Administrivia

- Communication
  - Add yourself to the mailing list
  - Join the wiki, contribute to the discussion

- Groups
  - Use the breaks today to start finalizing groups
  - Email Jeff Bigham your group info tonight
  - For those unassigned, we’ll match with groups on Friday

- Red Team project
  - Exploit buffer overflow vulnerability, discuss policy implications
  - Overview on course page, programming details on Friday
  - Out Friday 10/7, due Monday 10/24
Today

- Two goals
  - Overview of computer security topics
  - Provide context for remaining cybersecurity talks
- Your opportunity to ask computer security questions
  - I’ve always wondered what X is…
  - Who knows, maybe I can answer it
- Standard disclaimer
  - I’m not a computer security expert, but I play one on ConfXP
  - Channeling Steve Zdancewic, Dan Boneh, Butler Lampson, and John Mitchell via Stefan’s CSE 127 slides
Game Plan

- Computer security issues
  - Identity, risks, trust, ...

- Basic cryptography
  - Encryption, authentication, ...
  - What is the crypto behind SSL?

- Stepping back: Overall system security
  - We have systems like SSL…why aren’t we done?

- Evolution of security concerns
  - Why are we all in a huff about cybersecurity now?
  - Set the stage for future cybersecurity talks in class
Definition: The reasoning, mechanisms, policies, and procedures used to deal with someone else doing something that you don’t want them to do

There are a handful of key issues here (paraphrasing Butler Lampson)

- Identity
- Policy
- Risks/Threats
- Deterrence/Policy
- Locks
Identity

- What is it?
  - One def: The distinct personality of an individual regarded as a persisting entity; individuality (courtesy Black Unicorn)

- Why valuable?
  - Unique identifier – distinguishing mark (courtesy A.S.L. von Bernhardi)
  - Needed to establish an assertion about reputation
Reputation

- **What is it?**
  - A specific characteristic of trait ascribed to a person or thing:
    - e.g., a reputation for paying promptly

- **Why valuable?**
  - Potentially a predictor of behavior, a means of valuation, and as a means for third-party assessment

- **Issues**
  - Reliable identifiers
  - Binding identity to reputation
Due Diligence and Trust

- Due Diligence
  - Work to acquire multiple independent pieces of evidence establishing identity/reputation linkage; particularly via direct experience
  - Problem: Expensive

- Trust
  - *Reliance on something in the future; hope*
  - Allows cheap form of due-diligence: third-party attestation
  - Economics of third-party attestation? Cost vs limited liability
  - What is a third-party qualified to attest to?
  - Thompson: “Trusting Trust”
  - “Trust” vs. “Trustworthy” (Bruce Schneier)
Policy

- What is a bad thing?
- Sometimes simple
  - Steve and Ed can read my files
  - Never execute downloaded code
- Often remarkably tricky to define
  - There are >100 security options for IE
  - How should you set them?
Risks and Threats

- **Risk**
  - What is the cost if the bad thing happens?
  - What is the likelihood of the bad thing happening?
  - What is the cost of preventing the bad thing?
  - Example: Visa/Mastercard fraud

- **Threats**
  - Who is targeting the risk?
  - What are their capabilities?
  - What are their motivations?

- These tend to be well understood/formalized in some communities (e.g., finance sector) and less in others (e.g., computer science)
Deterrence

- There is some non-zero expectation that there is a future cost to doing a bad thing
  - going to jail, having a missile hit your house, having your assets seized, etc.
- Need meaningful forensic capabilities
  - Audit actions, assign identity to evidence, etc
  - Non-reputation
  - Must be cost effective
- Again channeling Butler: “lots of good locks on the Internet, few police”
  - We’ll come back to this at the end
Locks

- Mechanisms used to protect resources against threats
  - This is most of academic and industrial computer security
  - *Anderson*: Necessary, but not sufficient

- Several classes of locks
  - Cryptographic
  - Software security
  - Protocol security
κρυπτογραφία (Cryptography)

- Greek for “secret writing”

- Cryptographer: Invents cryptosystems
- Cryptanalyst: Breaks cryptosystems
- Cryptology: Study of cryptosystems
- Cipher: Mechanical way of encrypting text
- Code: Semantic translation
  - “eat breakfast tomorrow” = “attack on Thursday”
Cryptographic Properties

- Confidentiality
  - Obscure a message from eaves-droppers

- Integrity
  - Assure recipient that the message was not altered

- Authentication
  - Verify the identity of the source of a message

- Non-repudiation
  - Convince a 3rd party that what was said is accurate
“Secure” Cryptosystems

- If enemy intercepts ciphertext, cannot recover plaintext

- What else might your enemy know?
  - The kind of encryption function you are using
  - Some plaintext-ciphertext pairs from last year
  - Ciphertext for plaintext the enemy selected
  - Some information about how you choose keys

- What do we mean by “cannot recover plaintext”?
  - Ciphertext contains no information about plaintext
  - No “efficient” computation could make a reasonable guess
Computational Security

- 10,000 foot idea: not *impossible* to crack cipher, but very *difficult* to do so
  - Thus, an attacker with only *bounded resources* is extremely unlikely to crack it
  - Example: Assume attacker has only polynomial time, then encryption algorithm that can’t be inverted in less than exponential time is computationally secure
- This is 99% of crypto
  - Key issue: how sure are you about difficulty?
  - Relies on assumptions in theoretical computer science
SSL Motivation

- What is a secure cryptosystem that we all use daily?
- Secure Socket Layer (SSL)
  - “Web Encryption” used to protect credit card data
  - Note: Compare with using credit card in real world (which is riskier?)
- Use SSL to illustrate basic crypto properties
Shared Key Cryptography

- First step: Confidentiality (protect credit card)
  - Shared key & public key cryptography

- Sender & receiver use the same key
- Key must remain private (i.e., secret)
- Also called symmetric or secret key cryptography
- Examples
  - DES, Triple-DES, Blowfish, Twofish, AES, Rijndael, …
Shared Key Notation

- Encryption algorithm
  \[ E : \text{key} \times \text{plain} \rightarrow \text{cipher} \]
  Notation: \( K\{\text{msg}\} = E(K, \text{msg}) \)

- Decryption algorithm
  \[ D : \text{key} \times \text{cipher} \rightarrow \text{plain} \]

- \( D \) inverts \( E \)
  \[ D(K, E(K, \text{msg})) = \text{msg} \]

- Use capital “\( K \)” for shared (secret) keys

- Sometimes \( E \) is the same algorithm as \( D \)
Shared Keys Secure Channel

Alice  \[ K_{AB} \]  Bob

\[ K_{AB}\{\text{Hello!}\} \]
Shared Keys Secure Channel

Alice

K_{AB}\{Hello!\}

Bob

K_{AB}\{Hi!\}

K_{AB}
Shared Key Issues

- Compromised key means interceptors can decrypt any ciphertext they have acquired
  - Change keys frequently to limit damage
- Distribution of keys is problematic
  - Keys must be transmitted securely
  - Use couriers?
  - Distribute in pieces over separate channels?
- Number of keys is $O(n^2)$ where $n$ is # of participants
- We don’t have many “proofs” of computational security for shared key systems
Public Key Cryptography

- Sender encrypts using a public key
- Receiver decrypts using a private key
- Only the private key must be kept secret
  - Public key can be distributed at will (still tricky, though)
- Also called asymmetric cryptography
- Best known example: RSA
  - Ron Rivest, Adi Shamir, Leonard Adleman
    - Proposed in 1979
    - They won the 2002 Turing award for this work
  - Has withstood years of cryptanalysis
    - Not a guarantee of security…
    - But perhaps a strong vote of confidence
Public Key Notation

- Encryption algorithm
  \[ E : \text{keyPub} \times \text{plain} \rightarrow \text{cipher} \]
  Notation: \( K\{\text{msg}\} = E(K, \text{msg}) \)

- Decryption algorithm
  \[ D : \text{keyPriv} \times \text{cipher} \rightarrow \text{plain} \]
  Notation: \( k\{\text{msg}\} = D(k,\text{msg}) \)

- \( D \) inverts \( E \)
  \[ D(k, E(K, \text{msg})) = \text{msg} \]

- Use capital “\( K \)” for public keys
- Use lower case “\( k \)” for private keys
- Sometimes \( E \) is the same algorithm as \( D \)
Public Key Secure Channel

Alice

Bob

$K_B\{\text{Hello!}\}$

$K_A\{\text{Hi!}\}$

$K_A, K_B$

$k_A$

$k_B$
Public Key Crypto Pros/Cons

- More computationally expensive than shared key crypto
  - Algorithms are harder to implement
  - Require more complex machinery
  - RSA 1000x slower than DES (hardware implementations)

- More formal justification of difficulty
  - Hardness related to complexity-theoretic results

- A principal needs one private key and one public key
  - Number of total keys for pair-wise communication is $O(n)$
Diffie-Hellman Key Exchange

- Shared key systems are fast, but require pairwise secret key exchange
- Public key systems are slow, but allow easier distribution of keys
- Diffie-Hellman: Hybrid system
- Idea: Use public key system to distribute shared key
Crypto Summary

- We now have Confidentiality
  - Shared keys, public keys for encryption, decryption
  - Combine them for ease-of-use, efficiency
  - No one can eavesdrop and obtain credit card info

- Integrity property next
  - How does Alice know that what she received is what Bob sent?
  - How does Amazon know that no one corrupted credit card info in transit?
Message Authentication

- Issue: An attacker can reorder blocks in RSA
  - Decryption succeeds, but result is not what was encrypted
- You want some “evidence” that a message hasn’t been changed
- You’d prefer if the evidence wasn’t too expensive
  - To create, verify, or transmit
- It should be hard to forge the evidence
- The evidence should be unique
Cryptographic Hashes

- Map variable length string into fixed length digest
  - Sometimes called a Message Digest
- Hash functions $h$ for cryptographic use fall in one or both of the following classes
  - **Collision Resistant Hash Function (unique):** It should be computationally infeasible to find two distinct inputs that hash to a common value (i.e., $h(x) = h(y)$)
  - **One Way Hash Function (unforgeable):** Given a specific hash value $y$, it should be computationally infeasible to find an input $x$ such that $h(x) = y$
- Examples
  - Secure Hash Algorithm (SHA)
  - Message Digest (MD4, MD5)
Cryptographic Hash Uses

- **Modification Detection Codes (MDC)**
  - Compute and store hash (securely) of some data
  - Check later by recomputing hash and comparing
  - Has this file been tampered with?
  - Has this message stream been altered?

- **Message Authentication Code (MAC)**
  - Cryptographically keyed hash function
  - Send \((\text{msg}, \text{hash}(\text{msg}, \text{key}))\)
  - Attacker who doesn’t know the key can’t modify msg (or the hash)
  - Receiver who knows key can verify origin of message
Crypto Summary

- Confidentiality
  - Shared keys, public keys for encryption, decryption
  - Combine them for ease-of-use, efficiency
  - Credit card is private

- Integrity
  - Keyed hash (MAC) authenticates message
  - Credit card not corrupted

- Authentication next
  - How does Alice know that she is actually talking with Bob?
  - Are you actually sending your credit card to Amazon?
Signatures

- Consider a paper check used to transfer money from one person to another
- Signature confirms authenticity
  - Only legitimate signer can produce signature (true?)
- In case of alleged forgery
  - 3rd party can verify authenticity (maybe?)
- Checks are cancelled
  - So they can’t be reused
- Checks are not alterable
  - Or alterations are easily detected
Digital Signatures

- Only one principal can make, others can easily recognize
- Unforgeable
  - If P signs a message M with signature $S\{P,M\}$ it is impossible for any other principal to produce the pair $(M, S\{P,M\})$
- Authentic
  - If R receives the pair $(M, S\{P,M\})$ purportedly from P, R can check that the signature really is from P
- Not alterable
  - After being transmitted, $(M, S\{P,M\})$ cannot be changed by P, R, or an interceptor
- Not reusable
  - Duplicate messages will be detected by the recipient
Digital Signatures Using Public Keys

- Opposite from normal use as cipher
  - Let $K_A$ be Alice’s public key
  - Let $k_A$ be her private key
  - To sign msg, Alice sends $D(msg, k_A)$
  - Bob can verify the message with Alice’s public key
- Assumes encryption algorithm is commutative
  - $D(E(M, K), k) = E(D(M, k), K)$
  - RSA is commutative
Digital Signatures
Using Public Keys

- Authenticity: Only Alice has $k_A$
- Non-repudiation: Bob can keep $msg$, $k_A\{msg\}$, which only Alice could produce
Variations

- Timestamps (to prevent replay)
  - Signed certificate valid for only some time.

- Add an extra layer of encryption to guarantee confidentiality
  - Alice sends $K_B\{k_A\{msg\}\}$ to Bob

- Combined with hashes for performance
  - Send $(msg, k_A\{hash(msg)\})$
Authentication Protocols

- How do A and B convince each other that they are each A and B?
  - Despite the fact that A and B are paranoid
- Cryptographic authentication protocols
  - Participants can detect cheating, cannot dispute outcome
  - Resilient to attackers
  - Attacks: Replay, impersonation, usurpation, man-in-the-middle, transferability…
  - Techniques: nonces, sequence numbers, timestamps…
Crypto Summary

- Confidentiality
  - Shared keys, public keys for encryption, decryption
  - Combine them for ease-of-use, efficiency

- Integrity
  - Keyed hash (MAC) authenticates message (efficient)

- Authentication
  - Digital signatures authenticate participants
  - Non-repudiation, too (log signed messages)
  - We know we’re talking with Amazon (almost) and can prove it

- Subtle problem, though…
  - These properties are for keys, not identities!
Key Establishment

- How do we establish a trusted binding between keys and identities? (That the key is Bob’s key?)
  - Goes back to initial issues of identity, reputation, and trust
- Bilateral out-of-band: Meet, exchange secret key
- Centralized 3rd party: Trusted party gives secret keys
  - Needham-Schroeder, Kerberos
- Hierarchical: 3rd party public key crypto
  - PKI, SSL
- Distributed: Chained trust for public key crypto
  - PGP
- Anarchistic: Variety of options
  - SSH
Public Key Infrastructures

- Trusted third party, Certification Authority (CA), binds authentication data to a public key: Certificate
- The PKI Certificate X.509
  - Structured message with:
    - Public key
    - Identifier(s)
    - Lifetime
  - Digitally signed by a trusted third party
- Certification Authority (CA)
  - Binds identifiers to a public key
  - Expected to perform some amount of due diligence before vouching for this binding
  - Popular CA’s: Verisign, Thawte
You now know the crypto basics behind SSL

- Client contacts server
- Server identifies itself (digital signature) and provides certificate to client
- Client authenticates certificate (server public key) with CA
- Client validates certificate
  - If valid, client uses server public key to encrypt random session key (shared key) and sends to server
  - Client and server use session key to encrypt communication (protect your credit card number)

Note: Server does not authenticate client

- Why might this be ok? Why might it be a problem?
Overall System Security

- We have successful cryptosystems like SSL, so what’s the problem?
- Clearly, appropriate use of cryptography is essential
  - Even that is hard to get right (frequently problems are with implementations)
- …but even if cryptography is used appropriately, there are still plenty of possible vulnerabilities
  - 85% of CERT vulnerabilities could not be prevented with cryptography
When Is a System Secure?

Claim: Perfect security does not exist

- Security vulnerabilities are the result of violating an assumption about the software (or, more generally, the entire system).
- Corollary: As long as you make assumptions, you're vulnerable.

And: You always need to make assumptions! (or else your software is useless and slow)
What Can You Assume?

- Eavesdropping on light from CRTs (Kuhn)
- Light emitted by CRT is
  - Video signal combined with phosphor response
- Can use fast photosensor to separate signal from HF components of light
- Even if reflected off diffuse surface (wall)

Figure 1. Photomultiplier tube module.
Source Signal

CAN YOU READ THIS?

This image was captured with the help of a light sensor from the high-frequency fluctuations in the light emitted by a cathode-ray tube computer monitor which I picked up as a diffuse reflection from a nearby wall.

Markus Kuhn, University of Cambridge, Computer Laboratory, 2001
Bounced Off a Wall

CAN YOU READ THIS?

This image was captured with the help of a light sensor from the high-frequency fluctuations in the light emitted by a cathode-ray tube computer monitor which I picked up as a diffuse reflection from a nearby wall.

Markus Kuhn. University of Cambridge: Computer Laboratory. 2001
Practical Security

- **Anderson**: “Designers focused on what could possibly go wrong, rather than on what was likely to”
  - Not all threats are equal: What are the attacks being used?
  - Need data (tough, no one wants to admit it)
  - 2001: First published report of Denial-of-Service activity
  - 2001–today: Worms (hard to deny their existence…)

- **Lampson**: “The best is the enemy of the good”
  - Doing nothing until a perfect solution found is a bad idea
    - Especially since nothing is perfect…back to risk assessment
  - “I can think of a way to break your system”
  - IDS, worm defense, buffer overflow defense all have flaws
  - But we still want them
Kinds of Attacks

- Direct attacks
  - Attacks against the cryptosystem
    » e.g., timing attacks on SSL (Brumley and Boneh)
  - Typically requires high expertise

- Indirect attacks
  - Attacks on assumptions (light bouncing off walls)
  - Attacking interface to system, identity
  - Anderson: “most security failures are due to implementation and management errors”
    » Management of keys, usability of system
    » Buffer overflow, format string attacks
  - May not require much expertise
Identity, Reputation, Trust

- When using SSL, who are you trusting?
  - Ultimately, that Verisign did due diligence
  - You are trusting Verisign’s reputation to do the right thing
  - How do you know Verisign did?

- Establishing identity to a cryptosystem
  - User authentication is critical
Authenticating Humans

- Need to securely authenticate people into systems
  - If you get the keys, cryptosystem doesn’t know differently
- Authentication is based on one or more of the following:
  - **Something you know**
    - Password
  - **Something you have**
    - Driver’s license
  - **Something inherent about you**
    - Biometrics (fingerprint, retinal, voice, face), location
Text Passwords

- Shared code/phrase
- Client sends to authenticate
- Simple, right?
- How do you...
  - Establish them to begin with?
  - Stop them from leaking?
  - Stop them from being guessed?
- Brute force attacks can frequently break passwords
Usability

- No one wants security
  - Not the user, programmer, service, or attacker
- If security is burdensome to use, will be bypassed
  - Windows: always run with Administrator privileges
  - Anderson: “products are so complex and tricky to use that they are rarely used properly”
  - Whitten: users cannot encrypt mail (6th generation product)
  - Even password management a hassle
- Even the most secure system is defenseless if people do not use it correctly, or bypass altogether
  - Clicking on executables in email
Implementation Attacks

- Most common implementation attack is buffer overflow
  - 50% of all CERT incidents related to these
- Assumption (by programmer) that the data will fit in a limited-size buffer
- This leads to a vulnerability: Supply data that is too big for the buffer (thereby violating the assumptions)
- This vulnerability can be exploited to subvert the entire programming model
  - i.e., execute arbitrary code
- You will do precisely this in the Red Team project
Why CyberSecurity Now?

- Ever since the first computer we have had security
  - Why are we all in an uproar now?
Why CyberSecurity Now?

- Ever since the first computer we have had security
  - Why are we all in an uproar now?

- Transformation of threats and capabilities
Problem: Internet Succeeded

- Large, homogeneous software base
  - 300 million Internet hosts (7/04)
  - 80% run same OS family (Windows)
- Software vulnerabilities
  - Software is complex and it has bugs (weekly security updates)
- High-performance network (unrestricted connectivity)
  - Low latency: 16ms to UCB, 28ms to UW, 80ms to MIT
  - High bandwidth: UCSD has a multi-gigabit Internet connection
    » DSL and cable increasingly replacing dialup
- Incentives
  - Bragging, delinquency, anger, profit, terror
  - No deterrence: easy to do, difficult to get caught
The Threat Landscape
(courtesy David Aucsmith)

National Interest

Personal Gain

Personal Fame

Curiosity

Script-Kiddy  Hobbyist Hacker  Expert  Specialist
The Threat Landscape
(courtesy David Aucsmith)

National Interest
Personal Gain
Personal Fame
Curiosity

Script-Kiddy
Hobbyist
Vandal
Expert
Specialist
The Threat Landscape
(courtesy David Aucsmith)

- National Interest
- Personal Gain
- Personal Fame
- Curiosity

- Script-Kiddy
- Hobbyist Hacker
- Expert
- Specialist

- Vandal
- Trespasser
The Threat Landscape
(courtesy David Aucsmith)

National Interest

Personal Gain

Personal Fame

Curiosity

Script-Kiddy

Vandal

Hobbyist

Hacker

Trespasser

Expert

Author

Specialist
The Threat Landscape

(courtesy David Aucsmith)

- National Interest
- Personal Gain
- Personal Fame
- Curiosity

- Author
- Trespasser
- Thief
- Vandal
- Script-Kiddy
- Hobbyist
- Hacker
- Expert
- Specialist
The Threat Landscape
(courtesy David Aucsmith)

- National Interest
- Personal Gain
- Personal Fame
- Curiosity

- Vandal
- Script-Kiddy
- Hobbyist Hacker
- Expert
- Specialist
- Author
- Thief
- Trespasser
- Spy
The Threat Landscape

National Interest

Personal Gain

Personal Fame

Curiosity

Script-Kiddy

Hobbyist Hacker

Expert

Specialist

Fastest growing segment

Spy

Thief

Trespasser

Author

Vandal
The Threat Landscape

- National Interest
- Personal Gain
- Personal Fame
- Curiosity

TOOLS CREATED BY EXPERTS NOW USED BY LESS SKILLED ATTACKERS AND CRIMINALS

- Author
- Expert
- Specialist
- Hobbyist Hacker
- Vandal
- Trespasser
- Thief
- Spy

- Script-Kiddy

Hacking and the bad guys
The Threat Landscape

- National Interest
- Personal Gain
- Personal Fame
- Curiosity

Tools created by experts now used by less skilled attackers and criminals

Hacking and the bad guys
New Consequences

- New types of attacks
  - Viruses, DoS, worms, botnets, spyware, phishing, spam, extortion…

- With potentially severe consequences
  - Hundreds of thousands of infected machines
  - Worm propagation alone clogs Internet, down for day

- Billions of dollars lost
  - Lost data, commerce, communication
  - System management nightmare

- Spillover into other essential infrastructure
  - Public utilities, ATMs, 911 call centers, air traffic control…
Summary

- We can build cryptosystems
  - We use SSL daily
- But we cannot build perfect systems
  - Flaws in assumptions, implementation, usability, management
  - Practical security must address these
- Recent transformation of threats and capabilities
  - Software homogeneity + high-performance Internet = global risk
  - Buffer overflows + automated exploit software = lowers the bar
- Global scale attacks and consequences
  - DDoS, worms, spyware, viruses
  - Spillover into physical critical infrastructure
- Topics of remaining cybersecurity talks