Freescale Technology Forum


July, 2009

System Security
Protecting Systems from Hacking and Cloning

Rudan Bettelheim
Building Control Segment Marketing Manager
Agenda

- Introduction – Why Security?
- What Requires Protection
- Cryptography – Protecting Data
- Secure Systems – Preventing Hacking and Cloning
- Q & A
Introduction – Why Security?
Industrial Market Evolution

Last Decade

- Increasingly sophisticated electronic control systems
- Extensive spread of networking of industrial control systems
- Remote and mobile equipment becomes part of the control network
- Increasing concerns over equipment cloning

Future expectations

- The global industrial market for MPU/MCU/DSP is projected to grow from $2.8B in 2005 to $4.6B in 2011 (source: Semicast)
- Increasing use of Cryptography to protect communications
- Growing need and implementation of Secure Embedded Control Systems, facilitated by an ecosystem of hardware, software, and tools
- Emerging security standards such as EMV/Visa PCI
- A high profile event could lead to a sudden and strong market and regulatory requirement for system protection across many industrial applications
  - Suppliers that are ready to meet system security requirements are likely to gain significant market advantage and share
Example of Infrastructure Vulnerability

By SIOBHAN GORMAN

http://online.wsj.com/article_email/SB123914805204099085-lMyQijAxMDI5MzA5NzEwNDE4Wj.html

WASHINGTON -- Cyberspies have penetrated the U.S. electrical grid and left behind software programs that could be used to disrupt the system, according to current and former national-security officials.

The spies came from China, Russia and other countries, these officials said, and were believed to be on a mission to navigate the U.S. electrical system and its controls. The intruders haven't sought to damage the power grid or other key infrastructure, but officials warned they could try during a crisis or war.

"The Chinese have attempted to map our infrastructure, such as the electrical grid," said a senior intelligence official. "So have the Russians."

The espionage appeared pervasive across the U.S. and doesn't target a particular company or region, said a former Department of Homeland Security official. "There are intrusions, and they are growing," the former official said, referring to electrical systems. "There were a lot last year."
Example of Embedded System Vulnerability - Phlashing

Hardware: New 'Phlashing' Attack Sabotages Hardware

Posted by timothy on Tuesday May 20, 09:29AM
from the not-so-nice dept.

yaho writes

"A new type of denial-of-service attack, called permanent denial-of-service (PDOS), damages a system so badly that it requires replacement or reinstallation of hardware. A researcher has discovered how to abuse firmware update mechanisms with what he calls 'phlashing' — a type of remote PDOS attack."

hardware, it, security, brick (tagging beta)

Read More... hardware slashdot.org
What Requires Protection?
How Much Security?

When protecting a system you must consider:

- What are you trying to protect?
- What types of attack do you need to protect against?
- What are the likely attack points, and methods?
- How much security do you require?
  - How much are you willing to pay?
- How will security impact the underlying system?
- How will you upgrade/maintain the system and security over time?
Types of Attacks

Electrical
- Over/Under voltage
- Power analysis
- Frequency analysis
- Electrostatic discharge
- Circuit probing

Software
- Spy software insertion
- Flow analysis
- Trojan horse
- Virus

Physical
- Temperature variation (into extremes)
- Temperature analysis
- De-processing
- System theft
- Partial destruction
- Hardware addition/substitution
System Security

► Classic Security Requirements:
► Confidentiality - prevents eavesdropping
► Authentication - prevents impersonation
► Data Integrity - prevents tampering
► Non-repudiation - prevents denial
► Trusted Processing - enables trusted platform for authorized access to program and data
► IP Protection - prevent software/IP theft
Industrial System Security Requirements

► Industrial systems may have a wide range of security requirements:

► Secure communications key storage
  • To secure communications in a control system
  • For remote equipment authentication

► Program code authentication
  • To prevent unauthorized code from being executed
  • To prevent use of unlicensed software

► Program code protection
  • To prevent code from being copied and used on clone equipment
  • To prevent code tapering

► Data protection
  • To protect system or user data

► Reduce cost of physical system protection
Typical 32-bit System

Following Reset:
1. Transfer (and decompress) program code from Boot Flash to DRAM
2. Initialize system and peripherals
3. Transfer control to application program
<table>
<thead>
<tr>
<th>Application Security Levels</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Secure System, Server/Client Authentication</strong></td>
<td>Limits access to core system resources to OEM supplied and authorized software and data, and this is periodically authenticated with a secure server. Restricted execution of additional software without authorization is allowed.</td>
</tr>
<tr>
<td><strong>Secure Local System, External Software supported</strong></td>
<td>Limits access to core system resources to OEM supplied and authorized software and data, but restricted execution of additional software without authorization is allowed.</td>
</tr>
<tr>
<td><strong>Secure Local System, OEM Software only</strong></td>
<td>Ensures that only OEM supplied and authorized software and data can be used on the system, no other software can be executed.</td>
</tr>
<tr>
<td><strong>Software (IP) System Protection only</strong></td>
<td>Protection for system software and data IP, prevents software and data from being copied only</td>
</tr>
<tr>
<td><strong>No System Protection, fully open</strong></td>
<td>No system protection</td>
</tr>
</tbody>
</table>
Application Security Requirements Trends

- Secure System, Server/Client Authentication
- Secure Local System, External Software supported
- Secure Local System, OEM Software only
- Software (IP) System Protection only
- No System Protection, fully open
How are Systems Protected Today?

Physical security:
- Secure packaging
- Secure packaging with tamper detect (i.e. pressure monitoring)
- Secure packaging with tamper detect and destruction (i.e. dynamite)
- Obscured part numbers
- Hidden layers
- Protected location

Electronic Security:
- Security bit, to protect on-chip non-volatile memory (e.g. Flash), on MCUs
  - Prevent external access to on-chip resources:
    - Locks device into Single Chip mode (disables external parallel bus)
    - Disables Background Debug Mode
    - Disables Test Mode
    - Disables JTAG
    - Disables any (serial) “Bootstrap” functions
  - Memory array bulk erase turns security bit off
- Secure System (e.g. Freescale PISA)
  - Code signing to prevent software tampering
  - Assurance for stored IP
  - Data stored encrypted in external memory
  - Data decrypted and stored in on-chip private memory at runtime
    - How do you protect software IP?
- Proprietary (CPU) Design
- Silicon Obfuscation (e.g. obscuring metal layer)
- On-Chip Encryption Acceleration
  - How do you protect the key?
Cryptography – Protecting Data
Cryptography

Symmetric Key Cryptography:
• Same key used to encrypt and decrypt
• Very fast
  ▪ Typically used for bulk of encryption/decryption
• Same key must be at both end points

Asymmetric (Public) Key Cryptography:
• 2 related keys are required (known as a public and a private key)
• 1000 times slower than symmetric key
• Typically used for exchange of symmetric keys and sender authentication
• End points need have had no prior contact

Authentication:
• Necessary to know who you’re speaking to
• Certificates used to verify identity
Asymmetrical (Public) Key Cryptography (RSA)

Public key cryptography is based on a pair of keys:

► Public key for encryption (open padlock, anyone can lock)
  - Consists of the modulus \( n \), which is the product of two large prime numbers \( p \) and \( q \), which are kept secret, and the public exponent \( e \), typically \( 2^{16} + 1 = 65537 \)

► Private key for decryption (only the key can unlock the padlock)
  - Consists of the modulus \( n \), and the private exponent \( d \) which is based on the two large prime numbers \( p \) and \( q \)

For more information refer to:
http://en.wikipedia.org/wiki/RSA
The Code Book, by Simon Singh (Anchor)

RSA - Rivest, Shamir, Adleman
Cryptographic Acceleration Units

Freescale has a range of crypto modules, from slave units to descriptor driven bus mastering units

Typical Functionality:

- **Data Encryption Standard Execution Unit (DEU)**
  - DES, 3DES
  - Two key (K1, K2, K1) or three key (K1, K2, K3)
  - ECB and CBC modes

- **Advanced Encryption Standard Unit (AESU)**
  - Key lengths of 128, 192, and 256 bits
  - ECB, CBC, CTR, CCM modes

- **Message Digest Execution Unit (MDEU)**
  - SHA-1 160-bit digest
  - SHA-2 256-bit digest
  - HMAC with all algorithms
  - MD5 128-bit digest

- **ARC Four Execution Unit (AFEU)**
  - Compatible with RC4 algorithm

- **Hardware Random Number Generator (RNG)**
  - FIPS compliant (with appropriate software)
## Symmetrical (h)MACs and Random Numbers

<table>
<thead>
<tr>
<th>Cipher/Algorithm</th>
<th>Type</th>
<th>Block Size</th>
<th>Key Size</th>
<th>Common Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>Symmetric Block Cipher</td>
<td>64 bit</td>
<td>56 bit</td>
<td>CBC</td>
</tr>
<tr>
<td>3DES</td>
<td>Symmetric Block Cipher</td>
<td>64 bit</td>
<td>168 bit</td>
<td>CBC</td>
</tr>
<tr>
<td>AES</td>
<td>Symmetric Block Cipher</td>
<td>128 bit</td>
<td>128 bit, 192 bit, 256 bit</td>
<td>CBC</td>
</tr>
<tr>
<td>ARC-4</td>
<td>Symmetric Block Cipher</td>
<td>8 bit</td>
<td>40 - 128 bit</td>
<td>–</td>
</tr>
<tr>
<td>RSA</td>
<td>Asymmetric Stream Cipher</td>
<td>NA</td>
<td>Up to 2048 and 4096</td>
<td>–</td>
</tr>
<tr>
<td>MD-5</td>
<td>Hashing Cipher</td>
<td>512 bit</td>
<td>Up to 512 bit</td>
<td>HMAC</td>
</tr>
<tr>
<td>SHA-1/SHA-2</td>
<td>Hashing Cipher</td>
<td>512 bit</td>
<td>Up to 512 bit</td>
<td>HMAC</td>
</tr>
</tbody>
</table>
Security Options – IPsec/IKE

► IP Security
► IPsec uses encryption technology to provide data confidentiality, integrity and authenticity between participating peers in a private network.

► IPsec provides two choices of security services: Authentication Header (AH), which essentially allows authentication of the sender of data and Encapsulating Security Payload (ESP), which supports both authentication of the sender and encryption of data as well.

► The specific information associated with each of these services is inserted into the packet in a header that follows the IP packet header.

► Unlike Secure Sockets Layer, which provides services at layer 4 and secures two applications, IPsec works at layer 3 and secures everything in the network.
Security Options SSL/TLS

- Transport Layer Security
- SSLv3.1 is TLSv1.0.
- TLS is standardized by IETF and is a protocol intended to secure and authenticate communications across public networks by using data encryption.
- TLS is designed as a successor to SSL and uses the same cryptographic methods but supports more cryptographic algorithms. TLS is backwards compatible with SSL.
- It is designed to prevent eavesdropping, message forgery and interference.
Secure Shell

Secure Shell is a program to log into another computer over a network, execute commands on a remote machine and move files from one machine to another.

It provides strong authentication and secure encrypted communications between two hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can be forwarded over the secure channel.

It is a replacement for rlogin, rsh, rcp, rdist and telnet. SSH protects a network from attacks such as IP spoofing, IP source routing, and DNS spoofing.

By using ssh's slogin (instead of rlogin), the user initiates an entire login session, including transmission of password, that is encrypted; therefore it is almost impossible for an outsider to collect passwords.
Security Options – RADIUS Client

► RADIUS
► RADIUS comprises two pieces: authentication server code and client protocols.
► RADIUS Client implements a client/server mechanism that enables remote access servers to communicate with a central server to authenticate users and authorize their access to the requested system or service.
► RADIUS allows a company to maintain user profiles in a central database that all remote servers can share. It also allows a company to set up a policy that can be applied at a single administered network point.
► Having a central service also means that it's easier to track usage for billing and for keeping network statistics.
► Being able to centralize authentication and administration is especially attractive to embedded devices that need to verify user credentials and authorize users, without having the overhead of maintaining and administering a database of sensitive user information.
Security Options – EAP Authentication

► Extensible Authentication Protocol:
► Framework with hooks to support any authentication method
► Similar to RADIUS architecture
► Overcomes RADIUS shortcomings
  • Not limited to PAP/CHAP
  • Encryption between client and server
  • Future proof
► At the heart of all the wireless security protocols
► Spreading like wild fire
What Protocol Do You Need?

► Management information or data?
  • SSL or SSH
► Who initiates the communication?
► Which end of the channel needs to authenticate the other?
► What kind of threat are you protecting against?
  • Tampering or spying
► Where can the threats originate?
  • Internal network and or Internet
► How ‘open’ is your system?
  • Must it plan for connecting with new devices?
► What is the technical proficiency of the users?
► What protocols require security?
  • TCP and or UDP
### Options for Device and Communications Security

<table>
<thead>
<tr>
<th></th>
<th>IPsec</th>
<th>SSL/TLS</th>
<th>SSH</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type of Security</strong></td>
<td>Network</td>
<td>Transport/Session</td>
<td>Application</td>
</tr>
<tr>
<td><strong>Typical Usage</strong></td>
<td>Data Path</td>
<td>Management</td>
<td>Management/Control</td>
</tr>
<tr>
<td><strong>UDP Security</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Supports User Authentication</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Compatible with NAT &amp; Firewalls</strong></td>
<td>Limited</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Ease of provisioning</strong></td>
<td>Moderate</td>
<td>Extremely Easy</td>
<td>Extremely Easy</td>
</tr>
</tbody>
</table>

**IPsec/IKE** - IP Security, provides data confidentiality and node authentication, works at layer 3 and secures everything in the network.

**SSL/TLS** - Secure Socket Layer/Transport Layer Security, provides communications confidentiality and node authentication across public networks, works at layer 4 and secures applications.

**SSH** - Secure Shell, supports remote log into and control of a system with secure communications.
Secure Systems – Preventing Hacking and Cloning
Security bit is available today on most MCU products, provides good Cloning, and some Hacking Protection. Future developments will improve Hacking protection.
Secret key in combination with hardware crypto and secure RAM enables secure data storage in external memory.
Secure boot and integrity checker ensure that only authorized software will run on system.

Memory protection unit enforces restricted access to secure data.

Tamper detection destroys data and keys when system is threatened.
Secure Boot & Integrity Monitor

- System Reset
- Vector to Secure Boot ROM
- Determine Boot Mode/Memory
- System Integrity Check
- Authentication Hash
- Halt Execution
- User Code

System Secure?

Yes
- Periodic System Integrity Check

No
- Halt Execution

• Ensures that the system configuration is as expected
• Ensures that the Application Program and Data have not been Tampered with
• Only allows a fully verified system to execute application program
• Implemented in Hardware, operates continuously in the background
  • Continues to verify system integrity
  • Prevents a Trojan system from being inserted
Program Authentication

OEM System Provisioning

Application Program

- Message Digest Hash
- Private Key Encryption
- Private Key
- Public Key

Signature

Public Key

Secure Boot Authentication

Application Program

- Message Digest Hash
- Compare Hash Sum
- Public Key Decryption
- Authentication Result

Signature

Public Key

Verify Key

Fuse Box

Public Key Hash

Note: Program and Signature may also be encrypted for IP protection. Private Key has to be carefully managed and protected.
Following Reset:
1. Check initial system integrity
2. Authenticate program code
3. Transfer (decompress, and decrypt) program code from Boot Flash to DRAM
4. Initialize system and peripherals
5. Transfer control to application program
6. Protect sensitive data with secure RAM
7. Control memory accesses with MMU/MPU
8. Encrypt communications with CAU
9. Continuously hash memory with integrity checker
High Assurance Boot (HAB)

► A secure system’s foundation consists of the hardware platform and the critical code that executes on that platform. This foundation is built with an on-chip tamper resisted ROM based process that initiates validation of the platform.

► The High Assurance Boot process gains control of the system immediately after reset by executing a known boot code resident in on-chip ROM. The HAB process includes:

- **Health Check** - Validating the secure HW
- **Authenticity check** - Validating that the code image, stored in external memory, originated from a trusted authority
- **Integrity check** - Verifying that the code is in its original form
- **Versioning Control** - Checks the external code version (code revocation system)

► The boot process uses digital signatures to perform the validations.

► The boot sequence is flexible because it is controlled by authenticated scripts that reside in off-chip memory.
Run-Time Checks (RTIC)

- Protecting read only data from modification is one of the basic elements in trusted platforms

- Write protection can be achieved by using on-chip one time programmable (OTP) elements such as electrical fuses. Though OTP elements are write protected, their data capacity is limited and they are not flexible (once the element is programmed it cannot be modified)

- The Run-Time Integrity Checker (RTIC) mechanism periodically checks the integrity of code or data sections during normal OS run-time execution without interfering with normal operation

- The RTIC is an independent module that, once activated, cannot be stopped, unless the device is reset followed by a ROM boot sequence. The RTIC is initiated and enabled as part of the high assurance boot sequence
Debug Port Manipulation

- Debug port manipulation is one of the known hackers’ ways of executing unauthorized program code, getting control over secure applications and running code in privileged modes.

- Debug ports such as the IEEE standard 1149.1 (AKA JTAG) provides a hacker with all the means required to break the system’s security mechanisms and get control over the OS.

- Unauthorized debug port usage should be strictly forbidden in order to properly secure the system.

- However, a debug port must be available during platform initial laboratory development, manufacturing tests and software debugging.

- In order to prevent debug port manipulation while allowing access for manufacturing tests and software debugging, smartphone SoC incorporates a debug port access regulator that provides four different protection levels represented by four fuse modes.
Authenticated Debug

Fuse Box Security Levels:
1. No Debug Allowed
2. Random Challenge Required
3. Signed JTAG Request
4. No Security

SoC Security Controls

Random Number Generator

Challenge

Random Challenge
Signed Response

Internal Boot Detection

Customer Super Root Key

Allow Silicon Evaluation
Allowed Security Level

Response

Boot-time Verification Process

JTAG

Authorized Debug

Secure Server

Signed Debug Request
<table>
<thead>
<tr>
<th>Name</th>
<th>Issue</th>
<th>Solution</th>
<th>IP Used</th>
</tr>
</thead>
</table>
| Tamper Detection   | Hackers have used several approaches to gain access to data and to defeat security controls. | There are several techniques to prevent modification and exposure of information due to tampering:  
   - Test port protection and detection – BIST, JTAG, Scan  
   - Thermal detector, Clock monitor, Power supply monitor, Differential tamper detection  
   - Tamper Detect Input pin for Product level security violation detection and chip response (clear internal secrets) | ➤ GPIO pin  
   ➤ Secure BIST  
   ➤ ADM  
   ➤ Scan protection circuit  
   ➤ Optional SRTC |
Protecting a Program in External Memory

**Secure Boot & Integrity Monitor:**
- Authenticating application program prevents execution of a Trojan Horse program that would divulge the application program.

**Encryption:**
- Prevents program cloning.

**Data Encrypted**
- Portion of Program only could be read from DRAM.

**Data In Clear**
- Portion of Program In Clear could be captured from bus DRAM accesses.

**Hardware Crypto & Secret Key:**
- Decrypts application program in Boot Flash or Mass Storage for execution out of DRAM and internal SRAM.

**Tamper Detect:**
- Erases all SRAM Data if Tampering is detected.

**Secure SRAM**
- Critical portion of program is protected in internal SRAM.

**Secure System**
- Authenticating application program prevents execution of a Trojan Horse program that would divulge the application program.

**Clear**
- Portion of Program In Clear.

**Internal SRAM:**
- Critical portion of program is protected in internal SRAM.

**Secure Boot & Integrity Checker**
- Performs integrity checks on the application program.

**Peripheral**
- Provides additional functionality for the system.

**Boot FLASH**
- Stores the application program.

**SRAM**
- Stores the application program in clear.

**LAN**
- Enables communication over a local area network.

**PAN**
- Enables communication over a personal area network.

**WAN**
- Enables communication over a wide area network.

**CAU**
- Controls the application unit.

**RNG**
- Generates random numbers for cryptographic purposes.

**Secure Boot & Integrity Monitor:**
- Authenticating application program prevents execution of a Trojan Horse program that would divulge the application program.
Software Overview

► Well architected and developed application software will require minor adaptations for use in a secure system:
  ► All commonly used embedded system RTOSes may be used
  ► Application software should be evaluated for security weaknesses
  ► Access to sensitive data needs to be carefully assigned
  ► Movement of sensitive data has to be setup for correct encryption/decryption
  ► Security exceptions must be handled according to application requirements
  ► Changes to authenticated data must be re-hashed
  ► System memory allocation may need to be optimized for data and program protection
  ► Support for field system analyses requirements must be designed into the application
  ► Appropriate code signing procedures must be followed
Secure System Tools

Tools for supporting the development, maintenance and provisioning of Secure Systems

Required Functions:

- Code signing using private key
- Public/Private key generation, management and secure storage
- Password management
- Controlled environment for application software development
- Controlled environment for production Flash programming
- System deployment tracking, and update management
- Server authentication of valid systems
Secure System Management Phases

Application Development

- Need to restrict source code availability.
- Prevent un-authorized distribution.
- Prevent insertion of un-authorized code.

**Options:**
- Trust.
- Secure computer systems.

Code Signing

- Requires public/private key pair generation and management.
- Application code signing (Code hash and private key encryption of message digest).

**Options:**
- Locally on a secure computer.
- Contracted to a Code Signing service.
- Use run-time server authentication.

Production Programming

- Prevent un-authorized system (clone) programming.
- Ensure correct configuration of on chip secure system (fuses).
- Prevent un-authorized software distribution.

**Options:**
- Trusted premises programming.
- Secured production programmer.
- Production tracking system.
- Use run-time server authentication.

Maintenance

- Allow field firmware updates/upgrades.
- Prevent un-authorized system (clone) programming.
- Prevent un-authorized software distribution.
- Prevent un-authorized system use.

**Options:**
- Use Code signing process together with standard field firmware updates.
- Use run-time server authentication with software update.
What is Assurance?

Does the system do only what it was designed to do?

- Secure Boot & Executive
- High Assurance Software
- Memory Separation
- Secure Key Storage
- Monitor & Control
Thank you for attending this presentation. We’ll now take a few moments for the audience’s questions and then we’ll begin the question and answer session.