

Hands-on Laboratory Exercises for Teaching Software Security

Xiaohong Yuan¹, Joaquin Hernandez¹, India Waddell¹, Bill Chu², Huiming Yu¹
¹North Carolina A&T State University, Greensboro, NC
²University 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

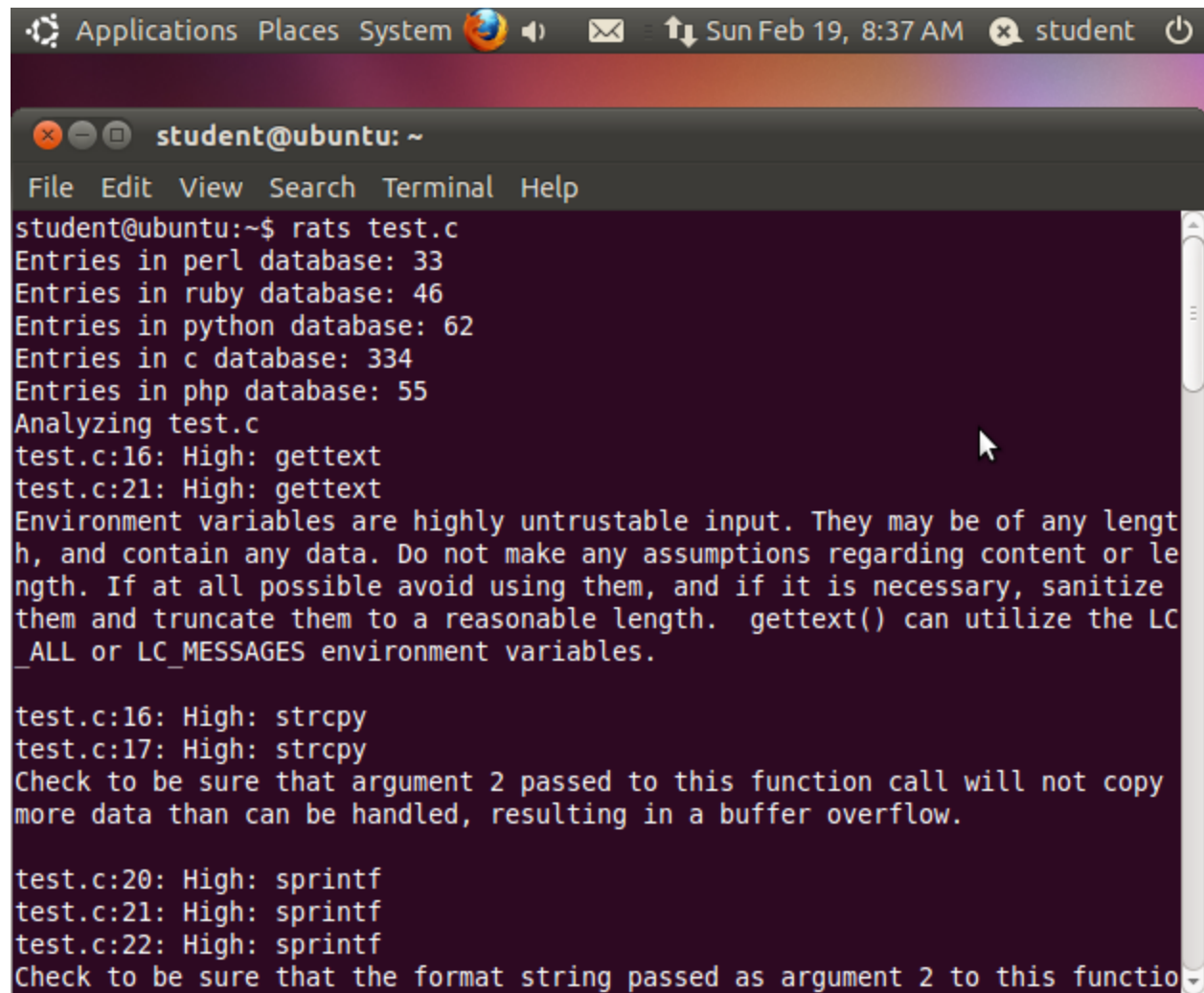
Hands-on Laboratory Exercises







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

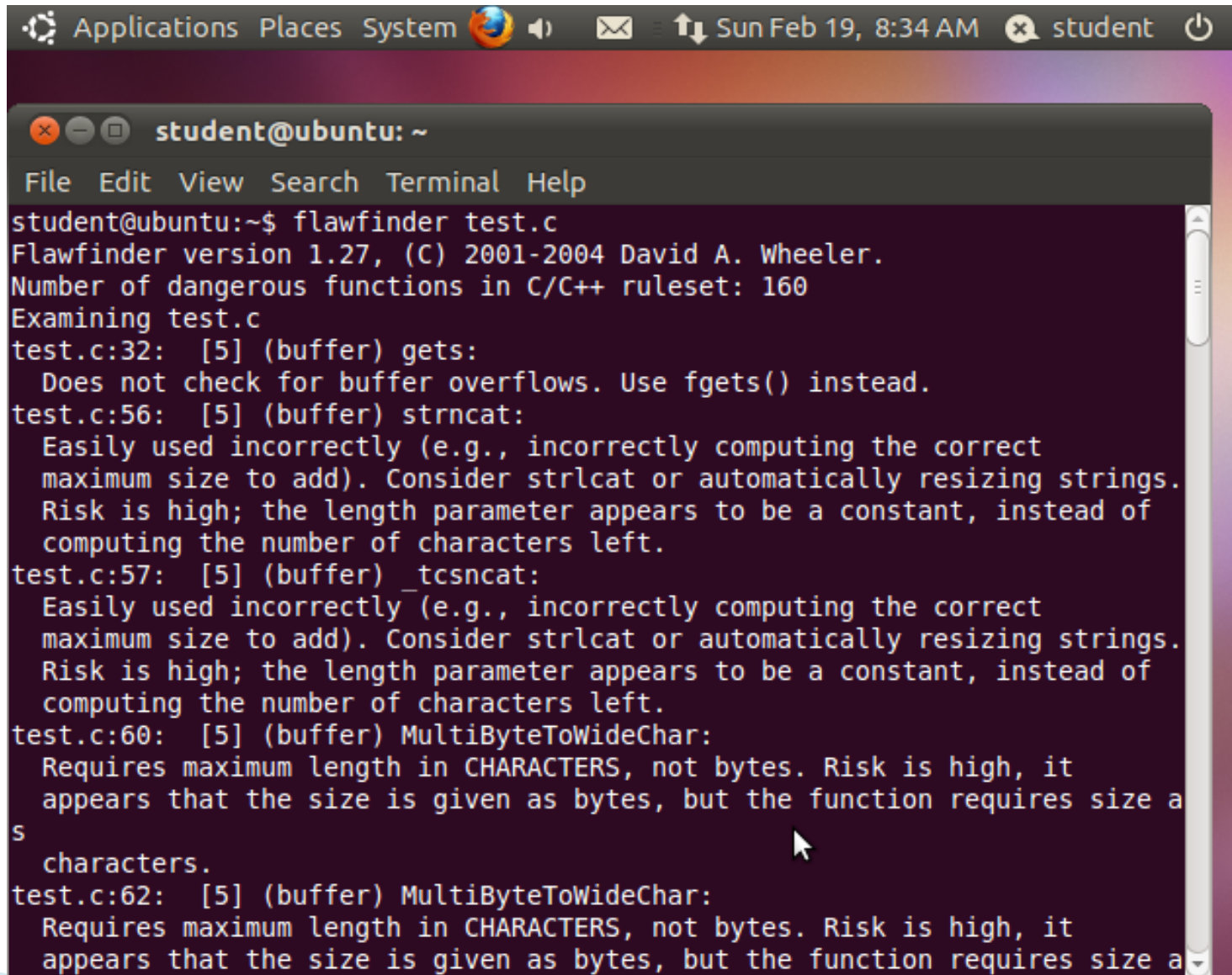
Code Review with RATS









```
Applications Places System     Sun Feb 19, 8:37 AM  student 
```

```
student@ubuntu: ~  
File Edit View Search Terminal Help  
student@ubuntu:~$ rats test.c  
Entries in perl database: 33  
Entries in ruby database: 46  
Entries in python database: 62  
Entries in c database: 334  
Entries in php database: 55  
Analyzing test.c  
test.c:16: High: gettext  
test.c:21: High: gettext  
Environment variables are highly untrustable input. They may be of any length, and contain any data. Do not make any assumptions regarding content or length. If at all possible avoid using them, and if it is necessary, sanitize them and truncate them to a reasonable length. gettext() can utilize the LC_ALL or LC_MESSAGES environment variables.  
  
test.c:16: High: strcpy  
test.c:17: High: strcpy  
Check to be sure that argument 2 passed to this function call will not copy more data than can be handled, resulting in a buffer overflow.  
  
test.c:20: High: sprintf  
test.c:21: High: sprintf  
test.c:22: High: sprintf  
Check to be sure that the format string passed as argument 2 to this function
```

Code Review with FlawFinder



```
Applications Places System     Sun Feb 19, 8:34 AM  student 
```

```
student@ubuntu: ~  
File Edit View Search Terminal Help  
student@ubuntu:~$ flawfinder test.c  
Flawfinder version 1.27, (C) 2001-2004 David A. Wheeler.  
Number of dangerous functions in C/C++ ruleset: 160  
Examining test.c  
test.c:32: [5] (buffer) gets:  
  Does not check for buffer overflows. Use fgets() instead.  
test.c:56: [5] (buffer) strcat:  
  Easily used incorrectly (e.g., incorrectly computing the correct  
  maximum size to add). Consider strlcat or automatically resizing strings.  
  Risk is high; the length parameter appears to be a constant, instead of  
  computing the number of characters left.  
test.c:57: [5] (buffer) _tcsncat:  
  Easily used incorrectly (e.g., incorrectly computing the correct  
  maximum size to add). Consider strlcat or automatically resizing strings.  
  Risk is high; the length parameter appears to be a constant, instead of  
  computing the number of characters left.  
test.c:60: [5] (buffer) MultiByteToWideChar:  
  Requires maximum length in CHARACTERS, not bytes. Risk is high, it  
  appears that the size is given as bytes, but the function requires size a  
s  
  characters.  
test.c:62: [5] (buffer) MultiByteToWideChar:  
  Requires maximum length in CHARACTERS, not bytes. Risk is high, it  
  appears that the size is given as bytes, but the function requires size a
```

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

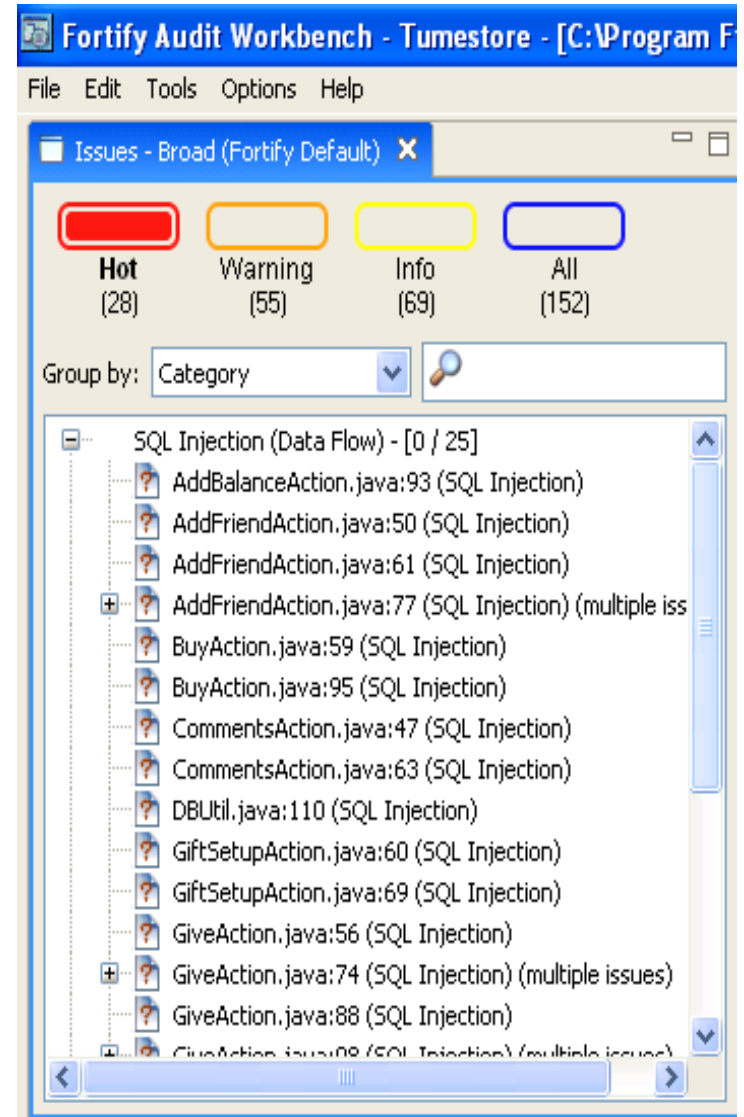
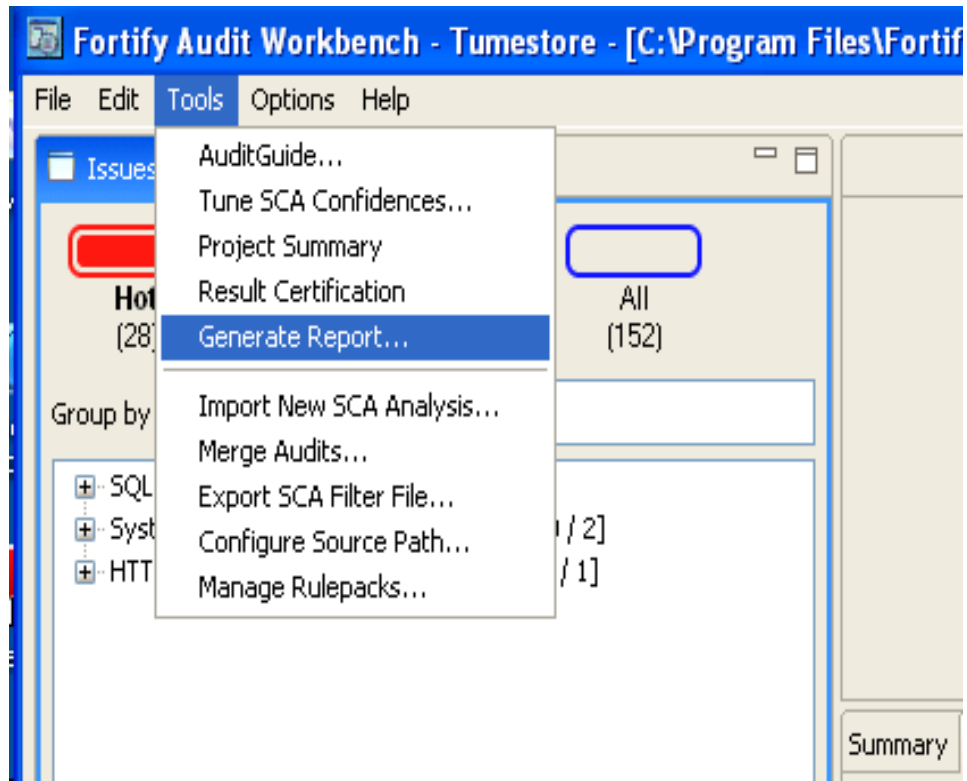
The screenshot displays the Eclipse IDE interface with a web browser window open. The browser shows the URL `http://localhost:8080/tunestore/list.do;jsessionid=4B606E66027EB73AEF48A87233B3ABAA`. The page content includes a header "the tunestore" with the tagline "buy some tunes - give some tunes". On the left, there is a login form with fields for "Username:" and "Password:", a "Stay Logged In?" checkbox, and a "Login" button. Below the login form is a link: "Don't have an account? [Register here](#)". On the right, there is a section titled "Tunestore::List" containing three music items:

Album Title	Artist	Price	Link
Classic Songs My Way	Paul Anka	\$9.99	Buy/Gift
The Ultimate Tony Bennett	Tony Bennett	\$9.99	Buy/Gift
Chumbawamba's Only Hit	Chumbawamba	\$9.99	Buy/Gift

Each item also has a "Comments" link. The Eclipse IDE interface includes a Project Explorer on the left showing the "Tunestore" project, and a Console at the bottom with the following log output:

```
Tomcat v6.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre6\bin\javaw.exe (Mar 2, 2011 12:26:44 AM)
INFO: Tiles process complete; forward to /WEB-INF/layouts/mainlayout.jsp
Mar 2, 2011 12:27:22 AM org.apache.struts.chain.commands.servlet.CreateAction createAction
INFO: Initialize action of type: org.springframework.web.struts.DelegatingActionProxy
Mar 2, 2011 12:27:22 AM org.apache.struts.util.PropertyMessageResources loadLocale
WARNING: Resource MessageResources_en_US.properties Not Found.
Mar 2, 2011 12:27:22 AM org.apache.struts.util.PropertyMessageResources loadLocale
WARNING: Resource MessageResources_en.properties Not Found.
```

Static Analysis with Fortify



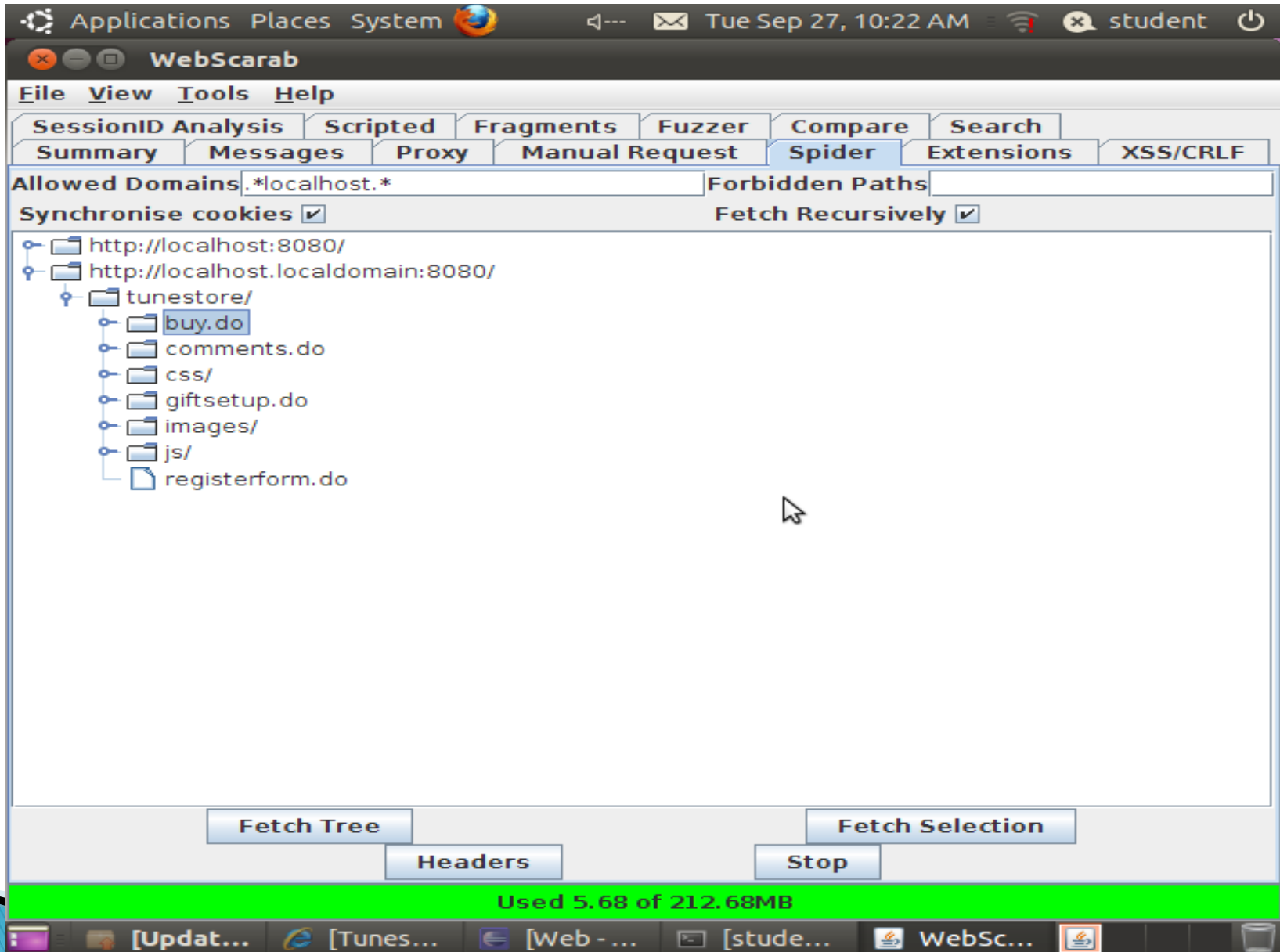
Web Application Vulnerability Assessment

- ▶ 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

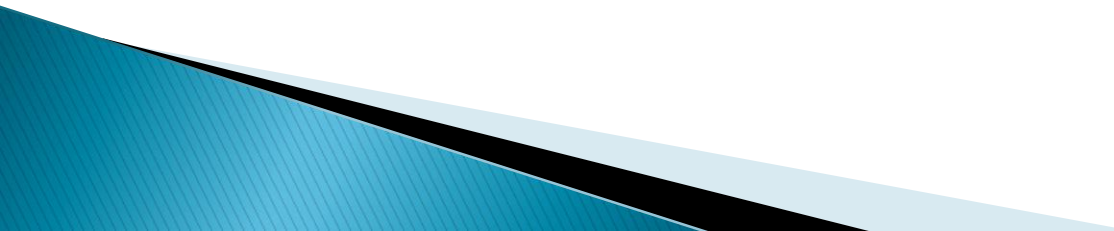
Lab 3. Information Gathering with WebScarab

- ▶ 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

The screenshot shows a web browser window titled "Baker Contact Management System Application - Windows Internet Expl" with the address bar displaying "http://localhost/Ecommerce/index.html". The page features a header with the word "ECOMMERCE" and a navigation menu on the left with buttons for "Home", "Buy Items", and "PayPal". The main content area displays two product listings:

Product: <input type="text" value="Sony VAIO A217S"/>	
Please enter your order quantity:	
Quantity: <input type="text" value="3"/> <input data-bbox="1110 906 1174 939" type="button" value="Buy!"/>	
	\$1224.95
Please enter your order quantity:	
Product: <input type="text" value="SONY VAIO VGN-C290"/>	
Quantity: <input type="text"/> <input data-bbox="1110 1135 1174 1168" type="button" value="Buy!"/>	
	\$1464.90

The status bar at the bottom of the browser window shows "Done".

Exploiting Hidden Value with WebScarab

The screenshot shows the WebScarab application window titled "Edit Request". At the top, there are checkboxes for "Intercept requests" (checked) and "Intercept responses" (unchecked). Below this, there are two tabs: "Parsed" (selected) and "Raw".

The "Parsed" view shows the following details:

- Method:** POST
- URL:** http://localhost:80/Ecommerce/order.php
- Version:** HTTP/1.1

Below the method and URL, there is a table of headers:

Header	Value
Accept	image/gif...
Host	localhost
Content-L...	51

To the right of the header table are "Insert" and "Delete" buttons.

Below the headers, there are three tabs: "URLEncoded" (selected), "Text", and "Hex". The "URLEncoded" view shows a table of variables:

Variable	Value
model	Sony VAIO VPCEG1BFX
quantit	5
Unit_Price	500.00

To the right of the variable table are "Insert" and "Delete" buttons.

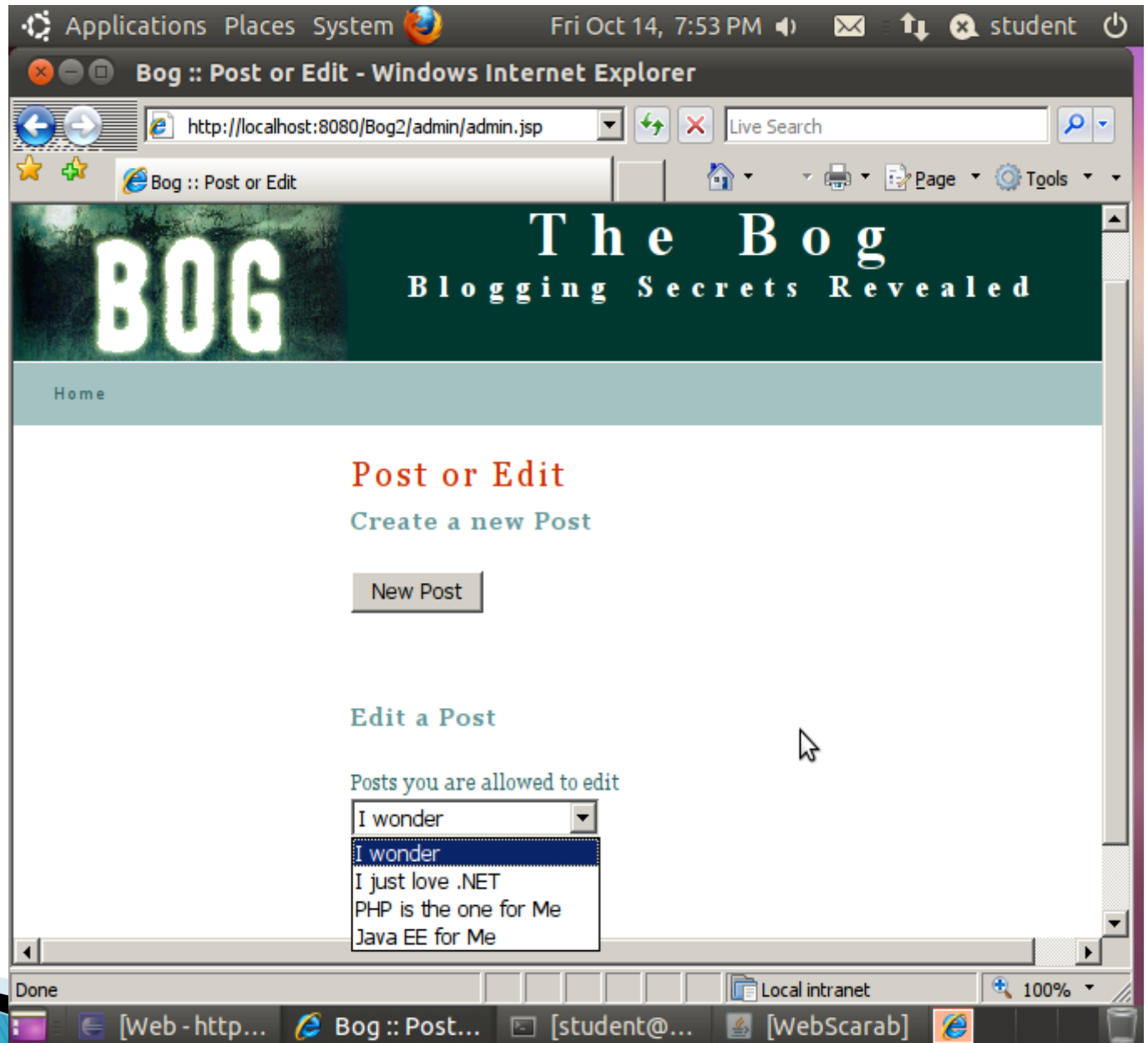
At the bottom of the window, there are four buttons: "Accept changes", "Cancel changes", "Abort request", and "Cancel ALL intercepts".

The system tray at the bottom shows the following icons: [student...], WebScarab, Baker Co..., Edit Requ..., and a trash can icon.

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

WebScarab

File View Tools Help

Summary Messages Proxy Manual Request WebServices Spider Extensions XSS/CRLF SessionID Analysis Scripted Fragments **Fuzzer** Compare Search

Method: POST URL: http://localhost:8080/tunestore2/login.do Version: HTTP/1.0

Header	Value
Accept	*/*
Referer	http://localhost:8080/tunestore2/logout.do
Accept-Language	en-US
User-Agent	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SIMBAR={9DE9CCE6-0A24-47AA-972B-6D...
Content-Type	application/x-www-form-urlencoded

Parameters

Location	Name	Type	Value	Priority	Fuzz Source
Cookie	JSESSIONID	STRING	92C4BD88B733BDABF11D14261...	0	
Body	username	STRING	sefrs	0	SQL Attack File
Body	password	STRING	sds	0	SQL Attack File

Total Requests: 5
Current Request: 5

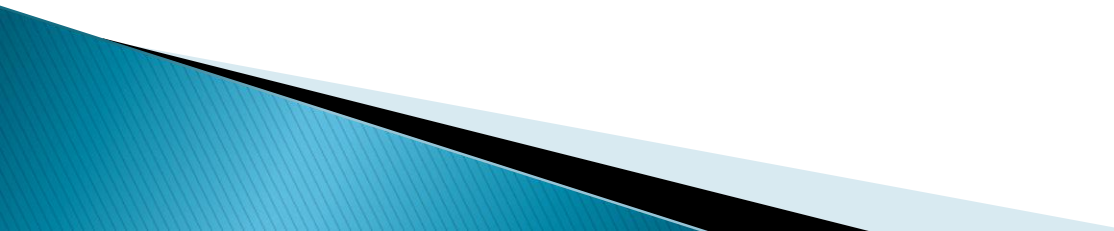
Sources Start Stop

ID	Date	Method	Host	Path	Parameters	Status	Origin
197	2010/11/26...	POST	http://localh...	/tunestore2/login.do		200 OK	Fuzzer
196	2010/11/26...	POST	http://localh...	/tunestore2/login.do		200 OK	Fuzzer
198	2010/11/26...	POST	http://localh...	/tunestore2/login.do		500 Interna...	Fuzzer
199	2010/11/26...	POST	http://localh...	/tunestore2/login.do		500 Interna...	Fuzzer
200	2010/11/26...	POST	http://localh...	/tunestore2/login.do		200 OK	Fuzzer

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
 - Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privileges
 - Risk Assessment based on **DREAD**
 - Damage potential, Reproducibility, Exploitability, # of Affected users, Discoverability

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

The screenshot displays the Threat Analysis and Modeling Tool (TAM) interface. The title bar reads "Threat Analysis and Modeling Tool - untitled.atmx*". The menu bar includes "File", "Edit", "Threat Model", "Analytics", "Visualizations", "Reports", "Tools", and "Help". The toolbar contains various icons for file operations and analysis.

The main window is divided into two panes. The left pane, titled "Threat Model", shows a tree view of the model structure:

- External Dependencies
- Application Use Cases
 - Administrator Create invoice
 - Administrator, Customer Read invoic
 - Administrator, Customer Update invo
 - Administrator Delete invoice
 - Administrator Create catalog
 - Administrator, Customer Read catalo
 - Administrator Update catalog
 - Administrator Delete catalog
 - Administrator Create inventory
 - Administrator, Customer Read invent
 - Administrator Update inventory
 - Administrator Delete inventory
- Threats
 - Confidentiality
 - Unauthorized disclosure of <sen
 - Unauthorized disclosure of <edit
 - Integrity
 - Illegal execution of <sends> usir
 - Illegal execution of <edits> using
 - Availability
 - Ineffective execution of <sends>
 - Ineffective execution of <edits>
- Attack Library
 - Attacks
 - Relevancies

The right pane, titled "Step 8 of 8 - Threats", displays the results of the threat analysis:

Step 8 - Threats

Based on the define calls, the following threats have been identified in the application.

- Threats
 - Confidentiality Threats
 - Unauthorized disclosure of <sends> using <Servlets> by <Administrator>
 - Unauthorized disclosure of <edits> using <Shopping Cart> by <Customer>
 - Integrity Threats
 - Illegal execution of <sends> using <Servlets> by <Administrator>
 - Illegal execution of <edits> using <Shopping Cart> by <Customer>
 - Availability Threats
 - Ineffective execution of <sends> using <Servlets> by <Administrator>
 - Ineffective execution of <edits> using <Shopping Cart> by <Customer>

At the bottom of the right pane, there are three buttons: "Cancel", "< Back", and "Next >".

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

New Model 1 - SDL Threat Modeling Tool v3.1.8

File Edit Actions View Help

Draw Diagrams

Diagrams

- Context

Help

Getting started creating a model

Either edit the diagram we've provided or create your own.

Start with a whiteboard diagram with:

Diagram Validation

No diagram validation errors found.

Shapes

Quick Shapes

Threat Modeling Stencil

Drop Quick Shapes here

- Process
- Multiple process
- External interactor
- Data store
- Data Flow
- Trust Boundary
- Process Boundary
- Machine Boundary
- Other Boundary

```
graph LR; User[User] -- Commands --> MyProcess((My Process)); MyProcess -- Responses --> User; MyProcess -- Configuration --> Data[Data]; Data -- Results --> MyProcess; subgraph TrustBoundary [Trust Boundary]; User; MyProcess; end
```

1 Draw Diagrams

2 Analyze Model

3 Describe Environment

4 Generate Reports

start New Model 1 - SDL T... 3:37 PM

Threat modeling with SDL – 2

Tunestore Application Model - SDL Threat Modeling Tool v3.1.8

File Edit Actions Help

Analyze Model

- All Elements
- Adm Create (Administrator to Song Selection)
- Adm Delete (Administrator to Profile Selection)
- Adm Delete (Administrator to Song Selection)
- Adm Read (Profile Selection to Administrator)
- Adm Read (Song Selection to Administrator)
- Adm Update (Administrator to Song Selection)
- Cust Buy Song (Customer to Song Selection)
- Cust Create (Customer to Profile Selection)
- Cust Delete (Customer to Profile Selection)
- Cust Read (Song Selection to Customer)
- Cust Read (Profile Selection to Customer)
- Cust Update (Customer to Profile Selection)
- Playing Song(s) (Customer Playlist Selection to Customer)
- Playlist/Song Request (Customer Playlist Selection to Customer Purchased Songs)
- Playlist/Song Retrieve (Customer Purchased Songs to Customer Playlist Selection)
- Profile Request (Profile Selection to Profiles)
- Profile Retrieve (Profiles to Profile Selection)
- Request Song Play (Customer to Customer Playlist Selection)
- Song Brought (Song Selection to Customer Purchased Songs)
- Song Request (Song Selection to Songs)
- Song Retrieve (Songs to Song Selection)
- Customer Purchased Songs
- Profiles
- Songs
- Administrator
- Customer
- Customer Playlist Selection
- Profile Selection
- Song Selection

Adm Create Data Flow from "Administrator" to "Song Selection"

Subject to: Tampering, Information Disclosure, Denial Of Service

Do not auto generate threats for this element because

⚠ Crosses boundaries: (TrustBoundary).

Threat type: Tampering [Learn more about Tampering](#)

Some questions to ask about this threat type

- Tampering is altering the bits on the wire or between processes [more](#)
- Is the dataflow timestamped/sequenced and integrity protected? [more](#)
- Do you check the dataflow for duplicate/overlapped data? [more](#)
- Are all endpoints mutually authenticated with keys obtained or validated out of band? [more](#)
- Is there a cryptographically strong message integrity system? [more](#)
- Is there a cryptographically strong channel integrity system? [more](#)

[Certify that there are no threats of this type](#)

ID:	Impact:	Solution:	Add Threat
59			Completion: <input type="text"/> <input type="checkbox"/> Finished
60			Completion: <input type="text"/>

Describe the threat impact and how you will mitigate it. Bug: [File bug](#) Delete Threat

Threat type: InformationDisclosure [Learn more about InformationDisclosure](#)

Some questions to ask about this threat type

- Information disclosure is when the information can be read by an unauthorized party. [more](#)
- Are all endpoints mutually authenticated with keys obtained or validated out of band? [more](#)
- Is there a cryptographically strong channel confidentiality system? [more](#)
- Have you performed a side channel analysis? [more](#)
- Is there a cryptographically strong message confidentiality system? [more](#)

[Certify that there are no threats of this type](#)

ID:	Impact:	Solution:	Add Threat
60			Completion: <input type="text"/>

start Tunestore Application... 3:56 PM

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

