COMPUTER SECURITY: THE GOOD, THE BAD, AND THE UGLY (with applications to embedded systems)

Catherine Meadows
Naval Research Laboratory
Code 5543
Washington, DC 20375
meadows@itd.nrl.navy.mil

Introduction

- A few years ago, took part in a panel on “the good, the bad, and the ugly”
- Each speaker asked to find three types of solutions in their domain of research
  - Good: sound and useful
  - Bad: sound but not useful
  - Ugly: messy but useful
- Instructive exercise
  - Here, I’m going to try to apply it to security in embedded systems

WHAT IS AN EMBEDDED SYSTEM?

- A computer system that is a component of a larger machine or system
- How is it different from a traditional computer system?
  - And how does it affect security?
- We'll see ….

OUTLINE OF TALK

- State of security in today's networked system described as a point of comparison
- Two examples of security problems in embedded systems
  - Cell phones
  - Multilevel security in embedded systems
- Conclusions and open research problems

CURRENT PARADIGM OF COMPUTER SECURITY

- Network of computers
- Each computer has
  - Internal protections (e.g. access control)
  - External protections (authentication, firewalls)
- Network itself has security policy and internal and external protections
- Usually a human in the loop
  - System manager responsible for setting and enforcing security policy
- Doesn’t work perfectly, but works well enough to use it
- Some problems, e.g. viruses, DoS, always with us
- Some problems (e.g. spam) seem intractable
- Don’t know how it will work if things get really bad

The “ugly” solution

HOW DID WE GET HERE?

- Started out with standalone computers
  - Some had internal access controls
  - Some had minimal external controls, e.g. passwords
  - Some had no controls at all, e.g. early personal computers
- Started hooking them up in networks
  - Naturally, problems began to appear
- Security solutions introduced (after the fact)
  - Cryptographic authentication
  - Firewalls
  - Intrusion detection

AN EXAMPLE EMBEDDED SYSTEM - CELL PHONES

- Little or no internal protection
  - Assumed to be single user
- Some external protection
  - Phone must be securely identified so that calls can be correctly charged to it
  - Can shut down cell phone if stolen
- Most protection provided in cellular infrastructure
  - Phone authenticates itself to infrastructure
  - Infrastructure manages accounting
- Some added constraints
  - Power
  - Mobility
  - Cost

AND WHAT'S HAPPENING ANYWAY?

- Exponentially growing complexity and connectivity
- You can now use phones to
  - Surf the web
  - Send and receive text messages
  - Exchange data directly via Bluetooth
    - Allows one device to talk directly to another
    - Bypasses infrastructure
- Now seeing beginnings of
  - Attacks on infrastructure
    - Cell phone spam
  - Direct attacks on phones
    - Cell phone worms
      - Cabir worm - laboratory proof-of-concept worm that got loose
      - Requires Bluetooth in discoverable mode

NIGHTMARE SCENARIO (Schneier)

- Car owner links her Bluetooth-enabled phone to her dashboard computer
  - Allows her to control phone via buttons on steering wheel
- As she drives down the road, phone connects to another in a passing car
- Suddenly, her navigational system fails

SAME STORY AS NETWORK SECURITY

- You start with something simple, start adding complexity and new kinds of connectivity

BUT

- Where do you put the firewalls? Where do you put the intrusion detection?
- Where does the sysadmin sit?
  - Will every cell phone user have to be a sysadmin?

POSSIBLE (PARTIAL) SOLUTIONS

- Offload security to larger, more stable part of the system
  - For cell phones, this is the cellular infrastructure
    - Already done to a large extent already
  - Drawbacks
    - Not useful when devices talk to each other directly
      - E.g. Bluetooth-enabled cellphones
- Improve security of protocols
- Involve users more in security decisions and risk assessment
- Make phone themselves more robust
  - And more expensive
- Problem may never go away entirely
  - New kinds of threats not prepared for by architecture

NEXT EXAMPLE: MULTILEVEL SECURITY

- Data processed and stored at different security levels
  - Unclass, Secret, Top Secret, etc.
- Separation very strict
  - Processes running at lower levels should, as much as possible, be completely ignorant about what goes on at higher levels
- May need some exceptions, however:
  - Data may need downgrading
  - Low data sent to high may need acks

MULTILEVEL SECURITY IN EMBEDDED SYSTEMS

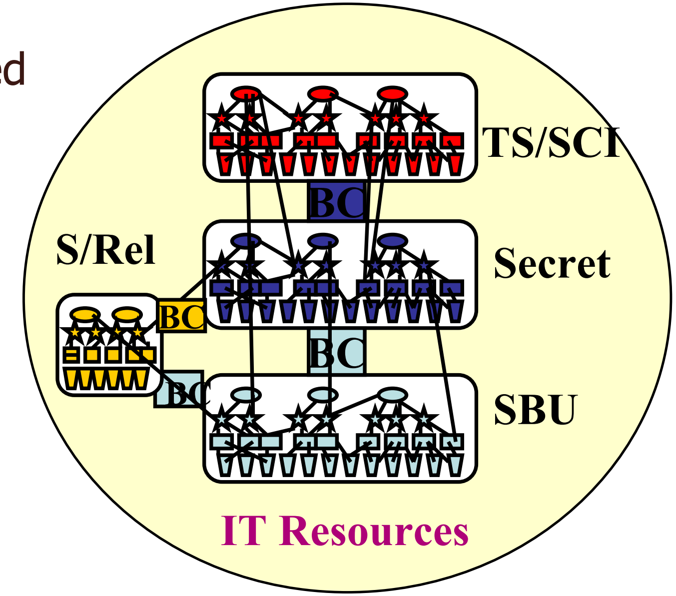
- The US DoD is going “net-centric”
- Networked data to be delivered directly to the warfighter
- This will require MLS embedded systems

MLS "ORANGE BOOK" ARCHITECTURE (1980's)

- Security kernel critical part of operating system
  - Kernel evaluates all access requests and grants or denies them according to security policy
- Two types of access control
  - Mandatory access control
    - Fixed rules governing different security levels
  - Discretionary access control
    - Rules covering everything else
- Security kernels tended to be large and difficult to evaluate
- This was the bad solution

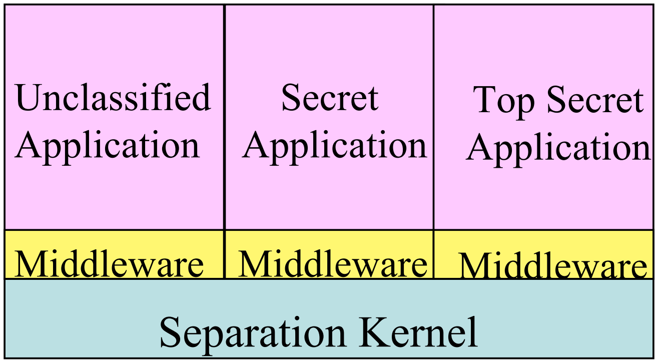
MSL (Multiple Single Level) ARCHITECTURE (1990's)

- Rely on physical separation to enforce separation between security levels
- Each machine has a single security level
- Data from machines at lower levels replicated at higher levels
- Critical trusted components are replicators and downgraders
- This was the good solution
- Advantages
  - Relatively easy to evaluate and modify
  - Works well in networked systems
- Disadvantages
  - Obviously no good for embedded systems!



MILS ARCHITECTURE

- Provide virtual instead of physical separation
  - Use separation kernel to provide independent virtual machines at different security levels
- Provide other security functionality at higher layers
- Separation kernel compact, good for resource-constrained systems
- Can add complexity without having to modify it



CONCLUSIONS WE CAN DRAW

- Necessary to anticipate complexity -- it will come whether you're expecting it or not
- Cell phone example shows that it is helpful to be able to anticipate the kind of complexity you'll get
- Figure out what your critical assets are and concentrate on protecting them first
  - MLS systems
    - protecting separation between security levels
  - Cell phones
    - Ability to make calls
      - Defense against DoS
    - Authentication of calls
- Realize that your critical assets may change, too

RESEARCH PROBLEMS

- Develop architectures for protecting critical assets that are
  - Compact
  - Hold up well under change and added complexity
- Develop avenues for change that respect the architectures we develop
  - Techniques for adding functionality while maximizing protection offered by architecture