The KTH Security Initiative

- Security is strategically important
- ICES did it for embedded systems ...
- ... so we can do it too

“We”:
- Groups of researchers at various KTH schools
  - CSC, EES, ICT Kista, ITM
  - SICS, FOI, Ericsson

For now:
- Something is brewing ...
- If interested contact one of us or mail mfd@kth.se
Embedded Systems Security

Security sensitive embedded applications are everywhere
HACKERS CAN TURN YOUR HOME COMPUTER INTO A BOMB

WASHINGTON — Right now, computer hackers have the ability to turn your home computer into a bomb and blow you to Kingdom Come — and they can do it anonymously from thousands of miles away!

Experts say the recent “Amack-ins” that paralyzed the Amazon.com, Buy.com and eBay websites are tame compared to what will happen in the near future.

Computer expert Arnold Vabnsson, president of the Washington-based non-profit group National CyberCrime Prevention Foundation (NCPF), says that as far as computer crime is concerned, we've only seen the tip of the iceberg.

“The criminals who knocked out these three major online businesses are the front of our worries,” Vabnsson told Weekly World News.

“Then, there are brilliant but unscrupulous computer hackers out there who have developed technology that can knock out an entire region's power grid, or even the entire country. This technology is already being used, and average person can't even dream of it. Even people who are familiar with how computers work have trouble getting their minds around the terrible things that can be done.

“It is already possible for an attacker to send someone an email with an innocent-looking attachment that looks like something they might want to open. When the receiver opens the attachment, the electrical current from the computer's power supply creates a small charge that can be directed at another computer. This can create a series of attacks that can be directed at many computers around the world.

Sickos can wreak death and destruction from thousands of miles away!

Arnold Vabnsson

KABOOM! It might not look like it, but an innocent home computer like this one can be turned into a deadly weapon.

“Just thinking about this is it shouldn't surprise anyone. It's just the next step in an ever-evolving progression of horrors committed and committed by hacks.

Vabnsson points out that these dangerous capabilities have already been demonstrated.

• Vandalized FBI and U.S. Army websites.
• Vandalized Chinese military networks.
• Vandalized and altered Chinese military networks.
• Vandalized Chinese military networks.

And worse, this e-mail bomb program will eventually find its way into the hands of anyone who wants it.

“After a while, anyone who has a computer and knows a few basic commands can download this program and use it to attack anyone's computer,” Vabnsson said.

“Soon it will be sold to terrorists and dangerous religious groups.

“Instead of blowing up a single plane, these groups will be able to blast into the computer network of a large corporation and blow up hundreds of planes at once.

And worse, this e-mail bomb program will eventually find its way into the hands of anyone who wants it.

“After a while, anyone who has a computer and knows a few basic commands can download this program and use it to attack anyone's computer,” Vabnsson said.

“Soon it will be sold to terrorists and dangerous religious groups.

“Instead of blowing up a single plane, these groups will be able to blast into the computer network of a large corporation and blow up hundreds of planes at once.

And worse, this e-mail bomb program will eventually find its way into the hands of anyone who wants it.”
Security Basics I - CIA

Confidentiality:
• Assurance that secret data is shared only among authorized persons or organizations

Integrity:
• Assurance that trusted information is authentic and complete, i.e. that it cannot be altered by untrusted agents

Availability:
• Assurance that the data accessible when and where we need it.
Security Basics II – A Cross-cutting Concern

End-to-end security involves it all! Sorry …
General Security Challenges

- Unknown adversaries
- Unknown environments
- Do not respect abstraction boundaries

And Countermeasures

- Rich processing capabilities
- Protected execution environments
- Well-developed and rich OS’s
- Audits and logs
- Intrusion detection capabilities
- Fast access to updates, alerts, and advisories
- Automatic update facilities
- Rich set of feedback channels
Embedded Security Challenges

Space / performance limitations
  • Crypto difficulties
  • No virus/intrusion detection

Hostile execution environment
  • Device can be subject to physical tampering and device level timing/power/em monitoring

Hidden executable
  • Once on device, executable often unavailable for inspection

Code longevity
  • No mechanism for auto updates

Lack of (e.g. visual) feedback
  • More vulnerable for man-in-the-middle attacks

And more, no doubt ...
Bart Preneel

- PhD 1993 on cryptographic hashing
- Leads COSIC group at K.U. Leuven
- President of the International Association for Cryptologic Research (IACR)
- Visiting positions at DTU, Bochum, Graz, Bergen, Gent, Berkeley
- Well-traveled lecturer at summer schools and conferences
- Very active in EU project contexts (ECRYPT II,...)
- World class expert in embedded system security
Software Security Techniques for Embedded Systems

Mads Dam
KTH/CSC/TCS
Embedded Systems Security

Security sensitive embedded applications are everywhere
Embedded Security Challenges

Space / performance limitations
  • Crypto difficulties
  • No virus/intrusion detection

Hostile execution environment
  • Device can be subject to physical tampering and device level timing/power/em monitoring

Hidden executable
  • Once on device, executable often unavailable for inspection

Code longevity
  • No mechanism for auto updates

Lack of (e.g. visual) feedback
  • More vulnerable for man-in-the-middle attacks
The Upshot

Developing secure embedded software is hard

Strong software quality techniques needed:
- Type safe languages
- Modeling and verification
- Static analysis
- Testing
- Sandboxing, runtime monitoring

To produce more secure + more reliable
- Platforms
- Applications
The Long-Term Vision

- Provably correct programming platforms
- Executing provably correct applications
- On provably correct hardware
- Using provably correct crypto
- Preferably used by provably correct users ;-) 

- Farfetched maybe, but ...
  - The L4 RTOS
  - SACEM RER Paris Metro
  - Mondex smart cards
  - Rockwell Collins AAMP5
  - Airbus A340 500/600
  - Praxis MULTOS/Tokeneer
Formal Methods at KTH/CSC

- Karl Meinke, PhD, prof Testing, formal methods
- Mads Dam, PhD, assoc prof Security, programming languages, networking
- Dilian Gurov, PhD, assoc prof Formal methods, control flow analysis
- Gurvan le Guernic, PhD, postdoc Language-based security
- Currently 8 PhD students
- Collaborations and projects:
  - VR, EU, Access, KTH, INRIA, Microsoft Research, SICS
  - EU projects: HATS, S3MS, VerifiCard, LOMAPS, ...
Security Research Themes

Runtime monitoring:
• How to specify/enforce rich security policies?

Information flow control:
• How to check fine-grained confidentiality + integrity properties?

Control flow analysis
• How to analyze, extract, and compose control flow graphs?

Secure compilation techniques

Security specification techniques
• Logics and model checking

Java bytecode security
• Static analysis and proof-carrying code
• Java smartcard case studies
Theme: Information Flow Analysis

Confidentiality: What can a low observer tell about the high inputs?
Integrity: How is a high observer affected by low input?
Examples

if h == 0
then l = 0
else l = 1
begin
if h == 0;
then print(
);  
else print(
);  
end

while h != 0 {h = h+1}

begin
l = 0 ;
while h != 0 {
    h = h -1} ;
l = 1
begin
lock(l);
process( );
unlock(l);
end
A Disclaimer

- Comprehensive static information flow control probably impossible ...
- C.f. Kocher et al timing/power/EM attacks
- Real processors offer rich opportunities for side channels:
  - Memory management
  - Caches
  - Pipelines
  - ...
- Still - don’t want to leave an empty playing field
Applications

Prevention of undesired side channels

- Don’t write user PIN to standard output
- Don’t write any secret key bit to any publicly readable output field
- More precisely: Do not allow any secret key bit to be correlated with any publicly readable output field

- Do not allow untrusted assignments to a trusted URL
- Do not allow brake status query signals to affect critical brake functionality
Security Type Systems

Idea: Assign safety types low/high to program text

|- c : t  means c does not produce flow below level t

|- skip: t

|- c : high

|- c: low

|- c₁ : t  |- c₂ : t

|- c₁ ; c₂ : t

|- x : t var  |- y : t

|- x = y : t

|- b : t  |- c₁ : t  |- c₂ : t

|- if b then c₁ else c₂ : t

Volpano, Smith: A sound type system for secure flow analysis, JCS’96
Precision

Efficient but too rigid:

```
if h == 0 {
  l = 0;
  l = 0
}
```

```
if h == 0 {
  l = 1; l = 0;
  l = 0;
}
```

But also not rigid enough:

```
while h != 0 h--
  l = 1;
while h != 0 {h--} ;
  l = 0
```

Research challenges:

- What are “good” conditions for real languages?
- Current work: Efficient algorithms for more precise + more robust information flow control
Utility

Multi-level security is too rigid

```plaintext
get(passwd) ;
if good(hash(passwd)) {enter}{exit}
get(secret) ;
send(public, encrypt(secret, myPrivateKey))
```

Research challenge:
- What are “good” models for declassification?

Dam: Decidability and Proof Systems ... , Proc. POPL’06
Theme: Security Monitoring

Security policy = automaton  [Schneider’00]

Enforcement is easy

```plaintext
proc foo( ... )

...  
  do
  ...
  end

get the state
is do enabled?
Y? update the state; return
N? abort

Allows rich security policies
```
SECURITY STATE

String requestorURL, String requestedFile;

BEFORE BluetoothToolkit.sendFile(String destURL, String file)

PERFORM

  requestorURL.equals(destURL) &&
  requestedFile.equals(file) -> {};

AFTER reply = JOptionPane.showConfirmDialog(String query)

PERFORM

  reply != 0 && goodFileQuery(query) -> {
    requestedFile = queryFile(reply);
    requestorURL = queryRequestor(reply);
    true -> {};
  }
Monitor Inlining

```
proc foo( ... )
  ...
  s = the_state() ;
  if do enabled in s then {
    s = update(s,do) ;
    do }
  else abort() ;
  ...
end
```

- Efficient implementation strategy
- Enforcement functionality weaved into application code in AOP style
- No need for monitoring infrastructure
- Applications made self-enforcing
- When can this be done?

Aktug, Dam, Gurov, Provably correct runtime monitoring, Journal of Logic and Algebraic Programming, 2009
Multithreaded Inlining Strategies

Shared security state needed

**Blocking**

```c
proc foo( ... )

... 
lock(state) ;
s = the_state() ;
if do enabled in s
then {
    s = update(s,do) ;
    do() ;
    unlock(state)}
else {
    unlock(state) ;
    abort()};
```

Sacrifices transparency

**Non-blocking**

```c
proc foo( ... )

... 
lock(state) ;
s = the_state() ;
if do enabled in s
then {
    s = update(s,do) ;
    unlock(state) ;
    do() } 
else {
    unlock(state) ;
    abort()};
```

Sacrifices security
Multithreaded Inlining

Blocking inliner:
- Secure but undesirable: May introduce deadlocks

Non-blocking inliner:
- Insecure – potential race

Results:
- Correct (secure and transparent) inlining not possible in general
- Correct inlining possible when restricted to class of race-free policies

Information Flow Monitoring

Monitoring one execution not sufficient

Need to observe that untaken branch assigns 1 to l here!

Possible solution: Combine static and dynamic analysis

In Conclusion

• New tools and techniques becoming available for static and dynamic software quality assurance
• Relevant for the embedded domain
• Two examples:
  • Information flow analysis
  • Runtime monitoring
• Feedback and industrial case studies welcome!
THANK YOU!
Embedded Security - Panel Discussion
Topics for Discussion

How can KTH/academia/institutes help?

What are needs of industry?
- Competent security specialists? New tools and methods?

What are the best ways of communicating?

What are the important research directions?