About Me

• Kevin Butler (PhD, Penn State)
• Assistant Professor, CIS - formerly interned with AT&T, Seagate, Symantec, worked & Telcordia & UUNET
• Research focus: Security of systems & networks
• Example Projects
  ‣ Kells portable storage security architecture
  ‣ Rootkit-resistant disk prototype
  ‣ Mobile phone and routing security mechanisms
• Office Hours: W 1-2, or by appointment
• Office: 245 Deschutes Hall
• Email: butler at cs.uoregon.edu
Security mechanisms and policies have been implemented at several system layers (app, OS, VM, network)

Are we now secure?
Most current security problems are based on the failure of people to deploy hosts securely

Botnets
Rootkits
Web attacks: XSS, SQL Inject, …
Worms (Conficker)
Password Guessing
Buffer Overflows
Arbitrary App Flaws
Security State

SANS Top Security Risks
http://www.sans.org/top-cyber-security-risks/

- Client-side software is unpatched (apps patched slower)
- Web servers are vulnerable (XSS are 80%)
- Application vulnerabilities exceed OS vulnerabilities
- Attacks on Mac systems (QuickTime)
- US is the major attack target (30:1)
- Still buffer (and heap) overflows

We will study the structure of attacks on hosts and a general procedure for their prevention
Cross-site Scripting (XSS)

- **Aim:** *Get a client to run an attackers’ code at higher privilege (Privilege Escalation)*

- **Attack:**
  - Attacker places content on trusted site
  - Client downloads content and that content attacks an unpatched client program (e.g., media player)
  - Attacker can run as client user
  - Install reverse shell backdoor (outbound HTTPS)
  - Download local privilege escalation program (again unpatched client code)
  - Attack other machines – Windows domain controller
Security Mythology

- Claim: *All these problems were solved in Multics*
- Is this claim true?
- Why not just use it?
- What is necessary?
- By whom?
- Can we make it happen?

- Claim: *We are still trying to solve the same security problems since Multics*
Who Has a Role?

- We want to examine what all the interested parties do/do not want for security/function and what they can/cannot know to resolve conflicts
- Programmers (may be multiple groups)
- OS Distributors
- Administrators
- Users
- Service Providers
- Content Providers
This course....

- Is a **systems** course that teaches principles for building a secure system and techniques for implementing those principles
  - Caveat: We are still trying to figure out the latter
  - Topics: What makes a system secure (mechanisms and policies); Example implementations of such principles (at OS, VMM, and application); Tools to assist in such implementations; How do we put it together; Recent research in secure systems design
Background

• Required:
  ‣ CIS 533 (or equivalent)

• Expected:
  ‣ Solid OS and some security background

• Additional:
  ‣ Willingness to read
    • We are going to read a lot of systems security papers
  ‣ Willingness to program
    • We are going to have some OS programming assignments (Linux)
  • Major deliverable: original research in the area
Course Materials

• Website
  ‣ [http://www.cs.uoregon.edu/Classes/11F/cis610syssec/](http://www.cs.uoregon.edu/Classes/11F/cis610syssec/)
  ‣ Course assignments, slides, etc. will be placed here
    • Check back often -- I may change some of the papers/assignments

• Course Textbook
  ‣ Augmented with research papers
Course Calendar

- The course calendar has all the details
- Links to online papers for readings
- Links to projects
- Please check the calendar frequently
  - it’s the real-time state of the course
Course Mailing List

• We’ll use the security@cs.uoregon.edu mailing list
  • Mixture of old and new folks on it

• I will send a test email
  ‣ Please reply if you do not receive by Fr
  ‣ May need to forward to your CIS account

• Can use to email about questions with readings/projects
  ‣ Please use “cis610” in the subject
Grading

- Midterm (15%)
  - Take home – do the readings

- Quizzes and Assignments (20%)
  ‣ (probably) 2 programming assignments
  ‣ OS analysis, security policy

- Participation (10%)

- Project (55%)
Lateness Policy

• Assignments and project milestones are assessed a **20% per-day late penalty**, up to a maximum of **4 days**. Unless the problem is apocalyptic, don’t give me excuses. Students with legitimate reasons who contact the professor before the deadline may apply for an extension.

• You decide what you turn in
Ethics Statement

• This course considers topics involving personal and public privacy and security. As part of this investigation we will cover technologies whose abuse may infringe on the rights of others. As an instructor, I rely on the ethical use of these technologies. Unethical use may include circumvention of existing security or privacy measurements for any purpose, or the dissemination, promotion, or exploitation of vulnerabilities of these services. Exceptions to these guidelines may occur in the process of reporting vulnerabilities through public and authoritative channels. Any activity outside the letter or spirit of these guidelines will be reported to the proper authorities and may result in dismissal from the class.

• When in doubt, please contact the instructor for advice. Do not undertake any action which could be perceived as technology misuse anywhere and/or under any circumstances unless you have received explicit permission from Professor Butler.
Road Map

• Introduction
  ‣ 1. What is security? 2. What are the fundamental principles of secure execution?

• Designing for Security and Protection
  ‣ 1. Experiences with Multics and UNIX/Windows

• Mandatory Access Control
  ‣ 1. Policy Models 2. Lattice Models in Depth

• Systems Security Architectures

• Assurance
  ‣ 1. Common Criteria 2. Program Analysis

• Practical System Integrity
  ‣ 1. System Integrity Models 2. Decentralized Label Model 3. Data/Control Flow Integrity

• Special Topics
  ‣ 1. Trustworthy Computing 2. Web Security
Review

- Are we speaking the same language?
- **General Terms**
  - Principals/Subjects and Adversaries/Attackers
  - Trust Model
  - Threat Model
  - Security Model
- We will develop (semi-)formal models for each
Reading papers . . .

- What is the purpose of reading papers?
- How do you read papers?
Understanding what you read

• Things you should be getting out of a paper
  ‣ What is the central idea proposed/explored in the paper?
    • Abstract
    • Introduction
    • Conclusions
    \[These \text{ are the best areas to find an overview of the contribution}\]
  ‣ How does this work fit into others in the area?
    • Related work - often a separate section, sometimes not, every paper should detail the relevant literature. Papers that do not do this or do a superficial job are almost sure to be bad ones.
    • An informed reader should be able to read the related work and understand the basic approaches in the area, and how they differ from the present work.
• What scientific devices are the authors using to communicate their point?
  ‣ Methodology - this is how they evaluate their solution.
    • Theoretical papers typically validate a model using mathematical arguments (e.g., proofs)
    • Experimental papers evaluate results based on test apparatus (e.g., measurements, data mining, synthetic workload simulation, trace-based simulation).
      ‣ Empirical research evaluates by measurement.
    • Some papers have no evaluation at all, but argue the merits of the solution in prose (e.g., paper design papers)
Understanding what you read (cont.)

• What do the authors claim?
  ‣ Results - statement of new scientific discovery.
    • Typically some abbreviated form of the results will be present in the abstract, introduction, and/or conclusions.
    • Note: just because a result was accepted into a conference or journal does not necessarily mean that it is true. Always be circumspect.

• What should you remember about this paper?
  ‣ Take away - what general lesson or fact should you take away from the paper.
  ‣ Note that really good papers will have take-aways that are more general than the paper topic.
Summarize Thompson Article

- Contribution
- Motivation
- Related work
- Methodology
- Results
- Take away
• **Contribution**: Ken Thompson shows how hard it is to trust the security of software in this paper. He describes an approach whereby he can embed a Trojan horse in a compiler that can insert malicious code on a trigger (e.g., recognizing a login program).

• **Motivation**: People need to recognize the security limitations of programming.

• **Related Work**: This approach is an example of a Trojan horse program. A Trojan horse is a program that serves a legitimate purpose on the surface, but includes malicious code that will be executed with it. Examples include the Sony/BMG rootkit: the program provided music legitimately, but also installed spyware.

• **Methodology**: The approach works by generating a malicious binary that is used to compile compilers. Since the compiler code looks OK and the malice is in the binary compiler compiler, it is difficult to detect.

• **Results**: The system identifies construction of login programs and miscompiles the command to accept a particular password known to the attacker.

• **Take away**: *What is the transcendent truth?????* (see next slide)
Turtles all the way down ...

• Take away: Thompson states the “obvious” moral that “you cannot trust code that you did not totally create yourself.” We all depend on code, but constructing a basis for trusting it is very hard, even today.

• ... or “trust in security is an infinite regression ...”

“A well-known scientist (some say it was Bertrand Russell) once gave a public lecture on astronomy. He described how the earth orbits around the sun and how the sun, in turn, orbits around the center of a vast collection of stars called our galaxy. At the end of the lecture, a little old lady at the back of the room got up and said: "What you have told us is rubbish. The world is really a flat plate supported on the back of a giant tortoise." The scientist gave a superior smile before replying, "What is the tortoise standing on?" "You're very clever, young man, very clever", said the old lady. "But it's turtles all the way down!"

Reading a paper

• Everyone has a different way of reading a paper.

• Here are some guidelines I use:
  ‣ Always have a copy to mark-up. Your margin notes will serve as invaluable sign-posts when you come back to the paper (e.g., “here is the experimental setup” or “main result described here”)
  ‣ After reading, write a summary of the paper containing answers to the questions in the preceding slides. If you can’t answer (at least at a high level) these questions without referring to the paper, it may be worth scanning again.

• Over the term, try different strategies for reading papers and see which one is the most effective for you.
Reading a systems security paper

• What is the security model?
  ‣ Who are the participants and adversaries
  ‣ What are the assumptions of trust (trust model)
  ‣ What are the relevant risks/threats

• What are the constraints?
  ‣ What are the practical limitations of the environment
  ‣ To what degree are the participants available

• What is the solution?
  ‣ How are the threats reasonably addressed
  ‣ How do they evaluate the solution

• What is the take away?
  ‣ key idea/design, e.g., generalization (not solely engineering)

• **Hint**: I will ask these questions when evaluating course project.
Course Projects

• The course project requires the students execute some limited research in security.
  ‣ Demonstrate applied knowledge
  ‣ Don’t try to learn some new non-security field
  ‣ Be realistic about what is possible in a one quarter.
  ‣ However, the work should reflect real thought and effort.

• The grade will be based on: novelty, depth, correctness, clarity of presentation, and effort.

• Structure
  ‣ 1-3 students per group
  ‣ Single person suggested if you will work in security.
Deliverables

• The chief product of the project will be a 10-15 page conference style paper. There will be several milestones:
  ‣ Project Choice (10/06/11)
  ‣ Abstract, Background and Related Work (10/20/11)
  ‣ Experiment Proposal (10/27/11)
  ‣ Project Status Slides (11/17/11)
  ‣ Project Presentation (12/01/11)
  ‣ Final Project Write-up (12/08/11)

• This is a very important factor in your grade (30%)
  ‣ An exceptionally good (poor) project may help (kill) grade
Project Choice

• Due on Oct 6, 5:00 PM

• Order list of projects
  ‣ Choose three projects in order of interest

• Choose up to 2 collaborators (optional)
  ‣ Get a sense of groupings

• I will approve/choose your project and group
  ‣ Hopefully, I can resolve the constraints implied
  ‣ One group per project
  ‣ A functional group
Topic Examples

• Web systems
  ‣ Evaluate the security of PHP, Apache extension ...

• Operating systems
  ‣ Create your own Linux security module to monitor all system calls and measure inter-process communication (*)

• Cloud Systems
  ‣ Design a cloud component for ensuring data security

• User Studies
  ‣ Measure the effectiveness of passwords, card systems

• Network security
  ‣ Build a intrusion detection system that watches IM msgs (*)

• Note: picking a topic is very important, and should almost certainly involve an area that you know well
Why write a paper?

• There are many reasons to write a paper:
  ‣ Articulate a new idea, thought, or observation ...
  ‣ Document your research ...
  ‣ Talk about new (observed) phenomenon ....
  ‣ Advance your career ...
  ‣ Because you have to ...

• Reality: publication is the coin of the realm in science, failure to do this successfully will lead to failure. You have to be effective at this to be a good (a) graduate student, (b) faculty member, or [sometimes] (c) researcher in professional research laboratory (IBM/AT&T/MS)
Where to publish?

- Venues for publication:
  - Tech report
  - Workshop
  - Conference
  - Journal

- Often your work will work through these from *preliminary* to *archival* versions of the work, sometimes branching or joining.

- *Book*: less frequent, more work.
Publication Tiers

• Not all publication venues are valued the same. Publication “tiers” tell the story

• 1st tier - IEEE S&P, USENIX Sec, CCS, TISSEC, JCS
  ‣ 1.5 NDSS

• 2nd tier - ACSAC, ACNS, ESORICS, CSF, RAID, TOIT

• 3rd tier - SecureComm, ICISS

• 4th tier - HICSS
  ‣ SCilgen (WMSCI 2005)
Journal publication

• The editor-in-chief (EIC) receives the papers as they are submitted.

• The papers are assigned to associate editors for handling.

• Anonymous reviewers rate the paper:
  ‣ Accept without changes
  ‣ Minor revision
  ‣ Major revision
  ‣ Reject
• The **PC Chair** is the person who marshals the reviewing and decisions of a conference. This is different than the *general chair*.

• **PC members** review, rate and discuss, the paper, then vote on which ones are accepted.

• The **acceptance rate** is the ratio of accepted to submitted papers.
Paper evaluation

• A paper is evaluated on
  ‣ Novelty
  ‣ Correctness
  ‣ Impact
  ‣ Presentation
  ‣ Relevance
  ‣ “hotness”
Parts of a paper

• Parts of paper (vast generalization)

1. Abstract
2. Introduction
3. Related Work/Background
4. Solution/Problem
5. Evaluation/Analysis/Experiment
6. Discussion (often, but not always)
7. Conclusions
Abstract

• One sentence each for:
  ‣ Area
    • Topic of work
  ‣ Problem
    • What’s the issue?
  ‣ Solution
    • How do you propose to address the problem?
  ‣ Methodology
    • What’s the experiment?
  ‣ Results
    • What did you find?
  ‣ Take Away: Lesson
Introduction

• **One paragraph each on:**
  - Area
    ‣ More elaborate
  - Problem
    ‣ Scenario
  - Why is problem not solved
    ‣ Brief of related work or the challenge
  - Proposed insight (“In this paper, …”)
    ‣ What is the experiment?
  - Contributions -- What will the reader learn?
  - Boilerplate outline (?)
Related work/Background

• This is a statement of the work that led to this one.
  ‣ who this work relies on
  ‣ who has done work in the area
  ‣ areas that inspired this work (not just technology)

• There are several reasons for related work section:
  ‣ Motivate the current work
  ‣ Differentiate from past work
  ‣ Establish “bona fides”

• Background
  ‣ Outline the Problem
    • May use an example scenario
  ‣ Material Related to the Solution
    • Why hasn’t it been solved
Background and Experiment

• Experiment
  ‣ Means of showing truth
  ‣ Big Insight -- Hypothesis -- Claim
    • Show why it is interesting
  ‣ Expected Results
    • Informal proof/argument that is true

• Experiment types
  ‣ Empirical - measure some aspect of the solution
  ‣ Analytical - prove something about solution
  ‣ Observational - show something about solution
Implementation and Results

• Implementation: Experimental Platform
  ‣ Exact specification of platform
  ‣ Design may have more than implementation -- what did you implement?
  ‣ How are key design features/mechanisms implemented?

• Results
  ‣ Summarize -- what do the results mean?
  ‣ Specific experiments
    • We did X, saw Y
  ‣ What do the experiments prove
  ‣ What other experiments would you want to do based on these results?
Conclusion

• Like the abstract in past tense

• Problem
  ‣ What was the problem?

• Solution
  ‣ What was the insight and why was it expected to work?

• Method and Results
  ‣ What did you find?

• Take away: Lesson

• Future work
Hint

• Intro: tell them what you are going to tell them
• Body: tell them
• Conclusion: tell them what you told them.