow?

Heartbleed

• Attack on SSL
  – CVE-2014-0160
• Could use
  – User input + lack of bounds checks
  – To get computers to export secrets
    – Like cryptography keys

xkcd Heartbleed Explanation

https://xkcd.com/1354/

Buffer Attack

int ud_connect(const char *name) {
  int fd;
  struct sockaddr_un {
    sa_family_t sun_family;
    char sun_path[108];
  } addr;
  ...
  sprintf(addr.sun_path, "%s", name);
  ...
  return fd;
}

What happens if name>108 characters?

[Averinos, CACM 2014] > 136?

https://xkcd.com/1354/
Buffer Overflow

- Unchecked buffers allow insertion/overwrite of data
- Unchecked buffers on stack allow overwrite of data controlling what code you execute
  - …and maybe even code to execute

Defenses

- Modern virtual memory systems will set code pages to be unwritable
- Can still control which code gets run
- Embedded systems without VM don’t have this option
- Modern compilers (like gcc) can be instructed to add sanity check code
  - Stack Guard: Canary data to catch overwrites
  - Attacker not change canary…

Memory Vulnerability

- Raw, undifferentiated memory
- Holding code, data, control structures
- Accessible from every memory operation
- Relying only on absence of out-of-bounds reference bugs to maintain integrity
  - …enables a host of security vulnerabilities

SoC

- Small, differentiated memories in SoC may help
  - Not live in single, unified address space
  - Tasks have access to limited, local memory (not everything)
- Hardware functions cannot be overwritten

DMA Master

- What can a DMA Master do to memory?
DMA

- May give USB/firewire/PCI DMA access to memory
- Without care can read/write anywhere in memory
- Allow malicious peripheral to
  - Steal data
  - Compromise integrity

ThunderStrike

Available for: OS X Yosemite v10.10 and v10.10.1, for: MacBook Pro Retina, MacBook Air (Mid 2013 and later), iMac (Late 2013 and later), Mac Pro (Late 2013)

Impact: A malicious Thunderbolt device may be able to affect firmware flashing

Description: Thunderbolt devices could modify the host firmware if connected during an EFI update. This issue was addressed by not loading option ROMs during updates.

CVE-2014-4498: Trammell Hudson of Two Sigma Investments

https://trmm.net/Thunderstrike_FAQ

Input

- Provides an opportunity to poke at system
  - Exploit vulnerabilities
  - Directly control system?
- Can mislead system
  - Lie to it

Input and Memory

- Memory section illustrates unchecked inputs can exploit vulnerabilities

Input Integrity (xkcd)

https://xkcd.com/327/
Input Data Integrity

- Bobby Tables illustrates
  - Must take care with any inputs that may be used in control
    - Interpret commands, where branch, specify what operate upon...
- As does
  - data[user_input()]
  - Heartbleed
  - Buffer bounds checks

Input Validity

- What happens if the CD-player sends a message to the brakes to stop?

![Car diagram]

Cars

- Modern cars contain many embedded controllers

Result

- If you can compromise any unit on the bus (like the MP3 player), can send control messages to any unit

Impact

- Were able to control the car
- More recent demo 2015 on 60 Minutes

[Checkoway, Usenix Security 2011]

[Figure 3. CAN packet structure. Extended frame format is shown. Raw frame format is similar.

Koscher, IEEE Security\&Privacy 2010]
Input Integrity

• What could we do to protect against this?

Input Lessons

• Need to carefully consider where our inputs come from
  – and how we know that
• If comes from untrustworthy source
  – Need to validate before use

Output

• Who should be able to see the outputs produced?
• What outputs is the system producing you may not have intended?
  – Radio-frequency, power, timing, audio…

Open Datastreams

• …and this is true of most internet connected cameras
  – Including baby monitors

AUTHOR: NOAH SHACHTMAN AND DAVID AXE.
NOAH SHACHTMAN AND DAVID AXE
SECURITY DATE OF PUBLICATION: 10.29.12.
10.29.12
TIME OF PUBLICATION: 4:00 AM.
MOST U.S. DRONES OPENLY BROADCAST SECRET VIDEO FEEDS

FOUR YEARS AFTER discovering that militants were tapping into drone video feeds, the U.S. military still hasn’t secured the transmissions of more than half of its fleet of Predator and Reaper drones, Danger Room has learned. The majority of the aircraft still broadcast their classified video streams “in the clear”— without encryption. With a minimal amount of equipment and know-how, militants can see what America’s drones see.

https://www.wired.com/2012/10/hack-proof-drone/
Side Channels

• Data-dependent behavior may leak information
  – Timing
  – Power
  – Radio Frequency emissions
• Ample demonstrations can harvest crypto keys from
  – Differential power analysis
  – RF emissions

Timing

for (i=0; i<LEN;i++)
if (passwd[i]!=input[i])
  break;

• How is timing of check related to data?
• If attacker can control address alignment of input, how can enhance timing difference?
  – In demand-fetched memory architecture

Data Independent

• Data independent computation
  – As we tend to need to do anyway for real-time computations
• Can reduce side channel vulnerabilities
• E.g.
  – fetch entire input local before compare
  – not report failure until entire input scanned
• ..but not sufficient to address RF, power

Output Lessons

• Think about how consumers will authenticate output data
  – Carry-over from Input integrity
• Think carefully about the privacy of output data, and how assure
• Watch unintended outputs

Cryptography

• Likely need
  – Encryption/decryption
    • Privacy
    • Authentication
  – Source of randomness
    • Often a physical random number generator
  – Some form of identity
    • Preferably unforgeable
Easy to screw it up

- Roll own insecure crypto algorithms
- Use weak algorithms or keys
- Use insufficiently random data
- Leak key-related data
  - (see memory, output)

Natural Streaming

Natural for spatial hardware

[Drimer, FCCM 2008]

Big Ideas

- SoC Designers need to be concerned about security
  - Confidentiality, integrity, availability
  - Important for many applications
- Hazards are real, understandable
  - Avoidable, tricky...

Admin

- Collect Zed Boards
  - Class Monday 4/24
- Final: Monday, May 1 9am—11am
- One more class on Wednesday