Hands-on Laboratory Exercises for Teaching Software Security

Xiaohong Yuan[1], Joaquin Hernandez[1], India Waddell[1], Bill Chu[2], Huiming Yu[1]
[1]North Carolina A&T State University, Greensboro, NC
[2]University of North Carolina at Charlotte, Charlotte, NC

CISSE 2012, June 11-13, 2012

Importance of Software Security

- SANS Institute Top 20 Internet Security Vulnerabilities are due to poor coding
- 90% of security vulnerabilities are the result of known software defects
  - According to a study conducted by SEI at Carnegie Mellon Univ.
- It is important to educate programmers to produce high quality, secure and reliable software

Secure Software Engineering Track

The required courses:

Course Number	Credit Hour	Course Title
COMP 710	(3)	Software Specification, Analysis & Design
COMP 727	(3)	Secure Software Engineering
COMP 725	(3)	Software Security Testing
COMP 796/797/COM711/COMP712/COMP620/COMP621/COMP626	(3)	Software Engineering or Information Assurance Elective

Courses on Software Security

- COMP727 Secure Software Engineering
  - Offered since Spring 2010
  - Textbook
    - Software Security: Building Security In, by Gary McGraw
- COMP725 Software Security Testing
  - Offered since Fall 2010, once a year
  - Textbook
    - The Art of Software Security Testing: Identifying Software Security Flaws by Chris Wysopal, *et al.*, 2007
    - The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws, by Dafydd Stuttard and Marcus Pinto, 2008

Software Security Topics

- The secure software development process
- Risk management for software security
- Architectural risk analysis
- Attack patterns and abuse cases
- Code review/static analysis tools
- Risk-based security testing with threat modeling
- Taxonomy of coding errors
- Common design and implementation vulnerabilities
- Security testing methods
- Web application vulnerability assessment
- Fuzz testing

Hands-on Laboratory Exercises

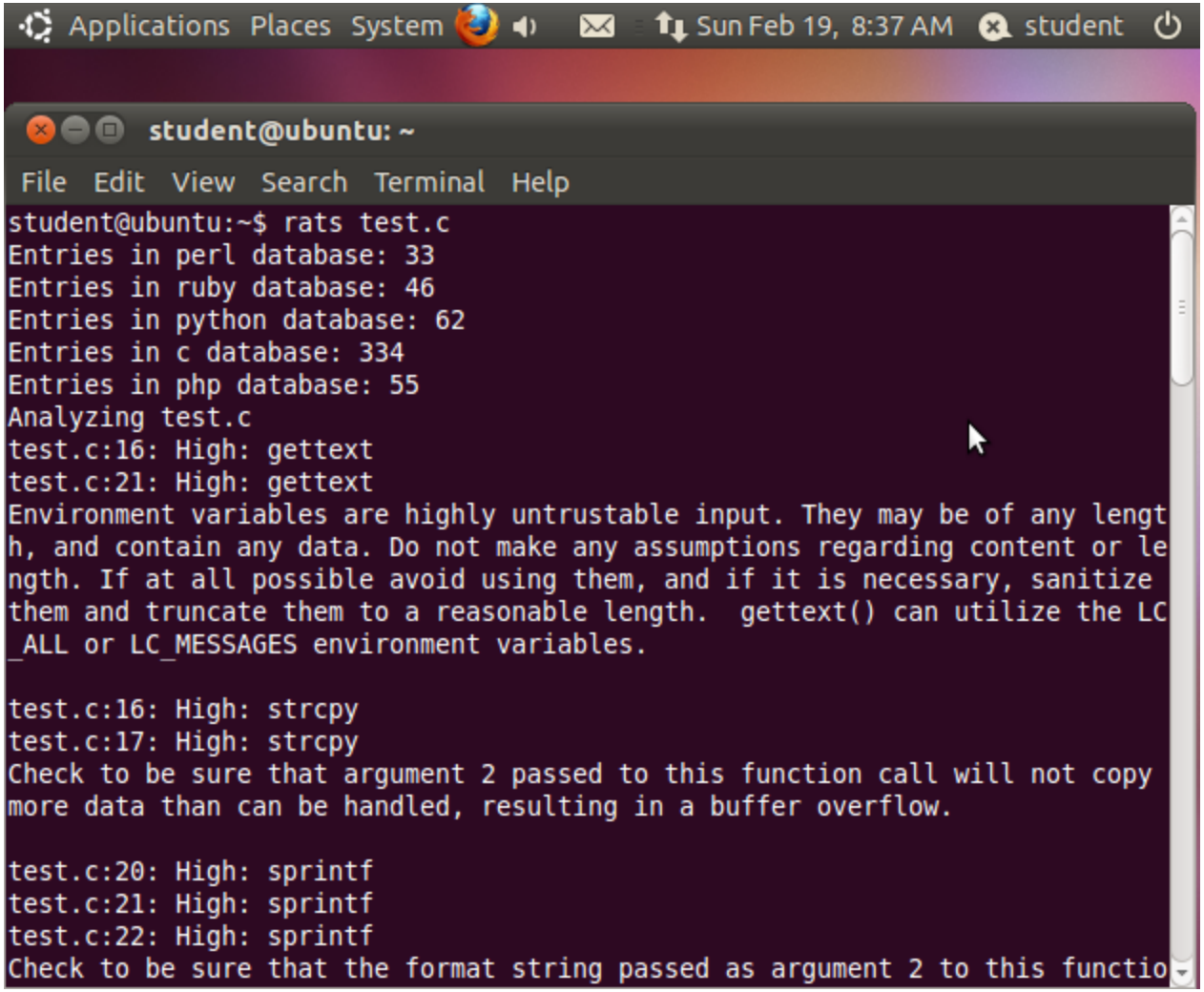
Topic	Laboratory Exercise
Code Review with Tools	Lab 1. Code review with RATs and Flawfinder
	Lab 2. Code review with Fortify
Web Application Vulnerability Assessment	Lab 3. Information gathering with WebScarab
	Lab 4. Exploiting hidden value
	Lab 5. Vulnerability assessment of Tunestore and BOG
Fuzz Testing	Lab 6. Fuzz testing with WebScarab
Threat analysis and modeling	Lab 7. Threat analysis with TAM
	Lab 8. Threat modeling with Microsoft SDL Threat modeling tool

Code Review with Tools

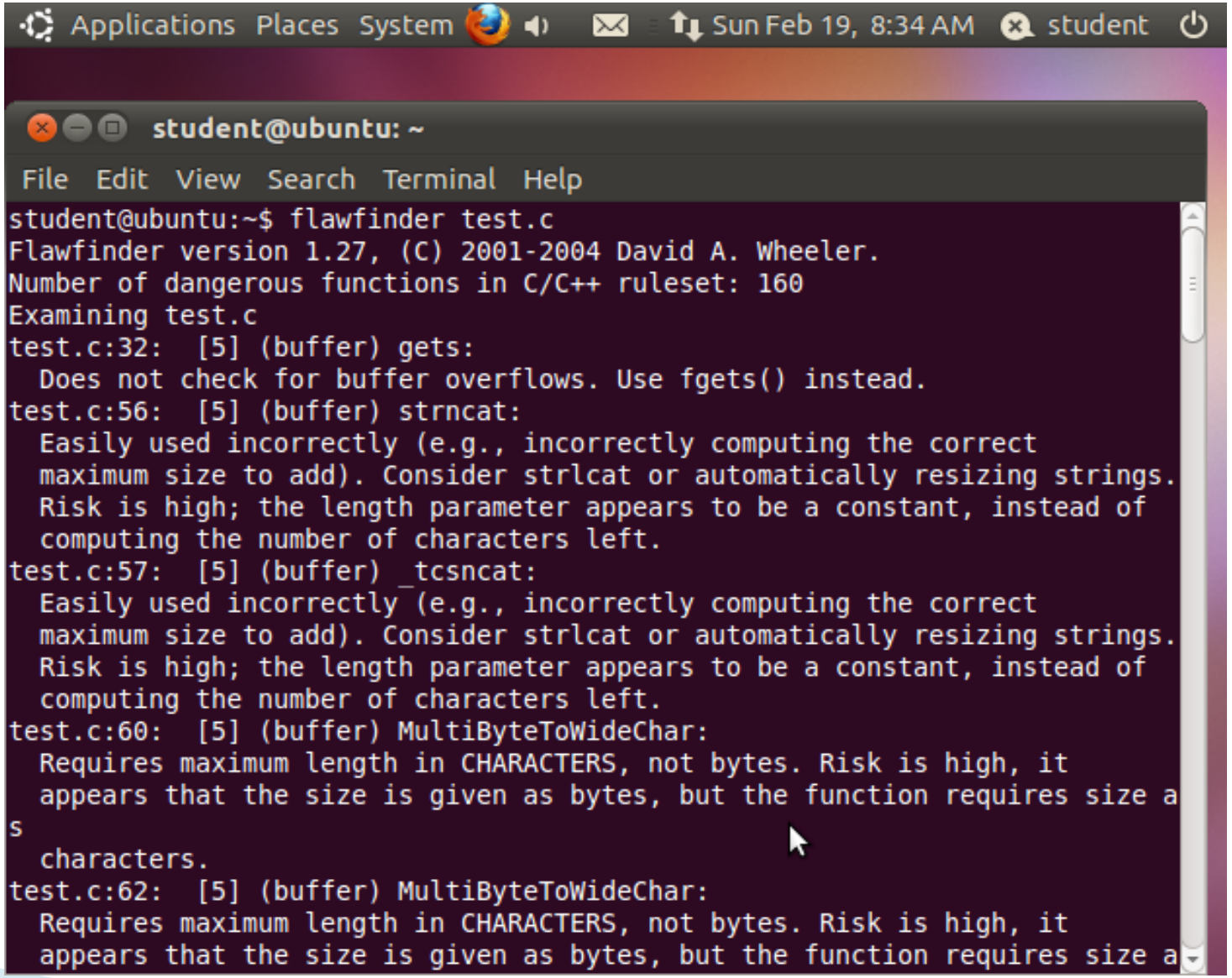
- Two methods of code review
  - Using security checklists (Security Injection Project)
  - Using tools
    - RATS (for C/C++, PHP code)
    - Flawfinder (for C/C++)
    - Fortify (for C/C++, Java, C# code)

- Lab 1. Code review with RATS and Flawfinder
  - Scan a C program with RATS and Flawfinder
  - Compare the two tools
  - Correct the code

Code Review with RATS



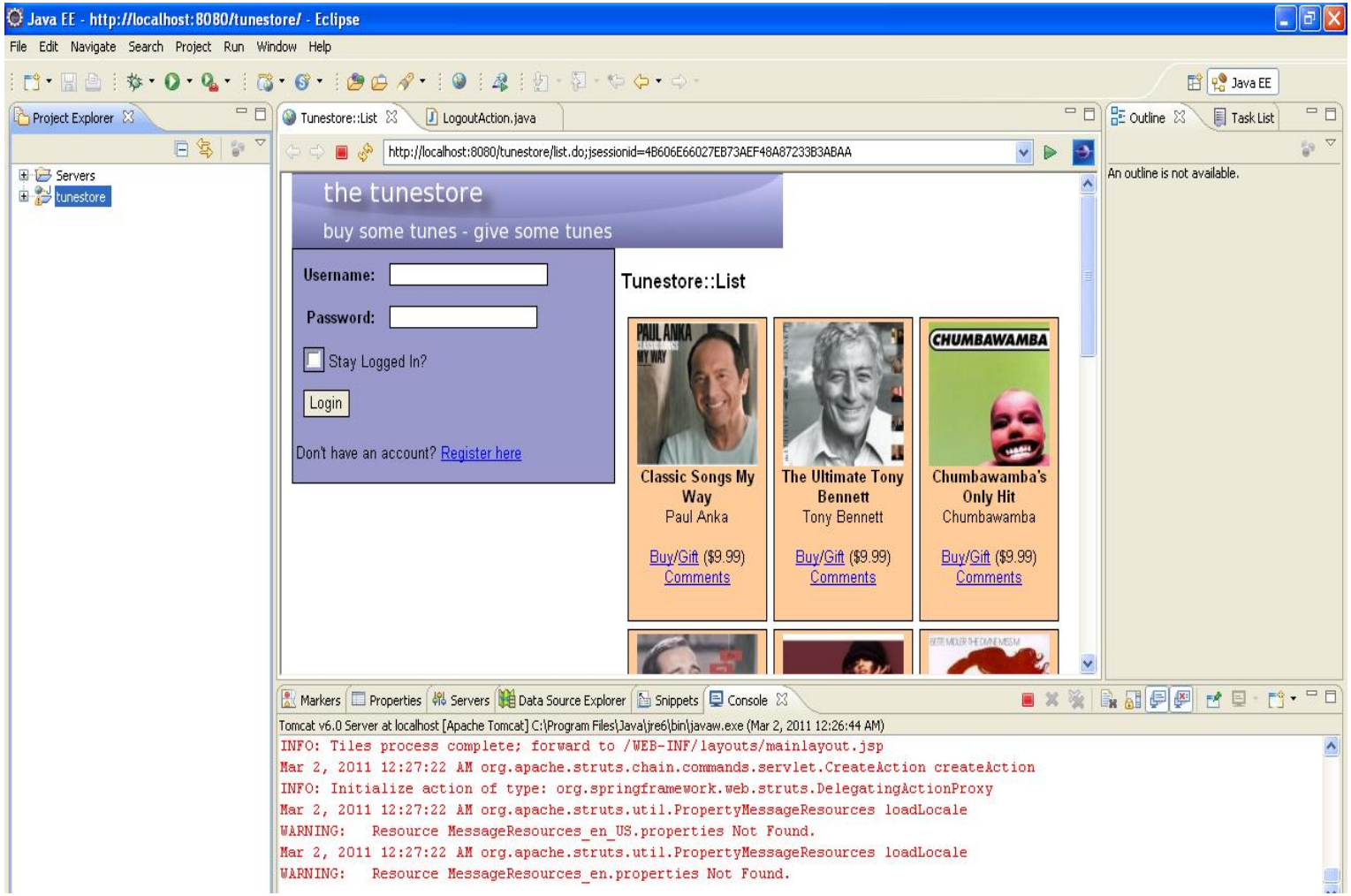
Code Review with FlawFinder



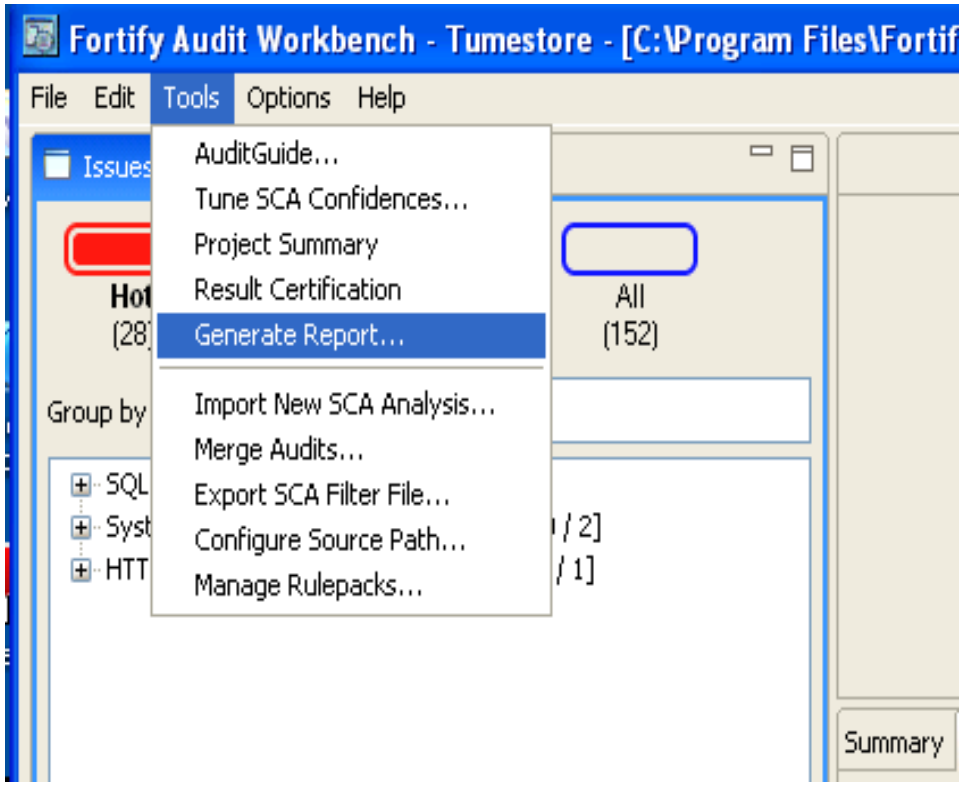
Lab. 2 Code Review with Fortify

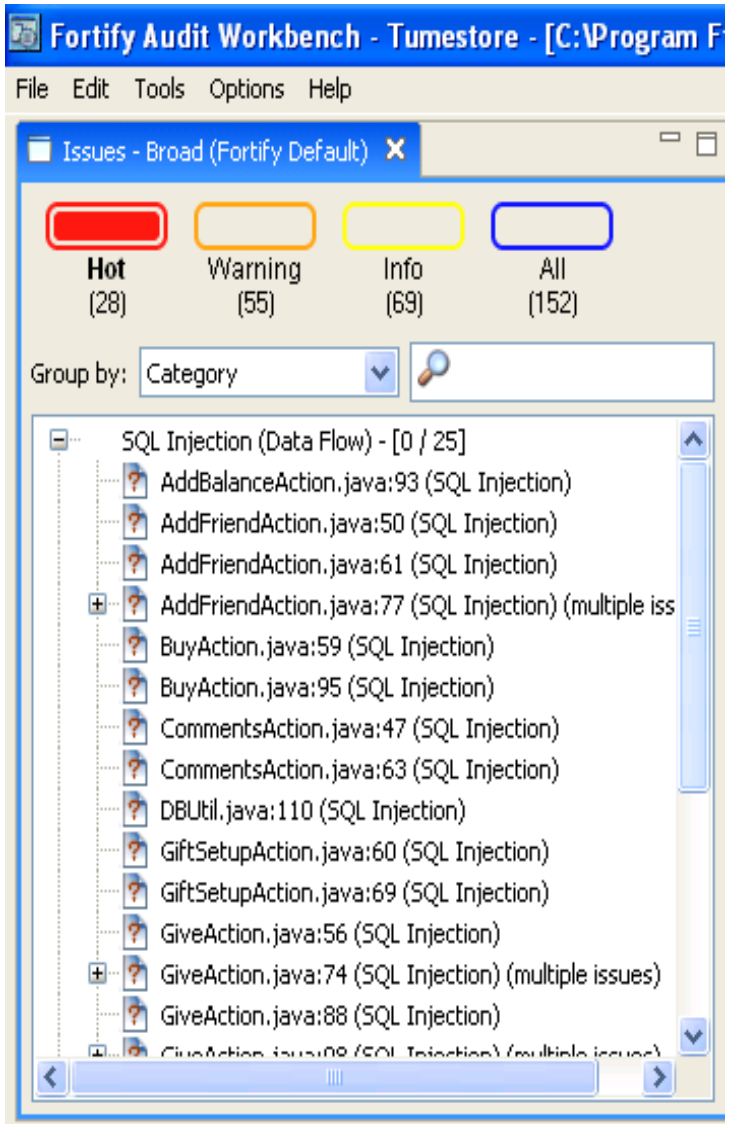
- Use Fortify Source Code Analyzer (SCA) and Audit Workbench to analyze a web application TuneStore
  - Scan source code and generate the vulnerability report
  - Correct code to prevent SQL injection attack
  - Scan code again and demonstrate SQL injection attacks are no longer successful

Example Application: Tunestore



Static Analysis with Fortify





Web Application Vulnerability Assessment

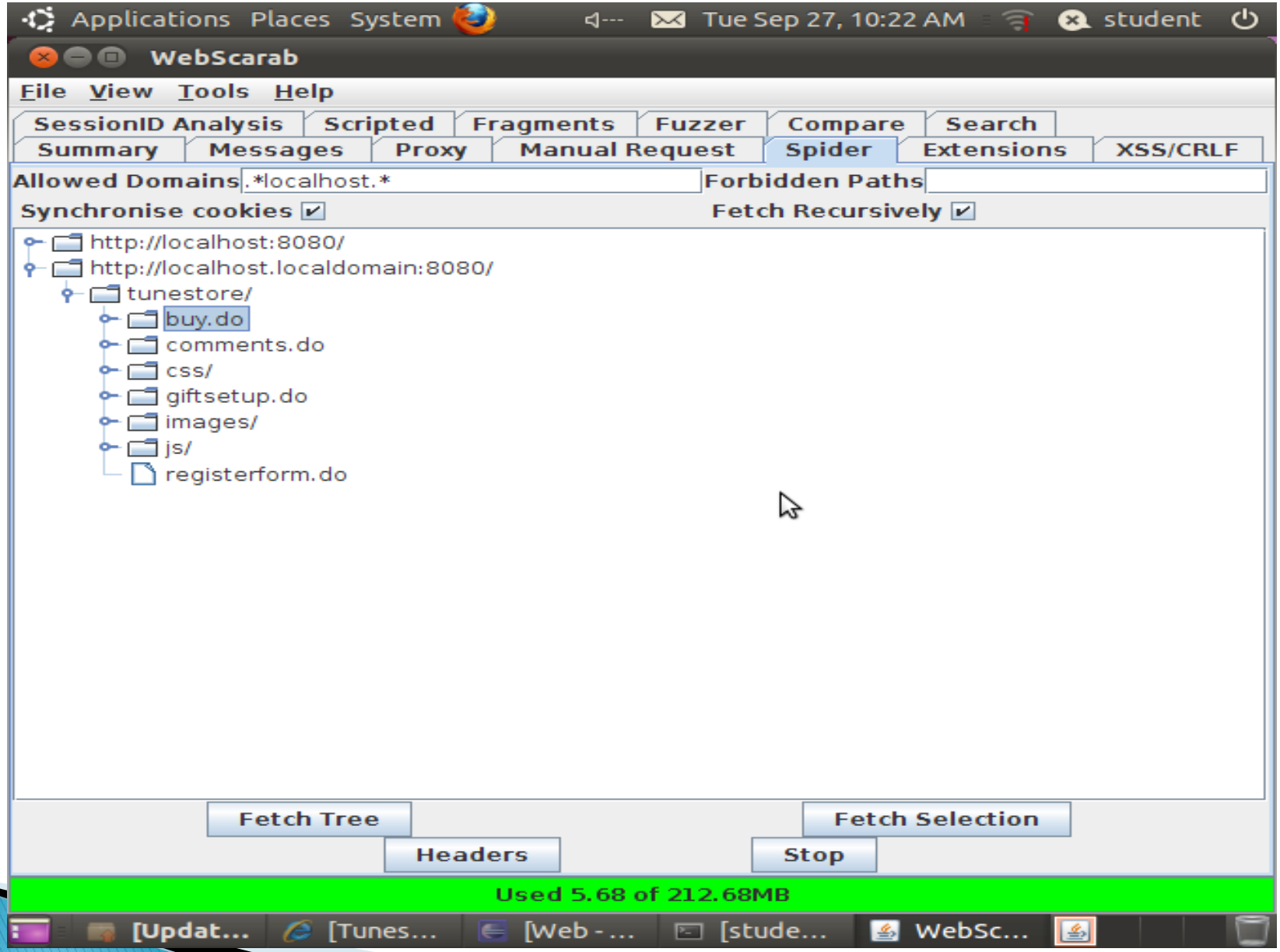
- Common web application vulnerabilities
  - Use hidden HTML form fields
  - Broken authentication
  - Broken access control
  - SQL injection
  - Cross site scripting
  - Information leakage
- WebScarab
  - An open source tool that implements
    - a basic intercepting proxy
    - a website spider
    - A rudimentary fuzzer

Lab 3. Information Gathering with WebScarab

- Use WebScarab's spidering function to enumerate TuneStore's content and functionality

- Compare mapping web application manually with using a spidering tool

Webspidering with WebScarab

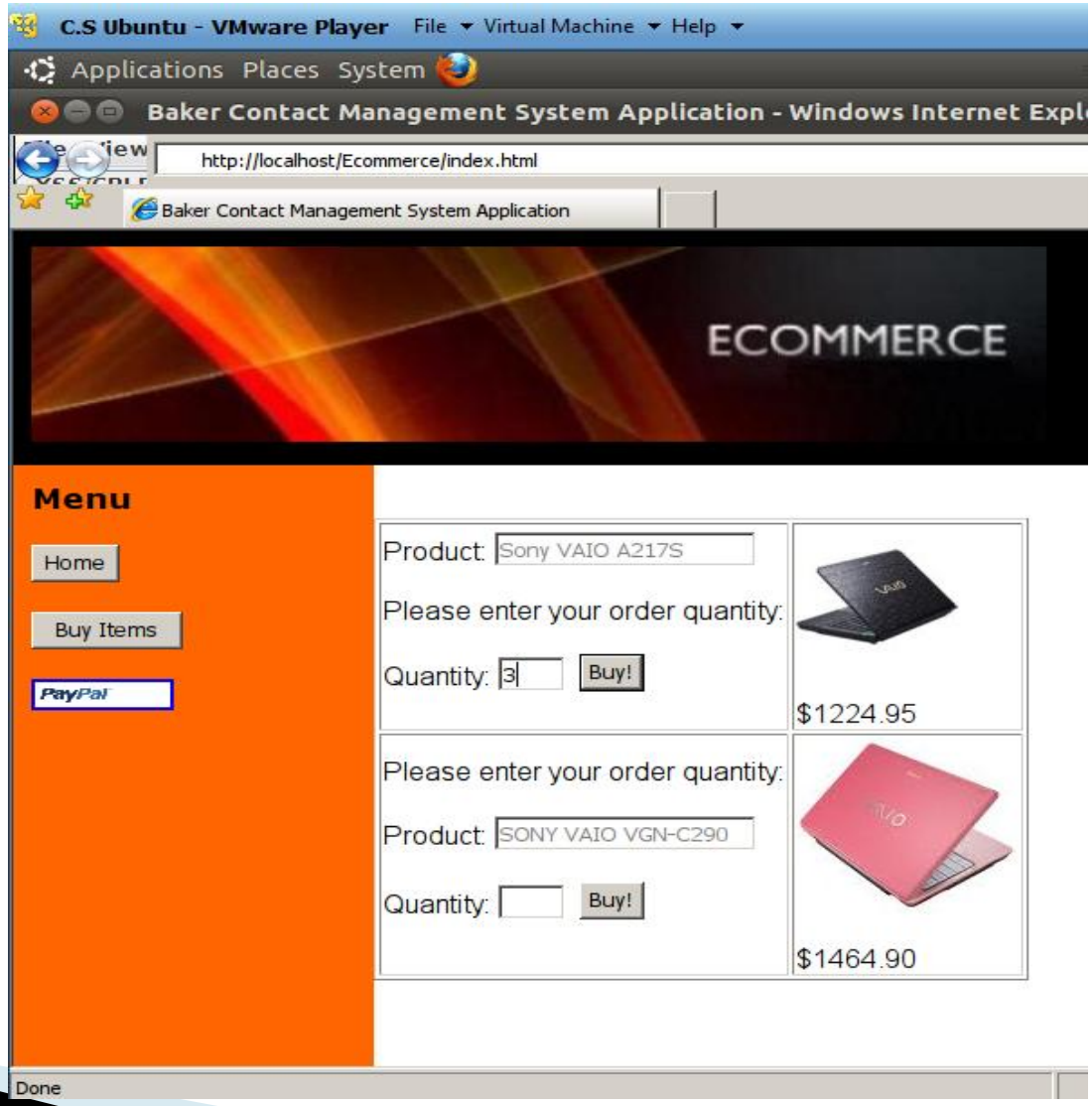


Lab 4. Exploiting Hidden Value

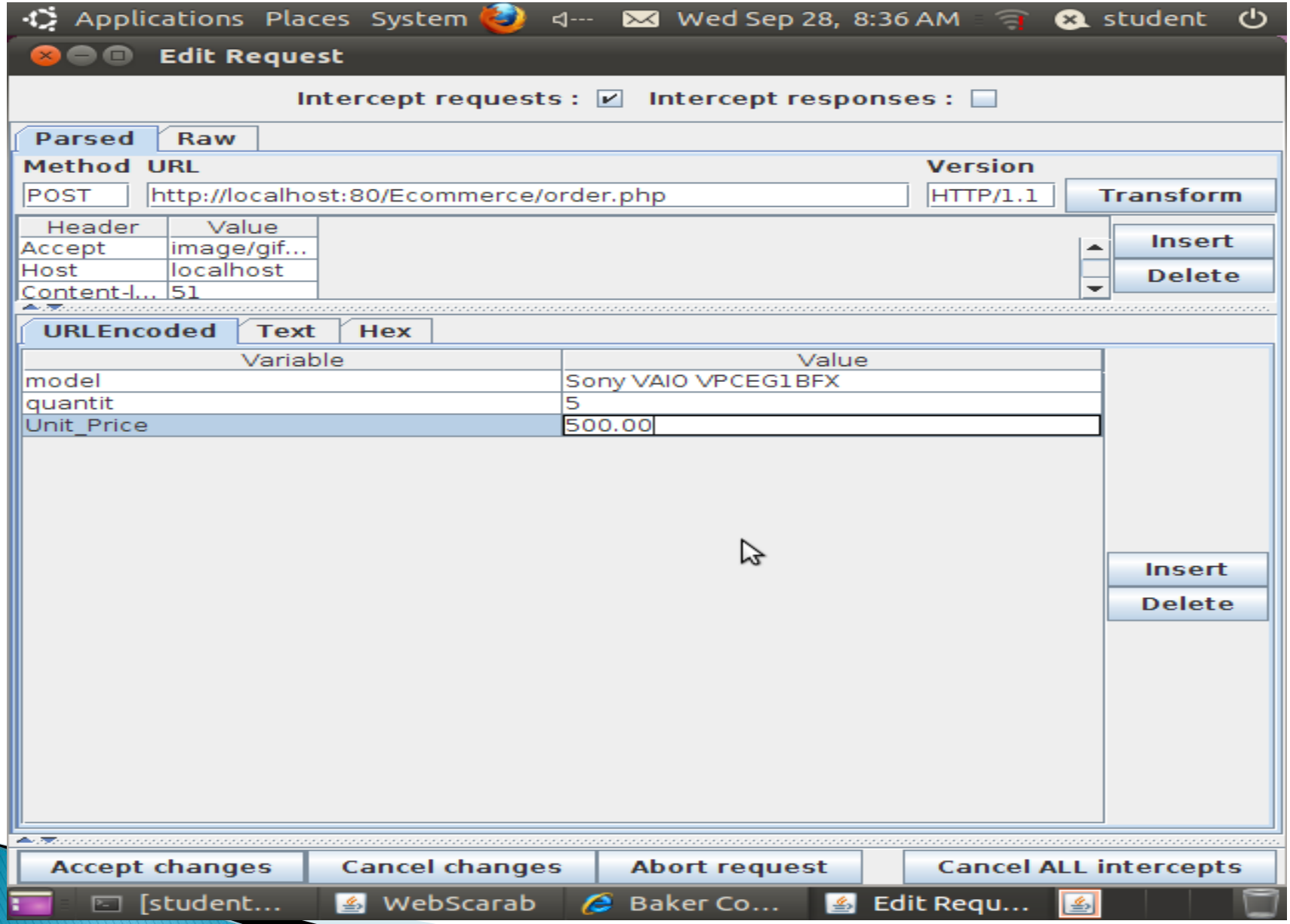
- HTML hidden form fields are not displayed on screen
- They are sent to the application when user submits the form
- Can be manipulated by intercepting proxy

- This lab uses WebScarab's intercepting proxy to manipulate hidden value

Example Application: ECommerce



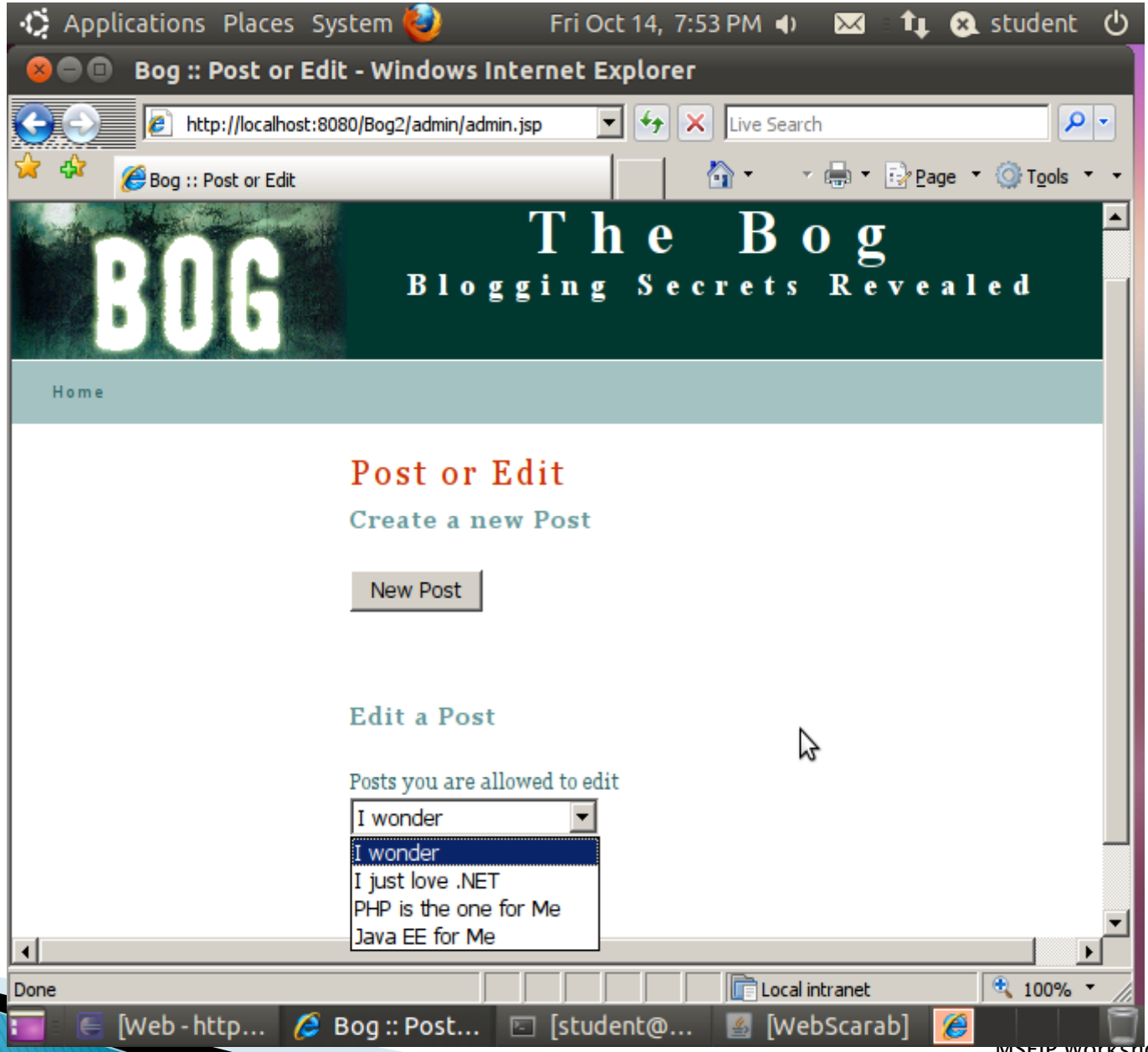
Exploiting Hidden Value with WebScarab



Lab 5. Vulnerability Assessment of TuneStore and BOG

- Assess the vulnerabilities in terms of
  - Authentication
    - Bad passwords, brute force login, verbose failure message, vulnerable transmission of credentials
  - Access control
    - Vertical and horizontal privilege escalation
  - SQL injection
  - Cross site scripting

Example Application: BOG

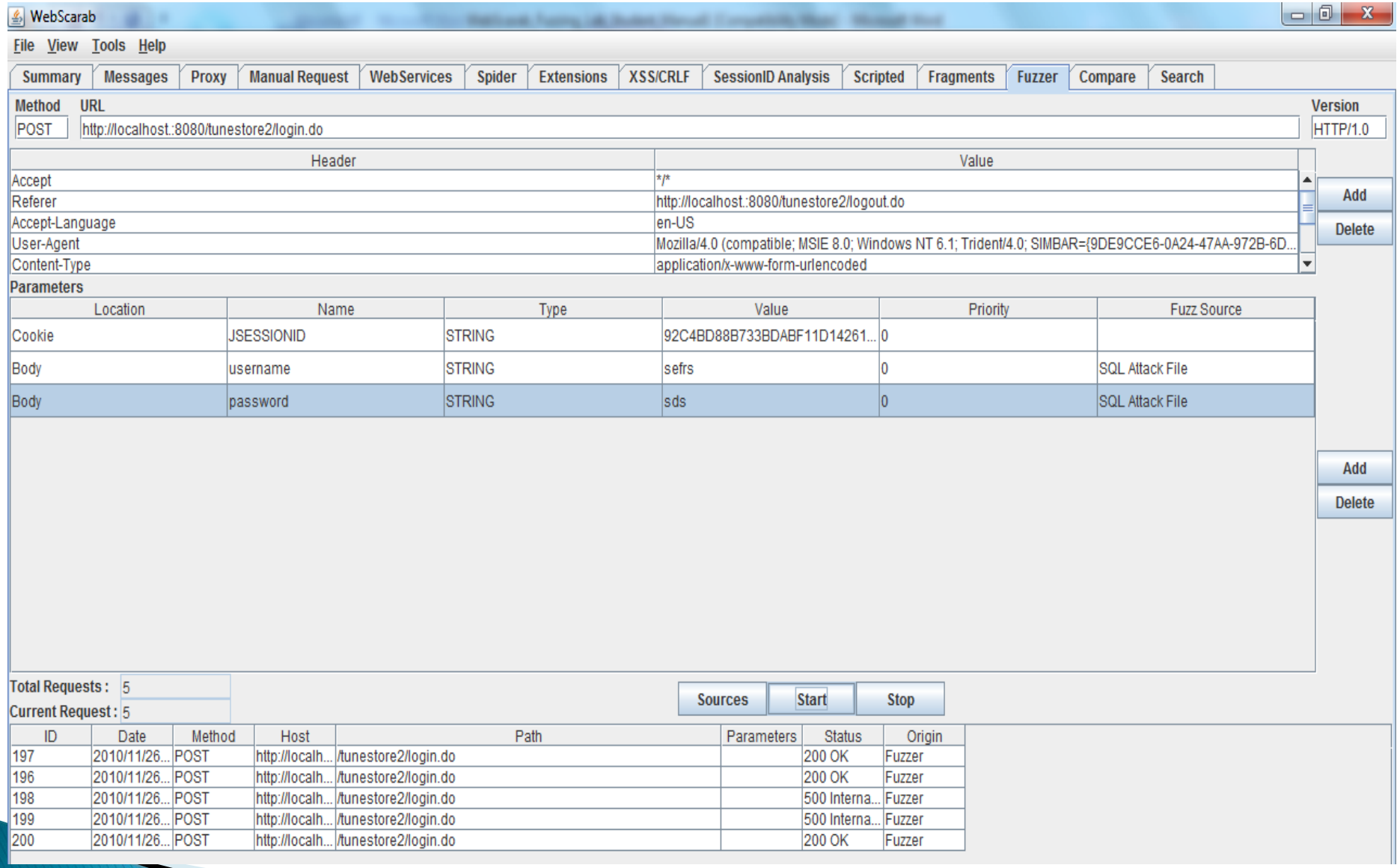


Lab 6. Fuzz Testing with WebScarab

- Tester supplies malformed and unexpected input data to a program in order to find defects

- Use WebScarab Fuzzer plugin to test WebScarab for SQL injection and XSS vulnerabilities
  - Create attack strings
  - Discuss testing results

Fuzz testing with Web Scarab



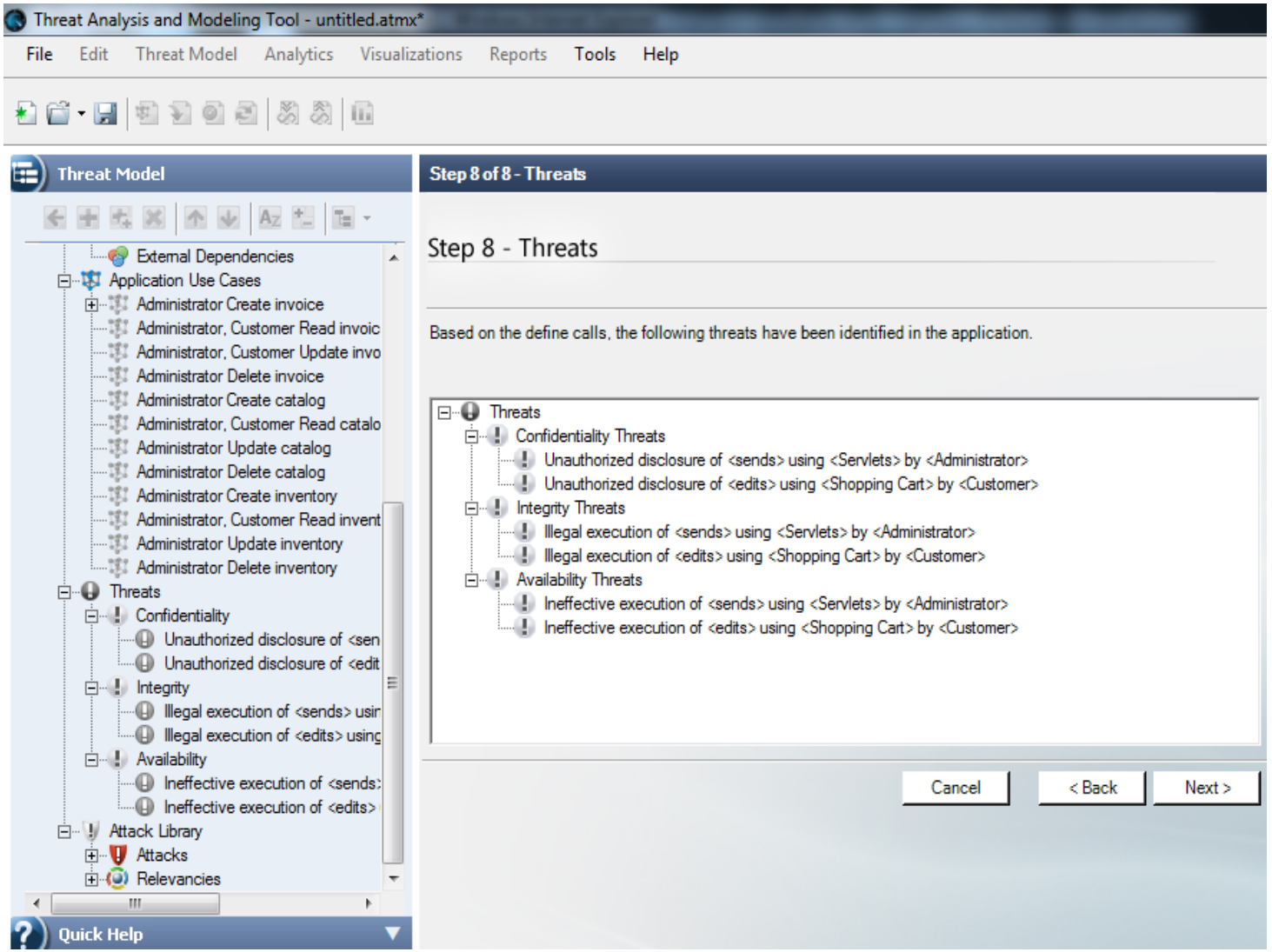
Threat Analysis and Modeling

- The Microsoft's Security Development Lifecycle (SDL)
  - is a **software development security assurance process**
- SDL Threat Modeling Process
  - Use **STRIDE** technique to classify the types of threats
    - **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privileges
  - Risk Assessment based on **DREAD**
    - **D**amage potential, **R**eproducibility, **E**xploitability, # of **A**ffected users, **D**iscoverability

Lab 7 Threat Analysis with TAM

- TAM – Microsoft Threat Analysis and Modeling tool
- Given the description of a system, or a real system
  - Create user roles, data, use cases, access control matrix, and architectural design
  - Use TAM to generate threat model, threat tree, and the comprehensive report

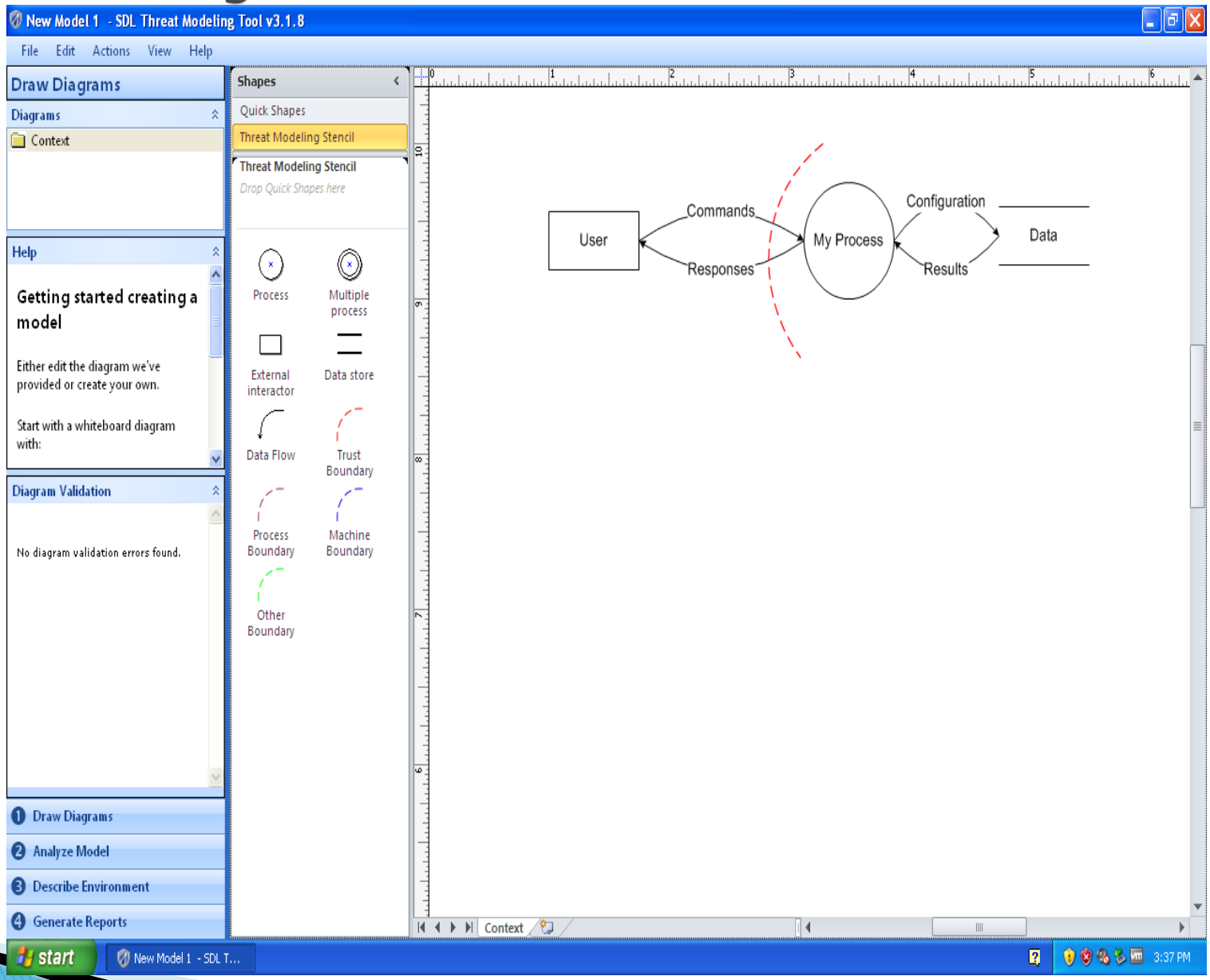
Threat Analysis with TAM



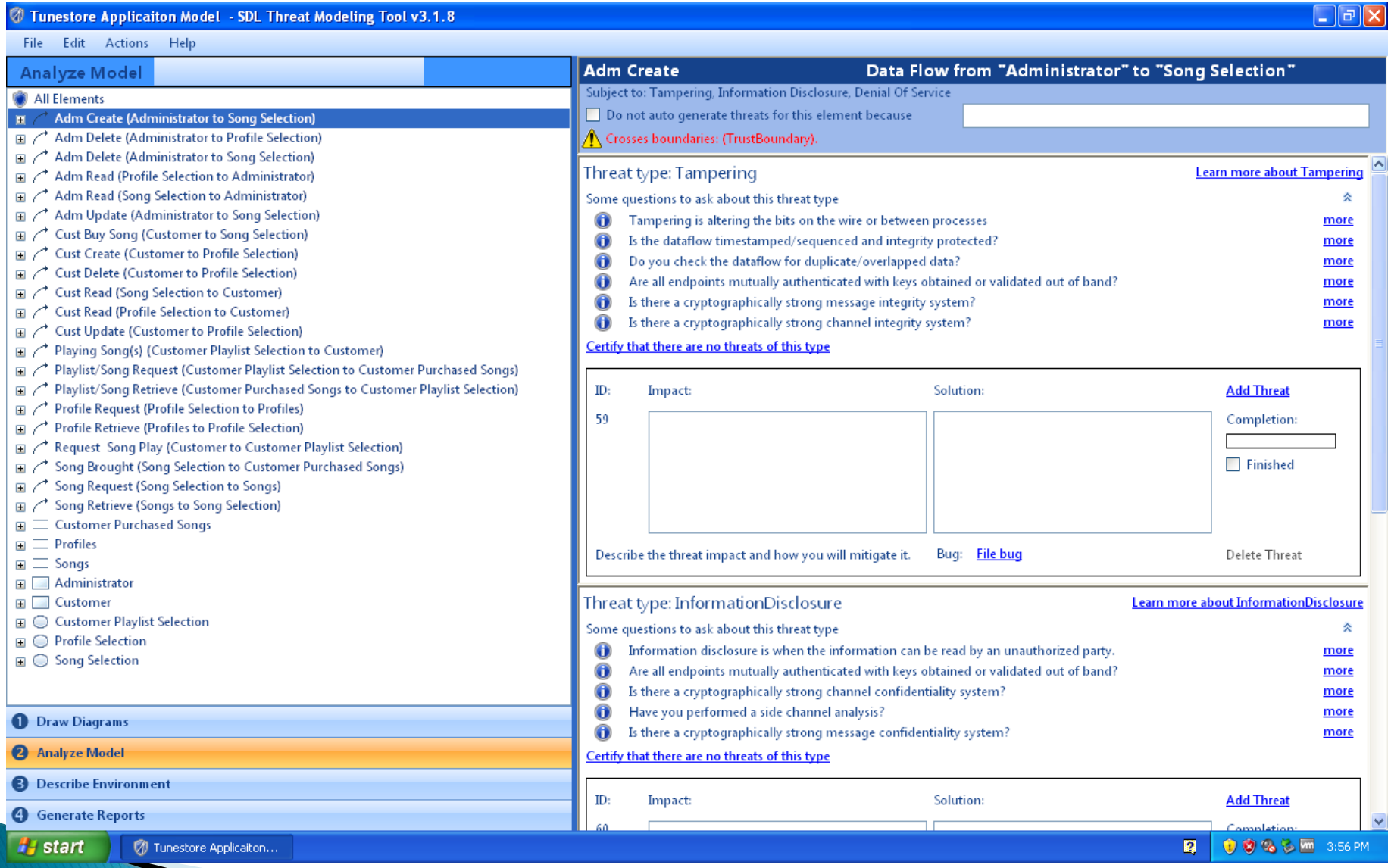
Lab 8: Threat Modeling with SDL

- Given a software system description/real system
  - draw data flow diagram
  - analyze model
  - describe environment
  - generate reports
- Discuss the generated fuzzing report

Threat modeling with SDL – 1



Threat modeling with SDL – 2



The Software and OS Environment for the Labs

Lab	Software Required	OS Environment
Lab 1.	RATS, Flawfinder	Linux
Lab 2.	Fortify	Windows
Lab 3.	WebScarab	Linux/Windows
Lab 4.	WebScarab ECommerce	Linux/Windows
Lab 5.	TuneStore BOG	Linux/Windows
Lab 6.	WebScarab TuneStore	Linux/Windows
Lab 7.	Microsoft Threat Analysis and Modeling Tool (v 3.0)	Windows
Lab 8	Microsoft's SDL Threat Modeling Tool v3.1.8	Windows

Teaching Experiences

- Students were asked to rate the degree of their agreement from 1(strongly disagree) to 5 (strongly agree)
  - I know better about putting the technologies or concepts/theories being taught in practice after finishing the related lab exercise/projects
    - Average rating: 4.25
  - The exercises stimulate my further interests in learning the technology and theories behind software security
    - Average rating: 4.44
- Students liked using virtual machines for these labs

Related Work

- The Security Injection project at Towson Univ.
  - Include lab exercises and security checklists for integer errors, input validation and buffer overflow
- The OWASP WebGoat Project
  - Provides a set of web application security lessons using the deliberately vulnerable application WebGoat
- The Secure Web Development Teaching Modules(SWEET)
  - Include labs that use Paros to crawl web pages, and intercept and tamper requests, labs on SQL injection and Cross Site Scripting attacks
- The SEED project
  - Include cross site scripting attack labs using a vulnerable web-based message board phpBB.

Conclusion

- We described eight hands-on exercises for teaching software security
- Our future work will be
  - Develop more hands-on labs and case studies for teaching software security
  - Evaluate the effectiveness of these hands-on labs

Questions