Introduction to Embedded Security

Black Hat USA 2004 Briefings
Wednesday, July 28, 1:45pm - 3:00pm

Joe Grand
Grand Idea Studio, Inc.
joe@grandideastudio.com
Agenda

- Goals
- Security in the Product Lifecycle
- Attack and Threat Classifications
- Practical Design Solutions
Goals

- Learn the concepts of designing secure hardware
- Become familiar with types of attacks and attackers
- Understand and accept that properly implemented security is extremely difficult
- Education by demonstration
Risk Assessment

• Nothing is ever 100% secure
  - Given enough time, resources, and motivation, an attacker can break any system

• Secure your product against a specific threat
  - What needs to be protected
  - Why it is being protected
  - Who you are protecting against (define the enemy)
Risk Assessment 2

And this house is even more secure! The front door is four feet thick and made of solid titanium...
Security in the Product Development Lifecycle

• Establish a sound security policy as the "foundation" for design
• Treat security as an integral part of system design
• Reduce risk to an acceptable level
  - Elimination of all risk is not cost-effective
• Minimize the system elements to be trusted
  - "Put all your eggs in one basket"
Security in the Product Development Lifecycle 2

- Strive for simplicity
  - The more complex the security, the more likely it is to contain exploitable flaws
- Implement layered security
- Do not implement unnecessary security mechanisms
  - Each mechanism should support a defined goal
Attack Types

- **Insider Attack**
  - Significant percentage of breaches
  - Run-on fraud, disgruntled employees

- **Lunchtime Attack**
  - Take place during a small window of opportunity

- **Focused Attack**
  - Time, money, and resources not an issue
Attacker Classification

- **Class I: Clever Outsiders**
  - Intelligent, but have limited knowledge of the system
  - Often try to take advantage of an existing weakness

- **Class II: Knowledgeable Insiders**
  - Substantial specialized technical experience
  - Highly sophisticated tools and instruments

- **Class III: Funded Organizations**
  - Specialists backed by great funding resources
  - In-depth analysis, sophisticated attacks, most advanced analysis tools
## Attacker Classification 2

<table>
<thead>
<tr>
<th>Resource</th>
<th>Curious Hacker (Class I)</th>
<th>Academic (Class II)</th>
<th>Organized Crime (Class III)</th>
<th>Government (Class III)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Limited</td>
<td>Moderate</td>
<td>Large</td>
<td>Large</td>
</tr>
<tr>
<td>Budget ($)</td>
<td>&lt; $1000</td>
<td>$10k - $100k</td>
<td>&gt; $100k</td>
<td>Unknown</td>
</tr>
<tr>
<td>Creativity</td>
<td>Varies</td>
<td>High</td>
<td>Varies</td>
<td>Varies</td>
</tr>
<tr>
<td>Detectability</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Target/Goal</td>
<td>Challenge</td>
<td>Publicity</td>
<td>Money</td>
<td>Varies</td>
</tr>
<tr>
<td>Number</td>
<td>Many</td>
<td>Moderate</td>
<td>Few</td>
<td>Unknown</td>
</tr>
<tr>
<td>Organized?</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Release info?</td>
<td>Yes</td>
<td>Yes</td>
<td>Varies</td>
<td>No</td>
</tr>
</tbody>
</table>
## Attack Difficulty

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None</td>
<td>No tools or skills needed. Can happen by accident.</td>
</tr>
<tr>
<td>2</td>
<td>Intent</td>
<td>Minimal skills. Universally available tools.</td>
</tr>
<tr>
<td>3</td>
<td>Common Tools</td>
<td>Technically competent. Tools available at retail computer/electronic stores.</td>
</tr>
<tr>
<td>4</td>
<td>Unusual Tools</td>
<td>Engineers using dedicated tools available to most people.</td>
</tr>
<tr>
<td>5</td>
<td>Special Tools</td>
<td>Highly specialized tools and expertise as found in academia or government.</td>
</tr>
<tr>
<td>6</td>
<td>In Laboratory</td>
<td>Major time and effort required. Resources available to few facilities in the world.</td>
</tr>
</tbody>
</table>
Product Accessibility

• **Purchase**
  - Attacker owns or buys the product

• **Evaluation**
  - Attacker rents or borrows the product

• **Active**
  - Product is in active operation, not owned by attacker

• **Remote Access**
  - No physical access to product, attacks launched remotely
Threat Vectors

- **Interception (or Eavesdropping)**
  - Gain access to protected information without opening the product

- **Interruption (or Fault Generation)**
  - Preventing the product from functioning normally

- **Modification**
  - Tampering with the product, typically invasive

- **Fabrication**
  - Creating counterfeit assets of a product
Attack Goals

- **Competition (or Cloning)**
  - Specific IP theft to gain marketplace advantage

- **Theft-of-Service**
  - Obtaining service for free that normally requires $$$

- **User Authentication (or Spoofing)**
  - Forging a user's identity to gain access to a system

- **Privilege Escalation (or Feature Unlocking)**
  - Gaining increased command of a system or unlocking hidden/undocumented features
Practical Design Solutions

- Enclosure
- Circuit Board
- Firmware
Product Enclosure

- Should prevent easy access to product internals
Product Enclosure 2

- External Interfaces
- Tamper Mechanisms
- Emissions and Immunity
External Interfaces

• Usually a product's lifeline to the outside world
  - Manufacturing tests, field programming, peripheral connections
  - Ex.: Firewire, USB, RS232, Ethernet, JTAG
External Interfaces 2

- Do not simply obfuscate interface
  - Will easily be discovered and exploited by an attacker
  - Ex.: Proprietary connector types, hidden access doors or holes

- Remove JTAG and diagnostic functionality in operational modes
  - Blown fuses or cut traces can be repaired by an attacker

- Protect against malformed, bad packets
  - Intentionally sent by attacker to cause fault
External Interfaces 3

- Only publicly known information should be passed
- Encrypt secret or critical components
  - If they must be sent at all...
  - Ex.: Palm OS system password decoding [1]
- Wireless interfaces also at risk
  - Ex.: 802.11b, Bluetooth
Tamper Mechanisms

- Primary facet of physical security for embedded systems
- Attempts to prevent unauthorized physical or electronic action against the product
  - Resistance
  - Evidence
  - Detection
  - Response
Tamper Mechanisms 2

- Most effectively used in layers
- Possibly bypassed with knowledge of method
- Costs of a successful attack should outweigh potential rewards
  - Ex.: Probing, machining, electrical attacks, physical barriers, tamper evident solutions, sensors, response technologies
Tamper Resistance

- Specialized materials to make tampering more difficult
  - Ex.: Hardened steel enclosures, locks, tight airflow channels
- Often tamper evident
  - Physical changes can be visually observed
Tamper Resistance 2

• Security bits/one-way screws
  - Can still be bypassed, but raises difficulty over standard screw or Torx

• Encapsulation
  - Cover circuit board or critical components with epoxy or urethane coating
  - Prevents moisture, dust, corrosion, probing
  - Difficult, but not impossible, to remove with solvents or Dremel tool (and wooden skewer as a "bit")
Tamper Resistance 3

- Sealed/molded housing
  - Ultrasonic welding or high-temperature glue
  - If done properly, will require destruction of device to open it
  - Consider service issues (if a legitimate user can open device, so can attacker)
Tamper Evidence

- Ensure that there is visible evidence left behind by tampering
- Major deterrent for minimal risk takers
- Only successful if a process is in place to check for deformity
  - If attacker purchases product, tamper evident mechanisms will not stop attack
Tamper Evidence 2

- Special enclosure finishes
  - Brittle packages, crazed aluminum, bleeding paint
- Passive detectors
  - Most common: seals, tapes, glues
- *Vulnerability of Security Seals* [3] explains that most can be bypassed with ordinary tools
  - All 94 seals tested were defeated
  - Ex.: Adhesive tape, plastic, wire loop, metal cable, metal ribbon, passive fiber optic
Tamper Detection

- Enable the hardware device to be aware of tampering
- Switches
  - Detect the opening of a device, breach of security boundary, or movement of a component
  - Ex.: Microswitches, magnetic switches, mercury switches, pressure contacts
Tamper Detection 2

• Sensors
  - Detect an environmental change, glitch attacks against signal lines, or probing via X-ray/ion beam
  - Ex.: Temperature, radiation, voltage, power supply
Tamper Detection 3

- Circuitry
  - Special material wrapped around critical circuitry to create a security perimeter
  - Detect a puncture, break, or attempted modification of the wrapper
  - Ex.: Flexible circuitry, nichrome wire, fiber optics, W.L. Gore's D3 electronic security enclosure
Tamper Response

- Countermeasures taken upon the detection of tampering
  - Works hand-in-hand with tamper detection mechanisms
- Erase critical portions of memory ("zeroize") or remove power
  - Contents not necessarily completely erased
  - Volatile memory (SRAM and DRAM) retains some data when power is removed [4]
Tamper Response 2

- Shut down or disable device
  - Extreme solution: Physical destruction using small, shaped explosive charge

- Logging mechanisms
  - Provide audit information for help with forensic analysis after an attack

- Accidental triggers are unlikely
  - User may still need to understand environmental and operational conditions
Emissions and Immunity

- All devices generate EMI (emissions)
- Can be monitored and used by attacker to determine secret information
  - Ex.: Data on a computer monitor [5], cryptographic key from a smartcard [6]
- Devices may also be susceptible to RF or ESD (immunity)
  - Intentionally injected to cause failure
Emissions and Immunity 2

• Aside from security, EMI emissions/immunity conditions part of many specifications
  - Ex.: FCC, FDA, UL, CE, IEC

• Install EMI shielding
  - Decrease emissions and increase immunity
  - Ex.: Coatings, tapes, sprays, housings
  - Be aware of changes in thermal characteristics that shielding may introduce (heating)
Circuit Board

- Physical Access to Components
- PCB Design and Routing
- Memory Devices
- Power Supply
- Clock and Timing
- I/O Port Properties
- Cryptographic Processors and Algorithms
Physical Access to Components

- Giving an attacker easy access to components aids in reverse engineering of the product
- Make sensitive components difficult to access
  - Ex.: Microprocessor, ROM, RAM, or programmable logic
- Remove identifiers and markings from ICs
  - Known as "De-marking" or "Black topping"
  - Use stainless steel brush, small sander, micro-bead blast, laser etcher, or third party
  - IC Master, Data Sheet Locator, and PartMiner allows anyone to easily find data sheets of components
Physical Access to Components 2

- Use advanced packaging types
  - Difficult to probe using standard tools
  - Ex.: BGA, Chip-on-Board (COB), Chip-in-Board (CIB)

- Epoxy encapsulation on critical areas
  - Prevent probing and easy removal
  - Ensure desired security goal is achieved
PCB Design and Routing

- Remove unnecessary test points
  - Use filled pad as opposed to through-hole, if necessary
- Obfuscate trace paths to prevent easy reverse engineering
  - Hide critical traces on inner board layers
- Use buried vias whenever possible
  - Connects between two or more inner layers but no outer layer
  - Cannot be seen from either side of the board
PCB Design and Routing 2

- Keep traces as short as possible
- Properly designed power and ground planes
  - Reduces EMI and noise issues
- Keep noisy power supply lines from sensitive digital and analog lines
- Differential lines aligned parallel
  - Even if located on separate layers
Bus Protection

- Address, data, and control bus lines can easily be probed
  - Ex.: Tap board used to intercept data transfer over Xbox's HyperTransport bus [7]
  - Be aware of data being transferred across exposed and/or accessible buses
Memory Devices

• Most memory is notoriously insecure
  - Serial EEPROMs can be read in-circuit [8]
  - RAM devices retain contents after power is removed, can also "burn in" [4]

• Security fuses and boot-block protection
  - Implement if available
  - Can be bypassed with die analysis attacks [9] using Focused Ion Beam
  - Ex.: PIC16C84 attack in which security bit is removed by increasing VCC during repeated write accesses
Programmable Logic

• In many cases, IP within PLD or FPGA is most valuable in the product
• SRAM-based FPGAs most vulnerable to attack
  - Must load configuration from external memory
  - Bit stream can be monitored to retrieve entire configuration
  - New devices: Actel Antifuse and QuickLogic FPGAs
Programmable Logic 2

• Protect against I/O scan attacks
  - Used by attacker to cycle through all possible combinations of inputs to determine outputs
  - Use unused pins on device to detect probing
    ◦ Set to input. If level change is detected, perform a countermeasure or response.

• Add digital "watermarks"
  - Features or attributes in design that can be uniquely identified as being rightfully yours

• If using state machine, ensure all conditions and defaults are covered
Power Supply

- Define minimum and maximum operating limits
  - Ex.: Comparators, watchdogs, supervisory circuits
- Do not rely on end user to supply a voltage within recommended operating conditions
  - Implement linear regulator or DC-DC converter
- Compartmentalize noisy circuitry
  - Easier to reduce overall EMI
  - Use proper filtering
  - Power supply circuitry as physically close as possible to power input
Power Supply 2

- Simple Power Analysis (SPA)
  - Attacker directly observes power consumption
  - Varies based on microprocessor operation
  - Easy to identify intensive functions (cryptographic)

- Differential Power Analysis (DPA)
  - Advanced mathematical methods to determine secret information on a device

- *Power Analysis Attack Countermeasures and Their Weaknesses* [10] proposes solutions
  - Ex.: Noise generator, active/passive filtering, detachable power supplies, time randomization
Clock and Timing

- Attacks rely on changing or measuring timing characteristics of the system
- Active timing attacks
  - Invasive attack: vary clock to induce failure or unintended operation
  - Monitor clock signals to detect variations
  - Implement PLL to reduce clock delay and skew
- Passive timing attacks
  - Non-invasive measurements of computation time
  - Different tasks take different amounts of time
I/O Port Properties

- Unused I/O pins should be disabled or set to fixed state
  - Use to detect probing of PLD or FPGA
  - Could introduce unwanted noise
- Prevent against ESD on exposed lines
  - Clamping diodes or Transient Voltage Suppressor
  - Ex.: Keypads, buttons, switches, display
Cryptographic Processors and Algorithms

- Strength of cryptography relies on secrecy of key, not the algorithm
- It is not safe to assume that large key size will guarantee security
- If algorithm implemented improperly, can be broken or bypassed by attacker
  - Without a secure foundation, even the best cryptosystem can fail
  - Test implementations in laboratory first!
Cryptographic Processors and Algorithms 2

- Do NOT roll-your-own crypto
  - Possibly the most common problem in engineering
  - Easily broken, no matter what you may think
  - Usually just "security through obscurity"
  - Ex.: Palm OS system password decoding [1], USB authentication tokens [8], iButton Dictionary Attack vulnerability [11]
Cryptographic Processors and Algorithms 3

- If possible, move cryptographic processes out of firmware and into FPGA
  - Harder to probe than ROM devices
  - Increased performance (more efficient)

- Or, use secure cryptographic coprocessor
  - Self-contained, hardware tamper response, layered design, self-initialization, authentication, general-purpose processor, randomness, API
  - Ex.: IBM 4758, PCI-X, Philips VMS747
Firmware

- Programming Practices
- Storing Secret Components
- Run-Time Diagnostics and Failure Modes
- Field Programmability
- Obfuscation (Security Through Obscurity)
Programming Practices

- Poor programming, flaws, and bugs can lead to security compromises
  - Ex.: Buffer overflows

- Remove unnecessary functionality and debug routines
  - Ex.: Palm Backdoor Debug mode [13]
Programming Practices 2

- Remove debug symbols and tables
  - As easy as a checkbox or command-line switch

- Use compiler optimizations
  - Possibly obfuscate easily identifiable code segments
  - Increase code efficiency
Storing Secret Components

• Difficult to securely and totally erase data from RAM and non-volatile memory [4]
  - Remnants may exist and be retrievable from devices long after power is removed or memory areas rewritten

• Limit the amount of time that critical data is stored in the same region of memory
  - Can lead to "burn in"
  - Periodically flip the stored bits
Run-Time Diagnostics and Failure Modes

• Make sure device is fully operational at all times
  - Periodic system checks
  - Ex.: Internal watchdog, checksums of memory
  - Failing device may open product to compromise

• Determine how the product handles failures
  - Set failure flags and continue
  - Halt or shutdown system
  - Zeroization of critical memory areas
Field Programmability

- Is your firmware accessible to everyone from your Web site?
  - Attacker can easily disassemble and analyze

- Code signing (DSA) or hashes (SHA-1, MD5)
  - Reduce possibility of loading unauthorized code
  - Will verify that firmware image has not been tampered with

- Encrypt firmware images
  - Compression routines are not encryption
  - Challenge is in protecting the private key
Obfuscation

- "Security through obscurity" does NOT work
  - May provide a false sense of security
  - Will temporarily discourage Class I attackers
- Encode fixed data
- Scramble address lines through extra logic
- Replace library functions with custom routines
- Write lousy code
- Add spurious and meaningless data ("signal decoys")
Conclusions

- Determine what to protect, why you are protecting it, and who you are protecting against
  - No one solution fits all
- Best defense is to make the cost of breaking the system greater than the value of your information
- Do not release product with a plan to implement security later
  - It usually never happens...
Conclusions 2

- Think as an attacker would
- Be aware of latest attack methodologies & trends
- As design is in progress, allocate time to analyze and break product
- Learn from mistakes
  - Study history and previous attacks
- Nothing is ever 100% secure
References


References 2


References 3


Thanks!

Grand Idea Studio, Inc.
http://www.grandideastudio.com
joe@grandideastudio.com