Hack-a-Vote: Studying Security Issues with E-Voting

Dan Wallach
Rice University

Collaborators:
Jonathan Bannet
David W. Price
Algis Rudys
Justin Singer

Perception vs. reality

- Voter feels that
  - Vote was counted
  - Vote was private
  - Nobody else can vote more than once
  - Nobody can alter others' votes
- People believe that the machine works correctly

☛These have to do with *perception*

It is also important that these perceptions are true.

Reliance on certification

Independent Testing Authorities

- Allowed to see the code
  - Nobody else looks
- Certify satisfaction of FEC standards
- Required by many states

Result: “Faith-based voting”

Inspiration

Have an e-voting system to "demonstrate" insider flaws

- Original idea from David Dill
- Original code by David W. Price
  - Written summer 2003
  - About 2000 lines of Java

Unnecessary after Diebold findings

Second application?

- How about in-class use?
- Old project: “smart card soda machine”
  1) design & formally model crypto protocol
  2) swap with other groups
  3) implement with real cards

☛ Real smart cards are painful

Hack-a-Vote project

- Remove “cheating” code
  - ~150 lines, mostly in one file
- Three phase assignment
  1) Be evil (2 weeks)
  2) Be an ITA (1 week)
  3) Design / formally model better version of Diebold smartcard (2.5 weeks)

Be evil?

- Students' role: corrupt developer inside vendor
- Code must still pass tests
- "Minimal" code changes
  - Multiple hacks encouraged

☛ Code should appear "normal"

Deliverables: Code + Written Report

Be an ITA?

- Swap code from groups
- Every group audits two versions
  - Honor code: no running **diff**
- Imperfect simulation of real ITAs
  - Student familiarity with code
  - Smaller codebase

Deliverables: Written Report

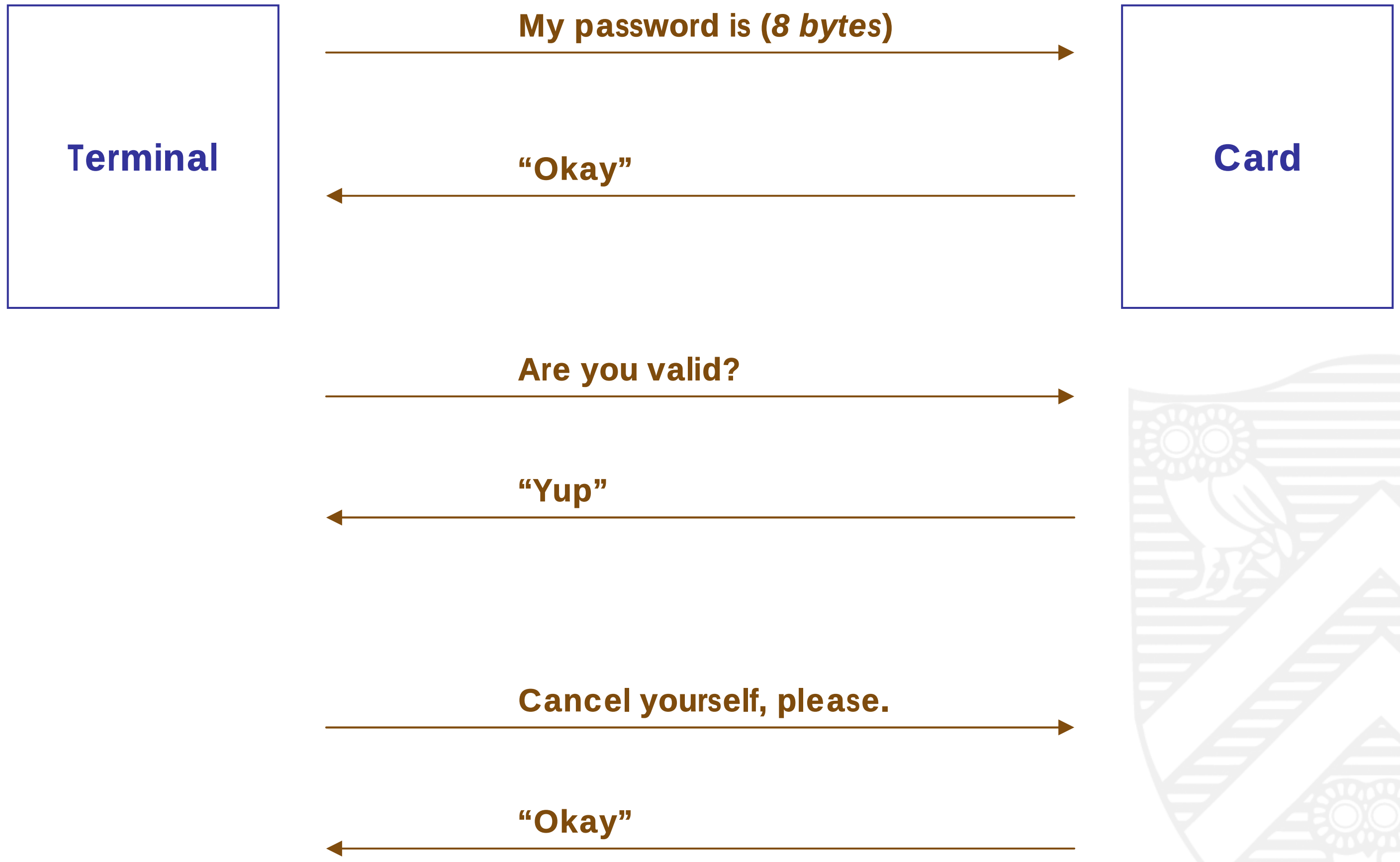
Better smartcard protocols?

- Lectures have prepared students
- **cryptyc** for protocol modelling
  - (Relatively) usable type checker

    cryptyc.cs.depaul.edu

Deliverables: Model + Written Report

Diebold's smart card protocol



My password is (*8 bytes*)

"Okay"

Are you valid?

"Yup"

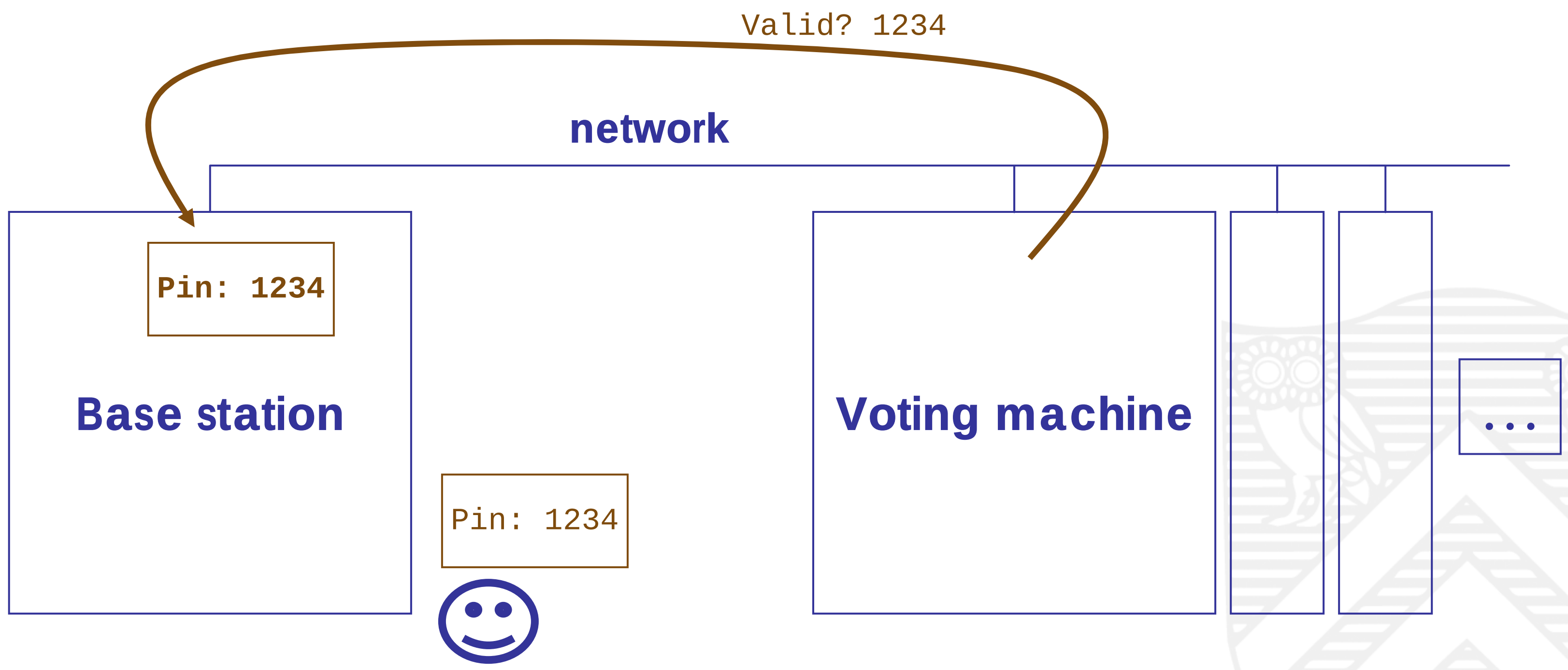
Cancel yourself, please.

"Okay"

Hack-a-Vote software

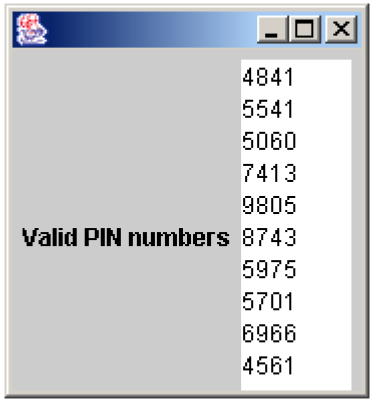
Inspiration: Hart InterCivic eSlate

eSlate protocol (hopefully)

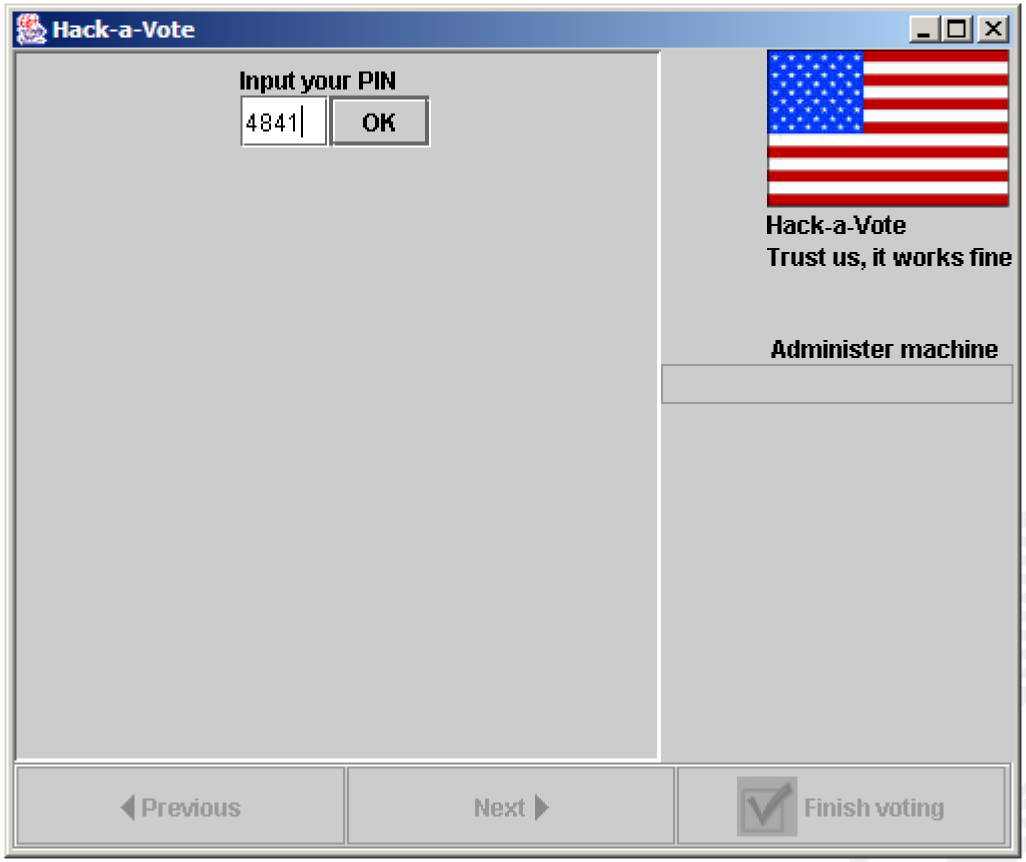


Hack-a-Vote live demo

Hack-a-Vote design



Valid PIN numbers	4841 5541 5060 7413 9805 8743 5975 5701 6966 4561

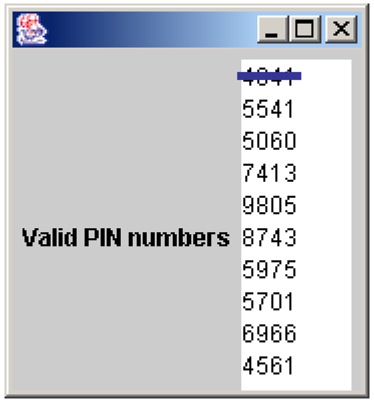


Hack-a-Vote

Input your PIN

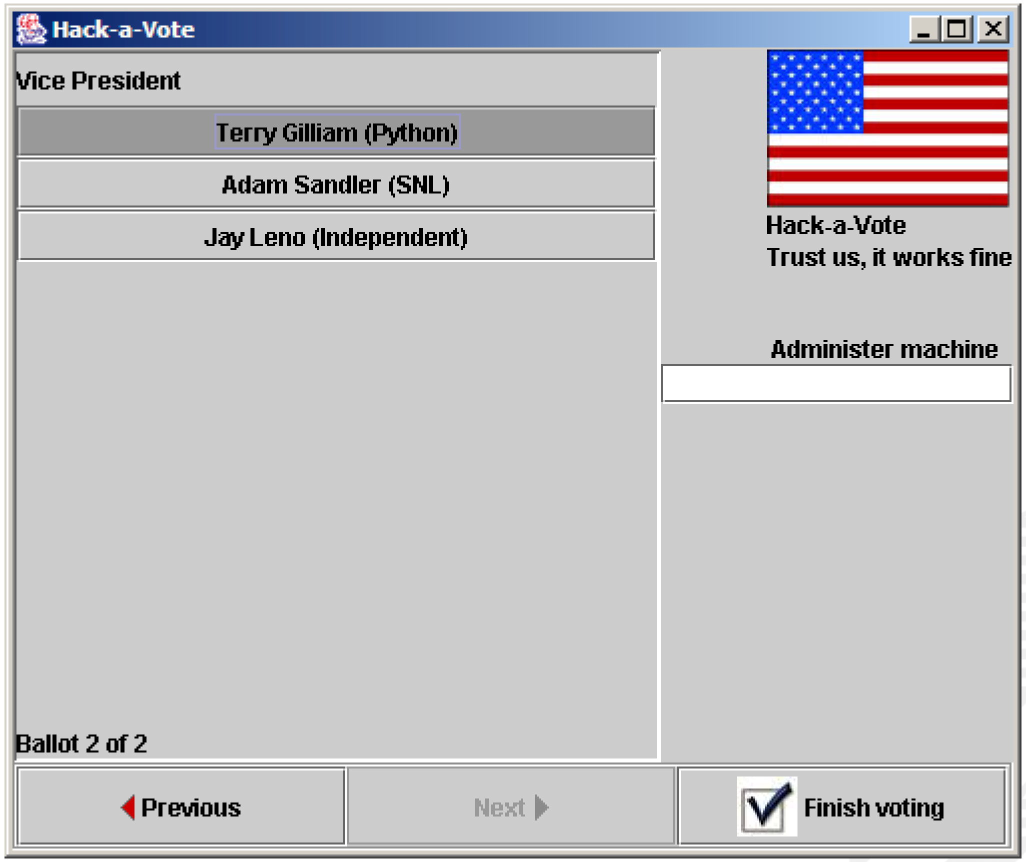
4841

Hack-a-Vote design



Valid PIN numbers

~~4041~~
5541
5060
7413
9805
8743
5975
5701
6966
4561



Hack-a-Vote

Vice President

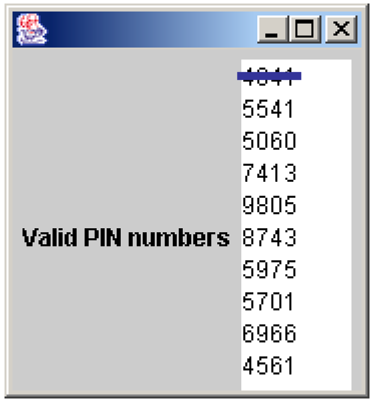
Terry Gilliam (Python)

Adam Sandler (SNL)

Jay Leno (Independent)

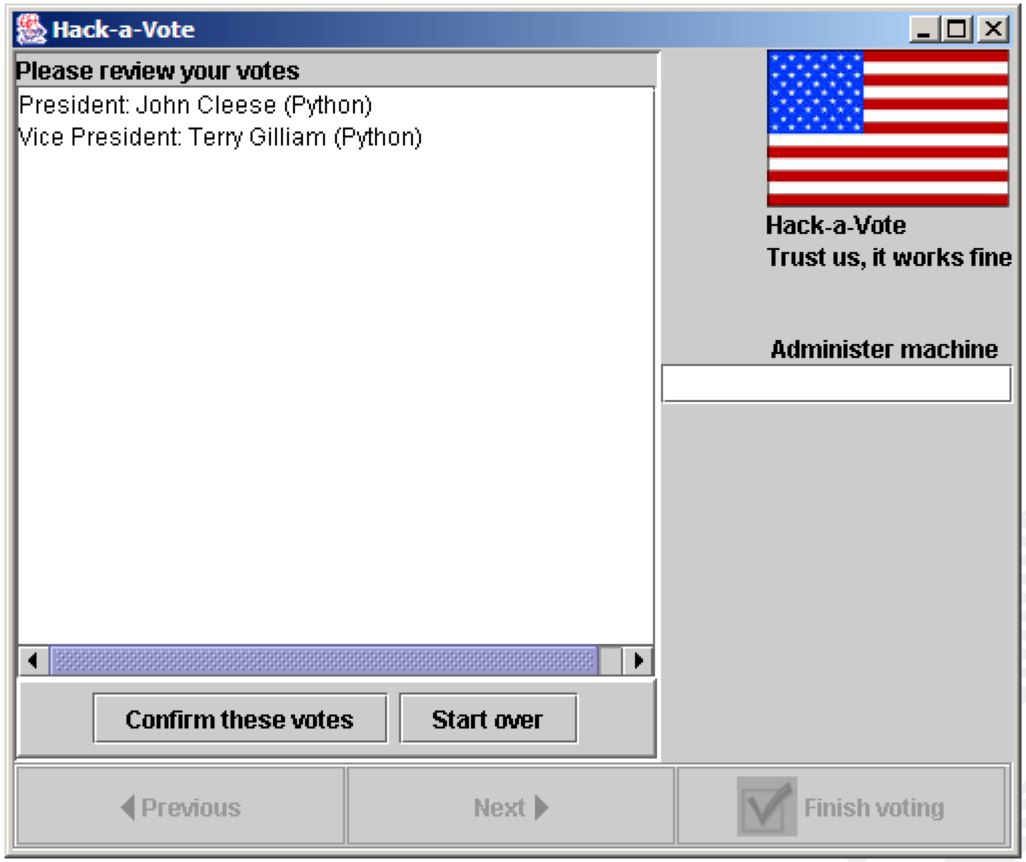
Ballot 2 of 2

Hack-a-Vote design



Valid PIN numbers

~~4641~~
5541
5060
7413
9805
8743
5975
5701
6966
4561

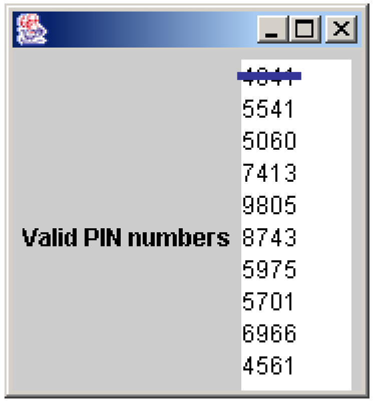


Hack-a-Vote

Please review your votes

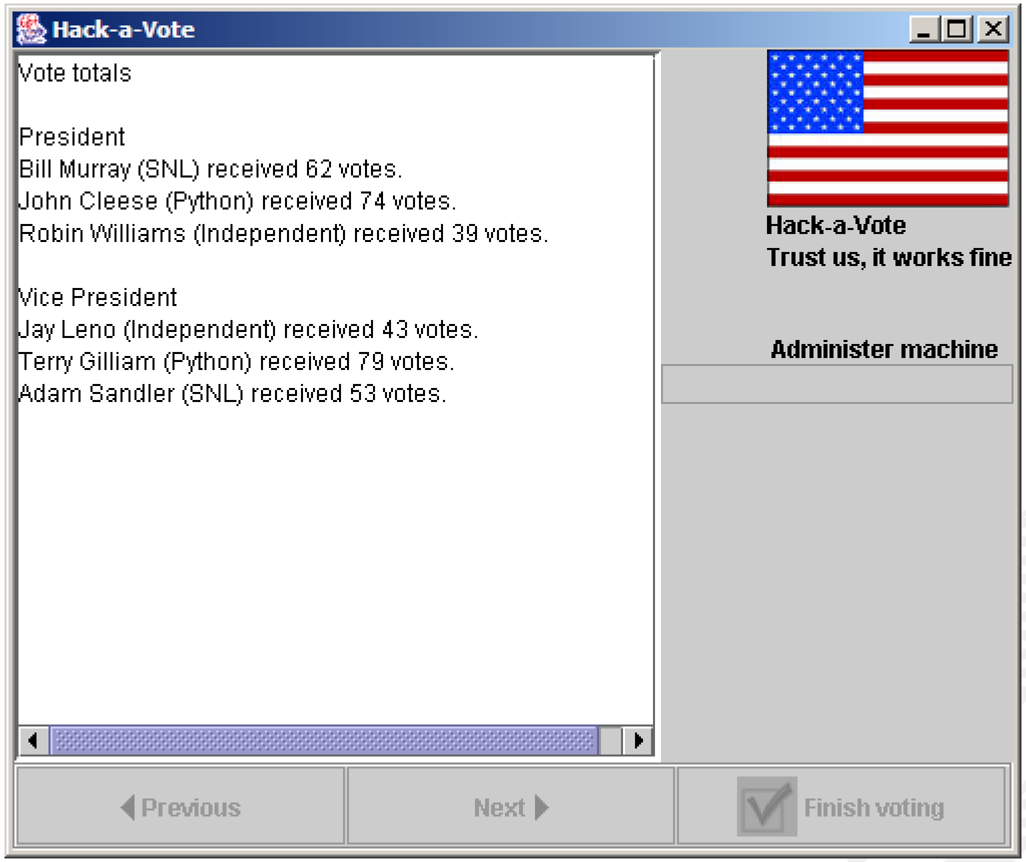
President: John Cleese (Python)
Vice President: Terry Gilliam (Python)

Hack-a-Vote design



Valid PIN numbers

~~4041~~
5541
5060
7413
9805
8743
5975
5701
6966
4561



```
Vote totals

President
Bill Murray (SNL) received 62 votes.
John Cleese (Python) received 74 votes.
Robin Williams (Independent) received 39 votes.

Vice President
Jay Leno (Independent) received 43 votes.
Terry Gilliam (Python) received 79 votes.
Adam Sandler (SNL) received 53 votes.
```

Wide gamut of attacks

- Manipulate election results
- Violate voter anonymity
- Crash / DoS voting machine

Clever hacks

- Overload **`equals()`** / **`hashCode()`**
- Variable with same name as class
  - ☛ Unusual control flows
- Reuse constants in the code
  - Network port: **`1776`**
  - Use as backdoor PIN
- “Start over” also submits a vote

Deeper hacks

- Weak random number generator
  - Easier to guess valid PINs
- RNG for vote shuffle seeded with terminal ID
  - Attacker can undo shuffle
- Only cheat if terminal ID > 2
  - Less likely to occur in testing

Did the ITAs catch the hacks?

Hack	Attempts	Found once	Found twice
Modify already-cast votes	6	6	5
Cast multiple votes	7	7	6
Violate voter anonymity	4	3	2
Denial of service	4	3	2

Implications for real ITAs

- Can real ITAs do better?
  - + They can run **diff**
  - + They can perform "parallel testing"
  - – Codebases are much larger
  - – Are they expecting Trojan Horses?
  - – How closely do they read the code?

- Very little support from tools

Uglier issues for certification

- Toolchain tampering (Thompson)
- Tampering with “embedded” OS
- Audited code = actual code in machine?

Publicity

IEEE Security & Privacy, Jan/Feb 2004

- Reprinted in *Computer User*
- Story on local TV news
- Impact on vendors / ITAs?

Choose Hack-a-Vote!

www.cs.rice.edu/~dwallach/courses/comp527_f2003/voteproject.html

BSD-style license

Trust us, it works fine