15-410

"…"Windows NT is C2 Secure"…"

Security Overview
Apr. 14, 2004

Dave Eckhardt

Bruce Maggs

Synchronization

15-412

- If this was fun...
- If you want to see how it's done “in real life”,
- If you want to write real OS code used by real people,
- Consider 15-412 (Spring '05)

Synchronization

Today

- Chapter 19, more or less

Next time

- Fun stuff not in the text

Some upcoming lectures – the “ECE invasion”

- Eno Thereska on advanced disk scheduling
- Joey Echeverria on comparative OS structure

Overview

Goals & Threats

Technologies

Next Time

- Applications
- Systems

U.S. DoD “Orange Book” Security Classifications

D – try again

C – authentication, controlled sharing

B – per-object sensitivity labels, user clearances

A – B-class system with formal spec, proofs

Sub-levels

- C2 = C1 + ACLs, audit logs, anti-tamper OS, ...

“Windows NT is C2 secure”

Windows NT is C2 secure

Wimpy old Unix is only C1

Use Windows, it's secure!

Windows NT is C2 secure

Windows NT is C2 secure

Wimpy old Unix is only C1

Use Windows, it's secure!

- *Melissa, Code Red, SQL slammer, SoBig, ...*
- What's wrong with this picture?

“Security Architecture” undermined by implementation

Physical Security

- Locked rooms, disable floppy booting
- In practice, isolate from Internet!

Goals & Threats

Authentication

- Threat: impersonation

Secrecy

- Threats: theft, eavesdropping, cipher breaking, ...

Integrity

- Threat: cracking

Signature

- Threats: impersonation, repudiation

...

Goals & Threats

Authentication

- Visitor/caller is Alice

Impersonation

- Act/appear/behave like Alice
- Steal Alice's keys (or “keys”)
- Maybe you can read Alice's secrets
- Maybe Alice goes to jail

Goals & Threats

Secrecy

- Only Bob can read Bob's data

Difficult secrecy threats

- Break a cipher (see below)
- Compromise a system (see below)
- Or...

Eavesdropping – get data while it's unprotected!

- Wireless keyboard
- Keystroke logger
- TEMPEST

TEMPEST

Code name for electromagnetic security standard

- The *criteria document* is classified

Problem

- Computers are *radios*
- Especially analog monitors
  - ~150 MHz signal bandwidth (“dot clock”)
  - Nice sharp sync pulses
- Surveillance van can *read your screen* from 100 feet

Goals & Threats

Integrity

- Only *authorized personnel* can add bugs to a system
- Or edit bank account balances
- Or edit high school grades

Threats

- Hijacking authorized accounts
- Bypassing authorization checks
  - Boot system in “administrator mode”?
  - Boot some other OS on the machine?
- Modifying hardware

Goals & Threats

Signature

- “Pay Bob $5 for his program” was uttered by Alice

Threats

- Alice repudiates message (after receiving program)
- Charlie signs “Pay Charlie $500 for his program”
  - *... with Bob's signature*

Goals & Threats

Anonymous communication

- “Whistle blowers”
- Secret agents

Threat

- “Traffic analysis”
  - Observe repeated “coincidence”
    - Node 11 sends a message, Nodes 1-10 attack
  - Which node is a good target?

Goals & Threats

Availability

- Web server is available to corporate customers
- Mailbox contains interesting mail

Threat

- DoS – Denial of Service
  - Flood server with bogus data
  - “Buries” important data
  - SYN flooding, connection resetting

Another DoS Attack

Automated Flight Data Processing System

- Transfers flight arrival/departure data
  - ...between radar tower in Elgin, IL (where's that?)
  - ...and tower at *O'Hare International*

Fallback system

- paper, pencil, telephone

Uh-oh...

- Chief engineer quit
  - after deleting *sole copy* of source code

Now What?

Police raided his house

Recovered code!

- Encrypted
- Cracked in 6 months

Summary

- http://news.airwise.com/stories/99/10/940530321.html

Lesson?

- People matter...

Malicious Programs (“malware”)

Trojan horse

Trapdoor

Buffer overflow

Virus/worm

Trojan, Trap Door

Trojan Horse

- Program with two purposes
- Advertised – “Here is the new security update!”
- Actual – Here is a hard-disk-wipe program!

Trap door

- login: anything
- Password: My hovercraft is full of eels!

#insert <reflections_on_trusting_trust>

Buffer overflow

```
GET  /default.ida?XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXX%u9090%u6858%ucbd3%u7801%u9090
%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u78
01%u9090%u9090%u8190%u00c3%u0003%u8b00%u5
31b%u53ff%u0078%u0000%u00=a  HTTP/1.0
```

Virus/Worm

Virus

- Program which cannot replicate itself
- Embedded in other programs, runs when they do
- Embeds self in other programs

Worm

- Breaks into remote machine
- Launches remote copy
- May not reside permanently on disk

Technologies

Scanning/intrusion detection/auditing

Hashing

Encryption (1-time, private, public)

Scanning

Concept

- Check your system for vulnerabilities
  - Before somebody else does!

Details

- Password scan
- Scan for privileged programs, extra programs
- Check for dangerous file permissions
- Check that program, config files have correct contents
- Are mysterious programs running?

Intrusion Detection

Concept

- Monitor system in secure state
- Summarize typical behavior
- Watch for disturbing variation

Examples

- Sudden off-site traffic to/from a machine
- Change in system call mix
  - Gee, my web server doesn't *usually* exec(“/bin/sh -i”)...

Issues – false positive, false negative

Auditing

Concept

- Estimate damage
  - What was taken?
- How to fix system?

Approach

- Log system actions off-board
  - paper printer
  - disk with hardware roll-back

Boring but useful *when* you're in trouble...

Hashing

Concept

- "One-way function"
- h_1 = f(message$_1$)
- h_1 != f(message$_2$), f(message$_3$), ...

Use

- Here is the OpenBSD CD-ROM image
- And here is the MD5 hash
- "Infeasible" to find malware with that hash

Hashing Issues

Verify data?

- Compute & check hash against official version

Say, what *is* the official version?

- The ***key distribution*** problem
- Easy if you're in a room with the OpenBSD release coordinator
- Otherwise, not easy

Don't trust MD5

- SHA-1 (for now)

Encryption

Concept

cipher = E(text, K_1)

text = D(cipher, K_2)

Algorithm E(),D()

- Should be *public*
  - Or else it will be cracked

Keys

- One (or maybe both) kept secret

“Random” Numbers

Recall back to Project 1...

- We encouraged you to quiz the user on random strings
- Some people turned in not-so-random behavior

Three concepts

- Pseudo-random number generator (PRNG)
  - Next = (Previous*L+I) mod M
  - Behaves *the same way every time* - not random *at all*
- Kind-of-random stuff
  - srand(get_timer());
  - Ok for games (where money isn't involved)
- Entropy pool

Entropy Pool

Goal (for security) is unguessability

- aka unpredictability, true randomness, entropy

Why “kind-of” doesn't work

- Netscape seeded SSL session key generator with
  - getpid(), getppid(), time of day
  - Time is a globally-known value
  - Process IDs occupy a small space
    - » ...especially if you are on the same machine!

Some things are genuinely random

- Which microsecond does the user press a key in?
- “Entropy Pool” is a queue of those events

One-Time Pad

Key

- *Truly random* byte string

Algorithm

- E(): XOR one key byte, one message byte
- D(): same process!
  - random XOR random = 0
  - msg XOR 0 = msg, so
  - (msg XOR random) XOR random = msg

One-Time Pad

Pad must be as long as message

Must be delivered securely

Never re-use pads!!

- (m1 XOR pad) XOR (m2 XOR pad) = (m1 XOR m2)
- Can be scanned very quickly

Private Key

Concept: ***symmetric*** **cipher**

```
cipher = E(text, Key)
text = E(cipher, Key)
```

Good

- **Fast, intuitive (password-like), small keys**

Bad

- **Must share a key *(privately!)* before talking**

Applications

- **Bank ATM links, secure telephones**

Public Key

Concept: *asymmetric* cipher (aka "magic")

```
cipher = E(text, Key1)
text = D(cipher, Key2)
```

Keys are *different*

- Generate *key pair*
- Publish "public key"
- Keep "private key" *very* secret

Public Key Encryption

Sending secret mail

- Locate receiver's public key
- Encrypt mail with it
- Nobody can read it
  - *Not even you!*

Receiving secret mail

- Decrypt mail with your private key
  - No matter who sent it

Public Key Signatures

Write a document

Encrypt it with your private key

- Nobody else can do that

Transmit plaintext *and ciphertext* of document

Anybody can decrypt with your public key

- If they match, the sender knew your private key
  - ...sender was you, more or less

(really: send E(hash(msg), K_p))

Public Key Cryptography

Good

- No need to privately exchange keys

Bad

- Algorithms are slower than private-key
- Must trust key directory

Applications

- Secret mail, signatures

Comparison

Private-key algorithms

- Fast crypto, small keys
- *Secret-key-distribution problem*

Public-key algorithms

- “Telephone directory” key distribution
- Slow crypto, *keys too large to memorize*

Can we get the best of both?

- Next time!

Summary

Many threats

Many techniques

“The devil is in the details”

Just because it “works” doesn't mean it's right!

Open algorithms, open source

Further Reading

Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations

- Markus Kuhn, Ross Anderson
- http://www.cl.cam.ac.uk/~mgk25/ih98-tempest.pdf

Optical Time-Domain Eavesdropping Risks of CRT Displays

- Markus Kuhn
- http://www.cl.cam.ac.uk/~mgk25/emsec/optical-faq.html

Further Reading

Reflections on Trusting Trust

- Ken Thompson
- http://www.acm.org/classics/sep96

Netscape random-number oops

- http://www.cs.berkeley.edu/~daw/netscape-randomness.html

Lava-lamp random numbers

- http://www.LavaRnd.org/