Lecture 1: Introduction, SE issues, open vs. closed source, etc

**CS 916 Spring 2004, Application Security**
- Professor: Gleb Naumovich
  - LC228 (Dibner library building)
  - gleb@poly.edu
  - (718) 260-3554
- online material: isis.poly.edu/courses/cs916/
  - Lecture notes
  - Code
  - Project descriptions
- office hours:
  - Tuesday and Thursday 4:30 - 5:50 PM

**Focus**
- What aspects of computer security does this course cover?
  - Configuration and deployment of distributed applications
  - Avoidance of exploits of bugs and features
  - Application of security mechanisms and cryptography in practice
- What aspects of computer security does this course not cover?
  - Intrusion detection
  - Details of cryptographic theory
  - Secure networking protocols and applications

**Topics of lectures**
1/29 Intro
2/5 Description of the application used in the project, intro to J2EE architecture
2/12 J2EE deployment and security policy
2/19 Security-related trade-offs in distributed computing
2/26 Class design for security
3/4 Buffer overflows and race conditions
3/11 Exception handling and data persistence
3/18 Securing communication channels
3/25 Authentication
4/1 Group presentations
4/15 J2SE security model
4/22 J2SE security model continued, class loading
4/29 Code obfuscation, watermarking, tamperproofing
5/13 Final reports

**Course organization**
- No midterm
- No final
- Project

**Project**
- Team project
  - 2-3 people per team
- Distributed Java application for running book auctions
  - Written by me, with no thought about security issues
  - Both Web and rich client UIs
  - Your goal is to re-design this program with security in mind
- Several stages
  1. Application deployment and configuration of J2EE security policy (2 weeks)
  2. Design and implementation for security (4 weeks)
  3. Vulnerability analysis and testing (3 weeks)
  4. Re-design and security policy configurations (3 weeks)

**Grading**
- Grades (A, B, C, F) will be based on project stages:
  1. Application deployment and configuration of J2EE security policy 20%
  2. Design and implementation for security 40%
  3. Vulnerability analysis and testing 20%
  4. Re-design and security policy configurations 20%
Lecture 1: Introduction, SE issues, open vs. closed source, etc

Textbook
- Online materials (usually as links from the Lectures page)

Lecture topics
- Motivation
- Software engineering aspects of computer security
- Open source, security by obscurity, etc

Security --- that’s where the money is these days, right?
- Spending on security technology grew by 28% in 2001 (compared to 2000) [UBS Warburg]
  - $6bln in 2001
  - Projected $13bln in 2005
- 24% of firms had increased their technology budget in 2002, but 73% increased spending on security [Meta group]
- Most companies spend 3% of their technology budgets on security; technology budgets are typically 3% of the revenues [Meta group]
- The number of unfilled [computer] security jobs in the US is 75,000 [Symantec]

Viruses
- The costs of computer viruses worldwide are $13.2bln [Computer Economics, a consultancy firm]
  - The figure is probably too high
  - Hard to quantify the cost of detection and clean-up
  - Following Code Red and Nimda, sales of anti-virus products at Symantec were 53% higher than the previous year

Intrusion
- Reliable figures are hard to come by
  - Many attacks go unreported or unreported
- Survey by CSI/FBI:
  - 503 large companies and government agencies surveyed
  - 40% detected system intrusions in 2001
  - 70% of these were website vandalism
  - 20% reported theft of proprietary information
  - 85% reported virus problems
  - 90% have anti-virus software installed
  - 89% have firewalls installed
  - 63% have intrusion detection software installed
  - The failure to responsibly patch applications resulted in 99% of the 5,823 Web site defacements in 2000 [Computer Emergency Response Center (CMU)]
  - Forensics are expensive. From Washington Unv. 2001 Forensic Challenge competition:
    - The winner took <1 min to break in the Univ. network, stayed less than 30 min
    - Finding out what the intruder did took 34 hours on average
  - In this light, security-related hype put out by companies is laughable
    - Sun: “We make the net secure”
    - Oracle: "Unbreakable database software", about 9i Database
    - One individual found 9 serious exploits a couple of weeks later

What is the root cause of intrusions being possible?
- Intrusion in computer systems from the outside is not supposed to be possible, so how does it become possible?
- Short answer: bugs in applications
  - Code Red exploited a buffer overflow error in code that handles input URLs in the Windows 2000 Indexing Service
- Most security holes exploit buffer overflows
  - Programs get “confused” into executing code supplied by the attacker
- In this light, security-related hype put out by companies is laughable
  - Sun: “We make the net secure”
  - Oracle: “Unbreakable database software”, about 9i Database
    - One individual found 9 serious exploits a couple of weeks later
Lecture 1: Introduction, SE issues, open vs. closed source, etc

Types of networked attacks on security of a system
Generally, a flaw in the design or implementation of a system is exploited by these attacks
- Eavesdropping
  - Encrypting transmissions may fail to defeat attacks
- Tampering with data
  - Malicious modification of transmissions
- Spoofing
  - Generating phony transmissions
- Hijacking
  - Replacing a legitimate transmission with a malicious one
- Capture/replay
  - Capture a transmission and later replay it

Why is security of applications important (compared to security of networks and protocols)
- Applications are increasingly distributed
  - Resources can be accessed remotely via the application
- Applications are increasingly complex
  - Vulnerabilities are usually bugs and bugs are unavoidable in complex programs
- Applications are increasingly extensible
  - Extension points are offered for third-party plug-ins
  - Plug-ins can be loaded at run time
    - E.g. browsers run applets
  - Related to complexity --- extensible systems are harder to understand and analyze than non-extensible ones

Top 10 vulnerabilities in Web applications (Open Web Application Security Project)
1. Unvalidated parameters
2. Broken access control
3. Broken account and session management
4. Cross-file scripting flaws
5. Buffer overflows
6. Command injection flaws
7. Error handling problems
8. Insecure use of cryptography
9. Remote administration flaws
10. Web and application server misconfiguration

Human factors
- Consumers are not educated about security and privacy
  - What do you do when your car loses a wheel on a highway?
- Consumers demand features
  - "Given the choice between dancing pigs and security, users will pick dancing pigs every time", Ed Felten
- Developers' attitude
  - "Not my job --- I'm designing this thing to run as fast as possible!"
- Fixation on security from the networking/cryptography standpoint
  - "If we encrypt all traffic, we have nothing to worry about, right?"
- Fixation on technology
- Companies started to re-evaluate the trade-off between security and functionality
  - Microsoft's web server software has most features turned off by default
  - Customers have to configure them in site-specific way
  - Some customers asked for a button "to turn everything on"

The weakest link of computer security
- PentaSafe Security Technologies survey of a number of companies (2002):
  - 86% reported abuse of Internet access by insiders
  - 66% reported laptop theft
  - 50% reported unauthorized access by insiders
  - 40% reported unauthorized access by outsiders
  - 2/3 of commuters in Victoria Station (London) revealed their computer password in return for a ballpoint pen
  - Almost half of British office workers used their own name or the name of a family member or a pet as their password
- Meta Group: The most common way to gain access to a system is to call internal tech support and pose as an employee who forgot their password
- CSI/FBI survey (using a small sample size):
  - Average external attack cost $57,000
  - Average attack by an insider cost $2.7m

Penetrate and Patch approach
- A form of the "leave it to the market" mentality: don't worry about security until vulnerabilities become known, then patch them
  - Advantages: cost, of course
  - Drawbacks:
    - Only publicized vulnerabilities can be patched
    - Patches may be rushed and introduce more problems
    - Patches degrade the program design
    - Many users never apply patches
    - Insufficient education about security and privacy is to blame
Lecture 1: Introduction, SE issues, open vs. closed source, etc

Number of intrusions for a security hole over time

Security is not an absolute measure
- In practice, providing very high levels of security is very costly
- A security risk is just like any other business or software development risk
  - Predicted, managed, mitigated
- Credit card companies could provide much tighter security to combat fraud
  - Consumers would be unhappy
  - Too expensive, it’s easier to absorb the cost of fraud
- Companies can use “security insurance” that pays damages if their systems are breached
  - In future, this could mean that security companies impose restrictions on what equipment and business/development processes the company can use
  - Btw, insurance companies quote higher premiums if Windows systems are used

Is “electronic Pearl Harbor” a real danger?
- Lamar Smith, congressman, in a report to a judiciary committee, Feb 2002: “Until we secure our cyber-infrastructure, a few keystrokes and an Internet connection is all one needs to disable the economy and endanger lives... A mouse can be just as dangerous as a bullet or a bomb”
- US Naval War College / Gartner group simulations, August 2002:
  - An “electronic Pearl Harbor” attack on the US would cause serious disruption, but
  - Would need five years of preparation
  - Would need $200m in funding

When is a software system secure?
- Depends on who you ask
- Depends on the context
- Viega-McGraw: A system is secure if its policy for accessing resources is enforced
- This definition is not very satisfactory:
  - Do we believe that a policy is always explicitly defined?
  - What enforcing the policy means exactly? That there may be no circumstance under which some resources can be accessed illegally?
    - Easy counterexample: stolen password
- We’ll see if we can give a definition of security at the end of the course

So, if software bugs can be exploited, can’t we avoid most problems by hiding source?
- Most commercial software is distributed in a compiled form, without source code
- In theory, it is much harder for attackers to find exploitable bugs in a program if they do not have source for this program
- In practice,
  - There are some skilled people out there, reading machine code
  - Debuggers can help
  - Some high-level and scripting languages are not compiled to machine code
    - E.g., Java bytecodes can be decompiled back to source
  - Attackers can find and exploit bugs by analyzing the program behavior, without looking at the source
    - A faulty encryption algorithm in Netscape, found by analyzing changes in encrypted form of data

Open source advocates: if source of an application is open, many friendly eyes will see it and help identify potential security holes
- “Given enough eyeballs, all bugs are shallow” - Eric Raymond
- In practice,
  - People have to have a look at the code
    - Usefull application, free, well-designed and written...
    - People reading code have to have training in software security
  - People reading code have to have security as a goal
  - People reading code may have to understand the design of the whole application
  - Identifying bugs by reading code is just hard
**Lecture 1: Introduction, SE issues, open vs. closed source, etc**

---

**What is the best way to make people aware of security holes?**
- Full disclosure advocates: security vulnerabilities should be published on the Web
  - True, potential attackers get their hands on this information
  - But this should force companies to react quickly and post patches
- In the real world, this may pose problems
  - Big websites get attacked ~40 minutes after the publication of a new vulnerability
  - The patch may not be available for some time
  - This is basically a “penetrate and patch approach”

---

**Selection of the implementation language**
- Michael Bacarella: “It should be a crime to teach people C/C++”
  - More precisely, it’s a crime to teach them to write high-level applications in these languages
  - Because of pointer arithmetic, of course
  - Accessing an array outside of its boundaries is allowed – it’s a straight road to buffer overflows
  - But you gotta use C because it runs fast! Right?
  - But how often do you need speed in high-level applications?
  - A well-designed Java program is only about 2 times slower than a well-designed C program doing the same thing (anecdotal evidence)

---

**So, why is Java better than C/C++ for security?**
- Object-oriented
- Better data abstraction than C
- No direct memory access
- Nasty programs will have a harder time trawling through memory
- Garbage collection
- Less obscure bugs resulting from dangling pointers
- Array bound checking at runtime
- Type safety
  - Object casting is restricted to static inheritance relationships
  - A method can only be called if it exists in the class
  - Variables cannot be used before they are initialized
- Final classes and methods (cannot be extended)
- Lots of features supporting security of mobile code
  - Bytecode verification
  - A malicious class will not be able to access a private field or method
  - A security model for giving specific permissions to untrusted code

---

**Are there reasons using Java can hurt the security of a system?**
- Sure – if the Java (EE or SE) security model is used incorrectly
  - It is complicated, so using it is not straightforward
- Java runtime implementations and application servers/containers can contain bugs too