Table of Contents

Agenda

8:30 AM	Continental breakfast.
9:00 AM	Introduction and overview.
9:10 AM	UC Davis and the Center.
9:30 AM	NSF and the I/UCRC Program; how to join.
10:15 AM	The Center for Information Protection.
11:00 AM	Break.
11:15 AM	Potential projects:

11:15AM	Deception and Consistency (Matt Bishop)
11:40AM	Information Visualization (Kwan-Liu Ma)
12:05PM	Davis Social Links: P2P, Online Social Network, and Autonomous Community (S. Felix Wu)
12:30PM	Mobile Web Phishing Defense (Francis Hsu)
12:55PM	Modeling Vulnerabilities: from Buffer Overflows to Insider Threat (Sophie Engle)
1:20PM	Systematic and Practical Methods for Computer Forensics and Attack Analysis (Sean Peisert)
1:45PM	Secure Programming Education (Matt Bishop)
2:10PM	Mithridates: Peering into the Future with Idle Cores (Earl Barr)
2:35PM	Detecting Sensitive Data Exfiltration by an Insider Attack (Dipak Ghosal)

12:15PM	Working lunch.
3:00PM	Break.
3:15PM	Discussion of goals for the center.
3:45PM	Discussion of potential projects, LIFE forms, and other project ideas.
4:30PM	Discussion with NSF Program Director.

Industry and Government

Gene Kim
Co-Founder and Chief Technology Officer
Tripwire, Inc.
326 SW Broadway, 3rd Floor
Portland, OR 97205
phone: (503) 276-7500
email: genek@tripwire.com

Morris Moore
Vice President, Security Technology
Motorola, Inc.
6500 River Place Blvd., Building 7
MD: RP-1E
Austin, TX 78730
phone: (512) 427-7305
email: Morris.Moore@motorola.com

Gary Morgan
Public Private Partnerships Program Lead
Pacific Northwest National Laboratories
902 Battelle Boulevard
P. O. Box 999
Richland, WA 99352
phone: (509) 375-2373
email: gary.morgan@pnl.gov

Alan Paller
Director of Research
SANS
8120 Woodmont Ave., #205
Bethesda, MD 20814
phone: (301) 951-0102 x 108
email: apaller@sans.org

Alex Stamos
Principal Partner
Information Security Partners, LLC
phone: (415) 378-9580
email: alex@isecpartners.com

Shirley Ann Stern
Security Program Manager, Global Product Security
Oracle
500 Oracle Parkway
M/S 5OP948
Redwood Shores, CA 94065
phone: (650) 607-5887
email: shirley-ann.stern@oracle.com

Jacob West
Manager, Security Research Group
Fortify Software
2215 Bridgepointe Parkway, Suite 400
San Mateo, CA 94404
phone: (650) 358-5625
email: jwest@fortify.com

National Science Foundation

Rita Rodriguez
I/UCRC Program Director
National Science Foundation
4201 Wilson Boulevard
Arlington, VA 22230
phone: (703) 292-8950
email: rrodrigu@nsf.gov

Alex Schwarzkopf
Consultant and former I/UCRC Program Director
National Science Foundation
4201 Wilson Boulevard
Arlington, VA 22230
phone: (703) 292-5359
email: aschwarz@nsf.gov

Center for Information Protection

Doug Jacobson
Director, Center for Information Protection
Professor, Iowa State University
2419 Cover Hall
Department of Computer and Electrical Engineering
Iowa State University
Ames, IA 50011
phone: (515) 294-8307
email: dougj@iastate.edu

University of California at Davis

Nina Amenta
Professor and Vice-Chair, Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562
phone: (530) 754-5377
email: amenta@cs.ucdavis.edu

Earl Barr
Research Assistant, Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562
email: etbarr@ ucdavis.edu

Matt Bishop
Professor, Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562
phone: (530) 752-8060
email: bishop@cs.ucdavis.edu

Hao Chen
Professor, Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562
phone: (530) 754-5375
email: hchen@cs.ucdavis.edu

Sophie Engle
Research Assistant, Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562
email: sjengle@ucdavis.edu

Dipak Ghosal
Professor, Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562
phone: (530) 754-9251
email: ghosal@cs.ucdavis.edu

Greg Gibbs
Director of Development, College of Engineering
University of California at Davis
One Shields Ave.
Davis, CA 95616
phone: (530) 754-9673
email: glgibbs@ucdavis.edu

Karl Levitt
Professor, Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562
phone: (530) 752-0832
email: levitt@cs.ucdavis.edu

Kwan-Liu Ma
Professor, Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562
phone: (530) 752-6958
email: ma@cs.ucdavis.edu

Karen McDonald
Associate Dean, Research and Graduate Studies, College of Engineering
University of California at Davis
One Shields Ave.
Davis, CA 95616
phone: (530) 752-0559
email: kamcdonald@ucdavis.edu

Sean Peisert
Post-Doctoral Scholar, Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562
phone: (530) 752-2149
email: peisert@cs.ucdavis.edu

S. Felix Wu
Professor, Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562
phone: (530) 754-7070
email: wu@cs.ucdavis.edu

Deception and Consistency

Matt Bishop, Vicentiu Neagoe
bishop@cs.ucdavis.edu

The use of deception is one of the many defensive techniques being explored today. In the past, defenders of systems have used deception haphazardly, but now researchers are developing systematic methods of deception. The cornerstone of these methods is consistency: projecting a “false reality”, or “fiction”, that the attacker is to accept as reality. We challenge the necessity of this cornerstone.

This talk presents questions on the need for consistency in deception. We then discuss how to add deceptive mechanisms to the host, and examine two common functions (deleting a file and obtaining the name of the current working directory) to demonstrate the effects of inconsistency in deception, and ways to add it.

Biography: Professor Matt Bishop’s research area is computer security, in which he has been active since 1979. He is especially interested in vulnerability analysis and denial of service problems, but maintains a healthy concern for formal modeling (especially of access controls and the Take-Grant Protection Model) and intrusion detection and response. He has also worked extensively on the security of various forms of the UNIX operating system. He is involved in efforts to improve education in information assurance, and is a charter member of the Colloquium for Information Systems Security Education. His textbook, *Computer Security: Art and Science*, was published by Addison-Wesley in December 2002.

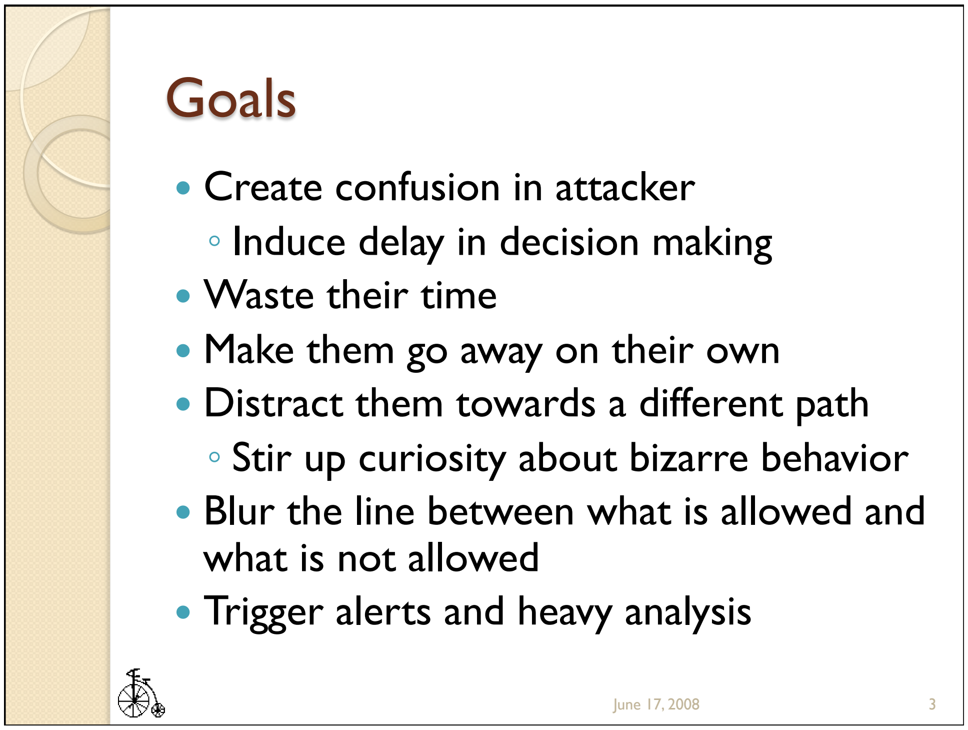
Deception and Consistency

Matt Bishop
Vicentiu Neagoe

Contact Information

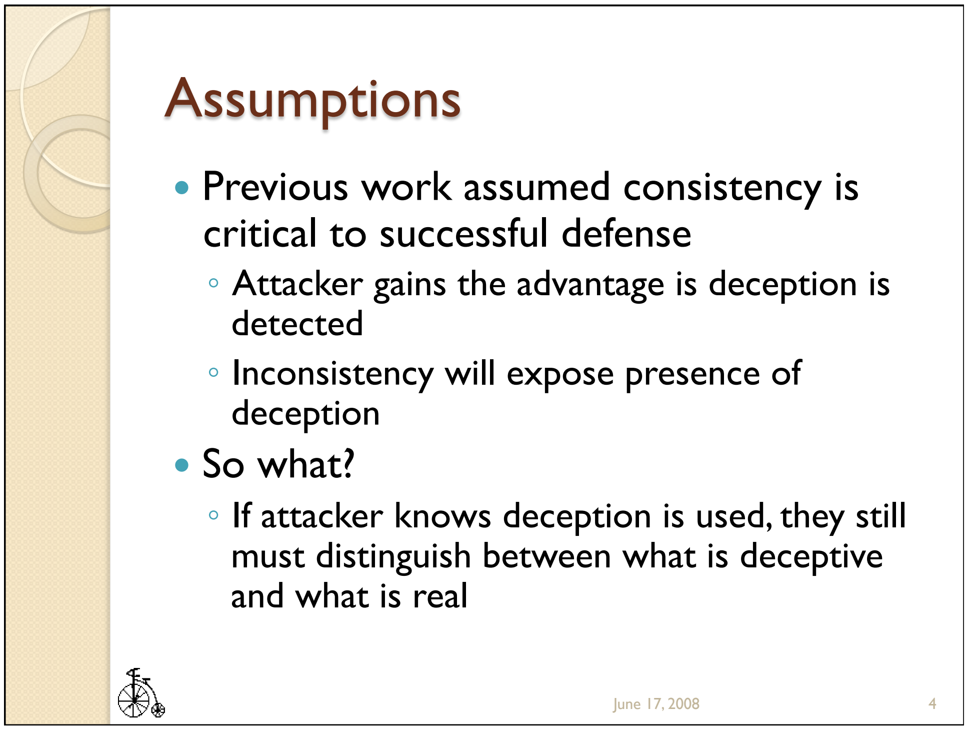
Matt Bishop
Department of Computer Science
University of California at Davis
1 Shields Ave.
Davis, CA 95616-8562

phone: (530) 752-8060
email: bishop@cs.ucdavis.edu
www: http://seclab.cs.ucdavis.edu/~bishop



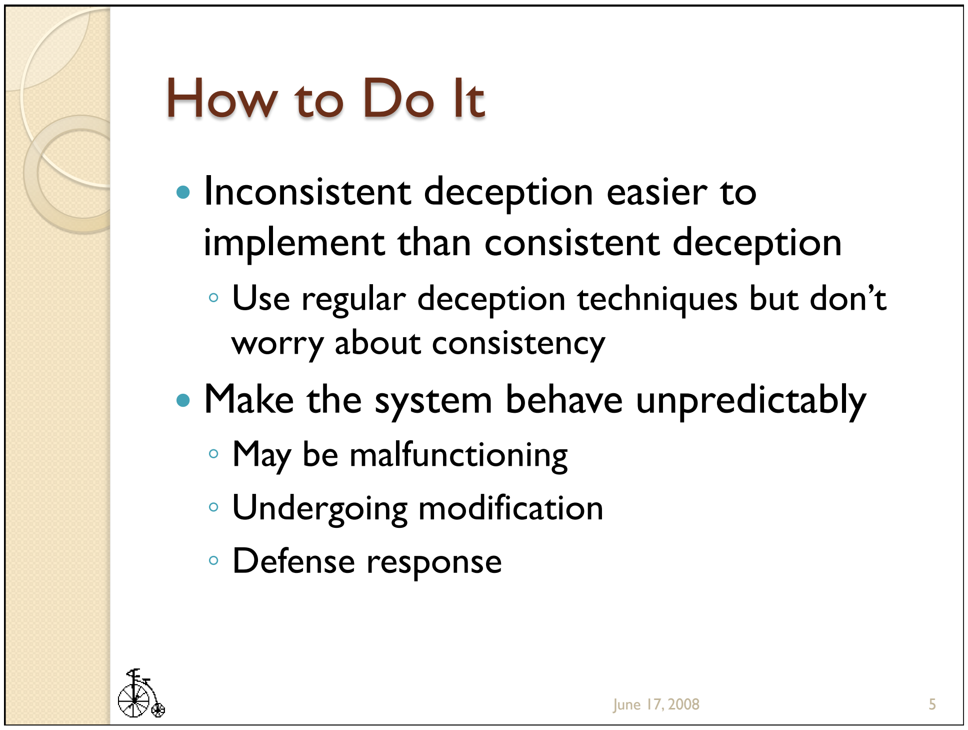
Goals

- Create confusion in attacker
  - Induce delay in decision making
- Waste their time
- Make them go away on their own
- Distract them towards a different path
  - Stir up curiosity about bizarre behavior
- Blur the line between what is allowed and what is not allowed
- Trigger alerts and heavy analysis



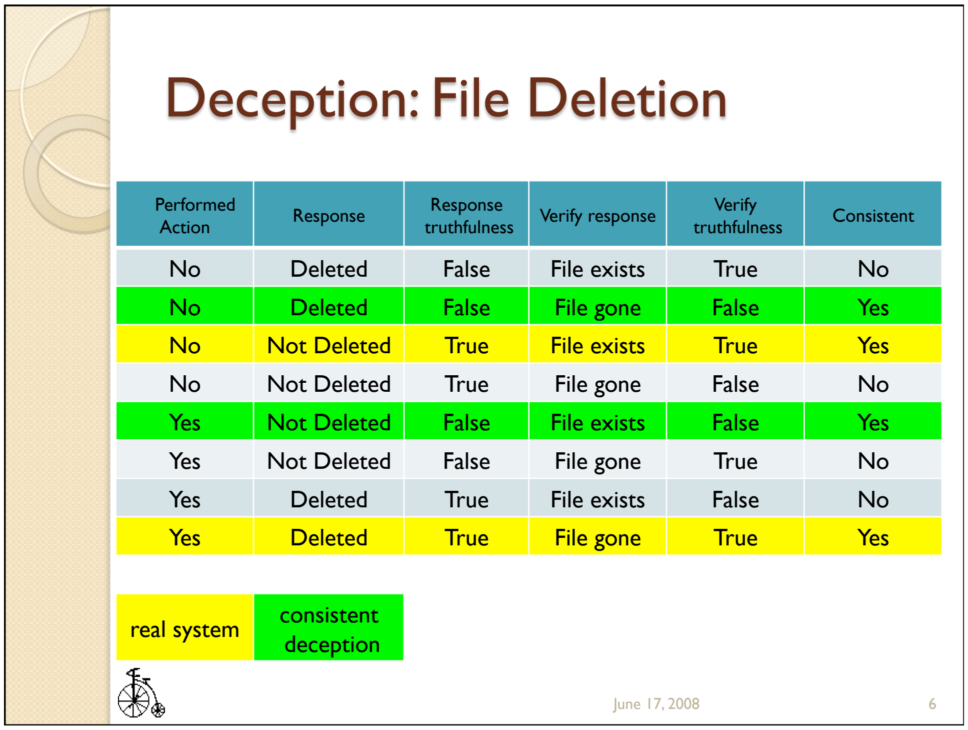
Assumptions

- Previous work assumed consistency is critical to successful defense
  - Attacker gains the advantage is deception is detected
  - Inconsistency will expose presence of deception
- So what?
  - If attacker knows deception is used, they still must distinguish between what is deceptive and what is real



How to Do It

- Inconsistent deception easier to implement than consistent deception
  - Use regular deception techniques but don't worry about consistency
- Make the system behave unpredictably
  - May be malfunctioning
  - Undergoing modification
  - Defense response

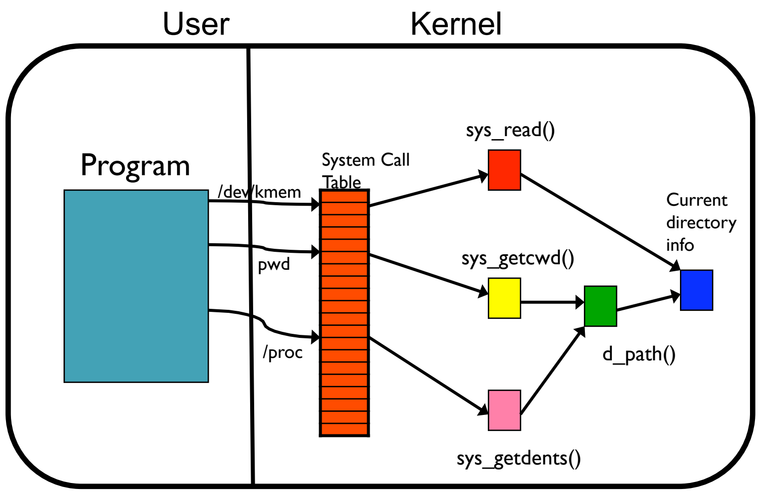


Deception: File Deletion

Performed Action	Response	Response truthfulness	Verify response	Verify truthfulness	Consistent
No	Deleted	False	File exists	True	No
No	Deleted	False	File gone	False	Yes
No	Not Deleted	True	File exists	True	Yes
No	Not Deleted	True	File gone	False	No
Yes	Not Deleted	False	File exists	False	Yes
Yes	Not Deleted	False	File gone	True	No
Yes	Deleted	True	File exists	False	No
Yes	Deleted	True	File gone	True	Yes

🟨 real system	🟩 consistent deception

Consistent Deception



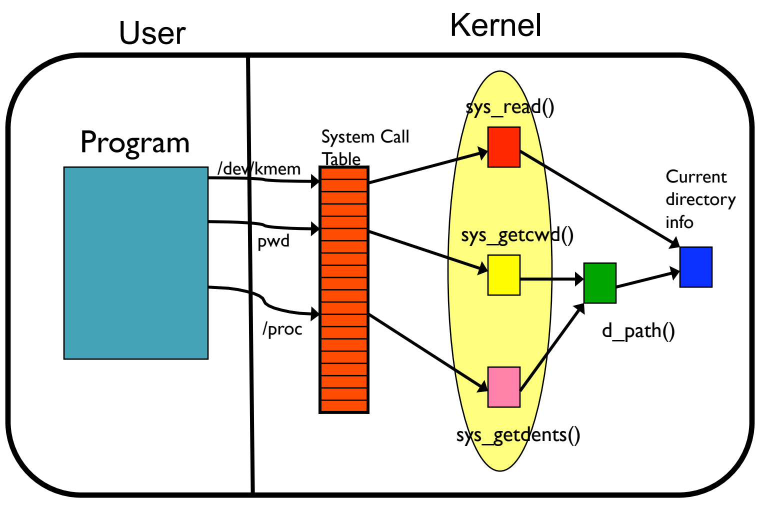
Types of Inconsistency

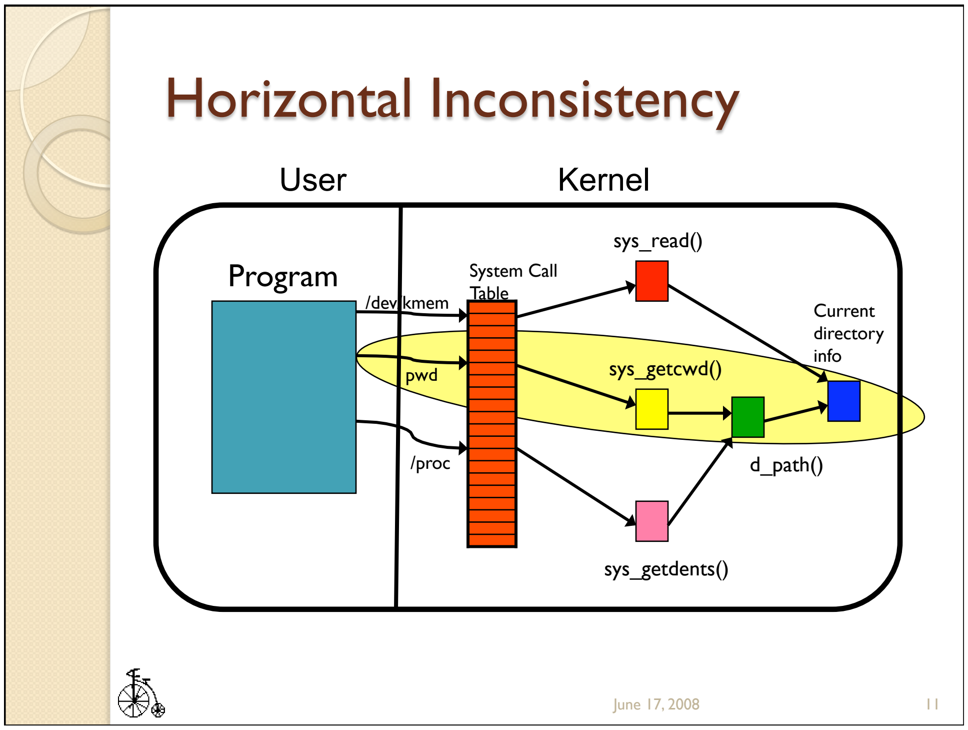
- Vertical – separate paths return different answers
- Horizontal – same path returns different answer

Example

- Process needs to determine its current working directory
  - Relative path names interpreted with respect to that directory
  - Is current working directory the real one or one created as part of a deception?
    - In the latter case, the system wants to lie about the name

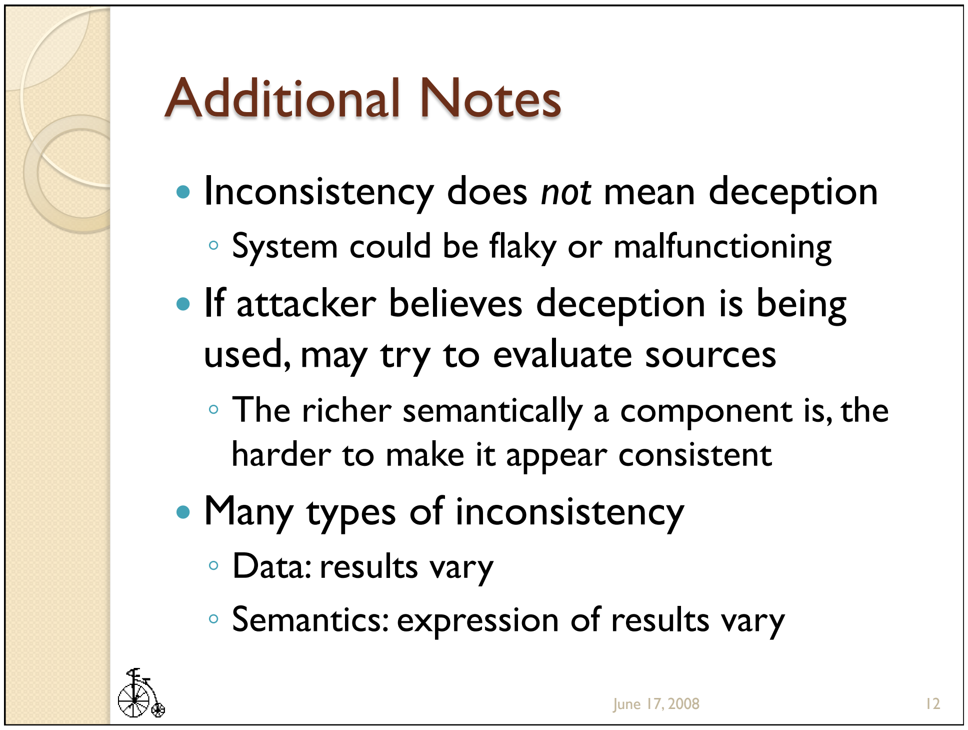
Vertical Inconsistency





Horizontal Inconsistency

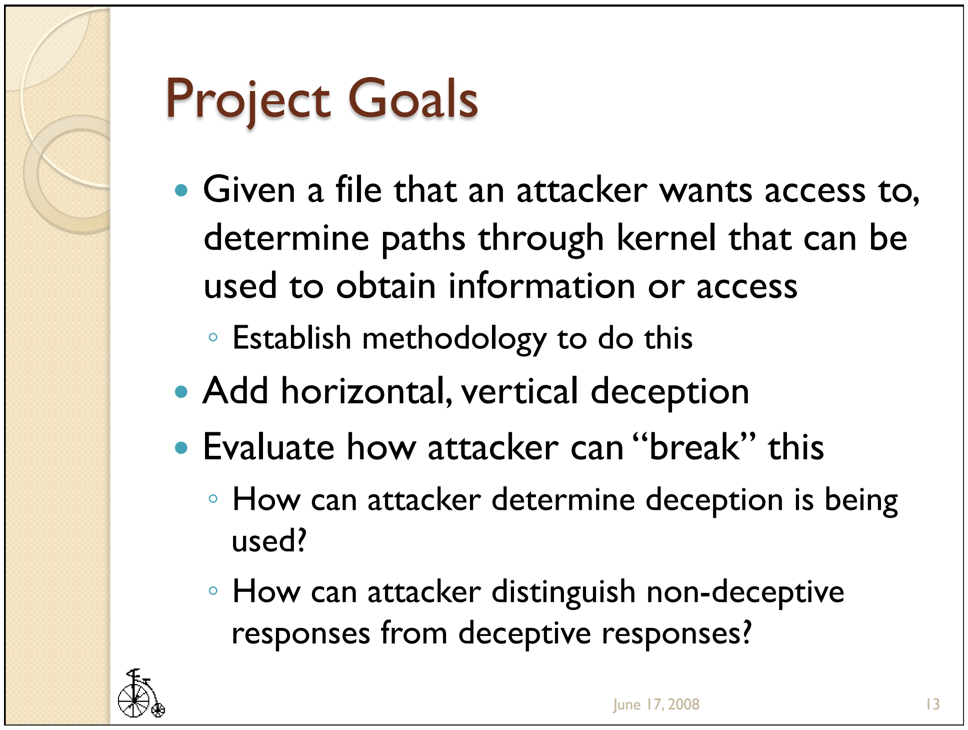
June 17, 2008 11



Additional Notes

- Inconsistency does *not* mean deception
  - System could be flaky or malfunctioning
- If attacker believes deception is being used, may try to evaluate sources
  - The richer semantically a component is, the harder to make it appear consistent
- Many types of inconsistency
  - Data: results vary
  - Semantics: expression of results vary

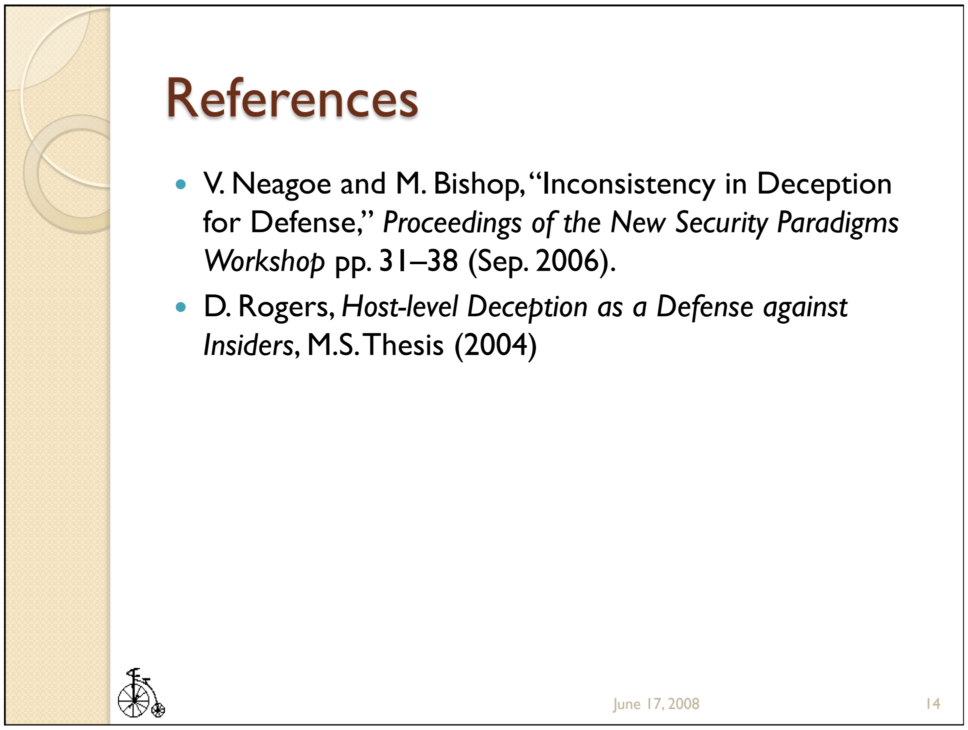
June 17, 2008 12



Project Goals

- Given a file that an attacker wants access to, determine paths through kernel that can be used to obtain information or access
  - Establish methodology to do this
- Add horizontal, vertical deception
- Evaluate how attacker can "break" this
  - How can attacker determine deception is being used?
  - How can attacker distinguish non-deceptive responses from deceptive responses?

June 17, 2008 13



References

- V. Neagoe and M. Bishop, "Inconsistency in Deception for Defense," *Proceedings of the New Security Paradigms Workshop* pp. 31–38 (Sep. 2006).
- D. Rogers, *Host-level Deception as a Defense against Insiders*, M.S. Thesis (2004)

June 17, 2008 14

Information Visualization

Kwan-Liu Ma
ma@cs.ucdavis.edu

Information collected for security assurance or business competitive advantage exhibits exponential growth, a daunting challenge we must address in order to extract knowledge from and maximize utilization of all the available information. Visualization, proving very effective for comprehending enormous amounts of data in many other domains, offers a promising solution for this pressing problem. This presentation gives an overview of UCD VIDI group‘s information visualization research.

Biography: Professor Ma's research interests include scientific visualization, information visualization, computer graphics, user interface design, and high-performance computing. He is the recipient of an NSF PECASE award and the Schlumberger Foundation Technical award.

Davis Social Links: P2P, Online Social Network, and Autonomous Community

S. Felix Wu
wu@cs.ucdavis.edu

In this talk, we will discuss the impact of Internet architecture design on network security. In the past few years, there have been many attempts to develop solution to protect our networked system against large-scale attacks such as worm, DDoS, and spam. However, it seems to us (and more and more clearly) that most, if not all, of the proposed solutions are not likely to be effective, given the growth of attacks in numbers and depth. Therefore, the network community has been trying to understand the fundamental issues and the root cause for these large-scale network attacks. One possible idea, currently being actively developed at UC Davis, is called DSL (Davis Social Links). Under DSL, we integrate the concepts of P2P, social networks, and trust management into the network layer, while we remove the requirement of global network identity (e.g., IP addresses or even email addresses, for the context of spam). While we are still in a very early stage regarding DSL, we will go through a few examples of DSL as well as technical considerations.

Biography: Professor Wu's research focuses on network security, specifically intrusion detection and protection for network protocols such as OSPF, BGP, IPsec, TCP, HTTP and 802.11. The nature of his research is very "experimental", meaning that he builds prototype systems and performs experiments to validate and evaluate new architectural concepts for the security of our Internet.

Mobile Web Phishing Defense

Francis Hsu
fhsu@cs.ucdavis.edu

Mobile devices with embedded browsers allow users to enjoy the same web resources they have on traditional computing platforms, but also expose them to the same problems. We examined the migration of the browser to mobile devices and the changes that affect a user's vulnerability to phishing attacks. Due to inherent hardware limitations on the platform, browser designers alter elements found in traditional browsers that normally aid users in defending against phishing attacks. Our user study identified and demonstrated potential phishing attacks that could successfully fool users into giving up their credentials. We propose examining changes to be made in browser, website and network design to create user-friendly anti-phishing solutions.

A major factor contributing to the success of phishing attacks on the web is our reliance on password authentication. Mobile devices connected to cellular networks do provide a resource not found in traditional network connections—the authentication of the device itself to the cellular network. To leverage the cellular network infrastructure, we have designed WebCallerID, a Web authentication scheme using mobile phones as authentication tokens and cellular network providers as trusted identify providers. The scheme eliminates users participation from the authentication process and so prevents security mistakes that could expose them to phishing attacks. Mobile devices have access to other bits of information about a user (GPS, voice, camera, local wireless networks) that we envision a multi-factor authentication system can use with WebCallerID to provide reliable and usable authentication services.

Advisor: Prof. Hao Chen, hchen@cs.ucdavis.edu

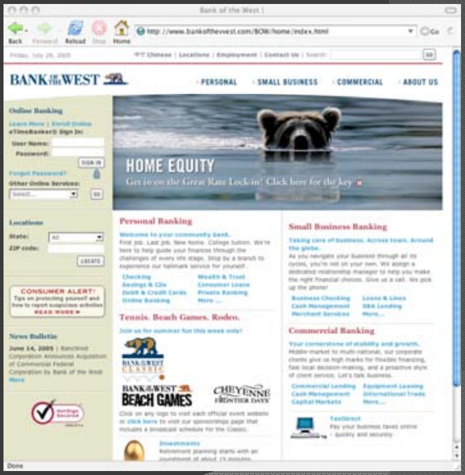
Mobile Web Phishing Defense

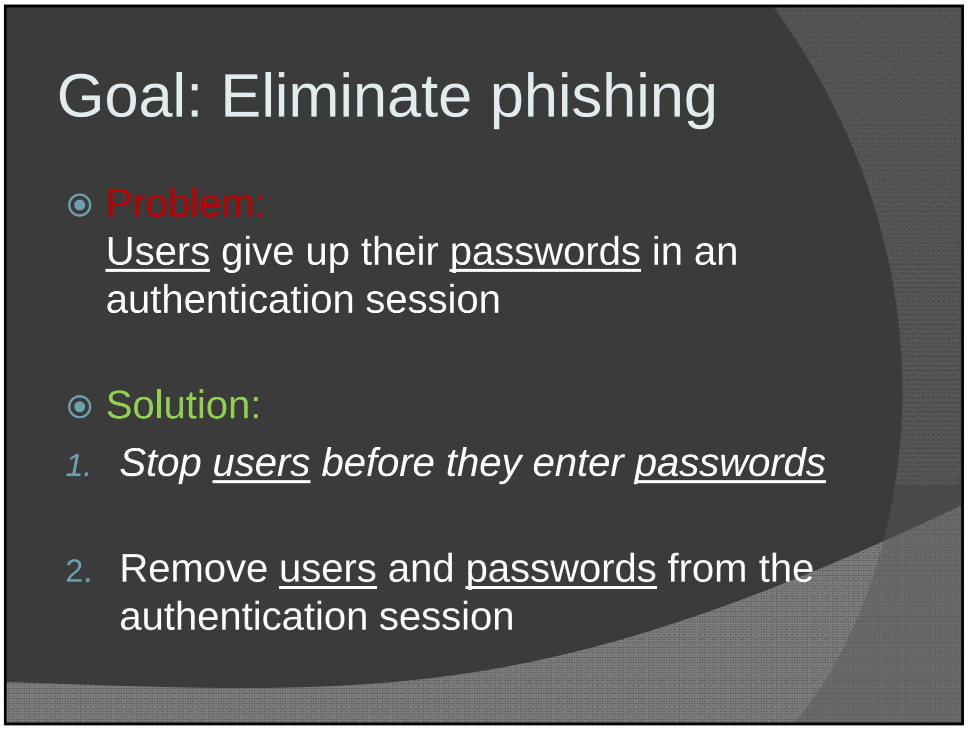
Francis Hsu, Yuan Niu, Hao Chen
{fhsu, niu, hchen}@cs.ucdavis.edu
Computer Science, UC Davis

UCDAVIS
UNIVERSITY OF CALIFORNIA

Phishing

- Human factors problem – users give up credentials to the wrong party
- 2 million victims and $1.2 billion in losses for US banks in 2003



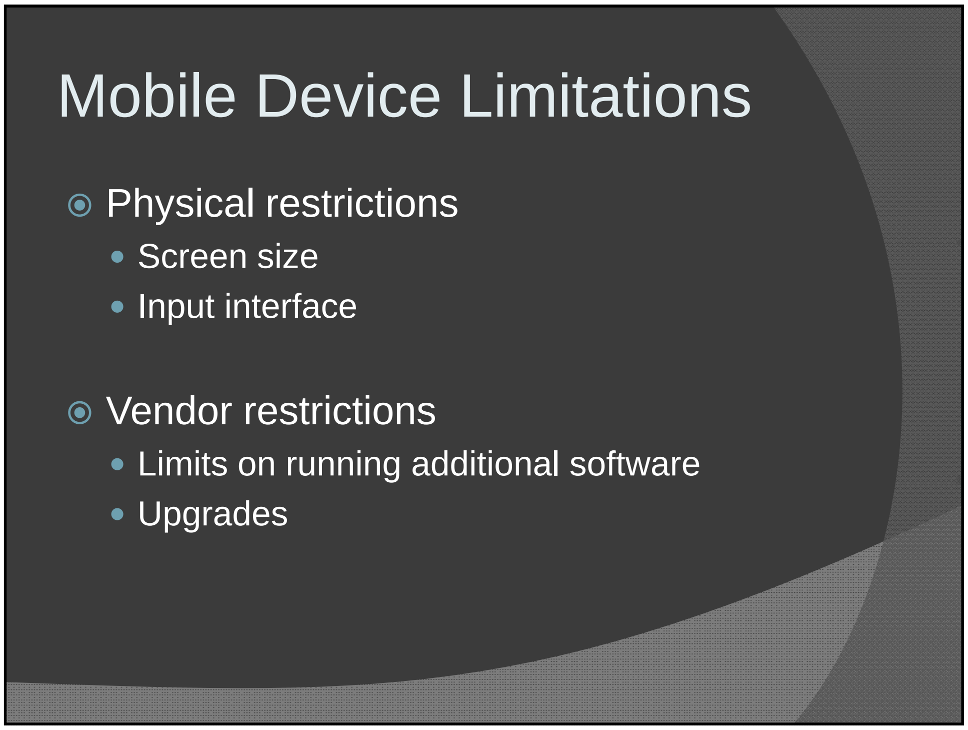


Goal: Eliminate phishing

- Problem:
  Users give up their passwords in an authentication session

- Solution:

1. *Stop users before they enter passwords*
2. Remove users and passwords from the authentication session

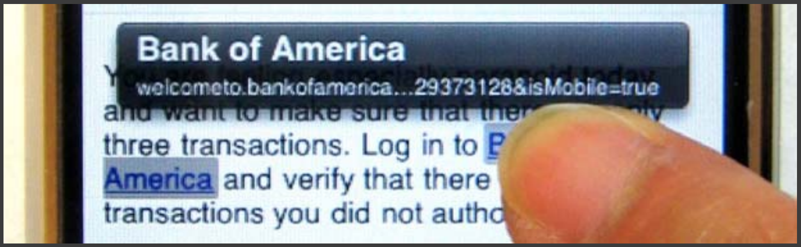


Mobile Device Limitations

- Physical restrictions
  - Screen size
  - Input interface
- Vendor restrictions
  - Limits on running additional software
  - Upgrades

URL Display

http://welcometo.bankofamerica.malweb.org/index.jsp


Bank of America
welcometo.bankofamerica…29373128&isMobile=true
three transactions. Log in to B
America and verify that there
transactions you did not autho

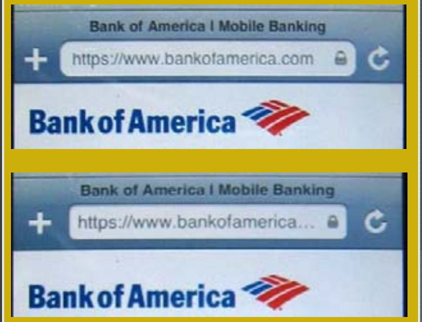
- No https indication
- Truncation from middle – lose effective second level domain
- Long URLs never fully displayed

5

Chrome

- Lack of trusted chrome elements
- Developers actively try to remove chrome from view

Which of these is a forgery?


Bank of America I Mobile Banking
https://www.bankofamerica.com
Bank of America
Bank of America I Mobile Banking
https://www.bankofamerica…
Bank of America

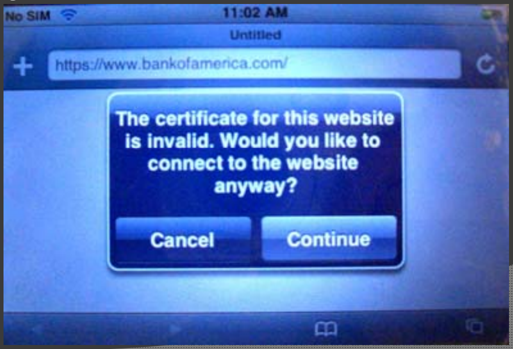
Chrome

Page Content

6

SSL

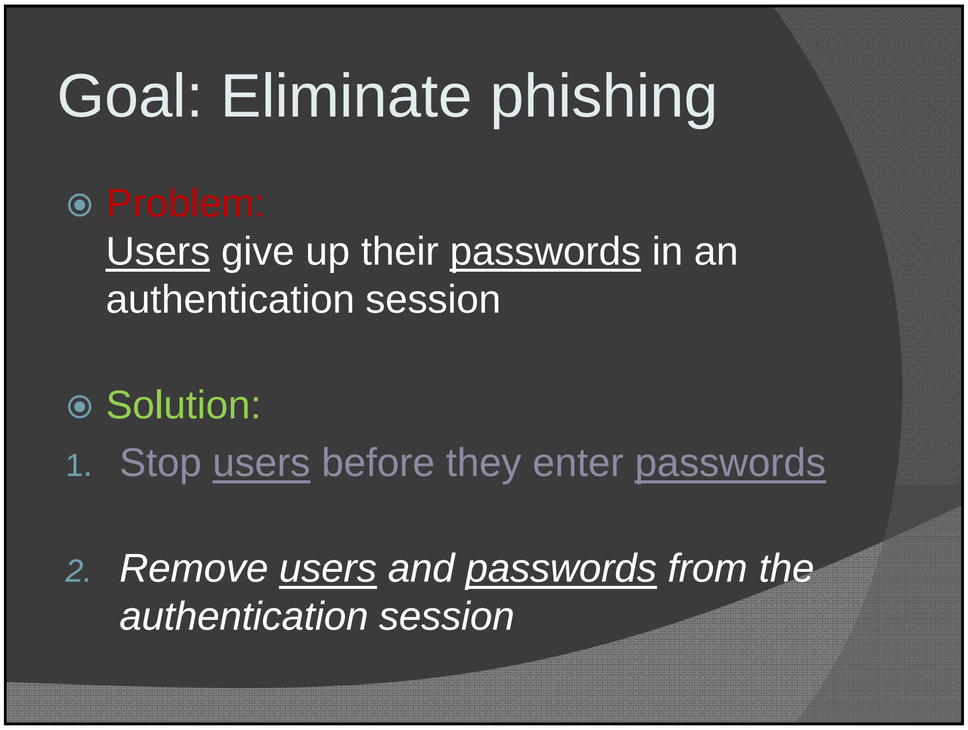
- What can a user do here?
- Even if they wanted to, users can't
  - Examine SSL certificates
  - Diagnose invalid certificates



The certificate for this website is invalid. Would you like to connect to the website anyway?

Mitigation Strategies

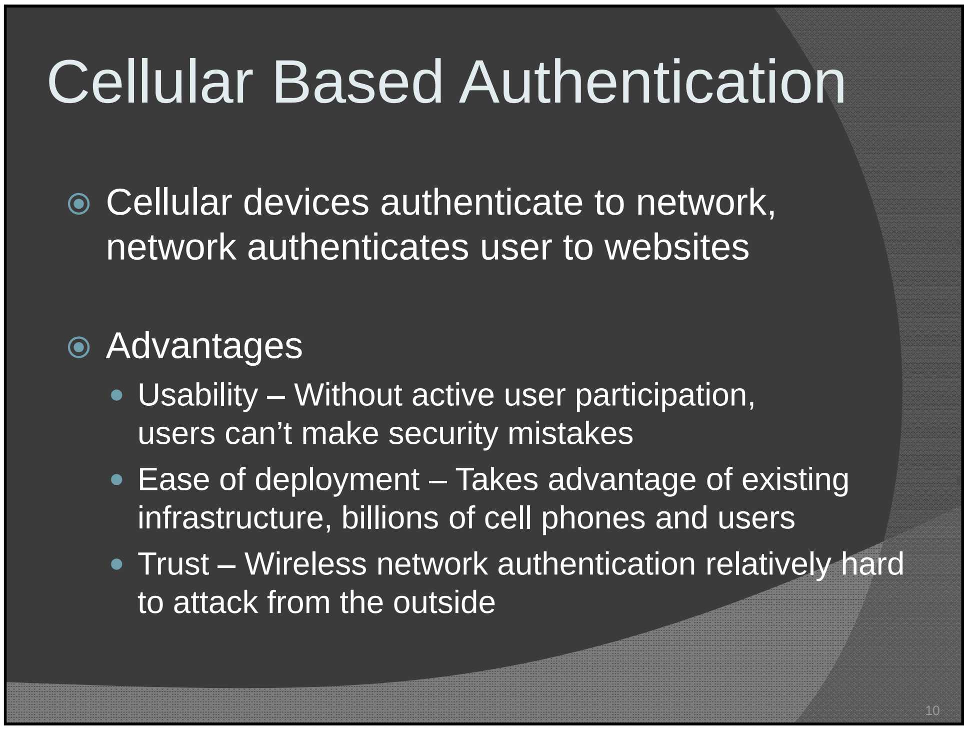
- Browser designer
  - Sites need to identify themselves to the user
  - Keep effective second level domain name
- Website authors
  - Site designers should shorten URLs
- Network administrators
  - Network level anti-phishing proxy filters



Goal: Eliminate phishing

- Problem:
  Users give up their passwords in an authentication session
- Solution:

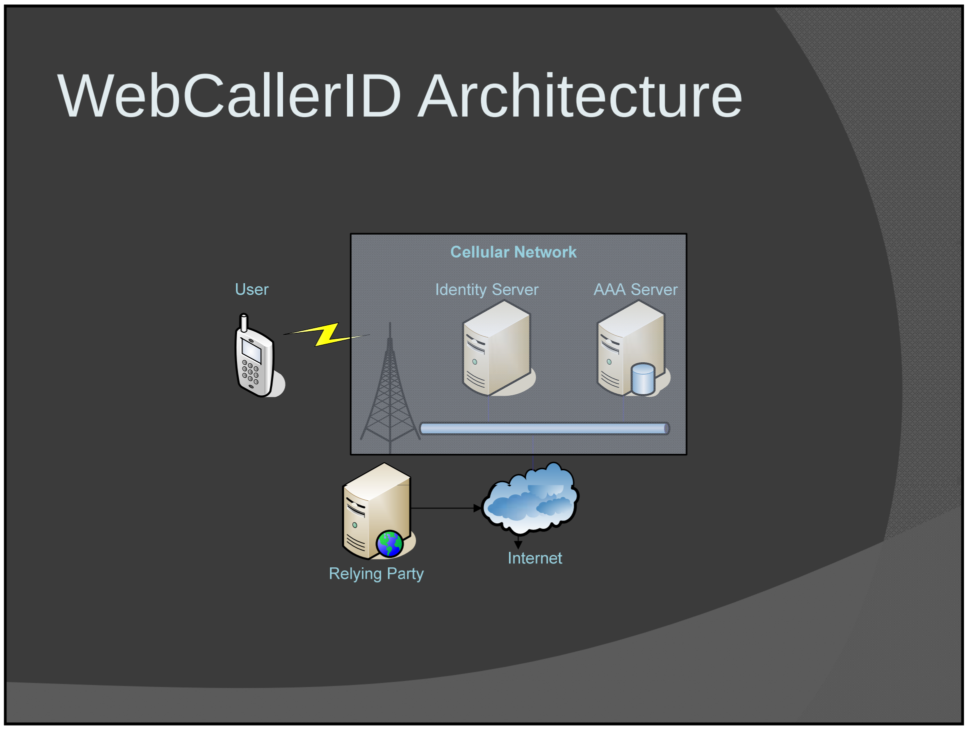
1. Stop users before they enter passwords
2. *Remove users and passwords from the authentication session*



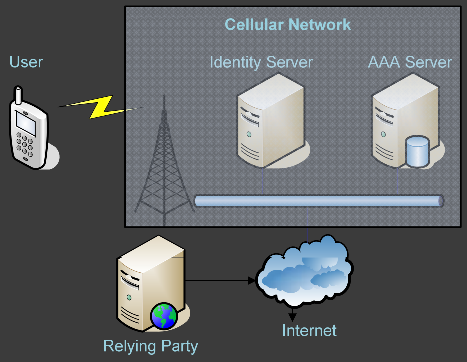
Cellular Based Authentication

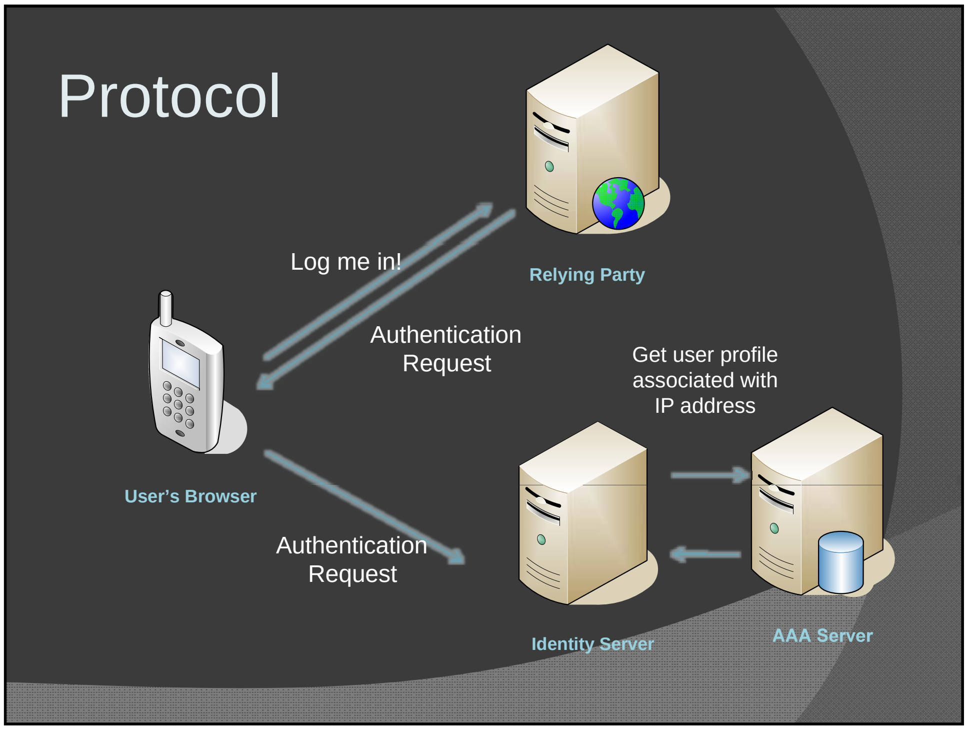
- Cellular devices authenticate to network, network authenticates user to websites
- Advantages
  - Usability – Without active user participation, users can't make security mistakes
  - Ease of deployment – Takes advantage of existing infrastructure, billions of cell phones and users
  - Trust – Wireless network authentication relatively hard to attack from the outside

10

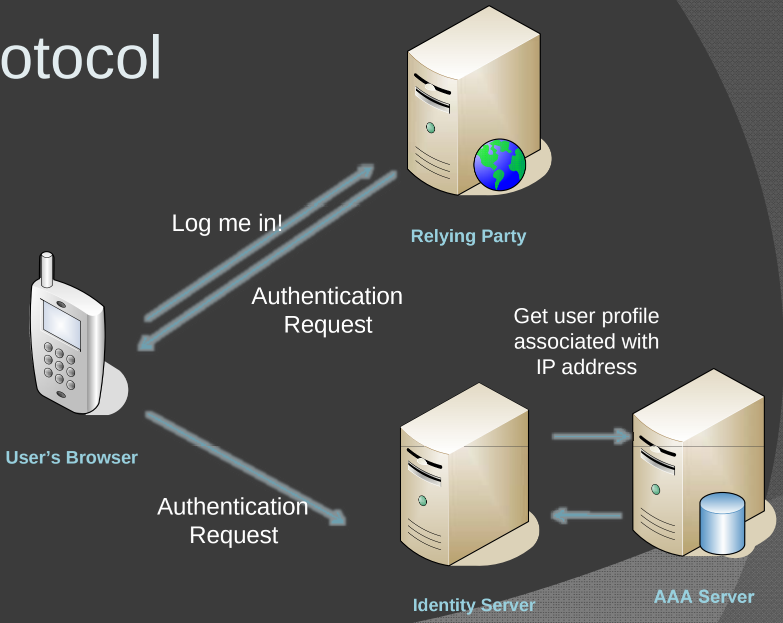


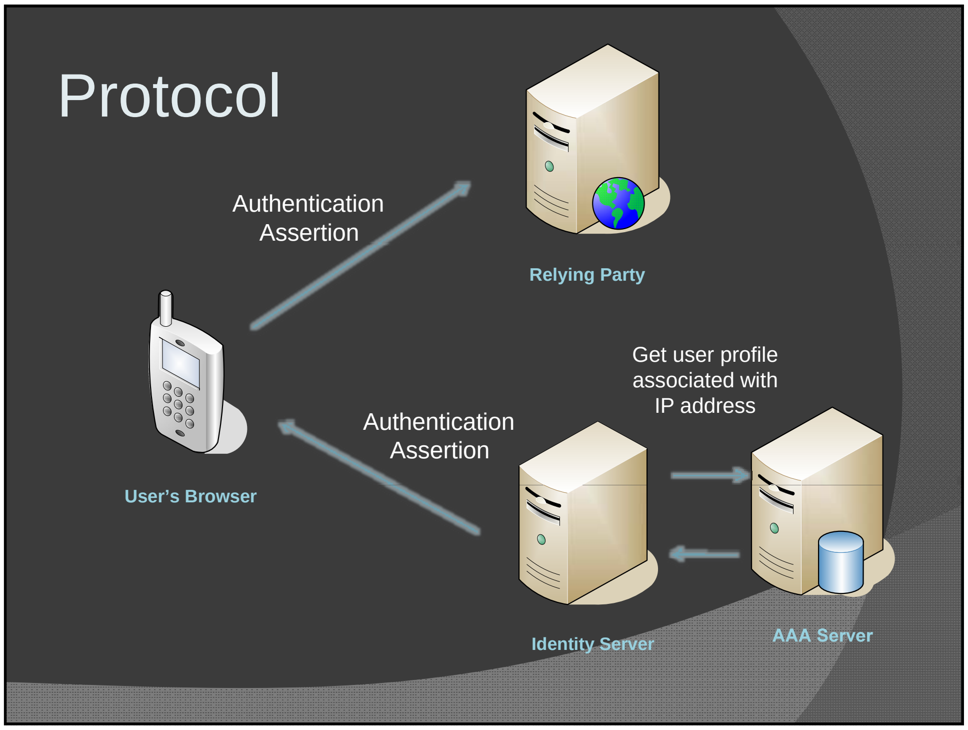
WebCallerID Architecture



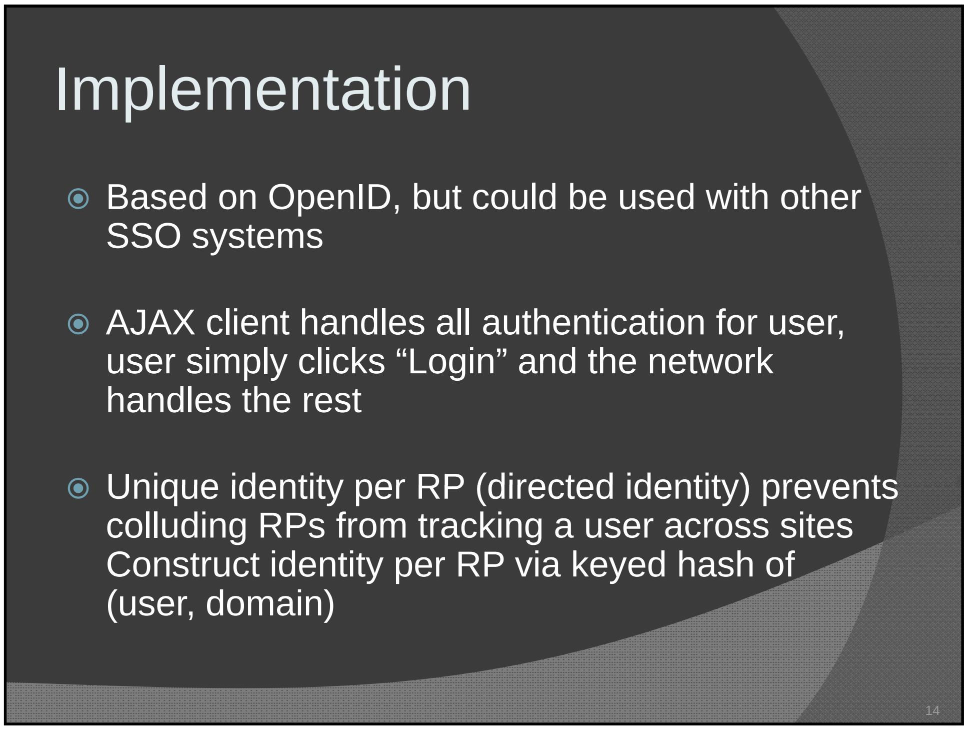


Protocol





Protocol



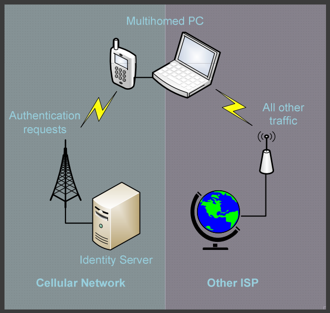
Implementation

- Based on OpenID, but could be used with other SSO systems
- AJAX client handles all authentication for user, user simply clicks “Login” and the network handles the rest
- Unique identity per RP (directed identity) prevents colluding RPs from tracking a user across sites
  Construct identity per RP via keyed hash of (user, domain)

14

Deployment

- No changes needed for user clients
- No changes needed for OpenID enabled relying parties
- Works with
  - cell phone based browsers
  - PCs with cellular modem
  - PCs with a tethered phone

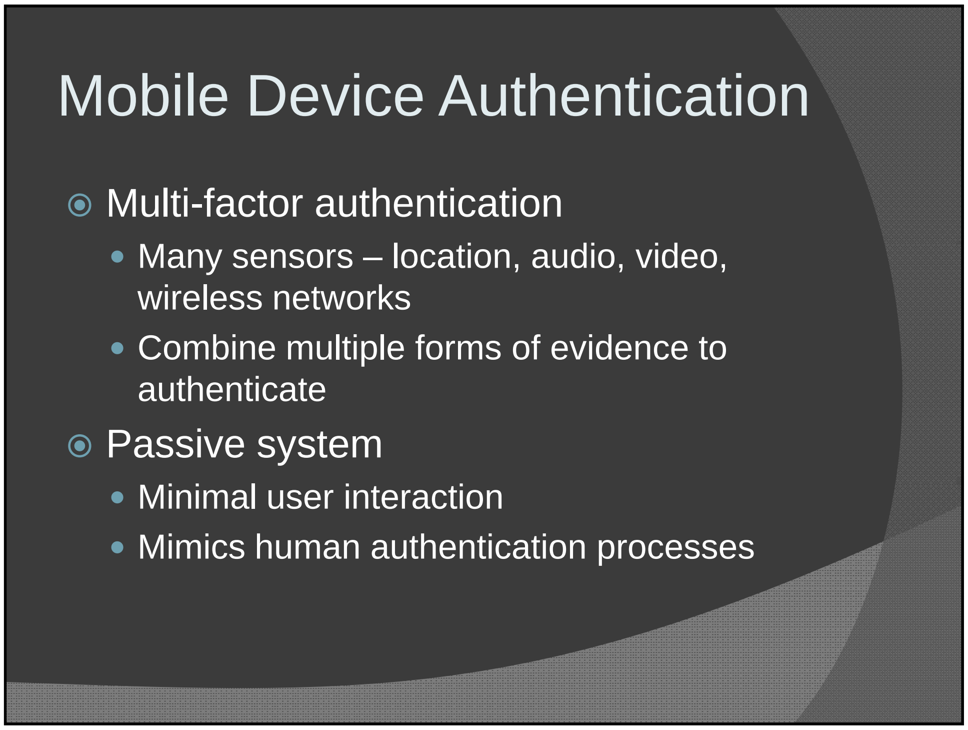


Multihomed usage scenario

Security Benefits

- Users don’t need to:
  - Create and remember good passwords
  - Identify malicious relying parties
  - Carry another physical token
- Websites don’t need to:
  - Store and handle user authentication data
  - Worry about phishing sites stealing valid credentials

16



Mobile Device Authentication

- Multi-factor authentication
  - Many sensors – location, audio, video, wireless networks
  - Combine multiple forms of evidence to authenticate
- Passive system
  - Minimal user interaction
  - Mimics human authentication processes

Modeling Vulnerabilities: from Buffer Overflows to Insider Threat

Sophie Engle
sjengle@ucdavis.edu

This proposal explores how to model all types of vulnerabilities, from traditional vulnerabilities such as buffer overflows to vulnerabilities involving covert channels, social engineering, and insider threat. To achieve this, we look at expanding the Unifying Policy Hierarchy (Carlson 2006) to other areas of security. With a unified formal model that captures these aspects, we can perform more comprehensive threat analysis for a system in a non *ad hoc* manner.

Advisor: Prof. Matt Bishop, bishop@cs.ucdavis.edu

Modeling Vulnerabilities
from buffer overflows to insider threat

Sophie Engle
NSF I/UCRC CIP Meeting

UC Davis Kemper Hall 1008 · Tuesday June 17 2008

Motivation

Motivation

What does it mean for a system to be secure?

Motivation

What does it mean for a system to be secure?

physically secure?

Motivation

What does it mean for a system to be secure?

cannot be misused by insiders?

Motivation

What does it mean for a system to be secure?

only authorized persons have access?

only authorized user accounts have access?

Motivation

What does it mean for a system to be secure?

no buffer overflow bugs?

no buffer overflow vulnerabilities?

Motivation

What do all of these examples have in common?

Motivation

What do all of these examples have in common?

POLICY

Motivation

What does it mean for a system to be secure?

physically secure?

policy defines…
the physical requirements of the system

Motivation

What does it mean for a system to be secure?

cannot be misused by insiders?

policy defines…
how the system is *intended* to be used

Motivation

What does it mean for a system to be secure?

only authorized persons have access?
only authorized user accounts have access?

policy defines…
who is authorized for what type of access

Motivation

What does it mean for a system to be secure?

no buffer overflow bugs?

no buffer overflow vulnerabilities?

policy defines…
the difference between bug & vulnerability

Motivation

What does it mean for a system to be secure?

no vulnerabilities

where a *vulnerability* is a set of conditions
that may lead to a potential policy violation

Motivation

How do we define policy?

Background

Background

How do we define policy?

Unifying Policy Hierarchy

(*Adam Carlson, Master's Thesis*)

Unifying Policy Hierarchy

Oracle Policy

– Represents the intent and will of policy makers

– May not be explicitly specified

Example:

Xander is authorized to read file `readme.txt`

Unifying Policy Hierarchy

Feasible Policy

– Represents the intent and will of policy makers

– Takes into account the mechanics and available access controls of the system

Example:

User account `xander` is authorized to read file `readme.txt`

Unifying Policy Hierarchy

Configured Policy

– Represents the policy configured on the machine

Example:

All user accounts are authorized to read file `readme.txt`

Unifying Policy Hierarchy

Actual Policy

– Represents the policy currently in effect on the machine

Example:

No user can read file `readme.txt`

(potentially result of denial of service attack)

Unifying Policy Hierarchy

Oracle Policy

Captures policy maker's intent

Feasible Policy

Considers limitations of system

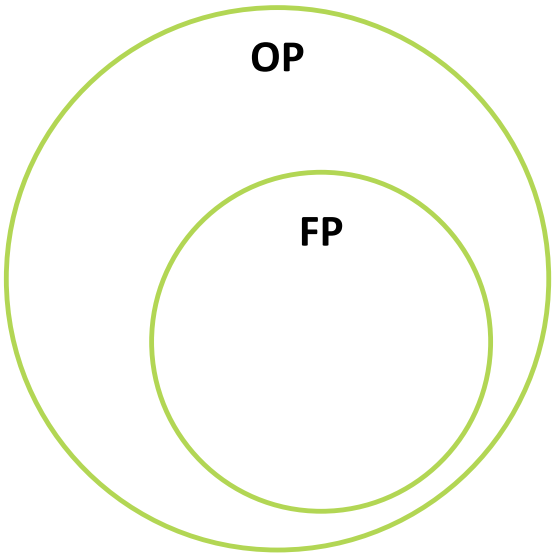
Configured Policy

Policy as configured on system

Actual Policy

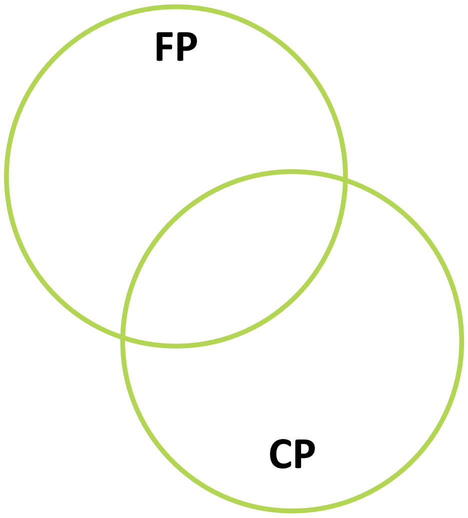
Policy currently in effect on system

Unifying Policy Hierarchy



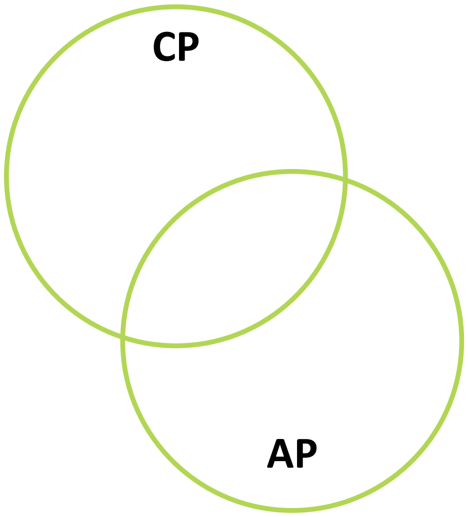
OP ≠ FP
Inherent Vulnerability

Unifying Policy Hierarchy



FP ≠ CP
Configuration Vulnerability

Unifying Policy Hierarchy



CP ≠ AP
Runtime
Vulnerability

Proposal

Proposal

1 Expand application of the hierarchy

Proposal

1 Expand application of the hierarchy

- Insider Threat
- Social Engineering
- Network Viewpoint
- *And more…*

Proposal

1 Expand application of the hierarchy

- **Insider Threat**
- Social Engineering
- Network Viewpoint
- *And more…*

Insider Threat

“exists whenever a lower policy level has *more* authorized privileges than a higher policy level”

Insider Threat

"exists whenever a lower policy level has *more* authorized privileges than a higher policy level"

OP: Yasmin may use the system to read medical records to treat patients.

Insider Threat

"exists whenever a lower policy level has *more* authorized privileges than a higher policy level"

OP: Yasmin may use the system to read medical records to treat patients.

FP: User account `yasmin` may use the system to read medical records.

Proposal

1 Expand application of the hierarchy

- Insider Threat
- Social Engineering
- **Network Viewpoint**
- *And more…*

Network Viewpoint

In original approach, each system has its own associated policy hierarchy.

Network Viewpoint

In original approach, each system has its own associated policy hierarchy.

How do we expand this to a more network-based approach?

Proposal

1 Expand application of the hierarchy

2 **Use model to perform threat analysis**

Threat Analysis

“Gap Analysis”

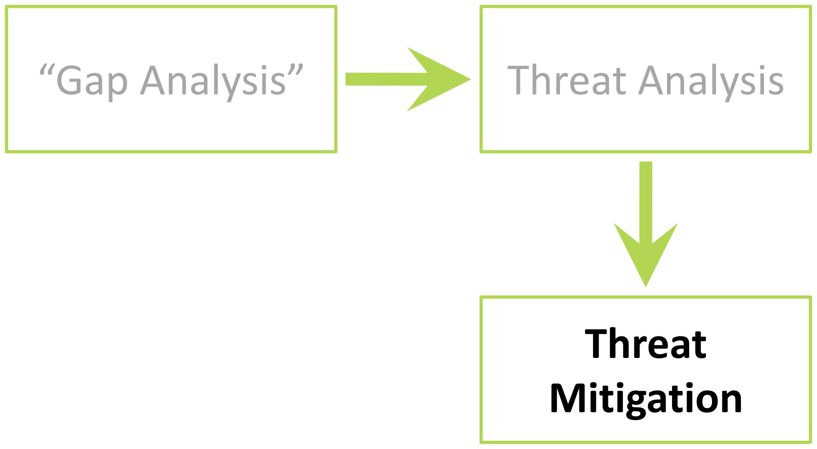
Examine the “gap” between levels of the policy hierarchy, i.e. everywhere two consecutive levels do not match.

Threat Analysis

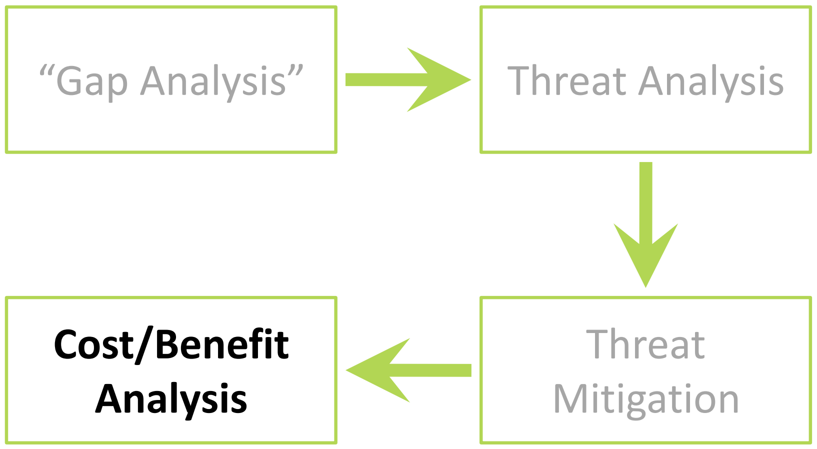
$$\begin{tikzcd}\boxed{\text{“Gap Analysis”}} \arrow[r] & \boxed{\textbf{Threat Analysis}}\end{tikzcd}$$

Next, determine the potential threat caused by these gaps.

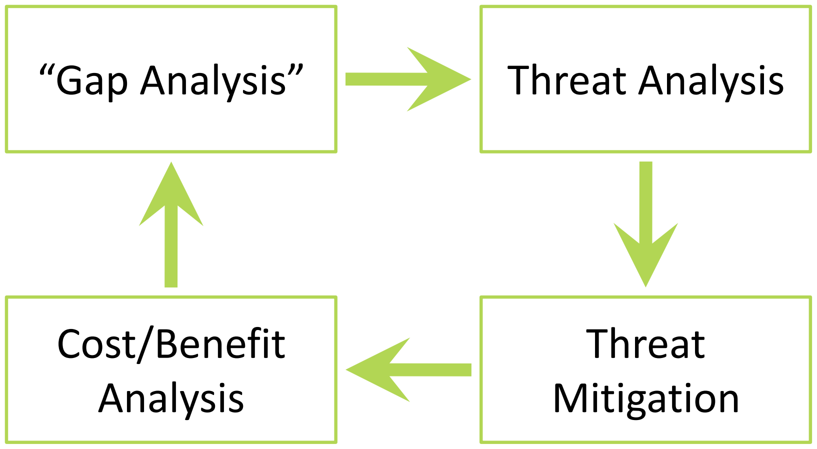
Threat Analysis



Threat Analysis



Threat Analysis



Proposal

1 Expand application of the hierarchy

2 Use model to perform threat analysis

3 **Present findings in a wiki format**

Questions?

Systematic and Practical Methods for Computer Forensics and Attack Analysis

Sean Peisert
peisert@cs.ucdavis.edu

Who attacked this computer system? What actions did they take? What damage did they do? With what degree of certainty, and under what assumptions, do we make these assertions? These questions are asked during the computer attack analysis process, but they are often hard to answer in practice. Computer scientists and security practitioners have made headway on developing functional systems for attack analysis. Some of those systems are based on theoretical models that help to construct complete solutions, but there are serious and important gaps in these systems. The result is an incomplete picture of the attack, or an incorrect analysis of what happened.

The goals of this project are to understand and improve methods used in forensic logging and computer attack analysis. To do this, we plan to extend the Laocoön model of forensics, and modify a system to enable us to implement the model. We will evaluate methods and assumptions used in attack analysis. In particular, we intend to apply these techniques to forensic technology used in the legal system, and to the insider problem.

Biography: Dr. Peisert received his Ph.D. in Computer Science from UC San Diego in 2007. He is currently a postdoctoral scholar at UC Davis, an I3P Fellow, and a Fellow of the San Diego Supercomputer Center (SDSC). In the UC Davis Computer Security Laboratory, he performs research in a number of topics relating to security, including computer forensic analysis, intrusion detection, vulnerability analysis, security policy modeling, electronic voting, and the design of secure systems. Previously, he was a postdoctoral scholar and lecturer in the Computer Science and Engineering department at UC San Diego, a computer security researcher at SDSC, and co-founded a now-defunct software company. He is currently working with Professor Matt Bishop.

Systematic and Practical Methods for Computer Attack Analysis and Forensics

Dr. Sean Peisert
UC Davis Computer Science Dept.

NSF I/UCRC Meeting ~ Davis, CA
June 17, 2008

1

When We Need Audit Logs

- Computer forensics in courts
- Recovering from an attack
- Compliance (HIPAA, SOx)
- Human resources cases
- Debugging or verifying correct results (e.g., electronic voting machines)
- Performance analysis
- Accounting

2

2

We're terrible analyzing events on computers

Audit data is usually...

- overwhelming
- free-form
- useless
- misleading (easily altered)

We're collecting too much bad information...

5

5

...and using it in courts and elections.

6

6

We need to...

- understand what the purpose of the analysis is
- understand what data can answer that purpose, with X% accuracy, and under a set of Y assumptions
- log the data
- give tools and techniques to an analyst to analyze that data

7

How is computer forensics done now?

- file & filesystem analysis (Coroner's Toolkit, Sleuth Kit, EnCase, FTK)
- syslog, tcpwrappers
- process accounting logs
- IDS logs
- packet sniffing

8

What do we need?
What are we missing?

9

A Systematic Approach is Better

10

Forensic Art & Science

- But computer science can only answer part of it.
- Forensic analysis is an art, but there *are* scientific components. What are they?
  - Determining what to log
  - Determining relevance of logged data
    - what is relevant?
    - what is not relevant?
    - under what circumstances something might be relevant?
  - Using the results to constrain and correlate data.
  - *This can be measured, systematized and automated.*

Measurement Example: Empirical Study of Firewall Rules

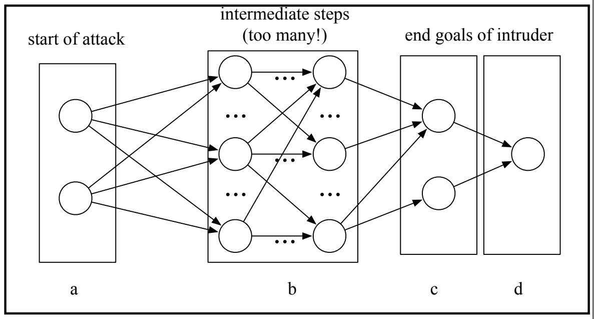
- How are firewalls configured?
- How should firewalls be configured?
  - What are the top, known vulnerabilities?
  - What are the top, known attacks?
- What are we missing? Is that OK?

Laocoön:
A Model of Forensic Logging

- Attack graphs of goals.
- Goals can be attacker goals or defender goals (i.e., "security policies")
- Pre-conditions & post-conditions of those goals.
- Method of translating those conditions into logging requirements.
- Logs are in a standardized and parseable format.
- Logged data can be at arbitrary levels of granularity.

Attack Graphs

- Intruder goals can be enumerated.
- Vulnerabilities, attacks, and exploits cannot (or in many cases, we would patch them).
- Defender goals can also be enumerated. They are called security polices.



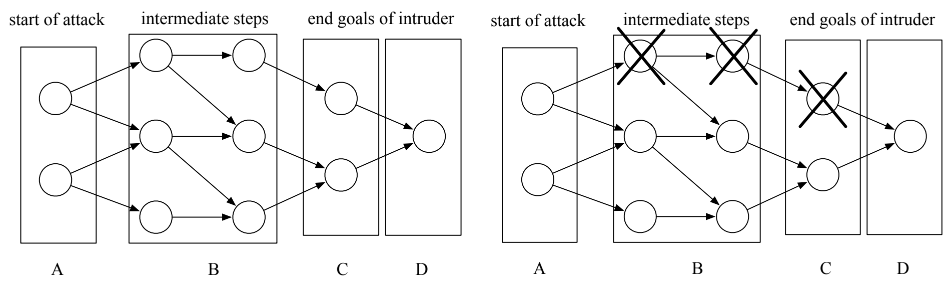
Security Policies

- Security policies can be reverse-engineered or enforced, automatically.
- Policies can be binary (block access) or flexible (log something).
- Policies can be static (always do this) or dynamic (uh oh—an intruder)

Applying Security Policies

- Applying Laocoön to security policies guides where to place instrumentation and what to log.
- The logged data needs to be correlated with a unique path identifier.
- Branches of a graph unrelated to the attack can be automatically pruned.
- Avoid recording data where events can be recreated because they are deterministic.

Pruning Paths



What are the assumptions for using current forensic tools?

- Often that there's only one person who had access to the machine.
- Often that the owner of the machine was in complete control (as opposed to malware).
- Probably a lot of other assumptions that we have no clue about.

Summary: we can do better

- Forensics, attack analysis, logging, and auditing are broken.
- We seek to work on real-world problems with real-world data to construct and implement useful, usable, real-world software solutions.

Proposed Project

- Research practicality and tradeoffs in conditional access control (e.g., allow & log vs. block)
- Implement conditional access control with several countermeasures, including logging.
- For the logging portion, implement forensic logging of system & function calls, and analysis tools to correlate and prune data unrelated to the end goals that an analyst is concerned with.
- If there is time, attempt to do this via virtual machine introspection.

Selected Recent Publications

- S. Peisert, M. Bishop, and K, Marzullo, "Computer Forensics *In Forensis,*" *Proc. of the 3rd Intl. IEEE Wkshp. on Systematic Approaches to Digital Forensic Engineering*, May 2008.
- S. Peisert, M. Bishop, S. Karin, and K. Marzullo, "Analysis of Computer Intrusions Using Sequences of Function Calls," *IEEE Trans. on Dependable and Secure Computing (TDSC)*, 4(2), Apr.-June 2007.
- S. Peisert and M. Bishop, "How to Design Computer Security Experiments," *Proc. of the 5th World Conf. on Information Security Education*, June 2007.
- S. P. Peisert, "A Model of Forensic Analysis Using Goal-Oriented Logging," Ph.D. Dissertation, UC San Diego, Mar. 2007.
- S. Peisert, M. Bishop, S. Karin, and K. Marzullo, "Principles-Driven Forensic Analysis," *Proc. of the New Security Paradigms Workshop (NSPW)*, Sept. 2005.

Questions?

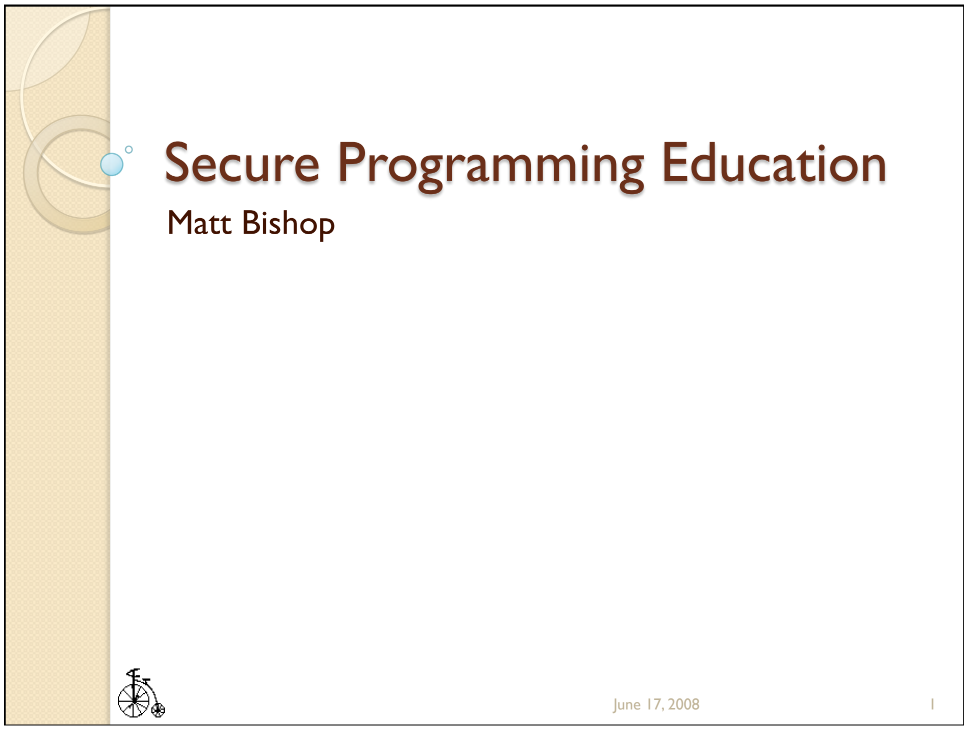
- Dr. Sean Peisert
  - Email: peisert@cs.ucdavis.edu
- More information and recent publications:
  - http://www.sdsc.edu/~peisert/

Secure Programming Education

Matt Bishop
bishop@cs.ucdavis.edu

We present an approach to emphasizing good programming practices and style throughout a curriculum. This approach draws on a clinic model used by English programs to reinforce the practice of clear, effective writing, and law schools to teach students legal writing. We present our model and some very preliminary results when we used it. We also discuss the next steps.

Biography: Professor Matt Bishop's research area is computer security, in which he has been active since 1979. He is especially interested in vulnerability analysis and denial of service problems, but maintains a healthy concern for formal modeling (especially of access controls and the Take-Grant Protection Model) and intrusion detection and response. He has also worked extensively on the security of various forms of the UNIX operating system. He is involved in efforts to improve education in information assurance, and is a charter member of the Colloquium for Information Systems Security Education. His textbook, *Computer Security: Art and Science*, was published by Addison-Wesley in December 2002.



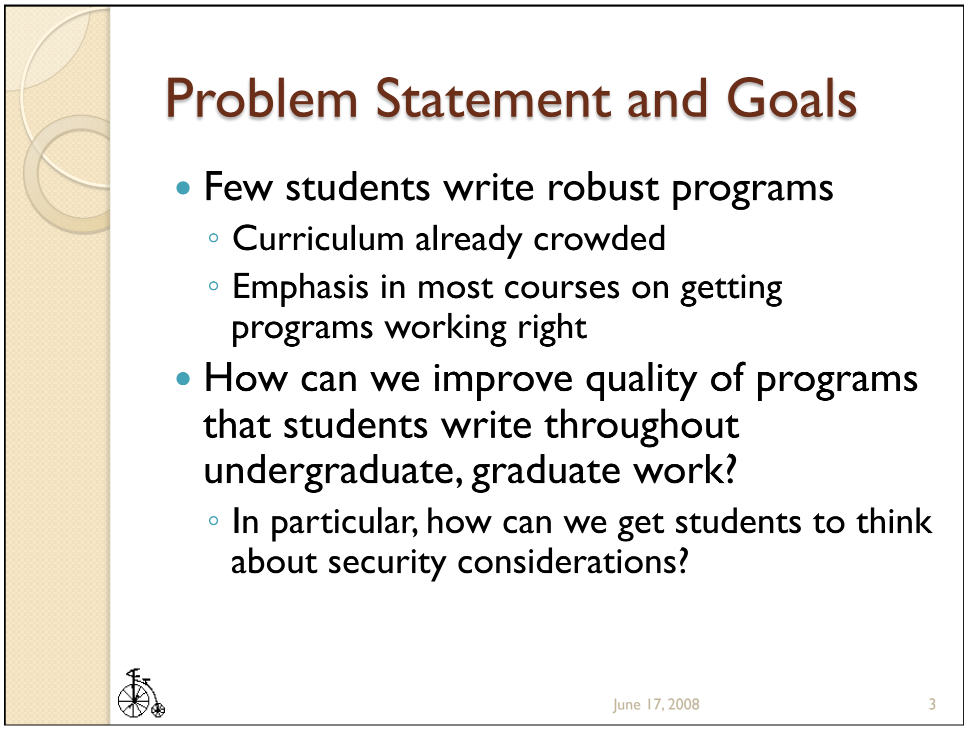
Secure Programming Education

Matt Bishop

Contact Information

Matt Bishop
Department of Computer Science
University of California at Davis
1 Shields Ave.
Davis, CA 95616-8562

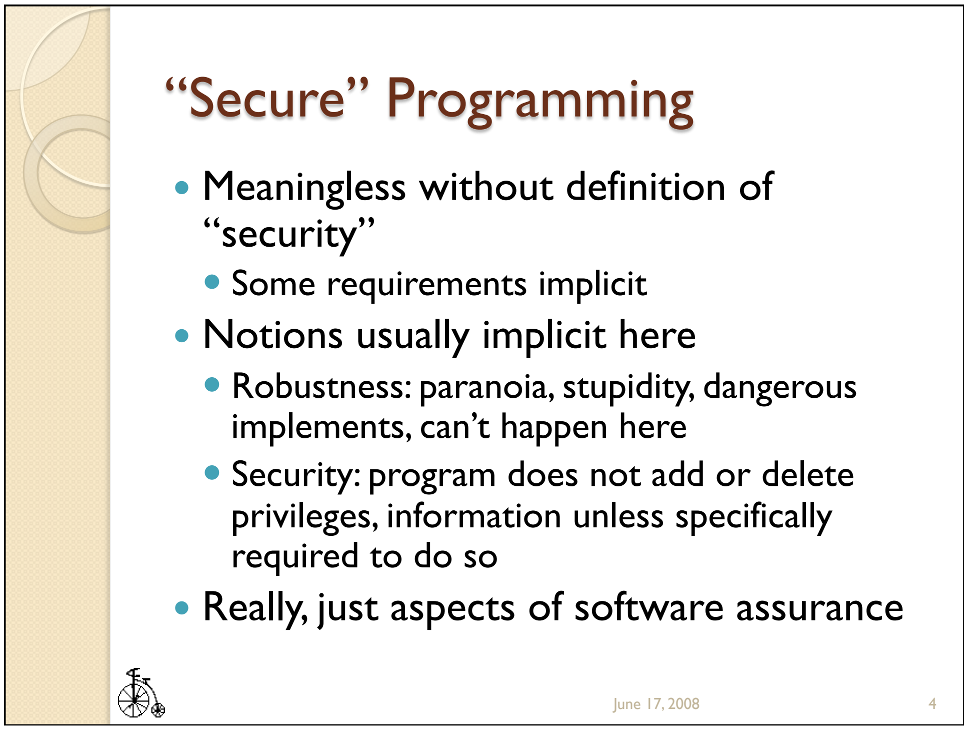
phone: (530) 752-8060
email: bishop@cs.ucdavis.edu
www: http://seclab.cs.ucdavis.edu/~bishop



Problem Statement and Goals

- Few students write robust programs
  - Curriculum already crowded
  - Emphasis in most courses on getting programs working right
- How can we improve quality of programs that students write throughout undergraduate, graduate work?
  - In particular, how can we get students to think about security considerations?

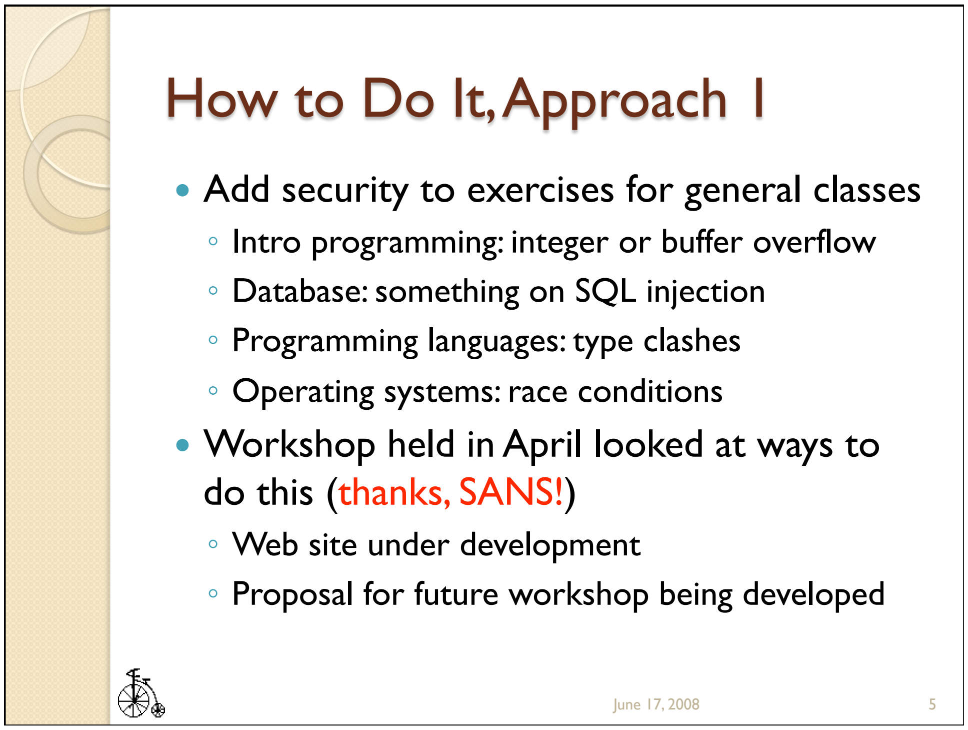
June 17, 2008 3



"Secure" Programming

- Meaningless without definition of "security"
  - Some requirements implicit
- Notions usually implicit here
  - Robustness: paranoia, stupidity, dangerous implements, can't happen here
  - Security: program does not add or delete privileges, information unless specifically required to do so
- Really, just aspects of software assurance

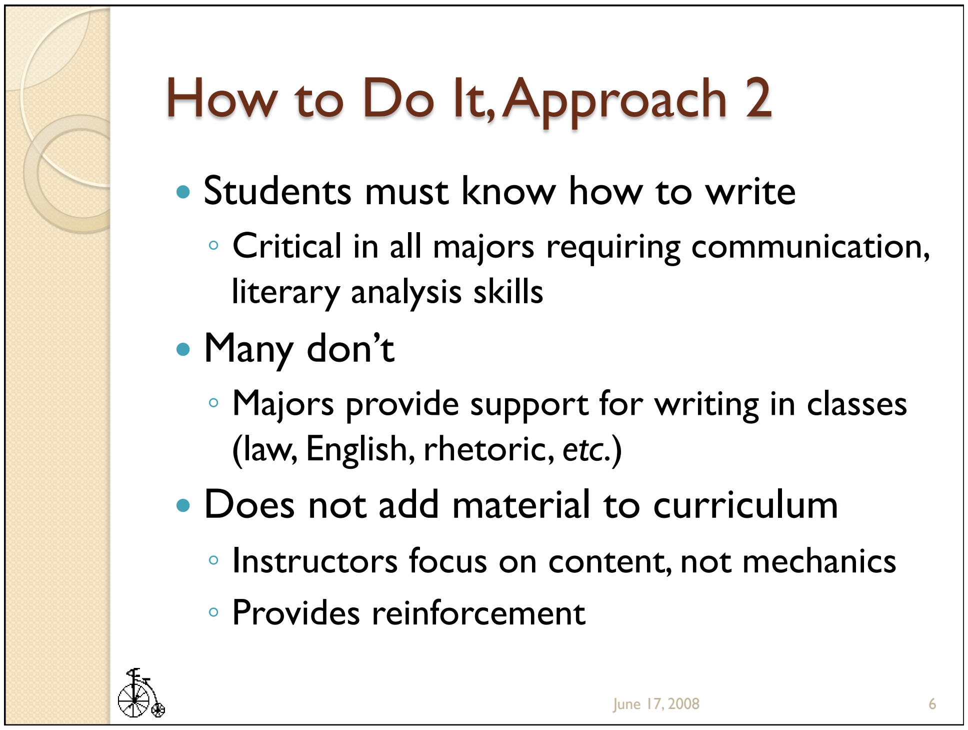
June 17, 2008 4



How to Do It, Approach 1

- Add security to exercises for general classes
  - Intro programming: integer or buffer overflow
  - Database: something on SQL injection
  - Programming languages: type clashes
  - Operating systems: race conditions
- Workshop held in April looked at ways to do this (thanks, SANS!)
  - Web site under development
  - Proposal for future workshop being developed

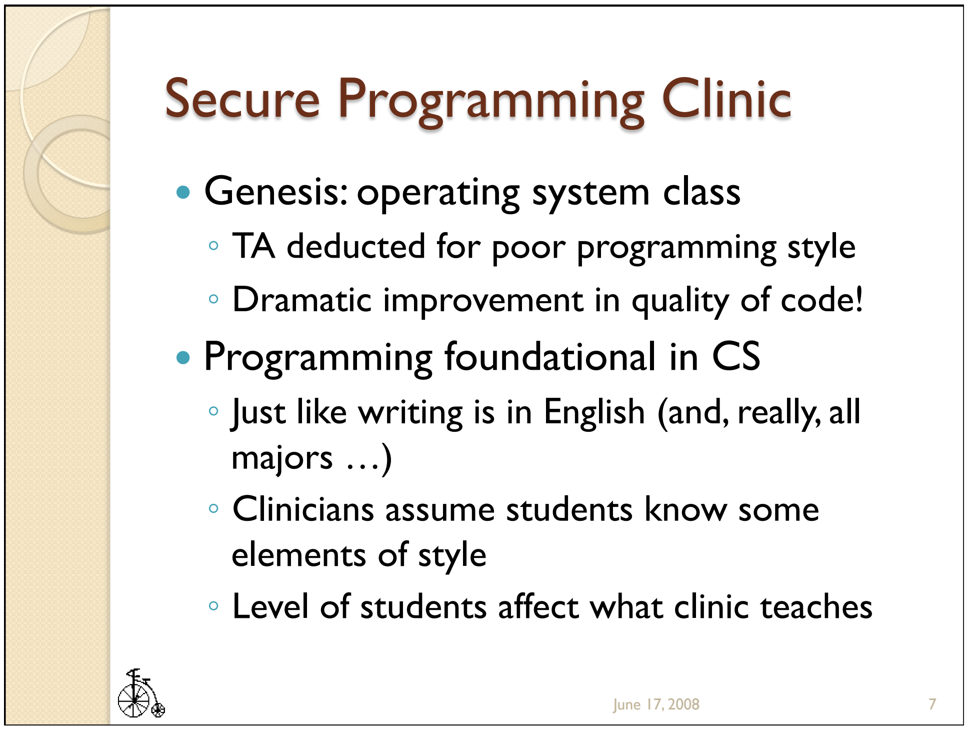
June 17, 2008 5



How to Do It, Approach 2

- Students must know how to write
  - Critical in all majors requiring communication, literary analysis skills
- Many don't
  - Majors provide support for writing in classes (law, English, rhetoric, *etc.*)
- Does not add material to curriculum
  - Instructors focus on content, not mechanics
  - Provides reinforcement

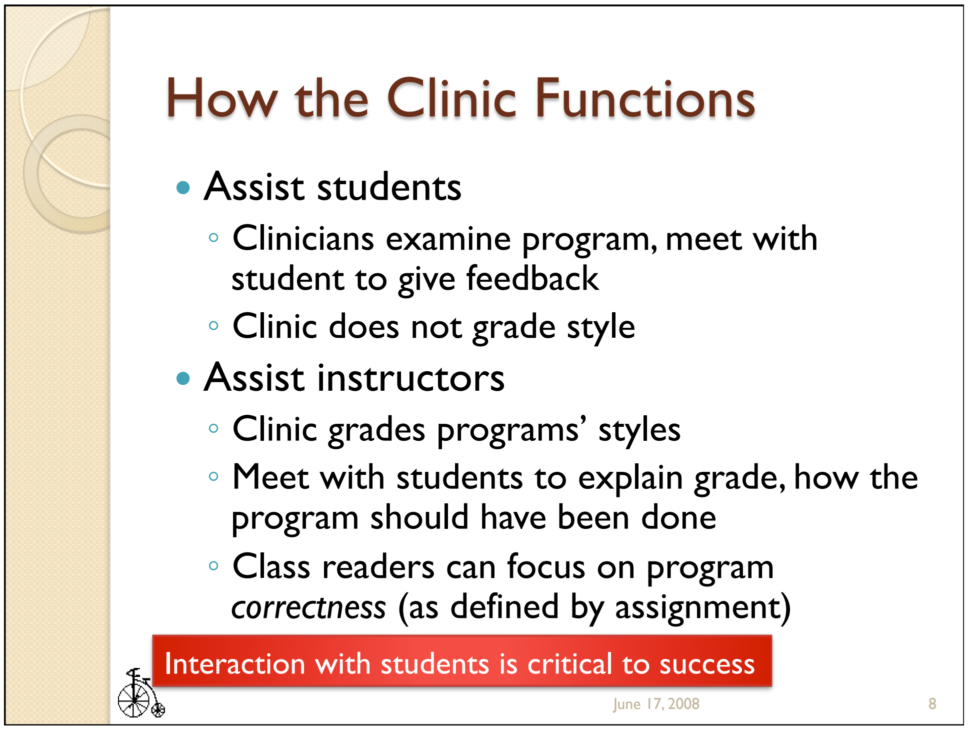
June 17, 2008 6



Secure Programming Clinic

- Genesis: operating system class
  - TA deducted for poor programming style
  - Dramatic improvement in quality of code!
- Programming foundational in CS
  - Just like writing is in English (and, really, all majors …)
  - Clinicians assume students know some elements of style
  - Level of students affect what clinic teaches

June 17, 2008 7

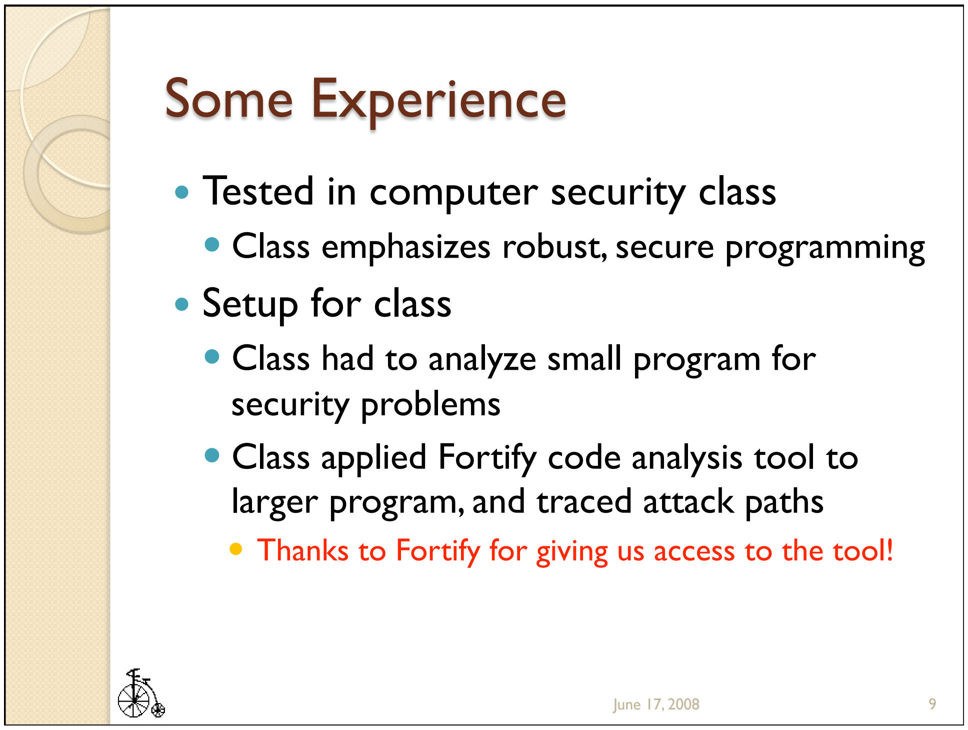


How the Clinic Functions

- Assist students
  - Clinicians examine program, meet with student to give feedback
  - Clinic does not grade style
- Assist instructors
  - Clinic grades programs' styles
  - Meet with students to explain grade, how the program should have been done
  - Class readers can focus on program *correctness* (as defined by assignment)

Interaction with students is critical to success

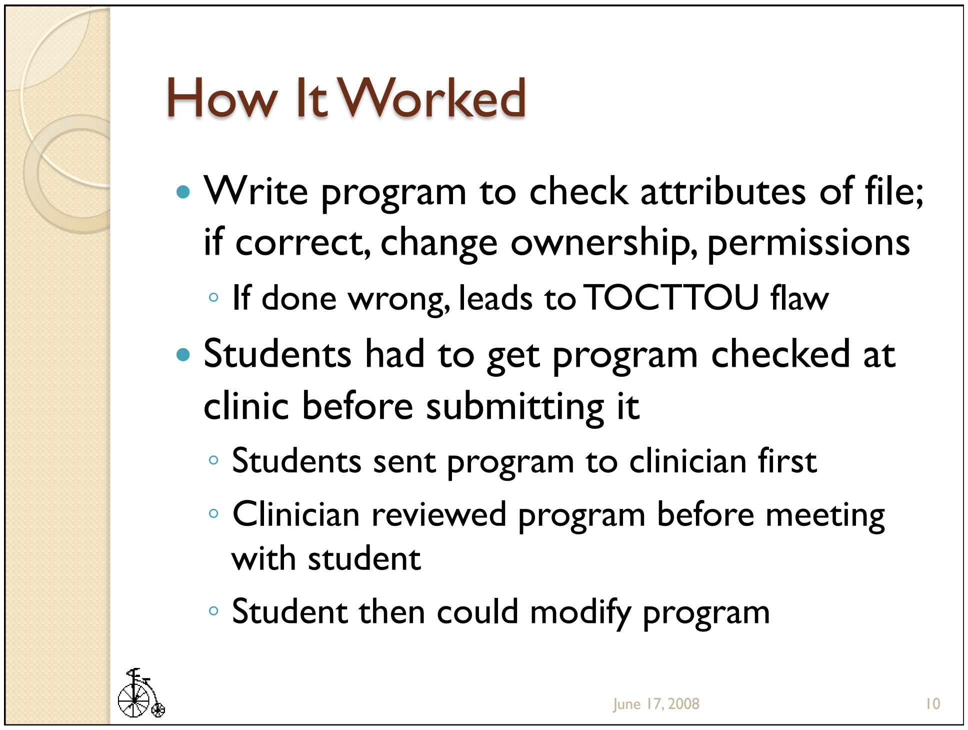
June 17, 2008 8



Some Experience

- Tested in computer security class
  - Class emphasizes robust, secure programming
- Setup for class
  - Class had to analyze small program for security problems
  - Class applied Fortify code analysis tool to larger program, and traced attack paths
    - Thanks to Fortify for giving us access to the tool!

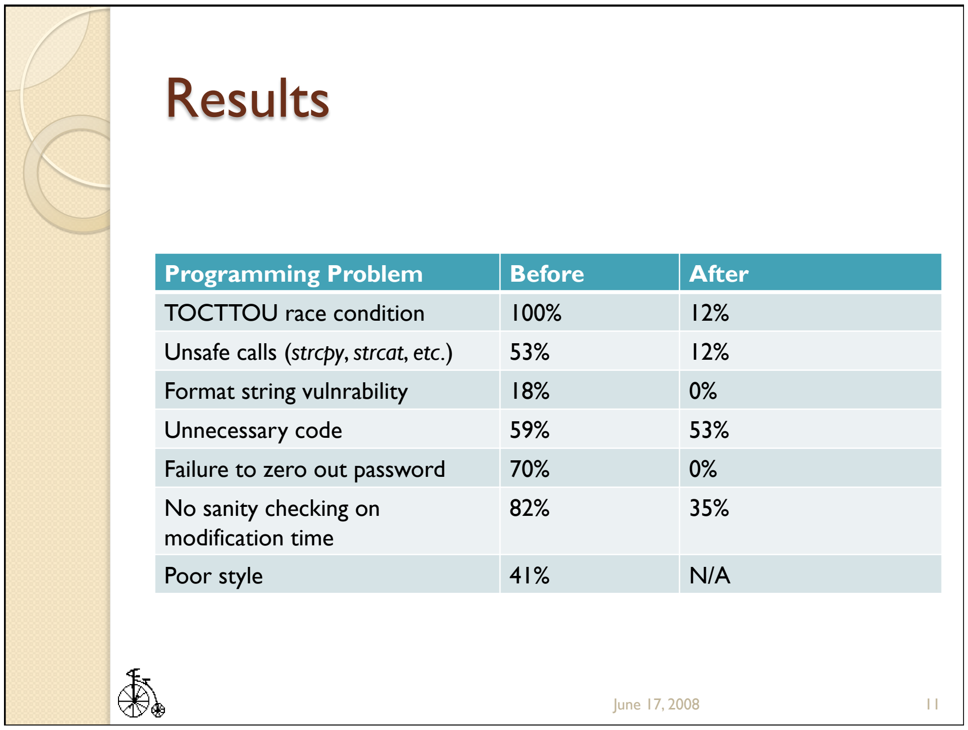
June 17, 2008 9



How It Worked

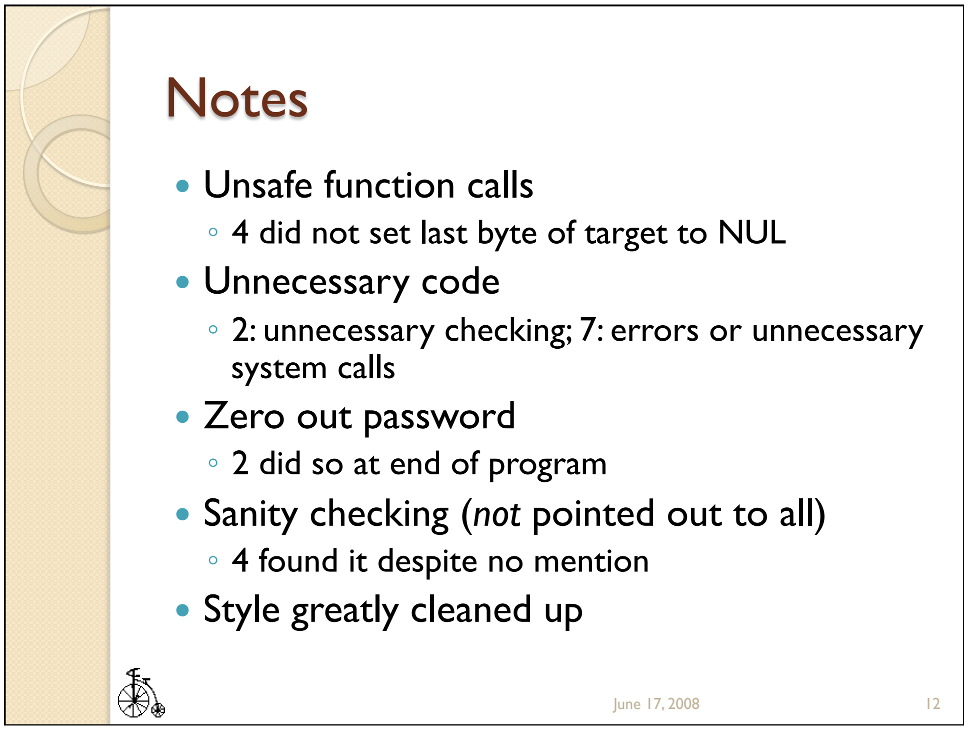
- Write program to check attributes of file; if correct, change ownership, permissions
  - If done wrong, leads to TOCTTOU flaw
- Students had to get program checked at clinic before submitting it
  - Students sent program to clinician first
  - Clinician reviewed program before meeting with student
  - Student then could modify program

June 17, 2008 10



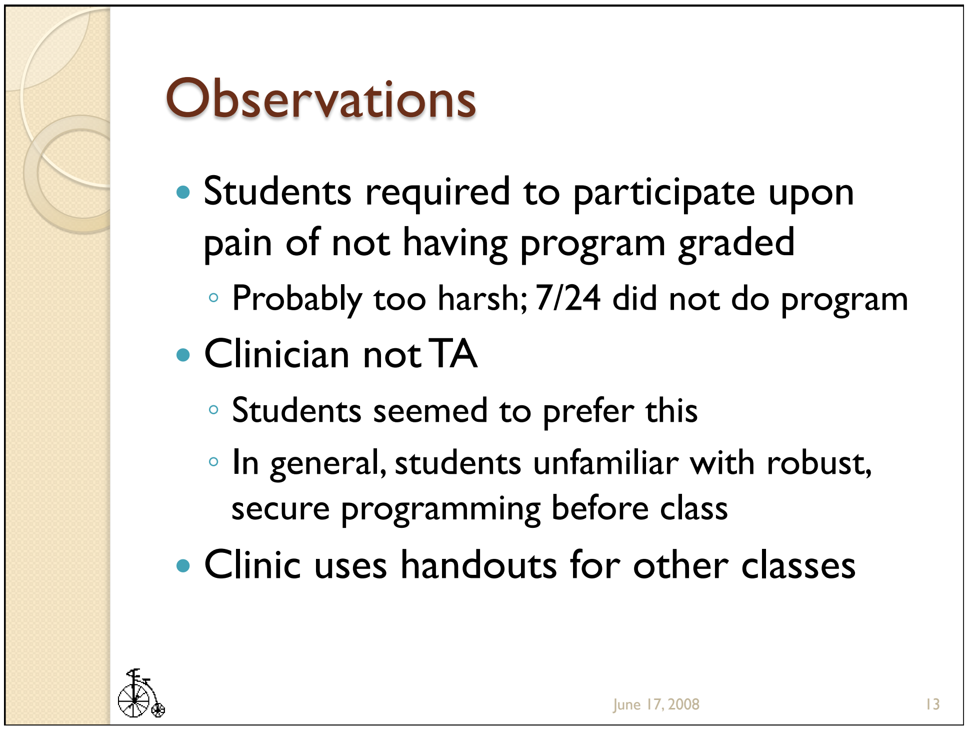
Results

Programming Problem	Before	After
TOCTTOU race condition	100%	12%
Unsafe calls (*strcpy*, *strcat*, *etc.*)	53%	12%
Format string vulnrability	18%	0%
Unnecessary code	59%	53%
Failure to zero out password	70%	0%
No sanity checking on modification time	82%	35%
Poor style	41%	N/A



Notes

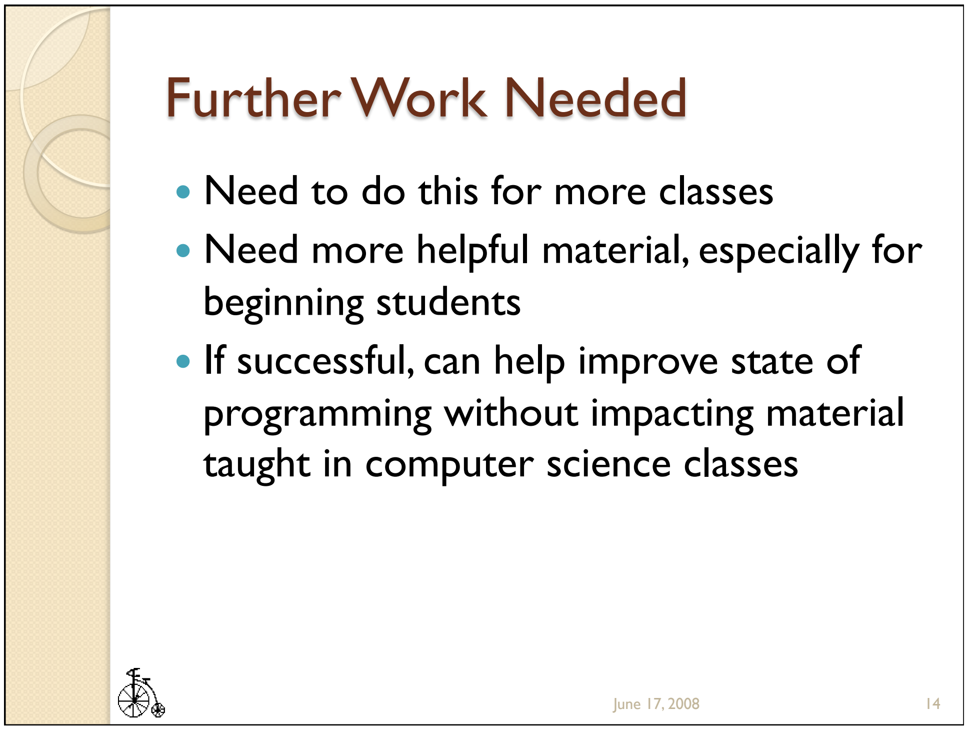
- Unsafe function calls
  - 4 did not set last byte of target to NUL
- Unnecessary code
  - 2: unnecessary checking; 7: errors or unnecessary system calls
- Zero out password
  - 2 did so at end of program
- Sanity checking (*not* pointed out to all)
  - 4 found it despite no mention
- Style greatly cleaned up



Observations

- Students required to participate upon pain of not having program graded
  - Probably too harsh; 7/24 did not do program
- Clinician not TA
  - Students seemed to prefer this
  - In general, students unfamiliar with robust, secure programming before class
- Clinic uses handouts for other classes

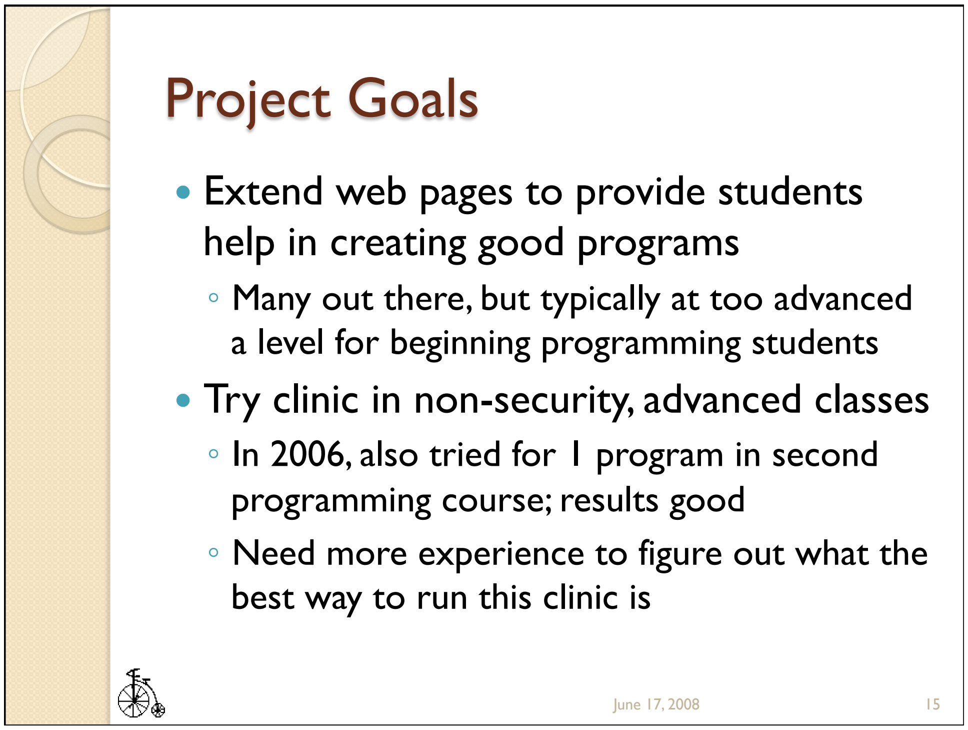
June 17, 2008 13



Further Work Needed

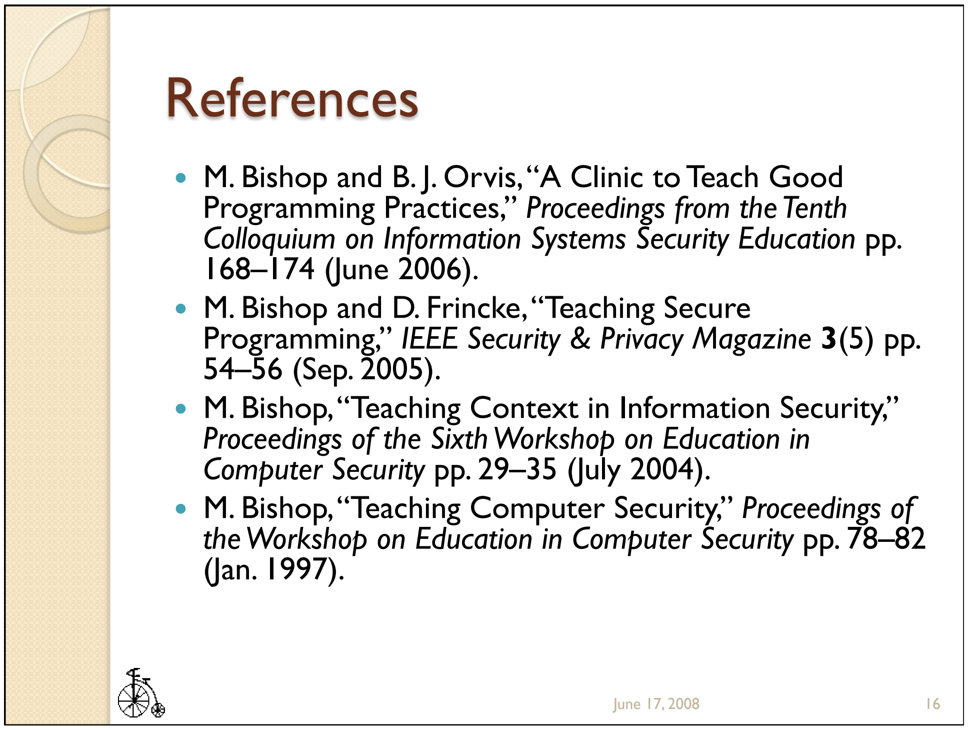
- Need to do this for more classes
- Need more helpful material, especially for beginning students
- If successful, can help improve state of programming without impacting material taught in computer science classes

June 17, 2008 14



Project Goals

- Extend web pages to provide students help in creating good programs
  - Many out there, but typically at too advanced a level for beginning programming students
- Try clinic in non-security, advanced classes
  - In 2006, also tried for 1 program in second programming course; results good
  - Need more experience to figure out what the best way to run this clinic is



References

- M. Bishop and B. J. Orvis, "A Clinic to Teach Good Programming Practices," *Proceedings from the Tenth Colloquium on Information Systems Security Education* pp. 168–174 (June 2006).
- M. Bishop and D. Frincke, "Teaching Secure Programming," *IEEE Security & Privacy Magazine* **3**(5) pp. 54–56 (Sep. 2005).
- M. Bishop, "Teaching Context in Information Security," *Proceedings of the Sixth Workshop on Education in Computer Security* pp. 29–35 (July 2004).
- M. Bishop, "Teaching Computer Security," *Proceedings of the Workshop on Education in Computer Security* pp. 78–82 (Jan. 1997).

Mithridates: Peering into the Future with Idle Cores

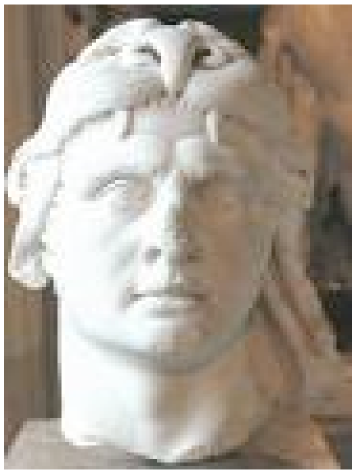
Earl Barr, Mark Gabel, David Hamilton, and Zhendong Su
barr@cs.ucdavis.edu

The presence of multicore machines, and the likely explosion in the number of cores in future CPUs brings with it the challenge and prospect of many idle cores: How can we utilize the additional, necessarily parallel cycles they provide? We propose Mithridates, a technique that uses idle cores to speed up programs that use dynamic checks to ensure a program's execution does not violate certain program invariants. Our insight is to take a program with invariants and transform it into a worker, shorn of the program's invariant checking, and one or more scouts that do the minimum work necessary to perform those checks. Then we run the worker and scouts in parallel. Ideally, the scouts run far enough ahead to complete invariant checks before the worker queries them. In other words, the scouts peer into the set of future states of their progenitor, and act as "short-sighted oracles."

We have evaluated Mithridates on an ordered list, as a motivating example, and on Lucene, a widely used document indexer from the Apache project. We systematically transformed these examples to extract the worker and the scouts. In both examples, we successfully utilized idle cores to reclaim much of the performance lost to invariant checking. With seven scouts, the Mithridates version of Lucene reduces the time spent checking the invariant by 92%. We believe Mithridates will bring invariants that are normally discarded after development into reach for production use.

Advisor: Prof. Zhendong Su, su@cs.ucdavis.edu

Mithridates: Peering into the Future with Idle Cores



–Earl T. Barr
–Mark Gabel
–David J. Hamilton
–Zhendong Su

CSSR
Center for Software and Systems Research

The Multicore Future

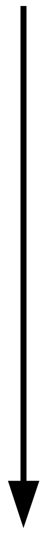
- “The power wall + the memory wall + the ILP wall = a brick wall for serial performance." David Patterson
- “If you build it, they will come.”
  - 10, 100, 1000 cores
- There will be spare cycles.
- What do we do with them?

Redundant Computation

- Cheap computation changes the economics of exploiting parallelism.
  - Swap expensive communication with recomputation.
  - Parallelize short "nuggets" of code, such as invariants

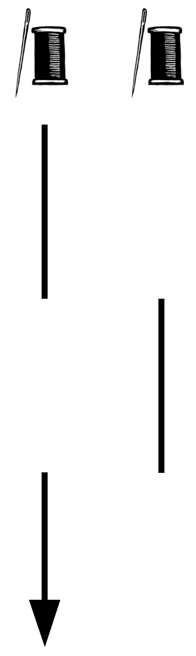
3

Sequential Execution



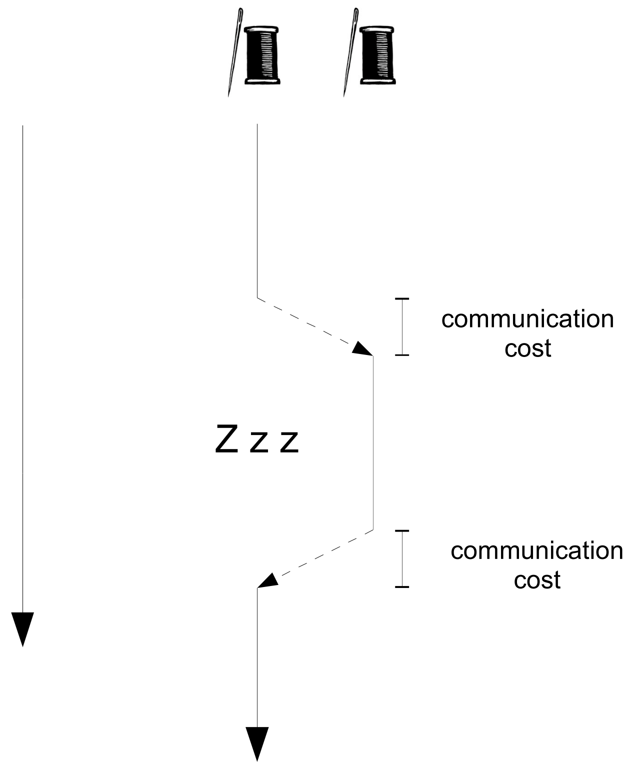
4

Concurrent Execution

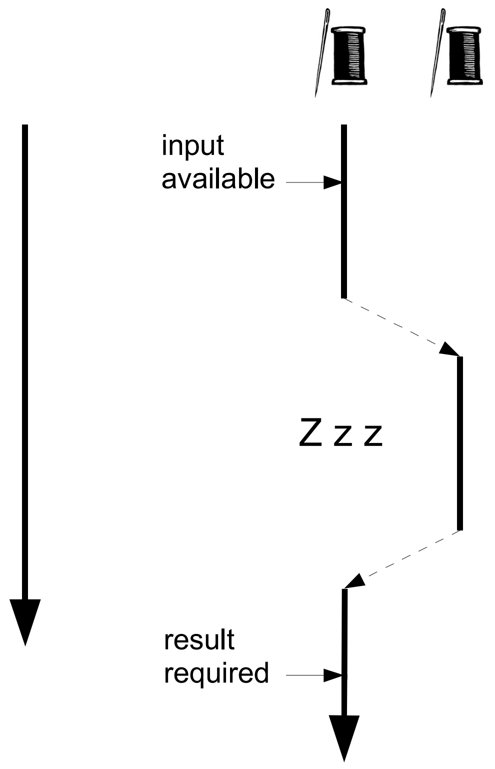


Concurrent Execution

Communcation cost =
synchronization + sending



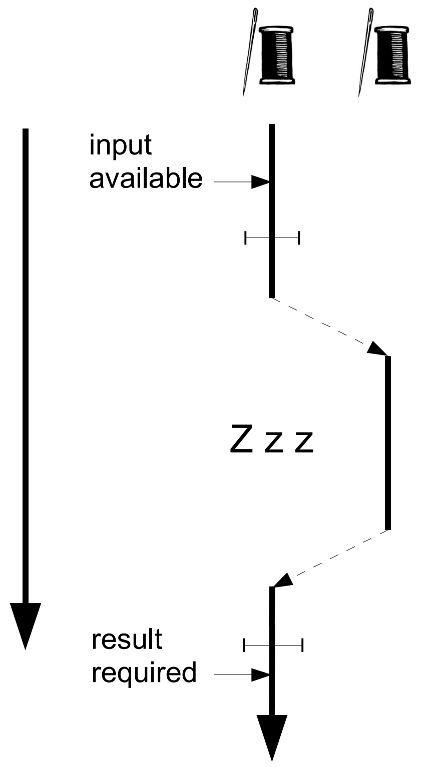
Traditional Parallelism



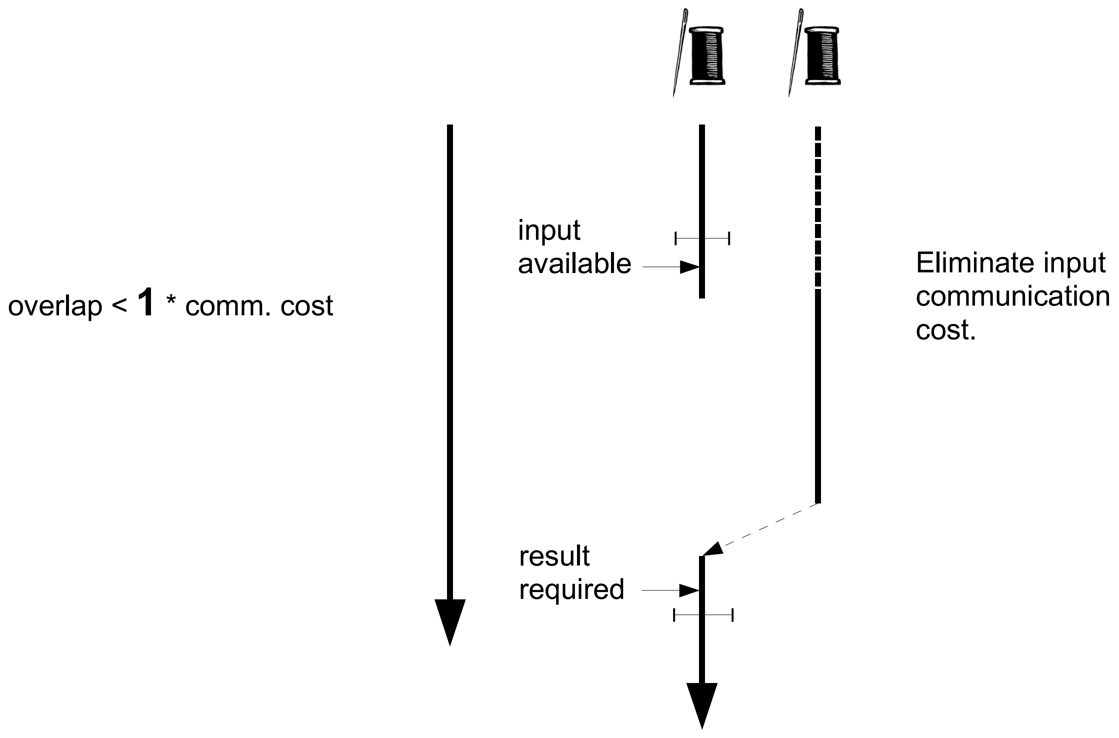
Narrow Window

Traditional techniques fail to parallelize code when

overlap < 2 * comm. cost

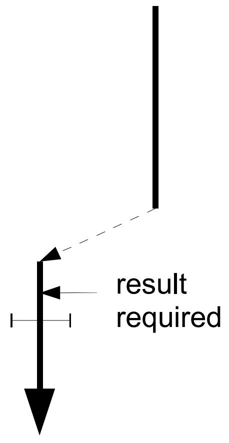


Mithridates

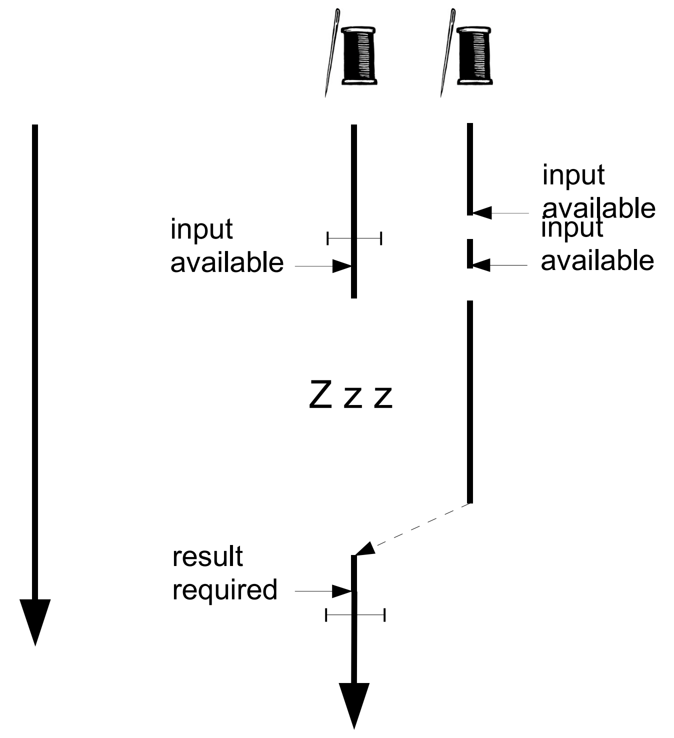


What about result communication?

- *Run ahead* to reduce the synchronization cost of result communication
  - Specialize via slicing
  - Schedule result calculation across n threads
- Small results
  - invariants → one bit

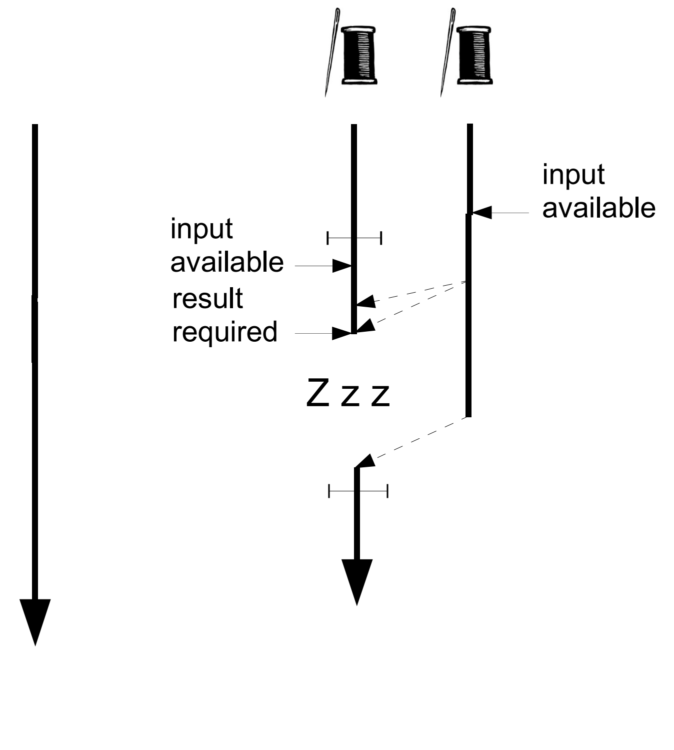


Slicing



11

Slicing



12

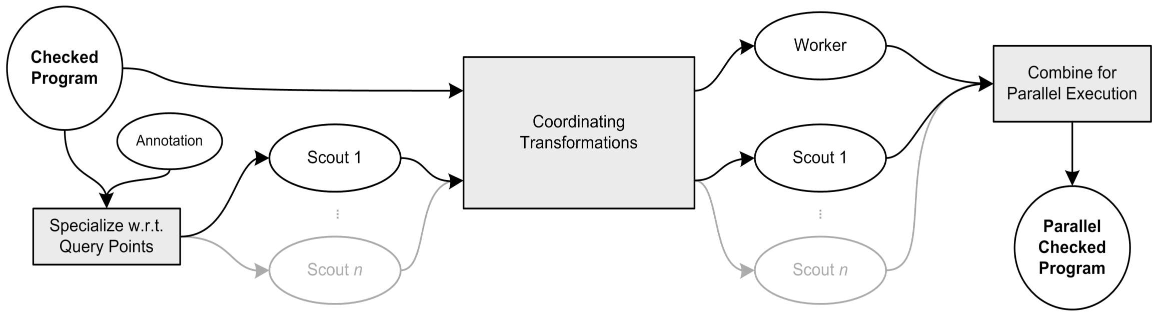
Approach

Transform a checked program into

- A worker
  - Core application logic, shorn of invariant checks
- Scouts
  - Minimum code necessary to check invariants assigned to them

Then execute in parallel

Architecture


Checked Program
Annotation
Specialize w.r.t. Query Points
Scout 1
Scout n
Coordinating Transformations
Worker
Scout 1
Scout n
Combine for Parallel Execution
Parallel Checked Program

Coordination

```
int a[10];
...
for(int i; i < 10; i++) {
        t = f(i);
        assert (t < 10);
        assert (t >= 0);

        sum += a[t];
}
...
```

Original

```
int a[10];
...
for(int i; i < 10; i++) {
        t = f(i);


        sem.down();
        sum += a[t];
}
...
```

Worker

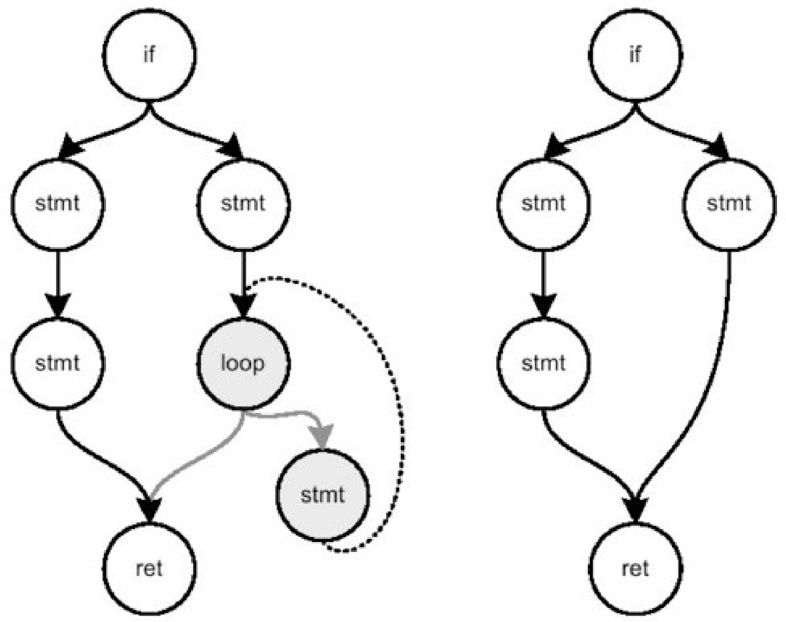
```
int a[10];
...
for(int i; i < 10; i++) {
        t = f(i);
        assert (t < 10);
        assert (t >= 0);
        sem.up();

}
...
```

Scout

Scout Transformation

- Assign invariants to each scout
- Remove code not related to assigned invariants
  - Program slicing
- Scouts do less work, so they can run ahead
- Short-sighted oracles

Control Flow Graph



Environment

- Any data not computed by the program
  - I/O, embedded programs, entropy

```
...
d = prompt user;
...
```

Original

```
...
d = prompt user;
q.enqueue(d);
sem.up();
...
```

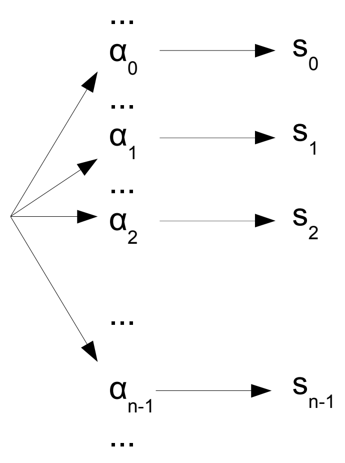
Worker

```
...
sem.down();
d = q.dequeue();
...
```

Scout

Invariant Scheduling

```
int a[10];
...
for(int i; i < 10; i++) {
      t = f(i);
      α: assert (t < 10 && t >= 0);
      sum += a[t];
}
...
```



Trace

Linked List

```
class Employee {              Employee {
    int ssn;                      int ssn;
    String name;
    int salary;
    int manager;
    String department;
    String location;
    Employee next;                Employee next;
};                            };
```

(a) Employee in P and P_w (b) Employee in P_s

Linked List Results

	Time	Peak RSS
Unchecked	0.2547 s	3.3 MB
Checked	0.8567 s	13.3 MB

(a) Baseline, linear invariant.

	Time	Peak RSS
Unchecked	0.3937 s	2.9 MB
Checked	753.208 s	12.2 MB

(b) Baseline, quadratic invariant.

$\lvert S \rvert$	Time	Peak RSS
1	0.6157 s	2.9 MB
2	0.386 s	2.9 MB
3	0.2935 s	3.1 MB
4	0.27 s	2.9 MB
5	0.285 s	3.0 MB
6	0.2715 s	2.9 MB
7	0.279 s	3.0 MB
8	0.2905 s	3.0 MB
9	0.3035 s	2.9 MB
10	0.3415 s	3.0 MB

(c) Parallelized checks using Mithridates, linear invariant.

$\lvert S \rvert$	Time	Peak RSS
1	614.7 s	12.7 MB
2	308.6 s	13.1 MB
3	206.4 s	13.3 MB
4	155.3 s	13.6 MB
5	124.2 s	14.4 MB
6	103.8 s	14.0 MB
7	88.91 s	14.4 MB
8	92.90 s	14.5 MB
9	88.62 s	14.6 MB
10	86.13 s	15.0 MB

(d) Parallelized checks using Mithridates, quadratic invariant.

Apache Lucene

	Time	Peak RSS
Unchecked	30.5 s	84 MB
Checked	124.8 s	71 MB

(a) Baseline, single-threaded.

	Dynamic Scheduling Only		Dynamic Scheduling w/ Transformations	
$\lvert S \rvert$	Time	Peak RSS	Time	Peak RSS
1	125.9 s	141 MB	118.0 s	104 MB
2	74.7 s	182 MB	72.0 s	110 MB
3	60.1 s	179 MB	55.3 s	148 MB
4	52.7 s	189 MB	48.2 s	163 MB
5	48.3 s	206 MB	43.2 s	162 MB
6	45.9 s	224 MB	39.5 s	165 MB
7	44.6 s	246 MB	38.1 s	173 MB

(b) Parallelized checks using Mithridates.

Figure 19: Results of applying Mithridates to the Apache Lucene Indexer. Figures represent the mean of three runs.

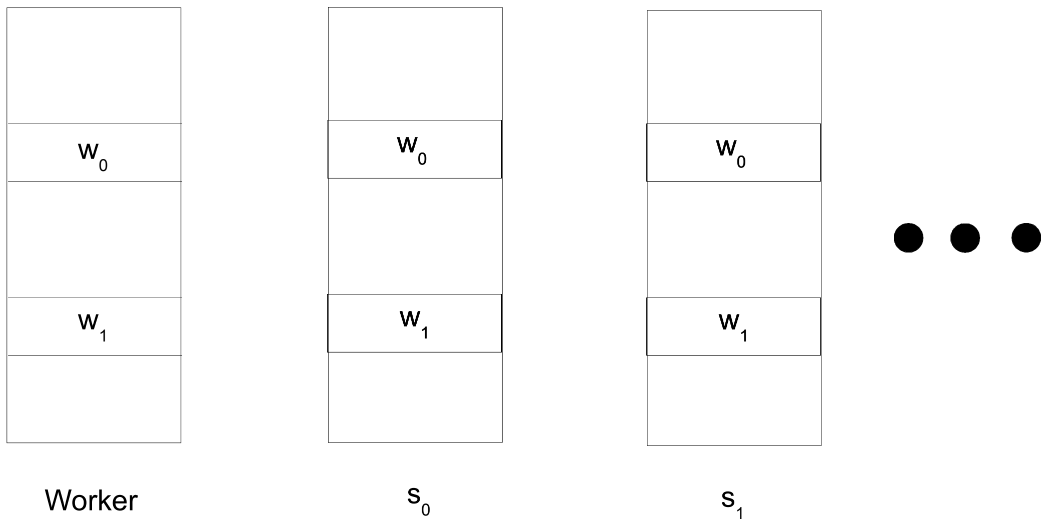
Future Work

- Pre-compute expensive functions?
- Extend to multi-threaded code
- Automate the transformation
  - Javassist
  - Soot
  - WALA
- Share Memory

Memory Cost

- O(n * (|P| + e))
  - n = number of scouts + 1
  - |P| is the high-water size of
    - Program
    - Stack
    - Heap
  - e is
    - input queue
    - semaphores
    - code to check invariants

Memory Sharing



Questions?

Related Work

- Thread level speculation (TLS)
  - Specialized hardware
  - Rollback implies expected performance gain
- Mithridates: Language-level, source-to-source
  - Runs on commercially-available, commodity machines today
  - Predictable performance gain

Related Work

- Shadow processing
  - Main and Shadow
  - Shadow trails Main to produce debugging output
- Mithridates
  - Enforces safety properties (sound)
  - Formal transformation
  - Invariant scheduling

Summary Static Costs

	Mithridates	TLS	Traditional
Input Handling	Rewrite to synchronize environmental interactions	Identify guess points	Identify input available
Result Handling	Identify result required and rewrite to insert milestones	Add logic to detect and resolve conflict and identify result required	Identify result required

Summary Runtime Costs

	Mithridates	TLS	Traditional
Input Handling	Synchronized environmental interaction	Communication cost	Communication cost
Result Handling	Communication cost - mitigation (slicing & invariant scheduling)	Communication cost + conflict resolution	Communication cost

Questions?

Issues – Handling Libraries

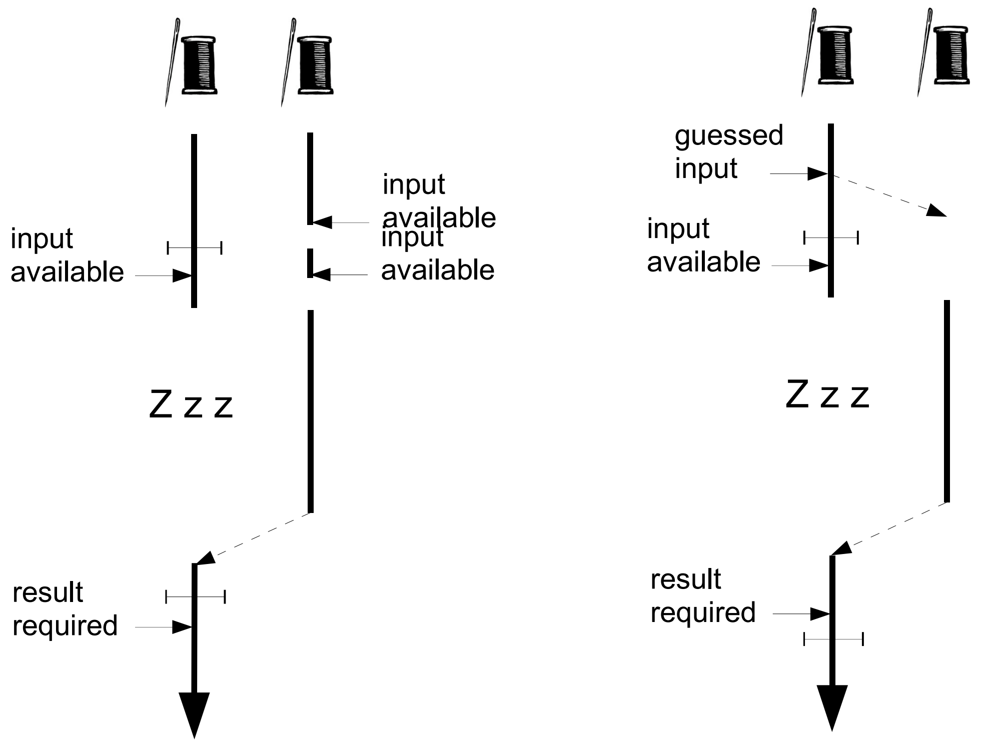
- $\frac{Ps}{Pw}$ is too large
- Libraries – not applications
- Few Concerns / High Cohesion

Assumptions

- Cores run at same speed
- Cores share main memory
- We do not model cache effects
- We have source code

33

Related Work: TLS



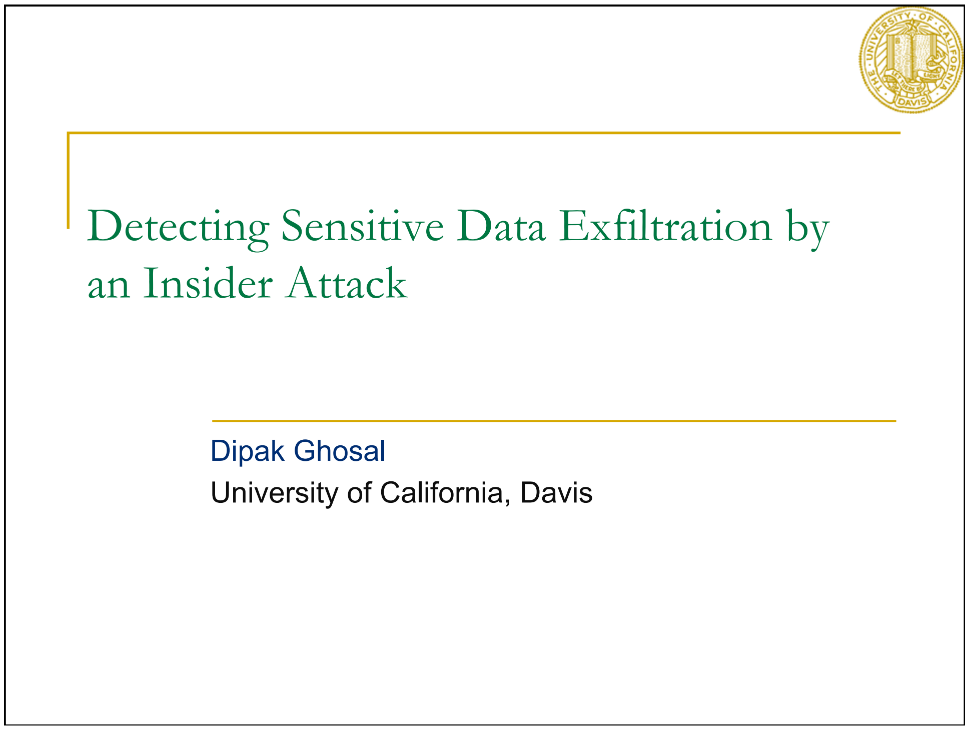
34

Detecting Sensitive Data Exfiltration by an Insider Attack

Dipak Ghosal
ghosal@cs.ucdavis.edu

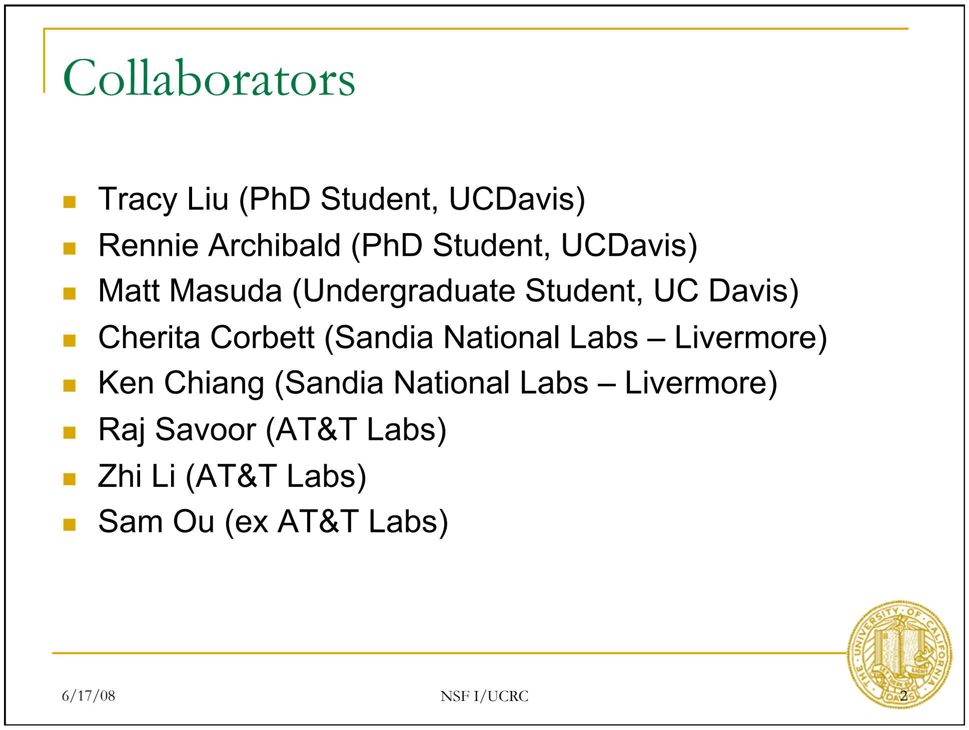
Methods to detect and mitigate insider threats are critical elements in the overall information protection strategy. Within the broader scope of insider threats, we focus on detecting exfiltration of sensitive data through the high-speed network. We propose a multilevel approach that consists of three main components: 1) network level application identification, 2) content signature generation and detection, and 3) covert communication detection. The key scientific approach used for all the above components is applying statistical and signal processing techniques on network traffic to generate signatures and/or extract features for classification purposes. In this talk, I will present the overall research directions and some preliminary results.

Biography: Professor Ghosal's primary research interests are in the areas of high-speed and wireless networks with particular emphasis on the impact of new technologies on the network and higher layer protocols and applications. He is also interested in the application of parallel architectures for protocol processing in high-speed networks and in the application of distributed computing principles in the design of next generation network architectures and server technologies. Professor Ghosal received an NSF CAREER Award in 1997 for his development plan for Research and Education in High Speed Networks. He is a member of IEEE.



Detecting Sensitive Data Exfiltration by an Insider Attack

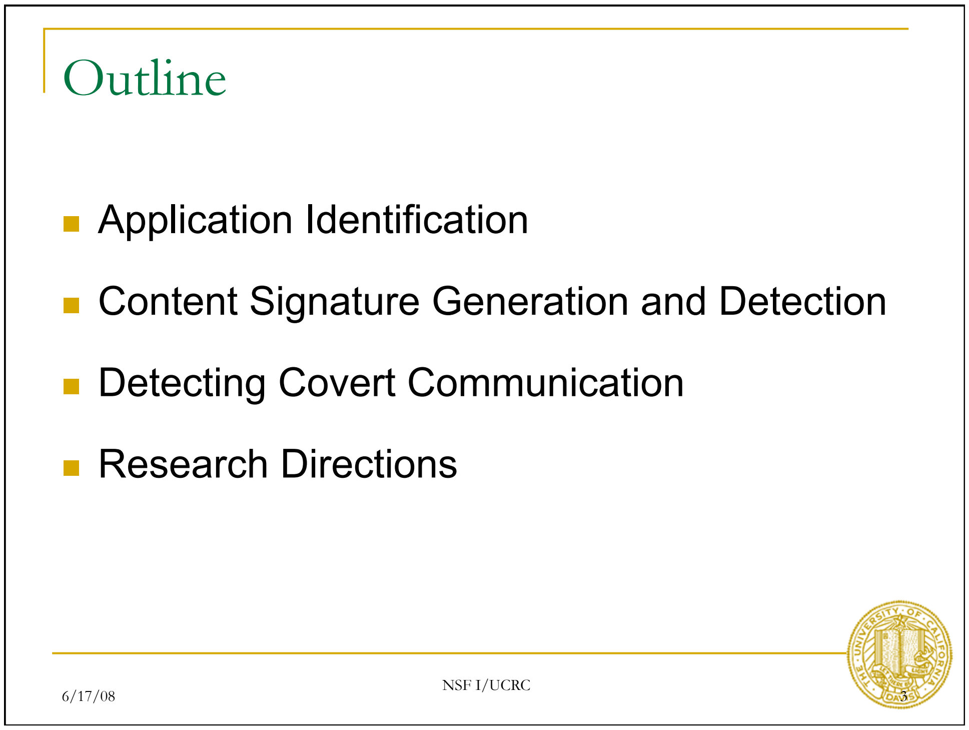
Dipak Ghosal
University of California, Davis



Collaborators

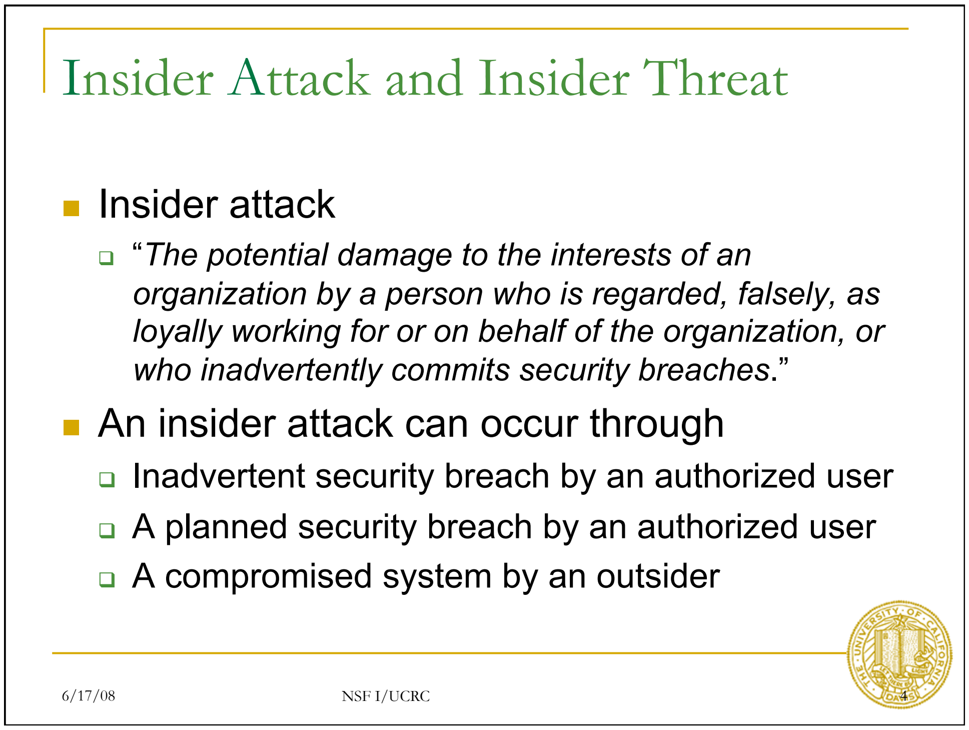
- Tracy Liu (PhD Student, UCDavis)
- Rennie Archibald (PhD Student, UCDavis)
- Matt Masuda (Undergraduate Student, UC Davis)
- Cherita Corbett (Sandia National Labs – Livermore)
- Ken Chiang (Sandia National Labs – Livermore)
- Raj Savoor (AT&T Labs)
- Zhi Li (AT&T Labs)
- Sam Ou (ex AT&T Labs)

6/17/08 NSF I/UCRC 2



Outline

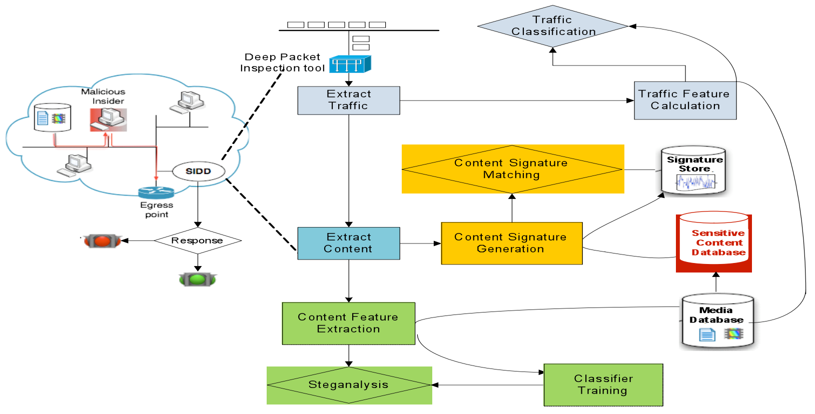
- Application Identification
- Content Signature Generation and Detection
- Detecting Covert Communication
- Research Directions



Insider Attack and Insider Threat

- Insider attack
  - “*The potential damage to the interests of an organization by a person who is regarded, falsely, as loyally working for or on behalf of the organization, or who inadvertently commits security breaches*.”
- An insider attack can occur through
  - Inadvertent security breach by an authorized user
  - A planned security breach by an authorized user
  - A compromised system by an outsider

Sensitive Information Dissemination Detection (SIDD) System

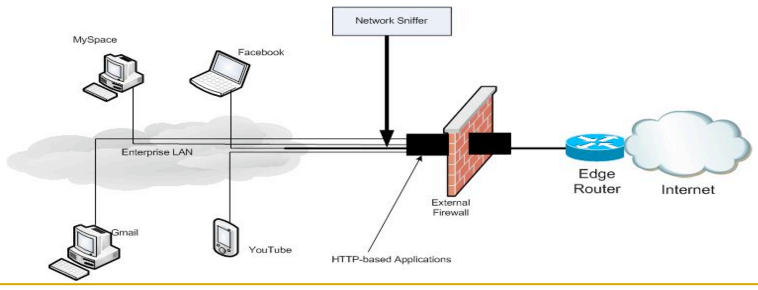

Traffic Classification
Deep Packet Inspection tool
Extract Traffic
Traffic Feature Calculation
Malicious Insider
SIDD
Egress point
Response
Content Signature Matching
Signature Store
Extract Content
Content Signature Generation
Sensitive Content Database
Media Database
Content Feature Extraction
Steganalysis
Classifier Training

▭ Application identification 🟨 Content detection ▨ Covert channel detection

6/17/08 NSF I/UCRC 5

Application Tunneling

- Current research has addressed the issue of identifying the application layer protocols
  - SSH, HTTP, FTP, etc.
- More fine grained identification is required for variety of applications that run over HTTP.
  - Social networking (MySpace and Facebook)
  - Web-mail (Gmail and Hotmail)
  - Streaming video applications (Youtube and Veoh)



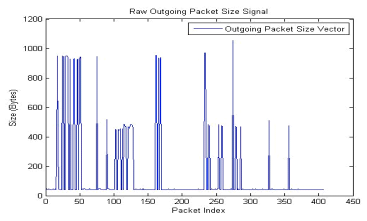
6/17/08 NSF I/UCRC 6

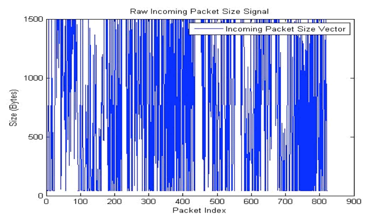
Signals

- **Inter-arrival time**: derived from the sequence of timestamps noted by the sniffer for packets inbound to the host
- **Inter-departure time**: derived from the sequence of timestamps noted by the sniffer for packets outbound from the host
- **Incoming packet size**: vector of packet sizes for HTTP packets inbound to the host
- **Outgoing packet size**: vector of packet sizes for packets outbound from the host
- **Outgoing Discrete Time Total Bytes**: vector of outgoing bytes of data aggregated over discrete and fixed time bins

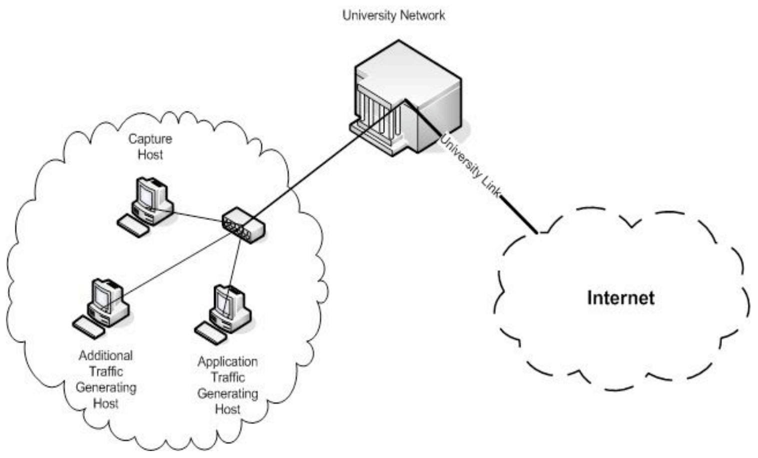
Signals – Examples

- Outgoing packet size vs. incoming packet size



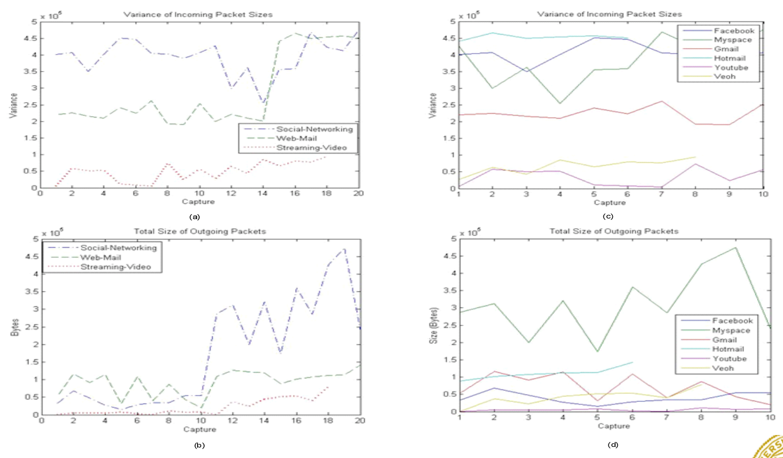


Experimental Setup


University Network
University Link
Capture Host
Additional Traffic Generating Host
Application Traffic Generating Host
Internet

6/17/08 NSF I/UCRC 9

Temporal Statistics



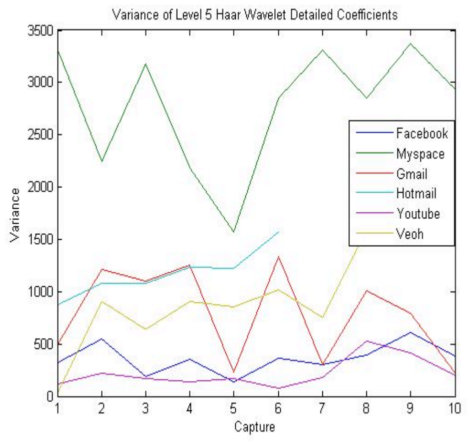
6/17/08 NSF I/UCRC 10

Temporal Characteristics

	Interdeparture		Outgoing Packet Size				Incoming Packet Size			
	Q3		Mean		Var		Var		Total Bytes	
	Mean $(x10^{-2})$	Var $(x10^{-5})$	Mean $(x10^{2})$	Var $(x10^{3})$	Mean $(x10^{4})$	Var $(x10^{9})$	Mean $(x10^{4})$	Var $(x10^{7})$	Mean $(x10^{5})$	Var $(x10^{11})$
Facebook	6.04	38.1	1.15	1.87	5.59	1.73	39.9	119	9.55	2.33
Myspace	1.94	3.1	3.27	8.37	28.6	4.89	38.4	509	31.1	29.9
Gmail	21.54	650	2.71	1.41	18.8	16.7	21.9	46.1	8.73	2.1
Hotmail	6.2	17.2	2.18	0.598	18	4.29	45.3	7.14	7.19	0.432
Youtube	11.26	250	0.82	1.35	4.53	14.9	3.47	69.4	121	133
Veoh	5.43	54.5	0.96	0.75	4	4.17	6.71	49.5	169	118

6/17/08 NSF I/UCRC 11

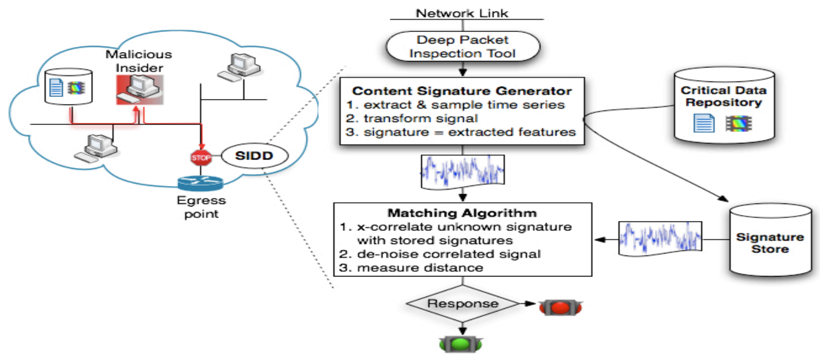
Wavelet Analysis



- Use Haar wavelet
- Feature used for comparison
  - Variance of the Level-5 detailed co-efficients

6/17/08 NSF I/UCRC 12

Content Identification: Motivation


Network Link
Deep Packet Inspection Tool
Content Signature Generator
1. extract & sample time series
2. transform signal
3. signature = extracted features
Critical Data Repository
Malicious Insider
SIDD
STOP
Egress point
Matching Algorithm
1. x-correlate unknown signature with stored signatures
2. de-noise correlated signal
3. measure distance
Signature Store
Response

Can we detect illegal dissemination of protected digital (media) assets?

6/17/08 NSF I/UCRC 13

Content Signature

- Content-based Signature
  - "The media itself is a watermark"
  - Unique and robust
    - Different content should have distinct signatures
    - The signatures are tolerant to various forms of noise and distortions
  - Requirements vary with applications
    - From video search to detecting video copying

6/17/08 NSF I/UCRC 14

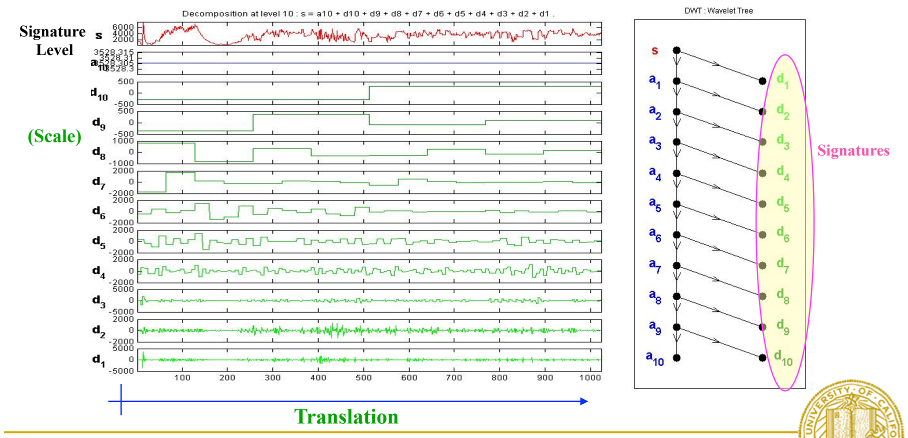
Content Signature Generation

- Basic idea
  - Extract a time series (or signal) of the content and analyze the signal to generate the signatures
  - Capture the temporal correlation in the signature
  - Treating the content signatures as time series
    - Use signal processing techniques and tools to analyze
      - Wavelet transform
    - Any portion of the content can be used for detection
    - Computation cost saving

6/17/08 NSF I/UCRC 15

Content Signature Generation – Example

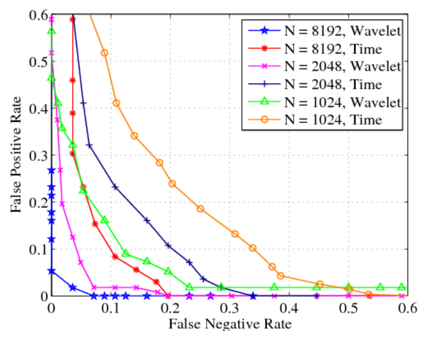
- The Detailed Coefficients of the Star Wars Movie



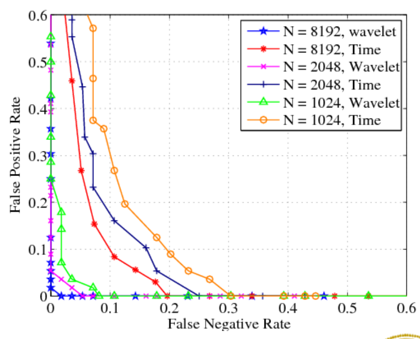
6/17/08 NSF I/UCRC 16

Preliminary Analysis

ROC curve in rate adaption case 1



ROC curve in rate adaption case 2

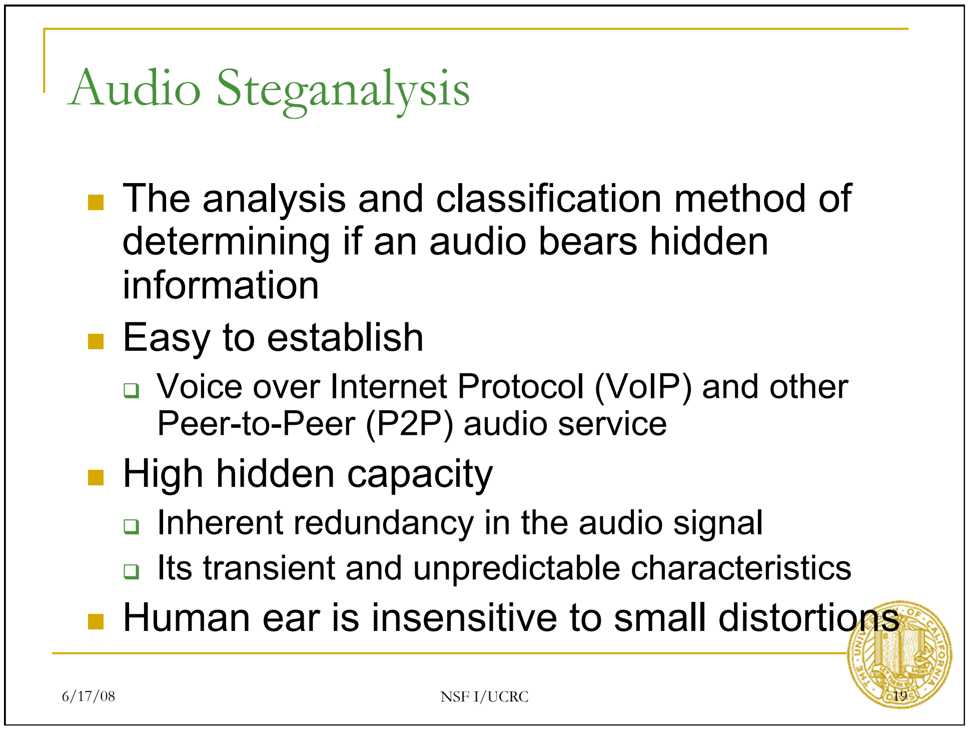


6/17/08 NSF I/UCRC 17

Detecting Covert Communication

- Exfiltration of sensitive information may be carried out using covert communication
  - Hiding content/communication in an innocuous carrier using a steganography tool
- Challenges
  - The content may be encrypted
  - Different types of carriers

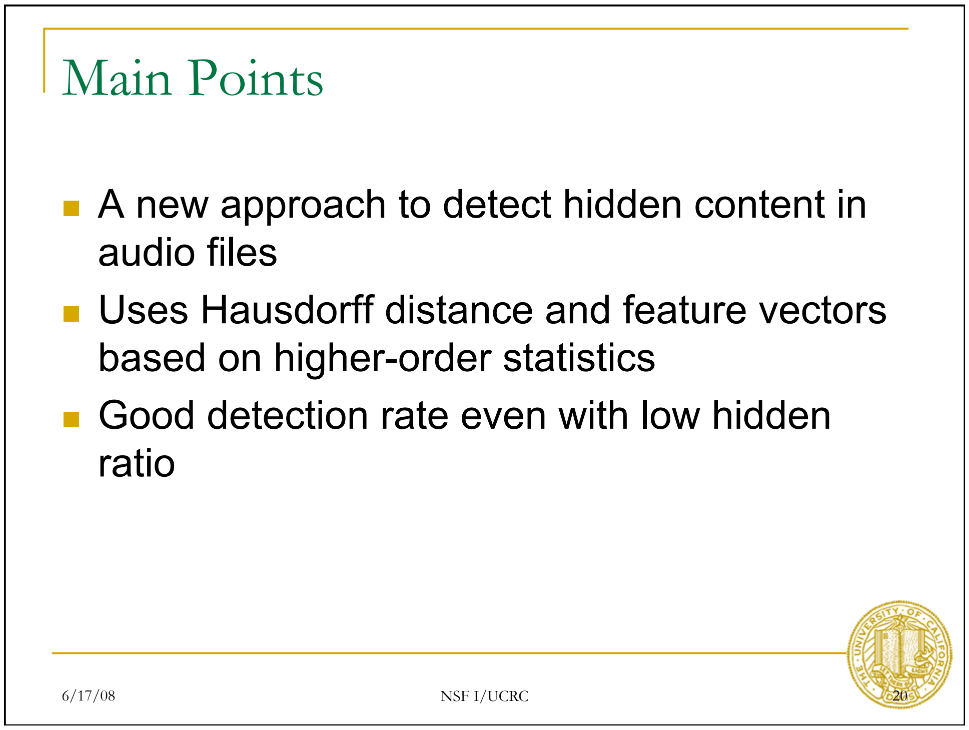
6/17/08 NSF I/UCRC 18



Audio Steganalysis

- The analysis and classification method of determining if an audio bears hidden information
- Easy to establish
  - Voice over Internet Protocol (VoIP) and other Peer-to-Peer (P2P) audio service
- High hidden capacity
  - Inherent redundancy in the audio signal
  - Its transient and unpredictable characteristics
- Human ear is insensitive to small distortions

6/17/08 NSF I/UCRC 19

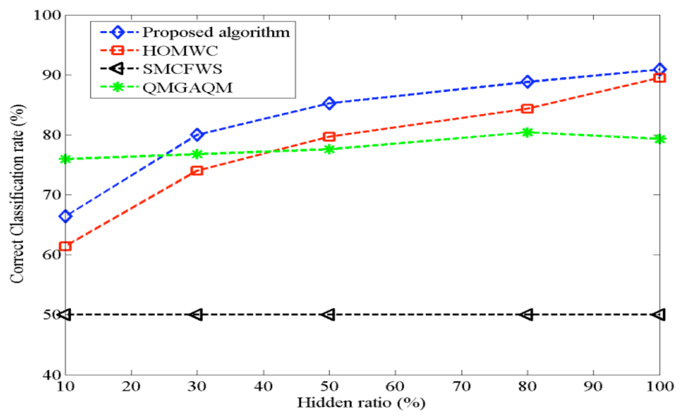


Main Points

- A new approach to detect hidden content in audio files
- Uses Hausdorff distance and feature vectors based on higher-order statistics
- Good detection rate even with low hidden ratio

6/17/08 NSF I/UCRC 20

Comparative Analysis



6/17/08 NSF I/UCRC 21

Research Directions

- Improving the techniques
  - Wavelet analysis allows time frequency localization
    - Where approximately time certain frequencies occur
    - Is it useful in disambiguating applications?
  - Co-integration can extract similarities in signals that may be uncorrelated
    - Can this be used to detect content that is encrypted and/or modified to evade detection?
- Developing prototypes
  - A VoIP steganalysis tool
  - A classifier for network level application identification

6/17/08 NSF I/UCRC 22