590.7 Network Security
Lecture 2: Goals and Challenges of Security Engineering

Xiaowei Yang
Roadmap

- What is security?
- Examples of secure systems
- Security properties
- Challenges
What is security?

- **System correctness**
  - If user supplies expected input, system generates desired output

- **Security**
  - If attacker supplies unexpected input, system does not fail in certain ways
What is security?

- System correctness
  - Good input $\Rightarrow$ Good output

- Security
  - Bad input $\Rightarrow$ Bad output
How to analyze a security system

1. **Policy**
   - What you are supposed to achieve

2. **Mechanism**
   - The techniques to meet the policy requirements
   - Ex: ciphers, access controls

3. **Assurance (security guarantees)**
   - The amount of reliance one can place on each mechanism

4. **Incentives**
   - Motive that good guys do their jobs right and bad guys defeat your policy
Ex: analyzing the 911 attack

- A failure of policy not mechanism
- Policy changed later
- Assurance is poor
Examples of security systems

- Home
- Hospital
- Bank
Home

- Home banking
- Remote car keys
- Mobile phones
- Wireless routers
Hospital

- Keeping patient records private
- Anonymizing patient records
- Web-based access to patient records
Bank

- Bookkeeping a customer’s transactions
- ATM
- Bank websites
- Messaging systems
- Bank offices
Security Properties

- **Confidentiality**
  - Information about system or its users cannot be learned by an attacker

- **Integrity**
  - Protected information not modified by attackers

- **Availability**
  - Actions by an attacker do not prevent users from having access to use of the system
Security is about

- Honest user (e.g., Alice, Bob, …)
- Dishonest Attacker
- How the Attacker
  - Disrupts honest user’s use of the system (Integrity, Availability)
  - Learns information intended for Alice only (Confidentiality)
Network security

Network Attacker
Intercepts and controls network communication

Alice
Web security

Web Attacker

Sets up malicious site visited by victim; no control of network
Operating system security

OS Attacker
- Controls malicious files and applications

Alice

OS Attacker
- Controls malicious files and applications
Confidentiality: Attacker does not learn Alice’s secrets
Integrity: Attacker does not undetectably corrupt system’s function for Alice
Availability: Attacker does not keep system from being useful to Alice
Challenges

- Buggy code
- Inexperienced users
- Poorly designed protocols
- Insider attacks
  - What can you trust?
Buggy code

![Graph showing total number of projected bugs through time. The x-axis represents years from 1994 to 2001, and the y-axis represents bug counts. The graph shows the growth of bugs over time, with different project versions at specific points.]
Market place for vulnerabilities

**Option 1:** bug bounty programs
- Google Vulnerability Reward Program: 3K $
- Mozilla Bug Bounty program: 500$
- Pwn2Own competition: 15K $

**Option 2:**
- Zero Day Initiatives, iDefense: 2K – 25K $
## Market place for vulnerabilities

- **Option 3**: black market

<table>
<thead>
<tr>
<th>Vulnerability/Exploit</th>
<th>Value</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Some exploits”</td>
<td>$200,000 - $250,000</td>
<td>A government official referring to what “some people” pay [9]</td>
</tr>
<tr>
<td>a “real good” exploit</td>
<td>over $100,000</td>
<td>Official from SNOsoft research team [10]</td>
</tr>
<tr>
<td>Vista exploit</td>
<td>$50,000</td>
<td>Raimund Genes, Trend Micro [8]</td>
</tr>
<tr>
<td>“Weaponized exploit”</td>
<td>$20,000-$30,000</td>
<td>David Maynor, SecureWorks [11]</td>
</tr>
</tbody>
</table>
Marketplace for owned machines

Pay-per-install (PPI) services

- Own victim’s machine
- Download and install client’s code
- Charge client

Cost:

- **US** 100-180$ / 1000 machines
- Asia 7-8$ / 1000 machines

Source: Cabalerro et al. (www.icir.org/vern/papers/ppi-usesec11.pdf)
Why own machines?
Steal IP addresses
Use the infected machine’s IP address for:

- **Spam** *(e.g. the storm botnet)*
  
  Spamalytics:
  
  - 1:12M pharma spams leads to purchase
  - 1:260K greeting card spams leads to infection

- **Denial of Service:**
  
  - Services: 1 hour (20$), 24 hours (100$)

- **Click fraud** *(e.g. Clickbot.a)*
Why own machines:
Steal user credentials

Keylog for banking passwords, web passwords, gaming pwds

Example: SilentBanker (2007)

User requests login page

Malware injects Javascript

Bank sends login page needed to log in

When user submits information, also sent to attacker

Similar mechanism used by Zeus botnet
Challenges

- Buggy code
- Gullible users
- Poorly designed protocols
- Insider attacks
  - What can you trust?
Inexperienced users

- Phishing attacks
  - “I am stuck in London… lost my wallet…”

- Poor choice of Passwords

- Unchanged default username/password
Poorly designed protocols

- **telnet**
  - Send plain passwords over the network

- **TCP**
  - Fixed initial syn numbers

- **BGP**
  - Unauthenticated messages
Insider attacks

- Hidden trap door in Linux (nov 2003)
  - Allows attacker to take over a computer
  - Practically undetectable change (uncovered via CVS logs)
What can you trust?

- What code can we trust?
  - Consider "login" or "su" in Unix
  - Is RedHat binary reliable?
  - Does it send your passwd to someone?

- Can't trust binary so check source, recompile
  - Read source code or write your own
  - Does this solve problem?
Compiler backdoor

- This is the basis of Thompson's attack
  - Compiler looks for source code that looks like login program
  - If found, insert login backdoor (allow special user to log in)
- How do we solve this?
  - Inspect the compiler source
C compiler is written in C

- Change compiler source S

```c
compiler(S) {
    if (match(S, "login-pattern")) {
        compile (login-backdoor)
        return
    }
    if (match(S, "compiler-pattern")) {
        compile (compiler-backdoor)
        return
    }
    .... /* compile as usual */
}
```
Avoid detection

- Compile this compiler and delete backdoor tests from source
  - Someone can compile standard compiler source to get new compiler, then compile login, and get login with backdoor

- Simplest approach will only work once
  - Compiling the compiler twice might lose the backdoor
  - But can making code for compiler backdoor output itself
    - (Can you write a program that prints itself? Recursion thm)

- Read Thompson's article
  - Short, but requires thought
Self-reproducing code example

- Code that prints itself
  ```java
class Quine {
  public static void main(String[] args) {
    char q = 34; // Quotation mark character
    String[] l = {
      "public class Quine",
      "{",
      "  public static void main(String[] args)
      {
        for(int i = 0; i < 6; i++) // Print opening code
          System.out.println(l[i]);
        for(int i = 0; i < l.length; i++) // Print string array
          System.out.println(l[6] + q + l[i] + q + ',');
        for(int i = 7; i < l.length; i++) // Print this code
          System.out.println(l[i]);
      }
    }
  }
}
```