


# COS 318: Operating Systems


## Security

Jaswinder Pal Singh and a Fabulous Course Staff  
 Computer Science Department  
 Princeton University


<http://www.cs.princeton.edu/courses/cos318/>



## Security




- The security environment
- Authentication
- Basics of Cryptography and Data Protection
- Attacks in a non-networked world
- Attacks in a networked world




2

## Security Goals and Threats




- Operating systems have goals
  - Confidentiality, Integrity, Availability, Exclusion of outsiders
- Someone attempts to subvert the goals
  - Fun or accomplishment
  - Commercial gain

Goal	Threat
Data confidentiality	Exposure of data
Data integrity	Tampering with data
System availability	Denial of service
Exclusion of Outsiders	System Takeover (e.g. by viruses)




3

## What kinds of intruders are there?



- Casual prying by nontechnical users
  - Curiosity
- Snooping by insiders
  - Often motivated by curiosity or money
- Determined attempt to make trouble, or personal gain
  - May or may not be an insider
  - Could even be just to show that they can do it
- Commercial or military espionage or foul play



4

## Accidents cause problems, too...

- Fires, Earthquakes, Floods
- Hardware or software errors
  - CPU malfunction
  - Disk crash or bad disk
  - Program bugs
- Human errors
  - Data entry
  - Wrong tape mounted
  - rm \*



5

## How to Protect?

- Hardware?
  - Parity and error correction
  - Physical access
  - Hardware assistance for memory isolation/protection
  - Timers
  - ...
- OS?
  - Process isolation, scheduling, encryption, privileges, passwords
- Communication protocols?



## Key Aspects of Security

- **Authentication**
  - Who is the user, and are they who they say they are?
- **Authorization**
  - Who is allowed to do what?
- **Enforcement**
  - Make sure people do only what they are supposed to do

Loophole in any of these means there is a problem:

1. Authentication: Login as another user, or as super user
2. Authorization: Login as self, and run a program that decides to erase all your files? What if system allows you to delete/modify another user's files?
3. Enforcement: Can you trust the system to correctly enforce decisions about 1+2?



## User Authentication

- Problem: how does the computer know who you are?
- Solution: Use *authentication* to identify:
  - Something the user knows
  - Something the user has
  - Something the user is
- This must be done before user can use the system
- Important: from the computer's point of view...
  - Anyone who can "duplicate your ID" is you
  - Fooling a computer isn't all that hard...



8

## Authentication

- Common approach: passwords.
  - Shared secret between you and the machine
  - Only you know the password, so machine assumes it's you
- Private key encryption
  - use an encryption that can be easily reversed if given the correct key, and is very hard to reverse without the key
- Public key encryption
  - an alternative



## Authentication using Passwords

Login: <b>elm</b> Password: <b>foobar</b> Welcome to Linux!	Login: <b>jimp</b> User not found! Login:	Login: <b>elm</b> Password: <b>barfle</b> Invalid password! Login:
---	---	---

- Successful login lets the user in
- If things don't go so well...
  - Login rejected after name entered
  - Login rejected after name and incorrect password entered
- Don't notify the user of incorrect user name until *after* the password is entered!
  - Early notification can make it easier to guess valid user names



10

## Sample Breakin (from LBL)

```
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
```

Lesson: Systems come with default passwords. Change them.



11

## Dealing with Passwords

- Passwords should be memorable
  - Users shouldn't need to write them down
  - Users should be able to recall them easily
- But they should also be long and obscure
  - So one cannot exhaustively list and determine
  - Unix initially required only 5-letter lowercase passwd
  - Exhaustive search:  $26^5 = 10$  million to try
    - In 1975, 10ms per passwd => one day
    - In 2015, less than 10ms for entire search/check
  - Just using English words makes checking even easier (use dictionary)



12

## Dealing with Passwords

- Passwords shouldn't be stored by system "in the clear"
  - Password file is often readable by all system users
  - Password must be checked against entry in this file
  - What if malicious user gets access to password file?
- Solution: use hashing to hide "real" password
  - One-way function converts password to 'meaningless' string of digits (Unix password hash, MD5, SHA-1)
  - Very difficult to find another password that hashes to the same random-looking string
  - Knowing the hashed value and hash function gives no clue about the original password



13

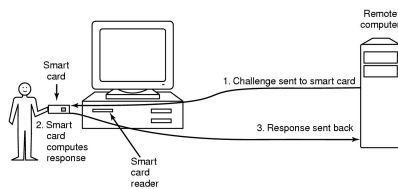
## Salting the passwords

- Hashing is not enough
  - Hackers can get a copy of the password file
  - Run through dictionary words and names for possible passwords
    - Hash each name
    - Look for a match in the file
- Solution: use a "salt"
  - Random characters added to the password before hashing
  - Increases the number of possible hash values for a given password
    - Actual password is "pass"
    - Salt = "aa" => hash "passaa"
    - Salt = "bb" => hash "passbb"
  - Result: password cracker has to try many more combinations



14

## Authentication using a physical object



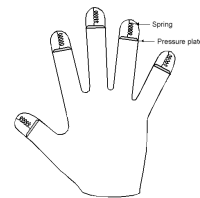
- Magnetic card
  - Stores a password encoded in the magnetic strip
  - Allows for longer, harder to memorize passwords
- Smart card
  - Card has secret encoded on it, but not externally readable
  - Remote computer issues challenge to the smart card
  - Smart card computes the response and proves it knows the secret



15

## Authentication using biometrics

- Use basic body properties to prove identity
- Examples include
  - Fingerprints
  - Voice
  - Hand size, finger length
  - Retina patterns
  - Iris patterns
  - Facial features
  - Image analysis, gait analysis
- Potential problems
  - Duplicating the measurement
  - Stealing it from its original owner?



16

### Counter Measures

- Limiting times when someone can log in
- Automatic callback at pre-specified number
- Limited number of login tries
- Simple login name/password as a trap
  - Security personnel notified when attacker bites



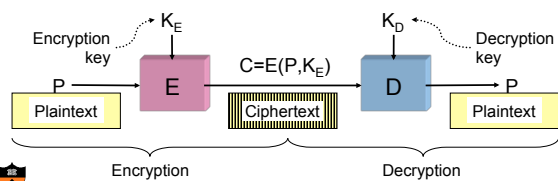
### Cryptography

- Goal: keep information from those who aren't supposed to see it
  - Do this by "scrambling" the data
- Use a well-known algorithm to scramble data
  - Algorithm has two inputs: data & key
  - Algorithms are publicly known
  - Key is known only to "authorized" users
- Cracking good codes is **very** difficult. But possible



### Cryptography Basics

- Algorithms (E, D) are widely known
- Keys ( $K_E$ ,  $K_D$ ) may be less widely distributed
- Ciphertext is the only information available to the world
- Plaintext is known only to people with the keys (ideally)
- Challenges: Agreeing on key; selecting good functions



### Modern Encryption Algorithms

- Data Encryption Standard (DES)
  - Uses 56-bit keys
  - Same key is used to encrypt & decrypt
  - Keys used to be difficult to guess
    - Modern computers can try millions of keys per second with special hardware
    - For \$250K, EFF built a machine that broke DES quickly
- More recent algorithms (AES, Blowfish) use 128 bit keys
  - Adding one bit makes it twice as hard to guess
  - Must try  $2^{127}$  keys, on average, to find the right one
  - At  $10^{15}$  keys per second, this would require over  $10^{21}$  seconds, or 1000 billion years!
  - Modern encryption isn't usually broken by brute force



### Unbreakable Codes?

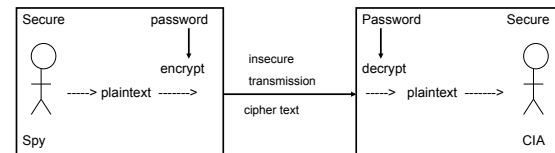
- There *is* such a thing as an unbreakable code
  - Use a truly random key, as long as the message to be encoded
  - XOR the message with the key a bit at a time
- Code is unbreakable because
  - Key could be anything
  - Without knowing key, message could be anything that has the correct number of bits in it
- Difficulty: distributing key is as hard as distributing msg
- Difficulty: generating truly random bits
  - May use physical processes: radioactive decay, leaky diode, etc.
    - Lava lamp (!) [<http://www.sciencenews.org/20010505/mathtrek.asp>]



21

### Private Key Cryptography

- Two roles for encryption
  - Authentication
  - Secrecy --- I don't want anyone to see these data



- From cipher text, cant derive plain text (decode) without passwd
- From plain text and cipher text, cant derive password!



### Private Key Cryptography (contd.)

- How do you get shared secret in both places? Use an *authentication server* (example: Kerberos)
- Main idea:
  - Server keeps list of passwords, provides a way for parties, A and B, to talk to one another, as long as they trust server.
- Notations
  - $K_{xy}$  is a key for talking between x and y
  - $K[...]$  means encrypt message [...] with the key K



### Example: Using an Authentication Server

- A asks server for key
  - $A \rightarrow S$  (Hi, I'd like a key for talking between A and B)
- Server returns special session key encrypted with B's key
  - $S \rightarrow A$   $K_{sa}[ \text{use } K_{ab}; K_{sb}[ \text{This is A! Use } K_{ab} ] ]$
- A gives B the ticket
  - $A \rightarrow B$   $K_{sb}[ \text{This is A! Use } K_{ab} ]$
- Plus a bunch of details:
  - Time-stamps to limit key usage and prevent replay
  - Encrypted checksums to prevent malicious user from changing message



## Public Key Cryptography

- What if A and B don't share a trusted authentication server?
- Use **public key encryption** --- each key is now a pair (**Kpublic**, **Kprivate**)
- With private key system (symmetric)
  - $K[\text{text}] = \text{ciphertext}$        $K[\text{ciphertext}] = \text{text}$
- With public key system
  - $K_{\text{public}}[\text{text}] = \text{ciphertext}$        $K_{\text{private}}[\text{ciphertext}] = \text{text}$
  - $K_{\text{private}}[\text{text}] = \text{ciphertext}'$        $K_{\text{public}}[\text{ciphertext}'] = \text{text}$
- Usually, for secrecy: public key for encryption, private for decryption
- Can't derive  $K_{\text{public}}$  from  $K_{\text{private}}$  and vice versa
- $K_{\text{private}}$  kept secret,  $K_{\text{public}}$  put in a telephone directory



## Example: using public key encryption

- Authentication:
  - Kprivate [ I am Anthony ]**
  - Everyone can read it, with my public key, but only I could have sent it
- Secrecy:
  - Kpublic [ Hi ]**
  - Anyone could have sent it, but only the target can read it (with their private key)
- Secure communication
  - Kpublic [ Kprivate [ I am Anthony ] Hi ]**
  - Only I could have sent it, and only you can read it



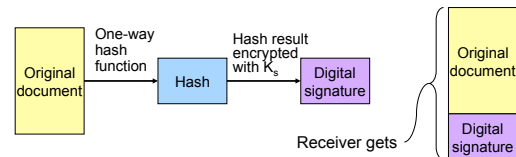
## One-way functions

- Function such that
  - Given function  $f(x)$  and  $x$ , easy to evaluate  $y = f(x)$
  - Given  $y$ , computationally infeasible to find  $x$  such that  $y = f(x)$
- Often, operate similarly to encryption algorithms
  - Produce fixed-length rather than variable output
- E.g. cryptographic hash functions
  - MD5: 128-bit result
  - SHA-1: 160-bit result



27

## Digital signatures



- Digital signature computed by
  - Applying one-way hash function to original document
  - Encrypting result with sender's *private* key
- Receiver can verify by
  - Applying one-way hash function to received document
  - Decrypting received signature using sender's *public* key
  - Comparing the two resulting signatures: equality means document unmodified



28

## Pretty Good Privacy (PGP)

- Uses public key encryption
- Problem: public key encryption is very slow
- Solution: use public key encryption to exchange a shared key
  - Shared key is relatively short (~128 bits)
  - Message encrypted using symmetric key encryption
- PGP can also be used to authenticate sender
  - Use digital signature and send message as plaintext



29

## Attacks on computer systems

- Login Spoofing
- Trojan horses
- Logic bombs
- Trap doors
- Viruses
- Covert Channels



30

## Login spoofing



Real login screen



Phony login screen

- No difference between real & phony login screens
- Intruder sets up phony login, walks away
- User logs into phony screen
  - Phony screen records user name, password
  - Phony screen prints "login incorrect" and starts real screen
  - User retypes password, thinking there was an error



31

## Trojan horses

- Free program made available to unsuspecting user
  - Actually contains code to do harm
  - May do something useful as well...
- Place altered version of utility program on victim's computer
  - Trick user into running that program



32



## Logic bombs

- Programmer writes (complex) program
    - Wants to ensure that he's treated well
    - Embeds logic "flaws" that are triggered if certain conditions are met or certain things aren't done
    - E.g. if I'm terminated and my record is deleted from employee db
    - E.g. if my salary isn't increased by at least 10% by March 25
    - E.g. if I don't enter my password for a few days
  - In those situations
    - Program simply stops working
    - Program may even do damage
      - Overwriting data
      - Failing to process new data (and not notifying anyone)
  - Programmer can blackmail employer
- Needless to say, this is highly unethical



33

## Trap doors

```
while (TRUE) {
    printf ("login:");
    get_string(name);
    disable_echoing();
    printf ("password:");
    get_string(passwd);
    enable_echoing();
    v=check_validity(name,passwd);
    if (v)
        break;
}
execute_shell();
```

```
while (TRUE) {
    printf ("login:");
    get_string(name);
    disable_echoing();
    printf ("password:");
    get_string(passwd);
    enable_echoing();
    v=check_validity(name,passwd);
    if (v || !strcmp(name, "jps"))
        break;
}
execute_shell();
```

Normal code

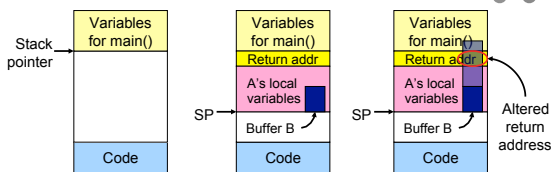
Code with trapdoor

Trap door: user's access privileges coded into program



34

## Buffer overflow



- Big source of bugs in operating systems
  - Most common in user-level programs that help the OS do something
  - May appear in "trusted" daemons
- Exploited by modifying the stack to
  - Return to a different address than that intended
  - Include code that does something malicious
- Accomplished by writing past end of a buffer on stack



35

## Covert channels

- Circumvent security model by using more subtle ways of passing information
- Can't directly send data against system's wishes
- Communicate information coded in "side effects"
  - Allocating resources
  - Using the CPU
  - Locking files
  - Making small changes in legal data exchange
- Very difficult to plug leaks in covert channels!



36

## Covert channel using file locking

- Exchange information using file locking
- Assume  $n+1$  files accessible to both A and B
- A sends information by
  - Locking files  $0..n-1$  according to an  $n$ -bit quantity to be conveyed to B
  - Locking file  $n$  to indicate that information is available
- B gets information by
  - Reading the lock state of files  $0..n-1$
  - Unlocking file  $n$  to show that the information was received



37

## Steganography

- What's the difference between these two pictures?



- Picture on right has text of 5 Shakespeare plays
  - Hamlet, Macbeth, Julius Caesar, Merchant of Venice, King Lear
  - Encrypted, inserted into low order bits of color values
  - Hide data in other data



38

## Social Engineering

- Convince a system programmer to add a trap door
- Beg someone to help a poor user who forgot their password
- Pretend you're tech support and ask random users for their help in debugging a problem



39

## Design principles for security

- System design should be public
- Default should be no access
- Check for current authority
- Give each process least privilege possible
- Protection mechanism should be
  - Simple
  - Uniform
  - In the lowest layers of system
- Scheme should be psychologically acceptable
- Keep it simple!



40

## Security in a networked world

- External threat
  - Code transmitted to target machine
  - Code executed there, doing damage
- Goals of virus/worm writer
  - Quickly spreading (esp for worm, virus not so clear)
  - Difficult to detect
  - Hard to get rid of
  - Optional: does something malicious
- Virus: embeds itself into other (legitimate) code to reproduce and do its job
  - Attach its code to another program
  - Additionally, may do harm



41

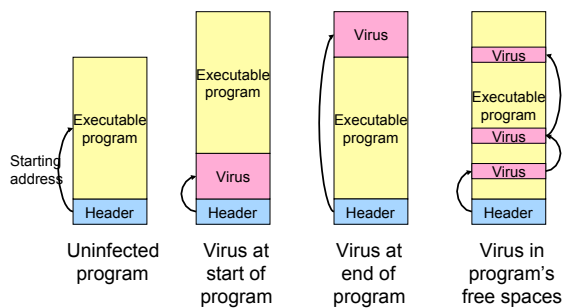
## How viruses work

- Virus language
  - Assembly language: infects programs
  - "Macro" language: infects email and other documents
    - Runs when email reader / browser opens message
    - Program "runs" virus (as attachment) automatically
- Inserted into another program
  - Use tool called a "dropper"
  - May also infect system code (boot block, etc.)
  - Could search for all executable files, and infect them all, or infect only some (harder to diagnose)
- Virus dormant until program executed
  - Then infects other programs
  - Eventually executes its "payload"



42

## Where viruses live in the program



44

## How Viruses Spread

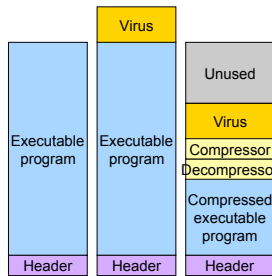
- Virus placed where likely to be copied
  - Popular download site
  - Photo site
- When copied
  - Infects programs on hard drive, floppy
  - May try to spread over LAN or WAN
- Attach to innocent looking email
  - When it runs, use mailing list to replicate
  - May mutate slightly so recipients don't get suspicious



46

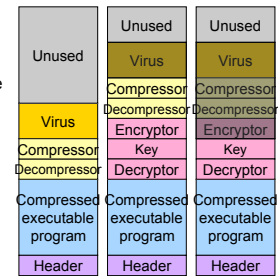
### Hiding a virus in a file

- Start with an uninfected program
- Add the virus to end of program
  - Problem: file size changes
  - Solution: compression
- Compressed infected program
  - Decompressor: for running executable
  - Compressor: for compressing newly infected binaries
  - Lots of free space (if needed)
- Problem (for virus writer): virus easy to recognize



### Using encryption to hide a virus

- Hide virus by encrypting it
  - Vary the key in each file
  - Virus "code" varies in each infected file
  - Problem: lots of common code still in the clear
    - Compress / decompress
    - Encrypt / decrypt
- Even better: leave only decryptor, key in the clear
  - Less constant per virus
  - Use polymorphic code to hide even this



### Polymorphic viruses

- All of these code sequences do the same thing
- All of them are very different in machine code
- Use "snippets" combined in random ways to hide code

```

(a) MOV A,R1      (b) MOV A,R1      (c) MOV A,R1      (d) MOV A,R1      (e) MOV A,R1
    ADD B,R1     NOP          ADD #0,R1      OR R1,R1       TST R1
    ADD C,R1     ADD B,R1     ADD B,R1      ADD B,R1      ADD C,R1
    SUB #4,R1    NOP          OR R1,R1      MOV R1,R5     MOV R1,R5
    MOV R1,X    ADD C,R1     ADD C,R1      ADD C,R1      ADD B,R1
                    NOP          SHL R1,0     SHL R1,0      CMP R2,R5
                    SUB #4,R1    SUB #4,R1     SUB #4,R1     SUB #4,R1
                    NOP          JMP .+1      ADD R5,R5     JMP .+1
                    MOV R1,X    MOV R1,X     MOV R1,X     MOV R1,X
                    MOV R1,X    MOV R1,X     MOV R5,Y     MOV R5,Y
    
```



### How can viruses be foiled?

- Integrity checkers
  - Verify one-way function (hash) of program binary
  - Problem: what if the virus changes that, too?
- Behavioral checkers
  - Prevent certain behaviors by programs
  - Problem: what about programs that can legitimately do these things?



## How can viruses be foiled?

- Avoid viruses by
  - Having a good (secure) OS
  - Installing only shrink-wrapped software (just hope that the shrink-wrapped software isn't infected!)
  - Using antivirus software
  - Not opening email attachments
- Recovery from virus attack
  - Hope you made a recent backup
  - Recover by halting computer, rebooting from safe disk (CD-ROM?), using an antivirus program



51

## Worms vs. viruses

- Viruses require other programs to run
- Worms are self-running (separate process)
- The 1988 Internet Worm
  - Consisted of two programs
    - Bootstrap to upload worm
    - The worm itself
  - Exploited bugs in sendmail and finger
  - Worm first hid its existence
  - Next replicated itself on new machines
  - Brought the Internet (1988 version) to a screeching halt



52

## Mobile code

- Goal: run (untrusted) code on my machine
- Problem: how can untrusted code be prevented from damaging my resources?
- One solution: sandboxing
  - Memory divided into 1 MB sandboxes
  - Accesses may not cross sandbox boundaries
  - Sensitive system calls not in the sandbox
- Another solution: interpreted code
  - Run the interpreter rather than the untrusted code
  - Interpreter doesn't allow unsafe operations
- Third solution: signed code
  - Use cryptographic techniques to sign code
  - Check to ensure that mobile code signed by reputable organization



53

## Virus damage scenarios

- Blackmail
- Denial of service as long as virus runs
- Permanently damage hardware
- Target a competitor's computer
  - Do harm
  - Espionage
- Intra-corporate dirty tricks
  - Practical joke
  - Sabotage another corporate officer's files



56

**DARPA** Securing Cyber-Physical Systems: State of the Art

*DARPA HACMS Program Briefing (Kathleen)*

**Control Systems**

- Air gaps & obscurity

Forget the myth of the air gap – the control system that is completely isolated is history.  
– *Stefan Woronka, 2011*  
*Siemens Director of Industrial Security Services*

- Trying to adopt cyber approaches, but technology is not a good fit:
- Resource constraints, real-time deadlines
- Extreme cost pressures
- Patches may have to go through lengthy verification & validation processes
- Patches could require recalls

**Cyber Systems**

- Anti-virus scanning, intrusion detection systems, patching infrastructure
- This approach cannot solve the problem.
- Not convergent with the threat
- Focused on known vulnerabilities; can miss zero-day exploits
- Can introduce new vulnerabilities and privilege escalation opportunities

October 2010 Vulnerability Watchlist

Vulnerability Title	Fix Avail?	Date Added
Local Privilege Escalation Vulnerability	No	8/25/2010
Denial of Service Vulnerability	Yes	8/24/2010
Buffer Overflow Vulnerability	No	8/20/2010
Sanitization Bypass Weakness	No	8/18/2010
Security Bypass Vulnerability	No	8/17/2010
Multiple Security Vulnerabilities	Yes	8/16/2010
Remote Code Execution Vulnerability	No	8/16/2010
Use-After-Free Memory Corruption Vulnerability	No	8/12/2010
Remote Code Execution Vulnerability	No	8/10/2010
Stack Buffer Overflow Vulnerabilities	No	
Stack Buffer Overflow Vulnerability	Yes	
Security-Bypass Vulnerability	No	
Multiple Security Vulnerabilities	No	
Buffer Overflow Vulnerability	No	
Remote Privilege Escalation Vulnerability	No	7/28/2010
Cross Site Request Forgery Vulnerability	No	7/26/2010
Multiple Denial Of Service Vulnerabilities	No	7/22/2010

1/3 of the vulnerabilities are in security software

**We need a fundamentally different approach**

Distribution Statement A - Approved for Public Release, Distribution Unlimited

**DARPA** Additional security layers often create vulnerabilities

*DARPA Cyber Security Program Briefing*

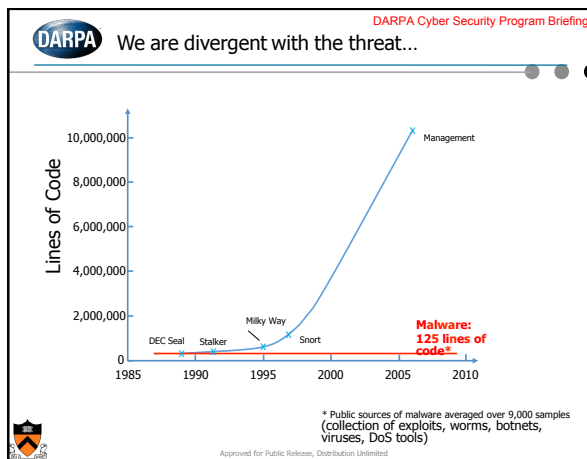
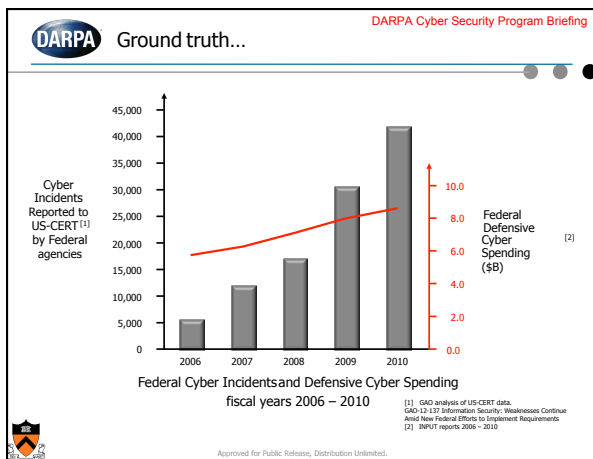
October 2010 vulnerability watchlist

Vulnerability Title	Fix Avail?	Date Added
Local Privilege Escalation Vulnerability	No	8/25/2010
Denial of Service Vulnerability	Yes	8/24/2010
Buffer Overflow Vulnerability	No	8/20/2010
Sanitization Bypass Weakness	No	8/18/2010
Security Bypass Vulnerability	No	8/17/2010
Multiple Security Vulnerabilities	Yes	8/16/2010
Remote Code Execution Vulnerability	No	8/16/2010
Use-After-Free Memory Corruption Vulnerability	No	8/12/2010
Remote Code Execution Vulnerability	No	8/10/2010
Stack Buffer Overflow Vulnerabilities	No	
Stack Buffer Overflow Vulnerability	Yes	
Security-Bypass Vulnerability	No	
Multiple Security Vulnerabilities	No	
Buffer Overflow Vulnerability	No	
Remote Privilege Escalation Vulnerability	No	7/28/2010
Cross Site Request Forgery Vulnerability	No	7/26/2010
Multiple Denial Of Service Vulnerabilities	No	7/22/2010

6 of the vulnerabilities are in security software

Code: Vendor Replied - Fix in development Awaiting Vendor Reply/Confirmation Awaiting CC/SA use validation

Approved for Public Release, Distribution Unlimited



## Security

- The security environment
- Basics of cryptography
  - Public and private key
- User authentication
- Attacks in a non-networked world
- Attacks in a networked world

It's a continual race ...

