CS162
Operating Systems and Systems Programming
Lecture 26

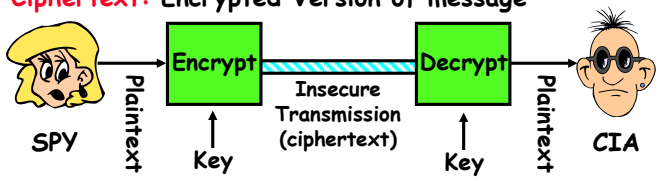
Protection and Security in Distributed Systems II

May 7, 2008

Prof. Anthony D. Joseph
http://inst.eecs.berkeley.edu/~cs162

Review: Private Key Cryptography

- Private Key (Symmetric) Encryption:
  - Single key used for both encryption and decryption
- Plaintext: Unencrypted Version of message
- Ciphertext: Encrypted Version of message



- Important properties
  - Can't derive plain text from ciphertext (decode) without access to key
  - Can't derive key from plain text and ciphertext
  - As long as password stays secret, get both secrecy and authentication
- Symmetric Key Algorithms: DES, Triple-DES, AES

Review: Public Key Encryption

- Key distribution without an authentication server
- Public Key Details
  - Don't have one key, have two: K_{public}, $K_{private}$
    - » Two keys are mathematically related to one another
    - » Really hard to derive K_{public} from $K_{private}$ and vice versa
  - Forward encryption:
    - » Encrypt: $(\text{cleartext})^{K_{public}} = \text{ciphertext}_1$
    - » Decrypt: $(\text{ciphertext}_1)^{K_{private}} = \text{cleartext}$
  - Reverse encryption:
    - » Encrypt: $(\text{cleartext})^{K_{private}} = \text{ciphertext}_2$
    - » Decrypt: $(\text{ciphertext}_2)^{K_{public}} = \text{cleartext}$
  - Note that $\text{ciphertext}_1 \neq \text{ciphertext}_2$
    - » Can't derive one from the other!
- Public Key Examples:
  - RSA: Rivest, Shamir, and Adleman
    - » K_{public} of form (k_{public}, N), $K_{private}$ of form $(k_{private}, N)$
    - » N = pq. Can break code if know p and q
  - ECC: Elliptic Curve Cryptography

Recall: Authorization: Who Can Do What?

- How do we decide who is authorized to do actions in the system?
- Access Control Matrix: contains all permissions in the system
  - Resources across top
    - » Files, Devices, etc…
  - Domains in columns
    - » A domain might be a user or a group of permissions
    - » E.g. above: User D_3 can read F_2 or execute F_3
  - In practice, table would be huge and sparse!

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

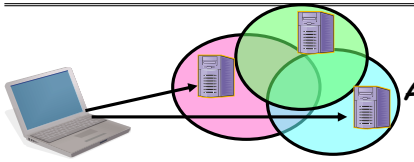
- Two approaches to implementation
  - Access Control Lists: store permissions with each object
    - » Still might be lots of users!
    - » UNIX limits each file to: r,w,x for owner, group, world
    - » More recent systems allow definition of groups of users and permissions for each group
  - Capability List: each process tracks objects has permission to touch
    - » Popular in the past, idea out of favor today
    - » Consider page table: Each process has list of pages it has access to, not each page has list of processes …

Goals for Today

- Distributed Authorization
- Enforcement
- Security Problems
  - Buffer Overflow
  - Morris Internet Worm
  - Password Checking
  - Self-Replicating Programs

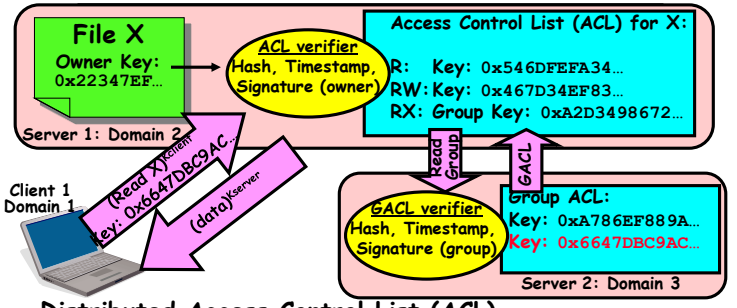
Note: Some slides and/or pictures in the following are adapted from slides ©2005 Silberschatz, Galvin, and Gagne. Many slides generated from my lecture notes by Kubiatowicz.

How to perform Authorization for Distributed Systems?



- Issues: Are all user names in world unique?
  - No! They only have small number of characters
    - » adj@mit.edu → adj@lcs.mit.edu → adj@cs.berkeley.edu
    - » However, someone thought their friend was adj@mit.edu and I got very private email intended for someone else…
  - Need something better, more unique to identify person
- Suppose want to connect with any server at any time?
  - Need an account on every machine! (possibly with different user name for each account)
  - OR: Need to use something more universal as identity
    - » Public Keys! (Called "Principles")
    - » People *are* their public keys

Distributed Access Control



- Distributed Access Control List (ACL)
  - Contains list of attributes (Read, Write, Execute, etc) with attached identities (Here, we show public keys)
    - » ACLs signed by owner of file, only changeable by owner
    - » Group lists signed by group key
  - ACLs can be on different servers than data
    - » Signatures allow us to validate them
    - » ACLs could even be stored separately from verifiers

Analysis of Previous Scheme

- Positive Points:
  - Identities checked via signatures and public keys
    - » Client can't generate request for data unless they have private key to go with their public identity
    - » Server won't use ACLs not properly signed by owner of file
  - No problems with multiple domains, since identities designed to be cross-domain (public keys domain neutral)
- Revocation:
  - What if someone steals your private key?
    - » Need to walk through all ACLs with your key and change…!
    - » This is very expensive
  - Better to have unique string identifying you that people place into ACLs
    - » Then, ask Certificate Authority to give you a certificate matching unique string to your current public key
    - » Client Request: (request + unique ID)^{Cprivate}; give server certificate if they ask for it.
    - » Key compromise⇒must distribute "certificate revocation", since can't wait for previous certificate to expire.
  - What if you remove someone from ACL of a given file?
    - » If server caches old ACL, then person retains access!
    - » Here, cache inconsistency leads to security violations!

Analysis Continued

- Who signs the data?
  - Or: How does the client know they are getting valid data?
  - Signed by server?
    - » What if server compromised? Should client trust server?
  - Signed by owner of file?
    - » Better, but now only owner can update file!
    - » Pretty inconvenient!
  - Signed by group of servers that accepted latest update?
    - » If must have signatures from all servers ⇒ Safe, but one bad server can prevent update from happening
    - » Instead: ask for a threshold number of signatures
    - » Byzantine agreement can help here
- How do you know that data is up-to-date?
  - Valid signature only means data is valid older version
  - Freshness attack:
    - » Malicious server returns old data instead of recent data
    - » Problem with both ACLs and data
    - » E.g.: you just got a raise, but enemy breaks into a server and prevents payroll from seeing latest version of update
  - Hard problem
    - » Needs to be fixed by invalidating old copies or having a trusted group of servers (Byzantine Agreement?)

Enforcement

- Enforcer checks passwords, ACLs, etc
  - Makes sure that only authorized actions take place
  - Bugs in enforcer ⇒ things for malicious users to exploit
- In UNIX, superuser can do anything
  - Because of coarse-grained access control, lots of stuff has to run as superuser in order to work
  - If there is a bug in any one of these programs, you lose!
- Paradox
  - Bullet-proof enforcer
    - » Only known way is to make enforcer as small as possible
    - » Easier to make correct, but simple-minded protection model
  - Fancy protection
    - » Tries to adhere to principle of least privilege
    - » Really hard to get right
- Same argument for Java or C++: What do you make private vs public?
  - Hard to make sure that code is usable but only necessary modules are public
  - Pick something in middle? Get bugs and weak protection!

State of the World

- State of the World in Security
  - Authentication: Encryption
    - » But almost no one encrypts or has public key identity
  - Authorization: Access Control
    - » But many systems only provide very coarse-grained access
    - » In UNIX, need to turn off protection to enable sharing
  - Enforcement: Kernel mode
    - » Hard to write a million line program without bugs
    - » Any bug is a potential security loophole!
- Some types of security problems
  - Abuse of privilege
    - » If the superuser is evil, we're all in trouble/can't do anything
    - » What if sysop in charge of instructional resources went crazy and deleted everybody's files (and backups)???
  - Imposter: Pretend to be someone else
    - » Example: in unix, can set up an .rhosts file to allow logins from one machine to another without retyping password
    - » Allows "rsh" command to do an operation on a remote node
    - » Result: send rsh request, pretending to be from trusted user→install .rhosts file granting you access

Involuntary Installation

- What about software loaded without your consent?
  - Macros attached to documents (such as Microsoft Word)
  - Active X controls (programs on web sites with potential access to whole machine)
  - Spyware included with normal products
- Active X controls can have access to the local machine
  - Install software/Launch programs
- Sony Spyware [Sony XCP] (October 2005)
  - About 50 recent CDs from Sony automatically install software when you played them on Windows machines
    - » Called XCP (Extended Copy Protection)
    - » Modify operating system to prevent more than 3 copies and to prevent peer-to-peer sharing
  - Side Effects:
    - » Reporting of private information to Sony
    - » Hiding of generic file names of form $sys_xxx; easy for other virus writers to exploit
    - » Hard to remove (crashes machine if not done carefully)
  - Vendors of virus protection software declare it spyware
    - » Computer Associates, Symantec, even Microsoft

Administrivia

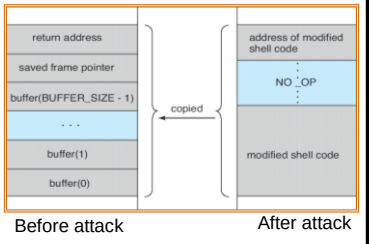
- No office hours on Tue 5/13
- Project #4 code deadline is Wed 5/14 at 11:59pm
- Final Exam
  - May 21st, 12:30-3:30pm
  - Exam will be comprehensive
  - Closed book, notes, slides
  - One 2-sided cheat sheet allowed
- Final Topics: Any suggestions?
  - Please send them to me by Thursday…

Other Security Problems

- Virus:
  - A piece of code that attaches itself to a program or file so it can spread from one computer to another, leaving infections as it travels
  - Most attached to executable files, so don't get activated until the file is actually executed
  - Once caught, can hide in boot tracks, other files, OS
- Worm:
  - Similar to a virus, but capable of traveling on its own
  - Takes advantage of file or information transport features
  - Because it can replicate itself, your computer might send out hundreds or thousands of copies of itself
- Trojan Horse:
  - Named after huge wooden horse in Greek mythology given as gift to enemy; contained army inside
  - At first glance appears to be useful software but does damage once installed or run on your computer

Security Problems: Buffer-overflow Condition

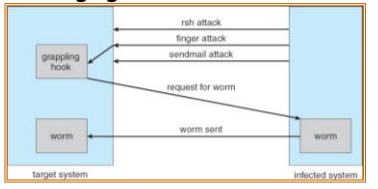
```
#define BUFFER_SIZE 256
int process(int argc,
            char *argv[])
{
  char buffer[BUFFER_SIZE];
  if (argc < 2)
       return -1;
  else {
       strcpy(buffer,argv[1]);
       return 0;
  }
}
```



- Technique exploited by many network attacks
  - Anytime input comes from network request and is not checked for size
  - Allows execution of code with same privileges as running program – but happens without any action from user!
- How to prevent?
  - Don't code this way! (ok, wishful thinking)
  - New mode bits in Intel, Amd, and Sun processors
    - » Put in page table; says "don't execute code in this page"

The Morris Internet Worm

- Internet worm (Self-reproducing)
  - Author Robert Morris, a first-year Cornell grad student
  - Launched close of workday on November 2, 1988
  - Within a few hours of release, it consumed resources to the point of bringing down infected machines



- Techniques
  - Exploited UNIX networking features (remote access)
  - Bugs in *finger* (buffer overflow) and *sendmail* programs (debug mode allowed remote login)
  - Dictionary lookup-based password cracking
  - Grappling hook program uploaded main worm program

Some other Attacks

- Trojan Horse Example: Fake Login
  - Construct a program that looks like normal login program
  - Gives "login:" and "password:" prompts
    - » You type information, it sends password to someone, then either logs you in or says "Permission Denied" and exits
  - In Windows, the "ctrl-alt-delete" sequence is supposed to be really hard to change, so you "know" that you are getting official login program
- Salami attack: Slicing things a little at a time
  - Steal or corrupt something a little bit at a time
  - E.g.: What happens to partial pennies from bank interest?
    - » Bank keeps them! Hacker re-programmed system so that partial pennies would go into his account.
    - » Doesn't seem like much, but if you are large bank can be millions of dollars
- Eavesdropping attack
  - Tap into network and see everything typed
  - Catch passwords, etc
  - Lesson: never use unencrypted communication!

Tenex Password Checking

- Tenex – early 70's, BBN
  - Most popular system at universities before UNIX
  - Thought to be very secure, gave "red team" all the source code and documentation (want code to be publicly available, as in UNIX)
  - In 48 hours, they figured out how to get every password in the system
- Here's the code for the password check:

```
for (i = 0; i < 8; i++)
  if (userPasswd[i] != realPasswd[i])
    go to error
```

- How many combinations of passwords?
  - 256^8?
  - Wrong!

Defeating Password Checking

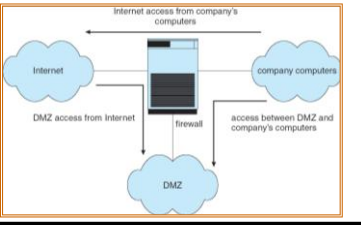
- Tenex used VM, and it interacts badly with the above code
  - Key idea: force page faults at inopportune times to break passwords quickly
- Arrange 1st char in string to be last char in pg, rest on next pg
  - Then arrange for pg with 1st char to be in memory, and rest to be on disk (e.g., ref lots of other pgs, then ref 1st page)

```
          a|aaaaaa
           |
page in memory| page on disk
```

- Time password check to determine if first character is correct!
  - If fast, 1st char is wrong
  - If slow, 1st char is right, pg fault, one of the others wrong
  - So try all first characters, until one is slow
  - Repeat with first two characters in memory, rest on disk
- Only 256 * 8 attempts to crack passwords
  - Fix is easy, don't stop until you look at all the characters

Defense in Depth: Layered Network Security

- How do I minimize the damage when security fails?
  - For instance: I make a mistake in the specification
  - Or: A bug lets something run that shouldn't?
- Firewall: Examines every packet to/from public internet
  - Can disable all traffic to/from certain ports
  - Can route certain traffic to DMZ (De-Militarized Zone)
    - » Semi-secure area separate from critical systems
  - Can do network address translation
    - » Inside network, computers have private IP addresses
    - » Connection from inside→outside is translated
    - » E.g. [10.0.0.2,port 2390] → [169.229.60.38,port 80]
      [12.4.35.2,port 5592] → [169.229.60.38,port 80]



Shrink Wrap Software Woes

- Can I trust software installed by the computer manufacturer?
  - Not really, most major computer manufacturers have shipped computers with viruses
  - How?
    - » Forgot to update virus scanner on "gold" master machine
- Software companies, PR firms, and others routinely release software that contains viruses
- Linux hackers say "Start with the source"
  - Does that work?

Ken Thompson's self-replicating program

- Bury Trojan horse in binaries, so no evidence in source
  - Replicates itself to every UNIX system in the world and even to new UNIX's on new platforms. No visible sign.
  - Gave Ken Thompson ability to log into any UNIX system
- Two steps: Make it possible (easy); Hide it (tricky)
- Step 1: Modify login.c

```
A: if (name == "ken")
      don't check password
      log in as root
```

  - Easy to do but pretty blatant! Anyone looking will see.
- Step 2: Modify C compiler
  - Instead of putting code in login.c, put in compiler:

```
B: if see trigger1
     insert A into input stream
```

  - Whenever compiler sees trigger1 (say /*gobbledygook*/), puts A into input stream of compiler
  - Now, don't need A in login.c, just need trigger1

Self Replicating Program Continued

- Step 3: Modify compiler source code:

```
C: if see trigger2
     insert B+C into input stream
```

  - Now compile this new C compiler to produce binary
- Step 4: Self-replicating code!
  - Simply remove statement C in compiler source code and place "trigger2" into source instead
    - » As long as existing C compiler is used to recompile the C compiler, the code will stay into the C compiler and will compile back door into login.c
    - » But no one can see this from source code!
- When porting to new machine/architecture, use existing C compiler to generate cross-compiler
  - Code will migrate to new architecture!
- Lesson: never underestimate the cleverness of computer hackers for hiding things!

Conclusion

- Authorization
  - Abstract table of users (or domains) vs permissions
  - Implemented either as access-control list or capability list
- Issues with distributed storage example
  - Revocation: How to remove permissions from someone?
  - Integrity: How to know whether data is valid
  - Freshness: How to know whether data is recent
- Buffer-Overrun Attack: exploit bug to execute code
- Want to learn more about security? Take CS 161