Security Overview

Dave Eckhardt
de0u@andrew.cmu.edu

Synchronization

- Today
  - Chapter 19
  - Plus extra fun stuff

Overview

- Goals & Threats
- Technologies
- Applications
- Systems

U.S. DoD “Orange Book” 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?
- Details matter
  - Disable floppy booting
  - No network connection

Goals & Threats

- Authentication (impersonation)
- Secrecy (theft, eavesdropping)
- Integrity (cracking)
- Signature (repudiation)
- ...

Goals & Threats

- Authentication
  - Visitor/caller is Alice
- Impersonation
  - Act/appear/behave like Alice
  - Steal Alice's keys (or “keys”)

Goals & Threats

- Secrecy
  - Only Bob can read Bob's data
- Break security (see below)
- 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
  - 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
    - What a coindicence!
    - Node 11 sends a message, Nodes 1-10 attack
    - Which node is a good target?

Goals & Threats

- Availability
  - Web server is available to corporate clients
  - 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 O'Hare International tower
    - and radar tower in Elgin, IL
- 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, trapdoor

- Trojan Horse
  - Program with two purposes
  - Advertised – “Here is the new security update!”
  - Actual – Here is a hard-disk-wipe program!
- Trap door
  - login: <u>anything</u>
  - Password: My hovercraft is full of eels!

Buffer overflow

- HTTP GET /index.html
- Host: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx$^@&#$^@#&**&/bin/sh $&$*@*$@

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
  - 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
- 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 trouble...

Hashing

- Concept
  - “One-way function”
  - h = f(message1)
  - h != f(message2), f(message3), ...
- 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
  - Verify *hash?*
  - The *key distribution* problem
- Don't trust MD5
  - SHA-1 (for now)

Encryption

- Concept

  ```
  cipher = E(text, K1)
  text = D(cipher, K2)
  ```

- Algorithm E(),D()
  - Should be ***public***
    - Or else it will be cracked
- Keys
  - One (maybe both) kept secret

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

  ```
  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?

Kerberos

- Goal
  - Authenticate, encrypt for N users, M servers
  - Fast private-key encryption
  - User remembers one ***small*** key
- Problem
  - Can't have system with NxM keys!
- Intuition
  - ***Trusted third party*** knows ***every*** user, server key

Not Really Kerberos

- Client sends to Key Distribution Center
  - {client, server, time}
- KDC sends client
  - $\{K_{session}$,server,time$\}K_c$
  - Ticket=$\{$client,time,$K_{session}\}K_s$
- Client decrypts session key, sends ticket to server
- Server decrypts ticket to $\{$client,time,$K_{session}\}$
- Client, server share a session key (and know so)

SSL

- Goal
  - Fast, secure commnication
- Problem
  - Public key algorithms are slow
  - ***There is no global key directory***
- Intuitions
  - Use private-key encryption for speed
  - Replace global directory with ***chain of trust***

Not SSL

- Server certificate
  - Whoever can ***decrypt*** messages ***encrypted*** with public key AAFD01234DE34BEEF997C is www.cmu.edu
- Client calls server, requests certificate
- Server sends certificate
- Client generates private-key ***session key***
- Client sends $\{K_{session}\}K_{server}$ to server
- If server can decrypt and use $K_{session}$, it must be legit

SSL Certificates

- How did we know to trust that certificate?
- Certificates signed by ***certificate authorities***
  - USPS, Visa, Baltimore CyberTrust, CMU
  - “Whoever can ***decrypt*** messages ***encrypted*** with public key AAFD01234DE34BEEF997C is www.cmu.edu
    - Signed, Baltimore CyberTrust”
- Certificate authority public keys ***ship in browser***
  - “Chain of trust”

PGP

- Goal
  - “Pretty Good Privacy” for the masses
  - Without depending on a central authority
- Approach
  - Users generate key pairs
  - Public keys stored “on the web”
  - Users sign each other's keys

PGP

- "***Web*** of trust"
  - Dave and Joey swap public keys (in my office)
  - Dave and Tadashi swap public keys (at lunch)
  - Dave signs Tadashi's public key (publishes signature)
  - Joey fetches Tadashi's public key
    - Verifies Dave's signature on it
  - Joey can safely send secret mail to Tadashi
  - Tadashi can sign mail to Joey

Password File

- Goal
  - User memorizes a small key
  - User presents key, machine verifies it
- Wrong approach
  - Store keys in file

Hashed Password File

- Better
  - Store hash(key)
  - User presents key
  - Login computes hash(key), verifies
- Vulnerable to ***dictionary*** attack
  - Cracker computes hash(“a”), hash(“b”), ...
  - Once computed, works for ***many users***
- Can we make the job harder?

Salted Hashed Password File

- Choose random number for new user
- Store #, hash(key,#)
- User presents key
- Login computes hash(typed-key,#) - no harder
- Cracker must compute a ***much larger*** dictionary
- Can we do better?

Shadow Salted Hashed Password File

- Protect the password file after all
- “Defense in depth” - Cracker must
  1. Compute enormous dictionary
  2. Break system security to get hashed password file
  3. Scan enormous dictionary
- Bribing user could be easier!

One-time passwords

- What if somebody ***does*** eavesdrop?
  - Can they undetectably impersonate you forever?
- Approach
  - System (and user!) store key ***list***
    - User presents head of list, system verifies
    - User and system ***destroy that item***
- Alternate approach
  - Portable cryptographic clock (“SecureID”)

Biometrics

- Concept
  - Tie authorization to *who you are*
    - Not what you know – can be copied
  - Hard to impersonate a retina
    - Or a fingerprint

Biometrics

- Concept
  - Tie authorization to *who you are*
    - Not what you know – can be copied
  - Hard to impersonate a retina
    - Or a fingerprint
- Right?

Biometrics

- Concept
  - Tie authorization to *who you are*
    - Not what you know – can be copied
  - Hard to impersonate a retina
    - Or a fingerprint
- Right?
- ***What about gummy bears?***

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

- Impact of Artificial "Gummy" Fingers on Fingerprint Systems
  - Matsumoto et al
  - http://cryptome.org/gummy.htm

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

- Kerberos: An Authentication Service for Computer Networks
  - B. Clifford Neuman, Theodore Ts'o
  - USC/ISI Technical Report ISI/RS-94-399