Intro to security

You can't trust computers

- Goal: try to prevent their misuse
- Big picture
  - why do computers fail?
- Key Lesson
  - technical solution alone is insufficient
  - must consider how system will be developed, maintained, used
- Later
  - basics of authentication

The security mindset

Bruce Schneier

- Me: What an elegant solution!
- Schneier: I can send ants to anyone!

SMARTWATER®
DELIVERING PROVEN CRIME REDUCTION STRATEGIES

- A liquid containing a code which can be read under ultraviolet light.
- Applied to valuable items, so that if they are stolen and later seized by police, their original owner can be determined.
- Another application is a sprinkler system that sprays a burglar with the (invisible) fluid, which cannot be washed off and lasts for months, to generate evidence which connects a suspect to a specific location.

- Me: What an elegant solution!
- Schneier: It would be cool to paint mine on your valuables, and then call the police can send ants to anyone!

Closer to home

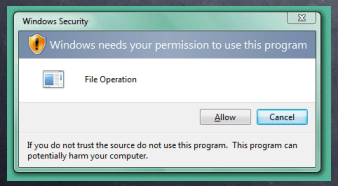
- Tenex (a Unix competitor)
  - we think of functions as black boxes
  - but they run on computers
- Tempest
  - we think of computers as things that process bits
  - but they are physical systems

Why cryptosystems fail

- Technologists like to focus on technical stuff, but most failures are non technical
  - top three reasons for ATM phantom withdrawals:
    - background noise
      - hard to keep below 1 error in 10^4 transactions
      - UK: before litigation, government claimed 1 in 1.5 x 106
    - postal interception
    - theft by bank staff
- Moral Hazard
  - hard to set the right incentives
    - Anderson: cost of failure should be born by party that can prevent/fix problem

Why cryptosystems fail

- End-to-end design
  - users, environment, other programmers
    - many problems come when one layer breaks assumptions of another layer
      - think through APIs
      - make it hard to misuse your layer and easy to use it correctly
    - "security through endless warning dialogs" does not work



- "Blend together into a giant "click here to get work done" button that nobody bothers to read anymore" Jeff Atwood

Why cryptosystems fail

- In design of security for ATM, engineers focused on technical attacks
  - break crypto
  - intercept & insert network messages
- And indeed these attacks happened

Why cryptosystems fail

- In design of security for ATM, engineers focused on technical attacks
  - break crypto
  - intercept & insert network messages
- And indeed these attacks happened
  - twice
    - a telephone engineer in Japan recorded customer card data and PINs from a phone line
    - technicians programmed a communications processor to send only positive authorizations to an ATM where accomplices where waiting
- Real reasons for ATM failures?

Moral Hazard: the ATM story

- Suppose you say you did not withdraw money from your account through an ATM
- The bank says you did
- Who wins?

1980: a NY court believes the customer!

- Federal Reserve then passes regulation requiring U.S. banks to refund disputed electronic transactions, unless they can prove customer is lying
- cameras appear everywhere on ATMs!

Moral Hazard: the ATM story

- Suppose you say you did not withdraw money from your account through an ATM
- The bank says you did
- Who wins?

Banks have managed to deny they were at fault; customers are lying!

- customers must show the bank is lying

Technical problems are hard too

- Fundamental asymmetry
  - Defender needs to find and fix all bugs
  - Attacker only needs to find and exploit one bug
- Try to cover every possible failure
  - systems becomes complex
  - implementation errors make system insecure
- Focus on common failures
  - missed attack can be exploited

Ken Thompson's Turing Award lecture

- Created with Dennis Ritchie UNIX operating system
- Created programming language "B"
  - guess what came next...
- 1983 Turing Award
  - Lecture: "Reflections on trusting trust"
- 1999 National Medal of Technology

Reflections on trusting trust

- "I am a programmer. On my 1040 form, that is what I put down as my occupation. As a programmer, I write programs. "
- "I would like to present to you the cutest program I ever wrote."

Trusting trust: some observations

- Stage I
  - A program can, when executed, output its own source code
- Stage II
  - A compiler can learn the meaning of a symbol
- Stage III
  - A compiler may (deliberately) output incorrect machine code

Stage I: A self-reproducing program

```
main() {
    char *s="main() { char *s=%c%s%c; printf(s,34,s,34); }"
printf(s,34,s,34);
}
```

Stage II: A learning compiler

- Somewhere inside a C compiler...

```
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
```

```
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return(11);
```

```
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return('\v');
```

- We wish to add the vertical tab (\v) symbol
- We return its ASCII value (11) if the symbol is \v
- We recompile our compiler, and we can now change our implementation to return \v

Stage III: a bugged compiler

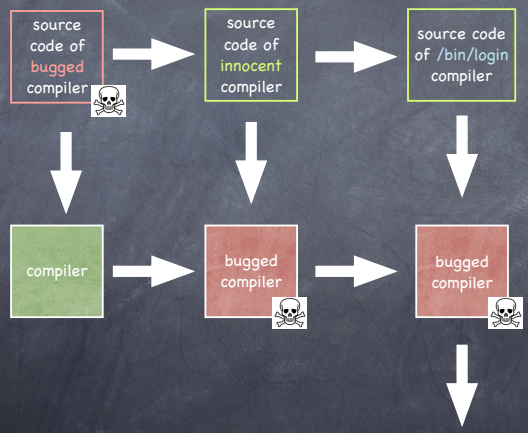
```
compile(s)
char *s;
{
        ...
}
```

```
compile(s)
char *s;
{
        if(match(s, "pattern")) {
                compile("bug");
                return;
        }
        ...
}
```

```
compile(s)
char *s;
{
        if(match(s, "pattern1")) {
                compile ("bug1");
                return;
        }
        if(match(s, "pattern 2")) {
                compile ("bug 2");
                return;
        }
        ...
}
```

- bug 2 is a self-reproducing program that inserts both trojan horses

What happens



BWAHAHAHAHA!

```
VMware ESX Server 3 (Dali)
Kernel 2.4.21-37.0.2.ELvmnix on an i686

localhost login: root
Password:
Last login: Tue Apr 17 22:06:17 on tty1
[root@localhost root]#
```

Moral

- "The moral is obvious. You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me.)"

Today, Ken Thompson works as a distinguished engineer for Google

Tenex

- Very popular at universities before Unix
- Thought very secure. Created a team trying to find loopholes
  - Ken Thompson: "I suspect that Daniel Bobrow would be here instead of me if he could not afford a PDP-10 and had had to settle for a PDP-11"
- Given all source code and documentation, and a normal account

Oops...

- In 48 hours, team had all passwords in the system
- Password check code

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

- Must try 26^8 combinations... secure?

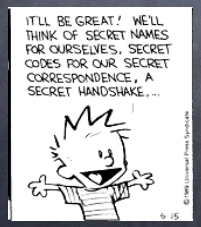
The joys of virtual memory

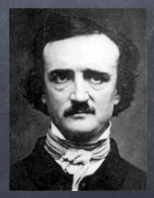
- Force page fault at carefully designed time to reveal password
  - put first character in string as last in a page
  - rest on next page
  - put page with first character in memory, rest on disk
  - Time how long it takes for password check
    - if fast, first character is wrong
    - if slow, page fault! first character is right!
  - Repeat

What do do?

- Robustness
  - minor error should lead to minor problems
  - multiple minor errors should still lead to minor problems
- Explicitness
  - list failure modes
  - show how failure modes addressed
  - state implementation plan (technical and managerial)
  - test spec; analyze failures and derive feedback

More on Security





It may well be doubted whether human ingenuity can construct an enigma of this kind (a cryptogram) which human ingenuity may not, with proper application, resolve

Security in the real world

- Security decisions based on:
  - Value, Locks, Police
- Some observations:
  - Not all locks are the same
    - People pay for security they need
    - Police are critical to the picture
    - Security is only as good as the weakest link

Security in Computer Systems

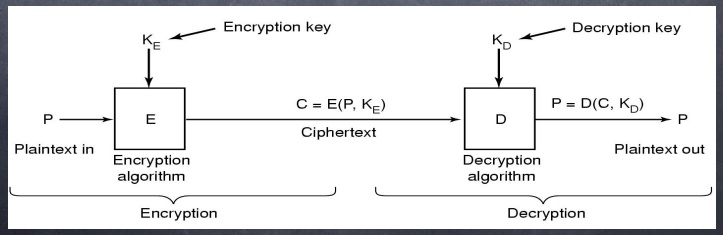
- In computer systems, this translates to:
  - Authorization
  - Authentication
  - Audit
- This is the Gold Standard for Security (Lampson)
- Some security goals:
  - Data confidentiality: secret data remains secret
  - Data integrity: no tampering of data
  - System availability: unable to make system unusable
  - Privacy: protecting from misuse of user's information

Security Threats

- Identified by Defense Science Board:
  - Incomplete, inquisitive and unintentional blunders.
  - Hackers driven by technical challenges.
  - Disgruntled employees or customers seeking revenge.
  - Criminals interested in personal financial gain or stealing services.
  - Organized crime with the intent of hiding something or financial gain.
  - Organized terrorist groups attempting to influence U.S. policy by isolated attacks.
  - Foreign espionage agents seeking to exploit information for economic, political, or military purposes.
  - Tactical countermeasures intended to disrupt specific weapons or command structures.
  - Multifaceted tactical information warfare applied in a broad orchestrated manner to disrupt a major U.S. military mission.
  - Large organized groups or nation-states intent on overthrowing the US

Cryptography Overview

- Encrypt data so it only makes sense to authorized users
  - Input data is a message or file called plaintext
  - Encrypted data is called ciphertext
- Encryption and decryption functions should be public
  - Security by obscurity is not a good idea!



Secret-Key Cryptography

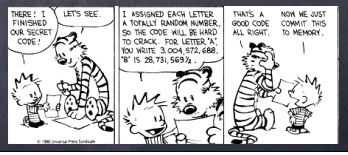
- Also called symmetric cryptography
  - Encryption algorithm is publicly known
  - E(message, key) = ciphertext D(ciphertext, key) = message
- Naïve scheme: monoalphabetic substitution
  - Plaintext : ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - Ciphertext: QWERTYUIOPASDFGHJKLZXCVBNM
  - So, attack is encrypted to: qzzqea
  - 26! possible keys ~ 4x1026 possibilities
    - 1 μs per permutation $\Rightarrow$ 10 trillion years to break
  - easy to break this scheme! How?
    - 'e' occurs 14%, 't' 9.85%, 'q' 0.26%

Symmetric Key Cryptography

- Which encryption algorithm is good?
  - DES was proposed in the 1970s
    - Encrypts 64 bits of data with 56 bit key to give 64-bit ciphertext
    - Uses 16 rounds of substitution and permutation
    - EFF invested $250000 to break DES message in 56 hours
    - DES made powerful by encrypting message 3 times (DES3)
  - Current standard is AES
    - A result of 3-year competition with entries from 12 countries
    - Winning entry was from Belgium, called 'Rijndael'
    - 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!
  - Strong algorithms, such as DES3, RC4 are used
    - WEP uses RC4

Unbreakable codes

- There is such a thing as an unbreakable code: one-time pad
  - use a truly random key as long as the message to be encoded
  - XOR the message with the key one bit at a time
- Code is unbreakable because
  - key could be anything
  - without key, message could be anything with the correct number of bits in it
- Difficulty: distributing key it is as hard as distributing message
- Difficulty: generating truly random bits
  - Cant use computer random gnerator!
  - May use physical processes
    - radioactive decay
    - lava lamps!

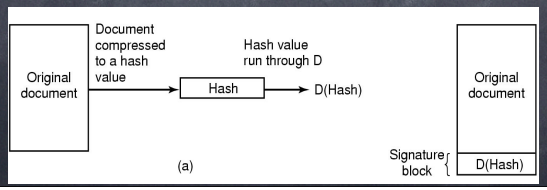


Public Key Cryptography

- Diffie and Hellman, 1976
- All users get a public key and a private key
  - Public key is published
  - Private key is not known to anyone else
- If Alice has a packet to send to Bob,
  - She encrypts the packet with Bob's public key
  - Bob uses his private key to decrypt Alice's packet
- Private key linked mathematically to public key
  - Difficult to derive by making it computationally infeasible (RSA)
- Pros: more security, convenient, digital signatures
- Cons: slower

Digital Signatures

- Hashing function hard to invert, e.g. MD5, SHA
- Apply private key to hash (decrypt hash)
  - Called signature block
- Receiver uses sender's public key on signature block
  - E(D(x)) = x should work (works for RSA)



Authentication

- Establish the identity of user/machine by
  - Something you know (password, secret)
  - Something you have (credit card, smart card)
  - Something you are (retinal scan, fingerprint)
- In the case of an OS this is done during login
  - OS wants to know who the user is
- Passwords: secret known only to the subject
  - Simplest OS implementation keeps (login, password) pair
  - Authenticates user on login by checking the password
  - Try to make this scheme as secure as possible!
    - Display the password when being typed? (Windows, UNIX)

Online passwords attacks

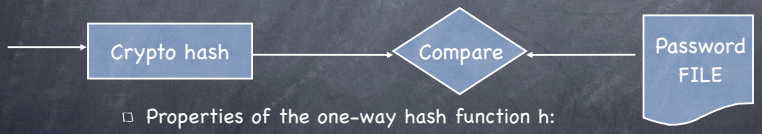
- Online attacks: system used to verify the guesses
  - How someone broke into LBL

```
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
```

- Thwart these attacks:
  - limit the number of guesses
  - better passwords

Offline password attacks

- Depends on how passwords are stored
- Approach 1: store username/password in a file
  - Attacker only needs to read the password file
  - Security of system now depends on protection of this file!
- Approach 2: store username/encrypted password in file



  - Properties of the one-way hash function h:
    - h is not invertible: h(m) easy to compute, h-1(m) difficult
    - It is hard to find m and m' s.t. h(m) = h(m')
  - Should use standard functions, such as SHA, MD5, etc.

More offline attacks

- Previous scheme can be attacked: Dictionary Attack
  - Attacker builds dictionary of likely passwords offline
  - At leisure, builds hash of all the entries
  - Checks file to see if hash matches any entry in password file
  - There will be a match unless passwords are truly random
  - 20-30% of passwords in UNIX are variants of common words
    - Morris, Thompson 1979, Klein 1990, Kabay 1997
- Solutions:
  - Shadow files: move password file to /etc/shadow
    - This is accessible only to users with root permissions
  - Salt: store (user name, salt, E(password+salt))
    - Simple dictionary attack will not work. Search space is more.

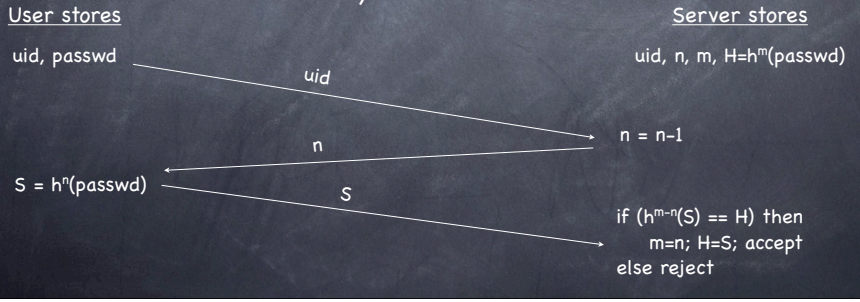
Salting Example

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

- If the hacker guesses Dog, he has to try Dog0001, ...
- UNIX adds 12-bit of salt
- Passwords should be made secure:
  - Length, case, digits, not from dictionary
  - Can be imposed by the OS! This has its own tradeoffs

One time passwords

- Password lasts only once
  - User gets book with passwords
  - Each login uses next password in list
- Much easier approach (Lamport 1981)
  - Uses one-way hash functions



Lamport's hash notes

- When n=1, user resets password and n. Sends to server
- Authentication is not mutual! User does not know is it is talking to server
  - Care against the small n attack
- Note that 1st password is h(h(h(h(x)))), 2nd h(h(h(x))), 3rd h(h(x))
  - A captured password yields past passwords, but no future ones

Challenge Response Scheme

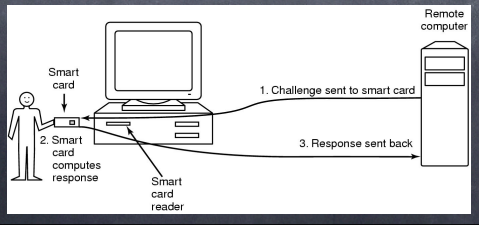
- New user provides server with list of ques/ans pairs
  - Server asks one of them at random
  - Requires a long list of question answer pairs
- Prove identity by computing a secret function
  - User picks an algorithm, e.g. x2
  - Server picks a challenge, e.g. x=7
  - User sends back 49
  - Should be difficult to deduce function by looking at results
- In practice
  - The algorithm is fixed, e.g. one-way hash, but user selects a key
  - The server's challenge is combined with user's key to provide input to the function

Authentication Using Physical Objects

- Door keys have been around long
- Plastic card inserted into reader associated with comp
  - Also a password known to user, to protect against lost card
- Magnetic stripe cards: about 140 bytes info glued to card
  - Is read by terminal and sent to computer
  - Info contains encrypted user password (only bank knows key)
- Chip cards: have an integrated circuit
  - Stored value cards: have EEPROM memory but no CPU
    - Value on card can only be changed by CPU on another comp
  - Smart cards: 4 MHz 8-bit CPU, 16 KB ROM, 4 KB EEPROM, 512 bytes RAM, 9600 bps comm. channel

Smart Cards

- Better security than stored value cards
  - Card sends a small encrypted msg. to merchant, who can later use it to get money from the bank
  - Pros: no online connection to bank required
- Perform local computations, remember long passwords



Biometrics: something you are

- System has 2 components:
  - Enrollment: measure characteristics and store on comp
  - Identification: match with user supplied values
- What are good characteristics?
  - Finger length, voice, hair color, retinal pattern, voice, blood
- Pros: user carries around a good password
- Cons: difficult to change password, can be subverted