Protection & Security

Today

- Security environment
- Basic of cryptography
- User authentication
- Protection mechanisms
- Attacks from inside/outside the system

Next

- Research in OS

The security environment

- Security
  - General problem involved in making sure files are not read/modified by unauthorized people;
  - It includes technical, administrative, legal and political issues
- Protection
  - Mainly OS mechanisms to safeguard information in the computer
- Security goals and threats

Goal	Threat	Description
Data confidentiality	Exposure of data	Secret data should remain secret
Data integrity	Tampering with data	Unauthorized users should not be able to modify data without owner's permission
System availability	Denial of service	Protect it from people making it unusable
Exclusion of outsiders	System takeover by virus	Increasing problem – takeovery to, for example, spam

- And … privacy: protecting people for misuse of info about them

Google Refuses Demand for Search Information Government Asked 4 Firms for Data in Effort to Revive Anti-Porn Law - January 20, 2006

washingtonpost.com

Slashdot
NEWS FOR NERDS. STUFF THAT MATTERS.
Log In | Create Account | Help | Subscribe | Firehose
Sections
Main
Apple
AskSlashdot
Books
Developers

Bush Demands Amnesty for Spying Telecoms

Posted by kdawson on Monday December 01, @09:14PM from the **courtroom-battles-not-ended** dept.

The Bush administration and the Electronic Frontier Foundation are poised to square off in front of a San Francisco federal judge Tuesday to litigate the constitutionality of legislation immunizing the nation's telecoms from lawsuits

Intruders & accidental data loss

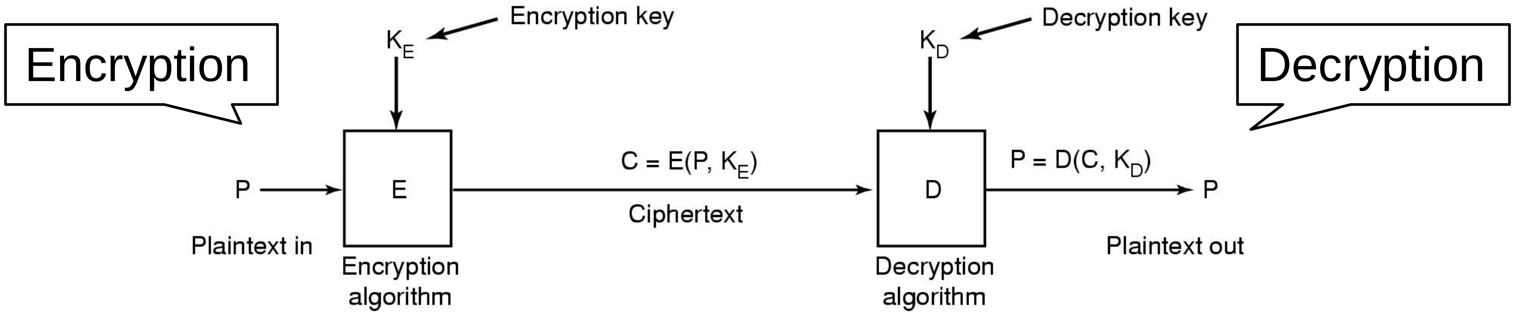
Know who/what you are dealing with

- Some common categories of intruders
  - Casual prying by nontechnical users
  - Snooping by insiders
  - Attempt to make $ (bank programmers' versions of rounding)
  - Commercial or military espionage
  - Virus – the writer is the intruder
- Beyond malicious intruders, plain accidents
  - Acts of God: fires, floods, earthquakes …
  - Hardware or software errors
  - Human errors
  - While seemingly mundane, most damage is probably due to accidental loss

Most can be dealt by maintaining adequate backups

Basics of cryptography

- Goal – make plaintext into ciphertext so that only authorized people can convert it back
- Kerckhoff’s principle
  - Encryption/decryption algorithms should be public – avoid *security by obscurity*
  - Secrecy should depend on keys (parameters)
- Relation between the different pieces
  - P is plaintext file, C is ciphertext
  - K_E/K_D is encryption/decryption key
  - E/D is encryption/decryption algorithm



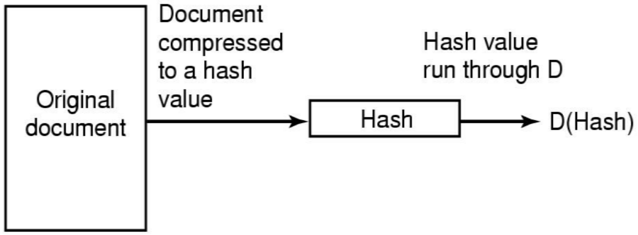
Secret- & public-Key cryptography

- Secret-key cryptography (or symmetric)
  - Simple example – monoalphabetic substitution
  - Given the encryption key, easy to find decryption key
    - In the example – statistical properties of natural languages
  - Could be ok if keys are long enough
- Public-key cryptography – e.g. RSA
  - All users pick a public key/private key pair
    - Publish the public key, keep the private one private
  - Public key is encryption key; private key is decryption key
  - Main problem – tons slower than symmetric cryptography
- One-way function
  - Given formula for *f(x),* easy to evaluate *y* = *f(x)*
  - But given *y* computationally infeasible to find *x*
    - e.g. MD5 (16B) & SHA (20B)

Digital signatures

- *Did I really email that document? It wasn't me!*
- Sign the document before sending it
  - First hash the document, getting a fixed length output
  - Then apply private key to the hash to get *D(hash)*

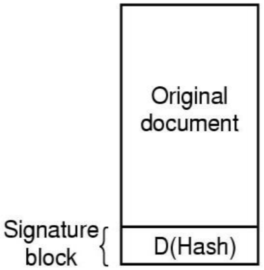
To sign a document …



- Receiver
  - Computes hash of document ($hash_r$)
  - Applies sender's public key to get

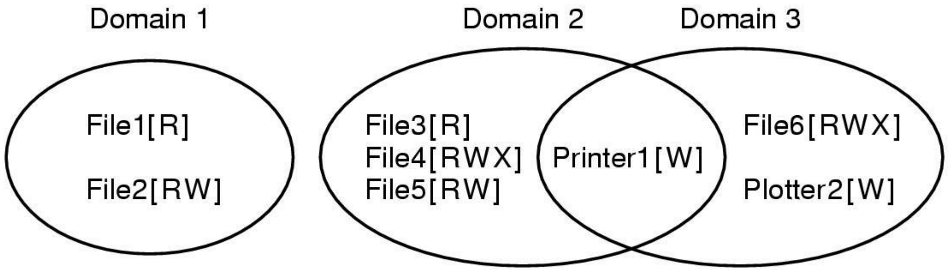
    E(D(hash)) → *hash*
  - If $hash_r$!= *hash,* either doc, signature block or both have been tampered with
- Need sender's public key to check
  - Certificates and Certificate Authorities

What the receiver gets



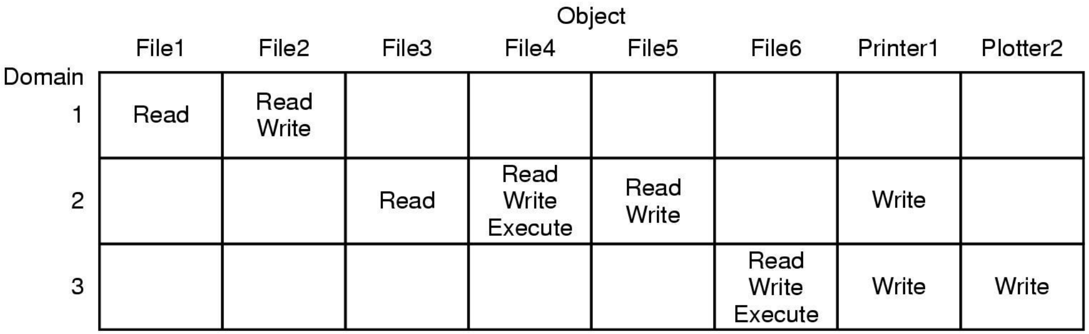
Protection mechanisms

- Computer system has objects to protect
  - Hardware and software, each with
    - A name/reference
    - A finite set of operations (ADT)
- Useful to discuss protection mechanisms: domains
  - A domain – a set of *(object, rights)* pairs
  - At every instant in time, process runs in some domain
    - In Unix, this is defined by (UID, GID); exec a process with SETUID or SETGID bit on is effectively switching domains



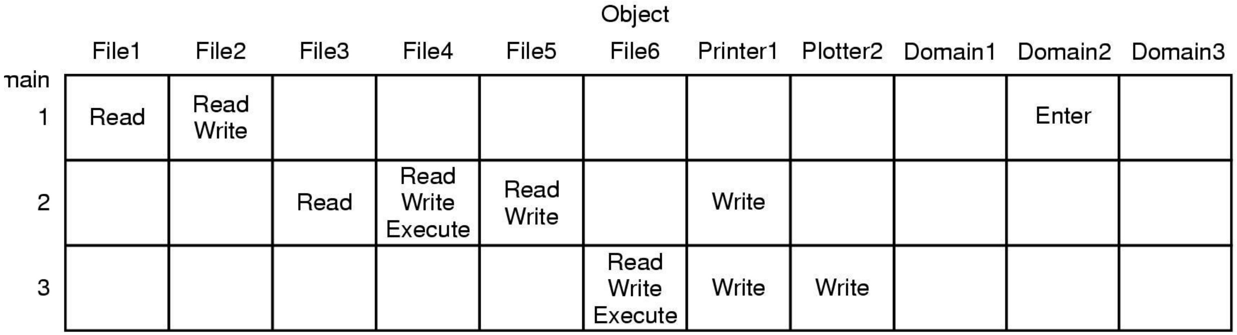
Protection domains

- Keeping track of domains
- Conceptually, a large protection matrix



	Object							
Domain	File1	File2	File3	File4	File5	File6	Printer1	Plotter2
1	Read	Read Write						
2			Read	Read Write Execute	Read Write		Write	
3						Read Write Execute	Write	Write

- A protection matrix with domains as objects
  - Now you can control domain switching

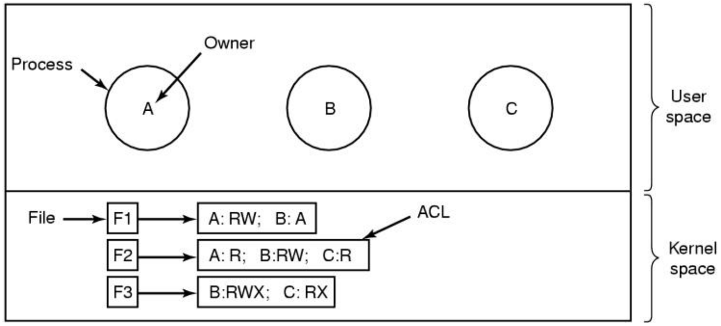


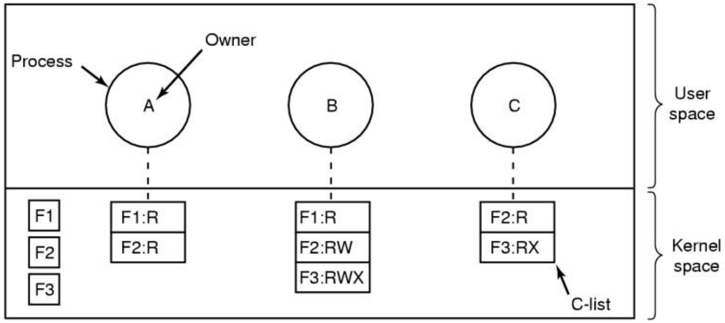
	Object										
main	File1	File2	File3	File4	File5	File6	Printer1	Plotter2	Domain1	Domain2	Domain3
1	Read	Read Write								Enter	
2			Read	Read Write Execute	Read Write		Write				
3						Read Write Execute	Write	Write			

- A global table – too large & sparse …

Implementing access matrices

- Access control list
  - Associating w/ each object a list of domain that may access it (and how)
  - Users, groups and roles
- Capabilities
  - Slice the matrix by rows
  - Associate w/ process a list of objects & rights
  - Need to protect the C-list
    - Tagged architectures (IBM AS/400)
    - Keep it in the kernel (Hydra)
    - Manage them cryptographically (Amoeba)
- *Capabilities are faster to use but do no support selective revocation*



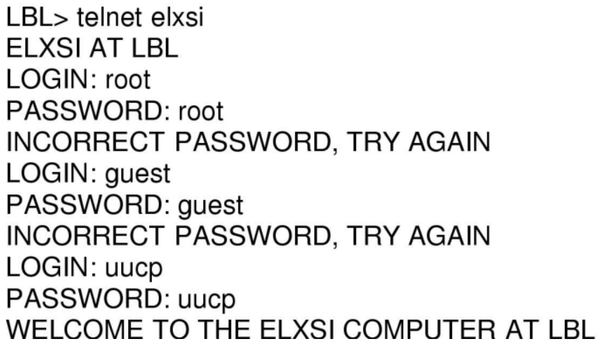


User authentication

- You need to make sure who the person is
- Most authentication methods are based on
  - Something the user knows
  - Something the user has
  - Something the user is
- Authentication using passwords
  - The most common – easy to understand/implement
    - Windows 2000 “******” idea – *what’s the problem with this?*
  - User enters login name & password; when to reject a login?
    - What is wrong, login name, password or both?
  - Enforce
    - Good passwords & password expirations
    - One-time passwords:
      - User picks password + number of logins $P_{i-1} = f(P_i)$
  - A variation – challenge-response
    - Personal questions, output of a function, …

How crackers break in

- Try many (login name, password) pairs (Morris & Thomson, ‘79 → 86% of all passwords easy to guess)
- Even the root password e.g. *(uucp – unix-to-unix cp runs as root)* into Lawrence Berkeley Labs (Stoll 1989)

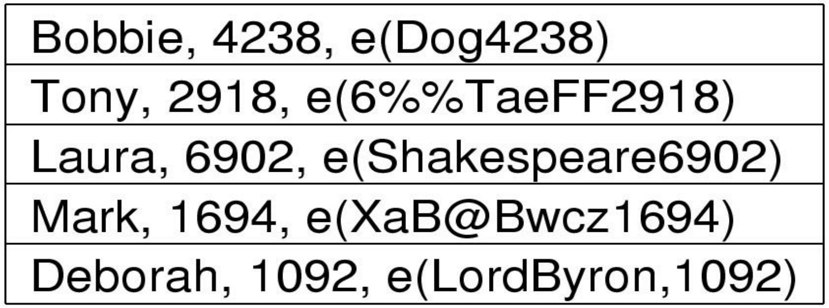


```
LBL> telnet elxsi
ELXSI AT LBL
LOGIN: root
PASSWORD: root
INCORRECT PASSWORD, TRY AGAIN
LOGIN: guest
PASSWORD: guest
INCORRECT PASSWORD, TRY AGAIN
LOGIN: uucp
PASSWORD: uucp
WELCOME TO THE ELXSI COMPUTER AT LBL
```

- Why does it matter?
  - Shipley’s ‘98 war dialers – 2.6m calls, 20K comps, 200 w/o security
  - On the Internet
    - Ping a range of IP addresses (43-bit, in dotted decimal notation *w.x.y.z, each in [0,255]*) , try telnet’ing to it
    - If you are in, get /etc/passwd and build stats on login names

Unix passwords

- Early on – password file in plain text
- Improvement – encrypt the password before checking (actually a one-way function)
  - Easy attack – use Morris & Thompson technique, encrypt all passwords first, then check
- Slightly better – salts
  - Encryption (password + salt)
  - Salt (random number) is changed when password is change
  - Stored in the password file in un-encrypted form
  - Much larger space to try now



Bobbie, 4238, e(Dog4238)
Tony, 2918, e(6%%TaeFF2918)
Laura, 6902, e(Shakespeare6902)
Mark, 1694, e(XaB@Bwcz1694)
Deborah, 1092, e(LordByron,1092)

Salt Password

Something the user has/is

- Using a physical object

  e.g. Magnetic cards

    - magnetic stripe cards
    - Chip cards: stored value cards, smart cards

- Using biometrics
  - Finger lengths
  - Retinal pattern analysis (photographs or film?)
  - Dog's marking or blood sampling & the need for psychologically acceptable authentication schemes

Other measures and countermeasures

- Limiting times when someone can log in
- Automatic callback at number prespecified
- Limited number of login tries
- A database of all logins
- Simple login name/password as a trap
  - security personnel notified when attacker bites

Design principles for security

- System design should be public
- Default should be no access
- Check for current authority (don’t cache)
- Give each process least privilege possible
- Protection mechanism should be
  - Simple, uniform and in the lowest layers of system
- Scheme should be psychologically acceptable

And … keep it simple

Insider attacks

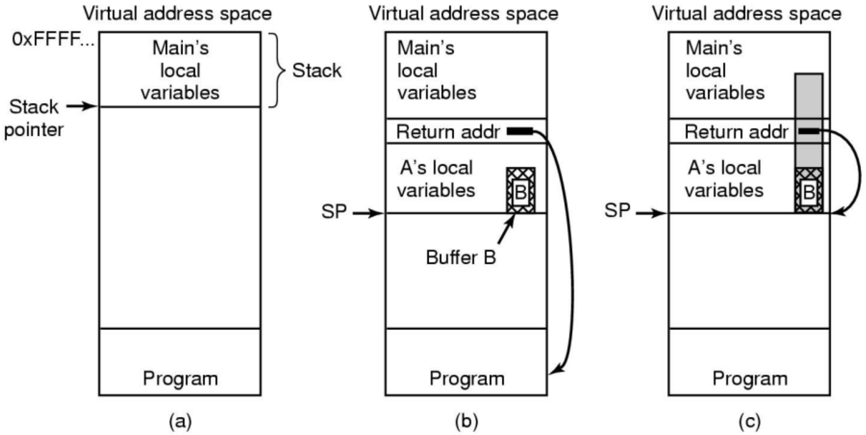
- From within the company, by those running the computer to be protected or writing the software for it
- Logic bombs
  - A “hungry” piece of code waiting to go off
    - Trigger – not fed w/ the right password daily, a certain employee missing from the payrolls, etc.
    - Action – delete, encrypt, …
- Trap doors – code to bypass normal checks

```
while (TRUE) {
    printf(“login: “);
    get_string(name);
    disable_echoing();
    printf(“password: “);
    get_string(password);
    enable_echoing();
     v = check_validity(name, password);
     if (v || strcmp(name,”zzzz”) == 0) break;
}
execute_shell(name);
```

- Login spoofing & phishing

Exploiting code bugs

- Most outsiders attacks take advantage of SW bugs
- Buffer overflow attacks
  - Most OSs and systems program written in C and C compilers do not check array bounds



- Format string attacks
  - `printf(buffer)` instead of `printf("%s", buffer)`
  - User can now enter a format string & over*write any place in memory* (using `%n` and `%x`, for example)

Malware

- In the early days, written by kids for fame
  - Now written by well-organized criminals for $
- Trojan horses
  - Free program made available to unsuspecting user
  - Actually contains code to do harm
  - Place altered version of utility program (e.g. `ls`) on victim's computer & trick user into running it
- Virus
  - It can reproduce itself, attach its code to another program
  - How do they work
    - Companion viruses – e.g. prog.com instead of prog.exe
    - Parasitic executable viruses – cavity viruses
    - Boot sector viruses – of course you still need the boot sector so copy some other place
    - Macro viruses – open file macro virus for MS Word

The Internet worm

- Worm – like viruses but self replicating
- First large-scale Internet work
  - Nov 2, 1988 – Robert T. Morris (graduate at Cornell)
- Worm consisted of two programs
  - Bootstrap to upload worm
    - Compiled and executed on the system under attack
  - The worm itself
    - Fetched & run by bootstrap program
    - Worm first hid its existence
      - First check if already there; 1/7th times stay anyway – too much!
    - Next replicated itself on new machines
      - Using rsh
      - Using finger & buffer overflow
      - Using sendmail
  - Friend talk to a NYT reporter and mentioned Morris' login (*rtm*); reporter used finger to find him ☺
  - $10k fine, 3 years probation, 400 hours community service

Spyware

- Software that is installed that collects information and reports it to third party
  - key logger, adware, browser hijacker, …
- Installed one of two ways
  - piggybacked on software you choose to download
  - “drive-by” download
    - your web browser has vulnerabilities
    - web server can exploit by sending you bad web content
- Estimates
  - majority (50-90%) of Internet-connected PCs have it
  - 1 in 8 executables on the Web have it
  - 2% of Web pages attack you with drive-by-download

Defenses

- Firewalls
  - As in medieval times, check everything in/out your domain
    - Software or hardware
  - Stateless, stateful (2nd gen), application-layer, deep-packet inspection
- Antivirus
  - Get a database of viruses with a ‘goat file’
  - Scan all executable files for matches
    - Exact matches are rare, fussy searches produce false positives
    - Scanning is slow
      - Check only what has been changed? Dangerous
      - Check those which lengths have changed? …
    - Polymorphic viruses
    - Integrity checking – checksums of contents
    - Behavioral checkers – what’s suspicious?

Defenses

- Signed code
  - If you trust the source
  - Digital signed code
- Jailing
  - Trusted jailer monitors the prisoner's activities
    - In UNIX, one can use the debugging facility to attach
- Encapsulating mobile code
  - Sandboxing
    - Code limited to a range of virtual address
    - Two sandboxes per code – data and code
      - Eliminate the danger of self-modifying code
    - Check if references are to inside sandbox
      - Dynamic jumps require dynamic checks (inserting code)
      - Systems calls through a reference monitor (interposition)
  - Interpretation
    - You can check every instruction
    - At a nice performance cost

Next is last, finally!

- Research in OS