Hands-on Security Tools

SecAppDev 2012

KRvW Associates, LLC

Caveats and Warnings

This is not a sales pitch for any product(s)

- If you want to talk to a sales person, tell me
- Otherwise, you will NOT get calls or spam

You are not authorized to “test” any systems other than your own

- If you do, then don’t call me from prison
- I don’t know you

Prerequisites

Computer (shared or solo)

- Windows, OS X, Linux
- Local admin access

Virtual machine environment (Vmware, Parallels, Virtual Box)

JRE 1.5+

Development environment (for source analysis tool)

- C or Java
- Make, Ant, Eclipse (3 or 2), Visual Studio, etc

Objectives and Intros

We'll look at several tools

Idea is to give everyone a glimpse of several tools

- NOT to turn anyone into an expert on any tool

Safe, sales-free env

Flow

- Describe each tool
- Demo (where applicable)
- Class tries tool with specific objectives
- Discuss results and applicability

Secondary Goals

Learn
Experiment with the tools
Judge for yourself
Have fun

Setup environment

We'll use a combination of stuff

- Virtual Machine - OWASP's WTE
- Desktop installation of Fortify

Virtual machine tips

- Allocate at least 1 Gb to the VM
- Either disable network or use shared net through host OS

Software security tools

Categories include

- Static code analysis tools
- Testing tools
  - Application proxy tools
  - Fuzzers

Web application testing

First, the manual approach
- A lot of times, there's no substitute for this
- Kind of like a single-stepping debugger

Testing proxies are useful
- Man-in-the-middle between browser and app

Examples
- WebScarab, ZAP

The tools we'll use

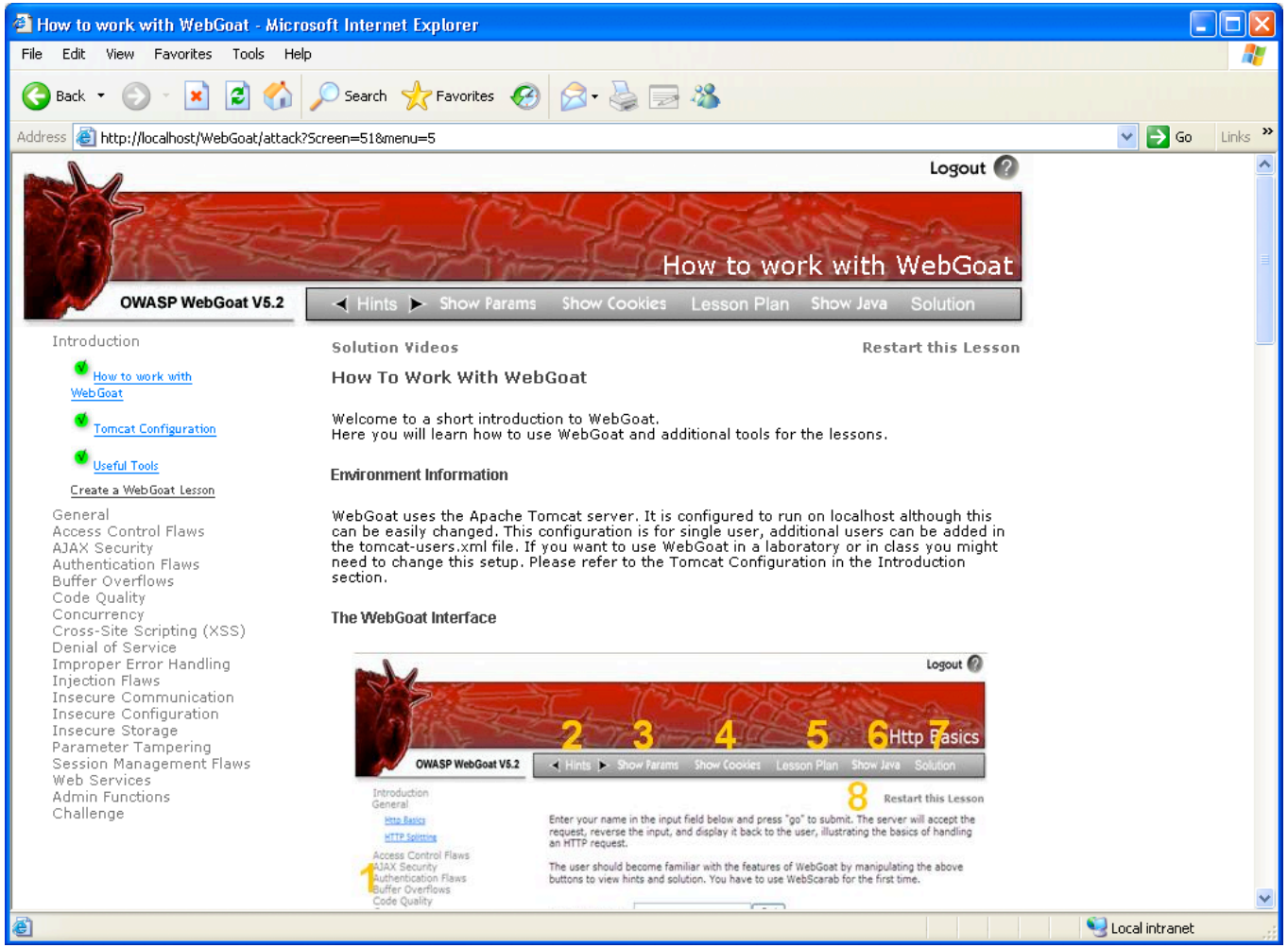
OWASP tools (freely available)

- Your web browser (IE or Firefox preferred)
- WebGoat -- a simple web application containing numerous flaws and exercises to exploit them
  - Runs on (included) Apache Tomcat J2EE server
- WebScarab -- a web application testing proxy

Instructor demo

Class installation of both tools

WebGoat



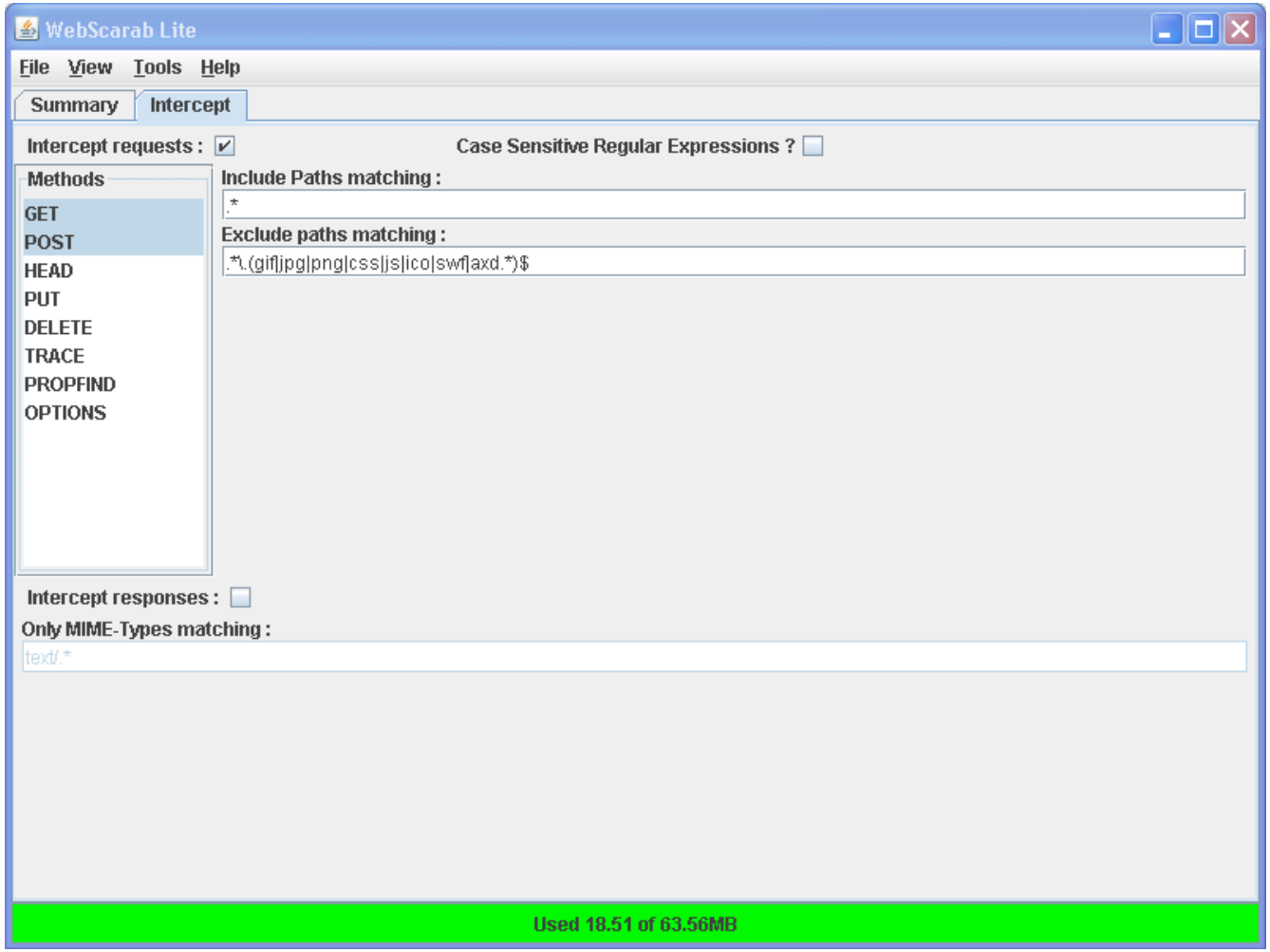
Setting up WebGoat (general)

Run WebGoat on TCP port 80

- From WebGoat folder (GUI or command line)
  - Windows: webgoat_80.bat
  - OS X or Linux: sudo ./webgoat.sh start80
    - (Will need to chmod +x webgoat.sh first)
  - Verify in browser http://localhost/webgoat/attack

At this point, WebGoat is running, but you'll still need a testing proxy to perform some attacks

WebScarab



Next, set up WebScarab

Run WebScarab

- Default listener runs on TCP port 8008
- Ensure listener is running within WebScarab

Configure proxy

- Set web browser proxy point to TCP port 8008 on 127.0.0.1 (localhost)
- Include proxy for localhost
- Connect once again to http://localhost:8080/WebGoat/attack

Troubleshooting

Scarab not running

- Listener turned off or on wrong port

Browser proxy not configured or misconfigured

- IE behaves differently than Firefox
  - IE 7 often "misbehaves"
- Make sure proxy is set for localhost and 127.0.0.1
- Try using 127.0.0.1. (note the "." at end)
- Turn off anti-phishing or "safe browsing" features
- Ensure JavaScript is running
- Try Firefox if you are an IE user, and vice versa

WebGoat tips

Report card shows overall progress

Don’t be afraid to use the “hints” button

- Show cookies and parameters can also help
- Show java also helpful
- None of these are typical on real apps…

Learn how to use it

Fabulous learning tool

Familiarizing Goat and Scarab

WebGoat

- Do “Web Basics” exercise
- Try Hints and other buttons
- Look at report card

Cross site scripting ("XSS")

Can occur whenever a user can enter data into a web app

- Consider all the ways a user can get data to the app

When data is represented in browser, "data" can be dangerous

- Consider this user input

```
<script>
alert(document.cookie)
</script>
```

Where can it happen?

- ANY data input

Two forms of XSS

- Stored XSS
- Reflected XSS

Two WebGoat exercises to see for yourself

Stored XSS

Attacker inputs script data on web app

- Forums, "Contact Us" pages are prime examples
- All data input must be considered

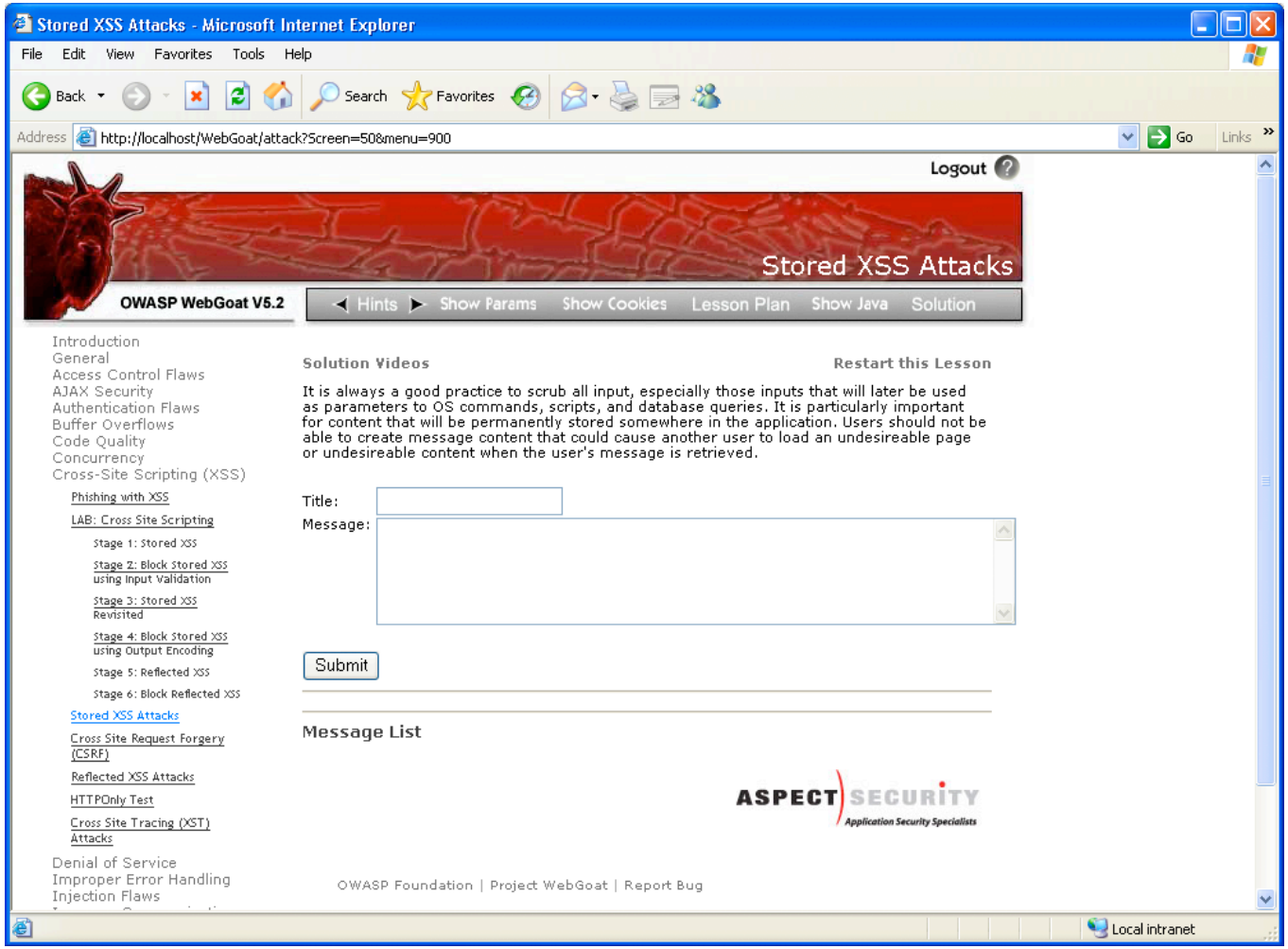
Victim accidentally views data

- Forum message, user profile, database field

Can be years later

- Malicious payload lies patiently in wait
- Victim can be anywhere

Stored XSS exercise



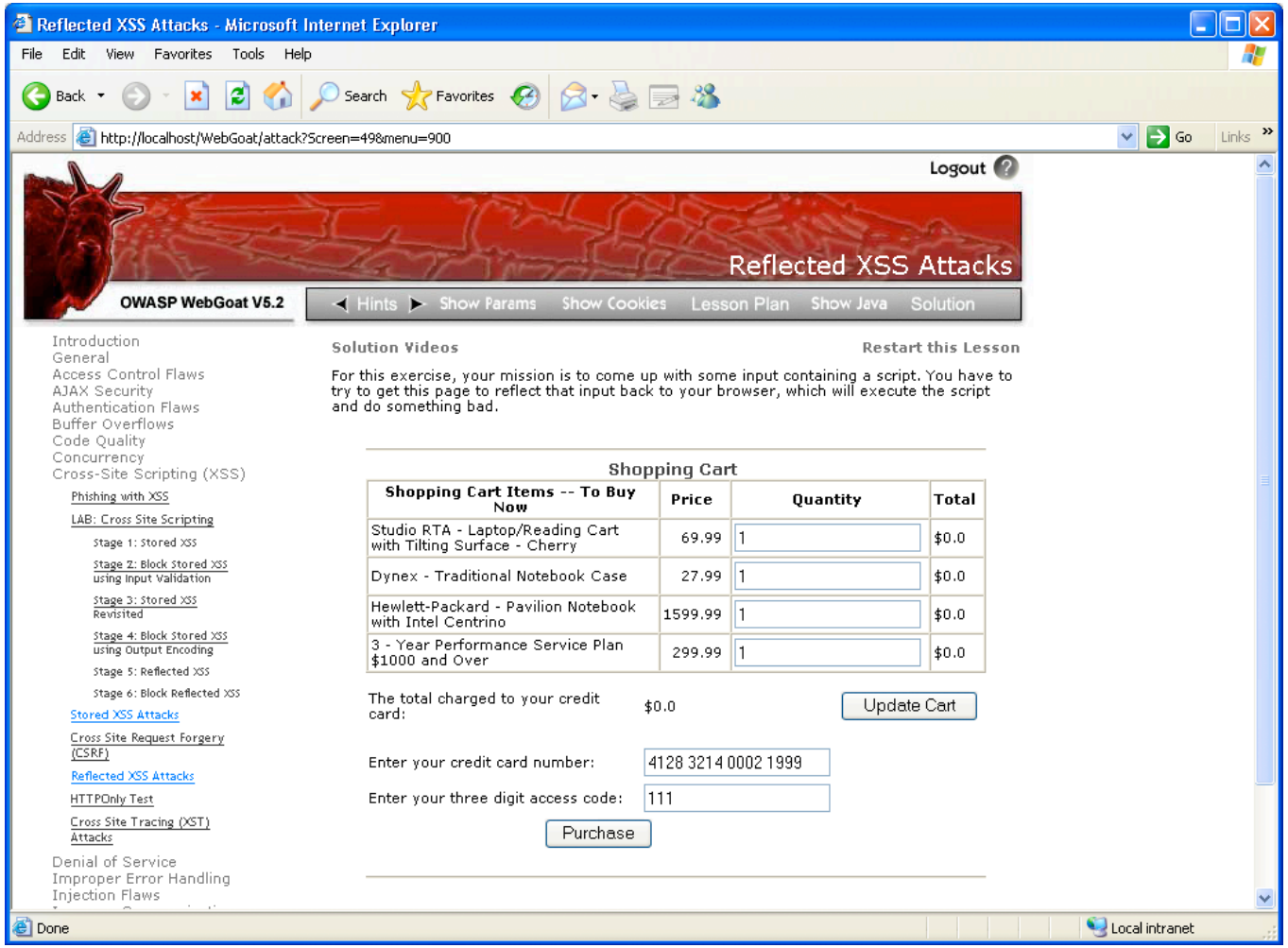
Reflected XSS

Attacker inserts script data into web app

App immediately "reflects" data back

- Search engines prime example
- "String not found"
- Generally combined with other delivery mechanisms
  - HTML formatted spam most likely
  - Image tags containing search string as HTML parameter
    - Consider width=0 height=0 IMG SRC

Reflected XSS exercise



Fuzzers –1

Growing field of app testing that involves sending malformed data to/from app

- Tools, frameworks, and APIs are popping up
- "One size fits all" approach is highly problematic
  - Informed fuzzing vs. uninformed fuzzing
- Still early adoption among pen testers (arguably)
- Dev knowledge is necessary to get most of it

Fuzzers –2

- Fuzzing can and should be done from unit to entire app tests
- QA test team needs to acquire and learn

Examples

- OWASP's JBroFuzz, PEACH, SPI Fuzzer, GPF, Codenomicon, Mu Security, SPIKE, Sulley

"At Microsoft, about 20 to 25 percent of security bugs are found through fuzzing a product before it is shipped"

JBroFuzz

http://www.owasp.org/index.php/Category:OWASP_JBroFuzz

Open source from OWASP

Simplistic, but can fuzz

- Fields in any web app form
- URL guessing

Static code analyzers −1

Review source code for common coding bugs

- A bit of history
  - 1999: First examples appear from research projects
    - E.g., ITS4, RATS, Flawfinder
    - Tokenize input streams and perform rudimentary signature analysis
    - Accurate at finding strcpy() and the like, but lacking context to really be useful

Static code analyzers −2

- 2001: “2nd generation” tools arrive
  - E.g., Fortify, Ounce Labs, Coverity
  - Parse and build abstract syntax tree for analysis
  - Enables execution flow, data flow, etc., traces
  - Significant leap forward, but much work remains
  - Hundreds of common bugs in several languages
  - Management tools for overseeing, measuring, and policy enforcement
  - Integration into popular IDEs
- Still, many are shelfware

Static code analyzers −4

- Then do large scale analysis at project completion
- Possibly using more than one tool set

Examples

- Fortify SCA, Ounce Labs Ounce 5, Coverity Prevent, Klocwork

Fortify SCA

http://fortify.com

Commercial source code analyzer

Supports numerous platforms, languages, compilers, and IDEs

License caveats for this class

Other features: security manager, rule builder

Kenneth R. van Wyk
KRvW Associates, LLC

Ken@KRvW.com
http://www.KRvW.com