Security Vulnerabilities in the open source Moodle eLearning System

Colton Floyd, Tyler Schultz and Steven Fulton
United States Air Force Academy

Undergraduate Research

- Final Project in Senior Level Computer Security Class
- Project was student identified with "suggestions" from instructor mentor
- Deliverables were outlined by course requirement (i.e. background paper, research design methodology, final paper).
- During background research, an older version of Moodle which identified some security issues was identified
- Implementation and vulnerability testing completed by 2 undergraduate students.

Background: Why E-Learning?

- Competitive Edge between institutions
- Corporate Training
- Education 'portals' for traditional classes
  - On-line quizzes
  - Electric File Submissions
  - Web-based lessons
  - Grade keeping
  - Student Feedback.

Background: Learning Suite Development

- As Internet Connectivity became more affordable and prevalent
  - Online education has increased
  - Learning Suites are typically:
    - Self Developed (i.e. WebTycho from UMUC)
    - COTS software (BlackBoard or WebCT)
    - Open Source (MOODLE: Modular Object-Oriented Dynamic Learning System)

Moodle Functionality

- Templates for common formats
  - Powerpoint
  - Flash Presentation
  - HTML
- Testing Modules
  - Multiple Guess, True False, Matching, short answer
- Workshop Module
  - Peer assessment tool
- Miscellaneous components
  - Chat Rooms, Wiki Modules, integration with PayPal.

Known Moodle Security Issues (Version 1.X)

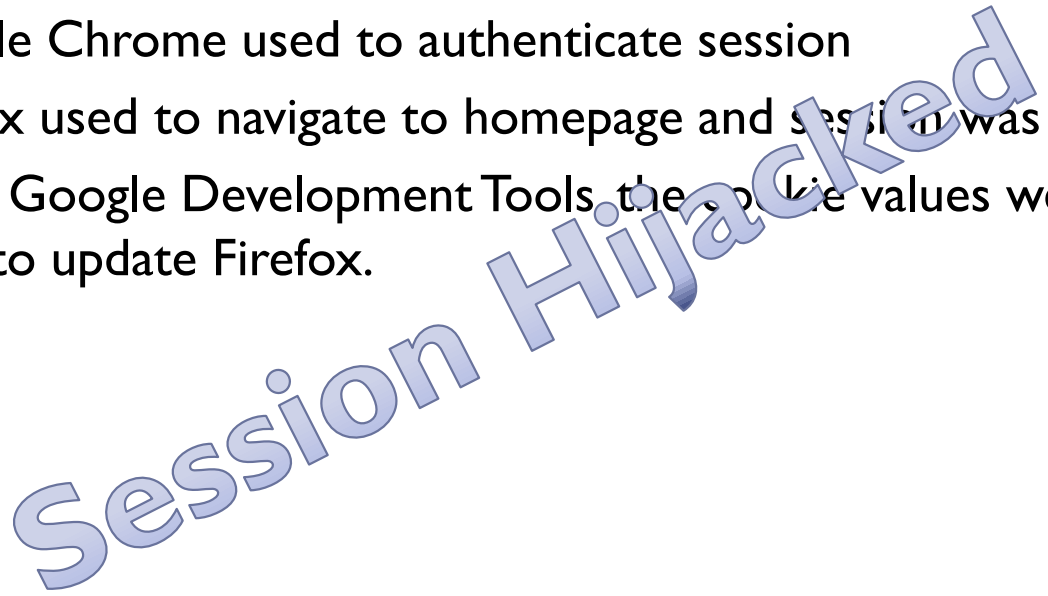
- Open Source means just that … Open Source
- Session Management not inherently secure
- Communications not always completed using SSL
- Re-authentication not required
- Malicious files can be uploaded

What we did:

- Moodle Server (On Virtual Machine)
  - Moodle 2.1
  - My Sql Server
  - Ubuntu
  - Apache

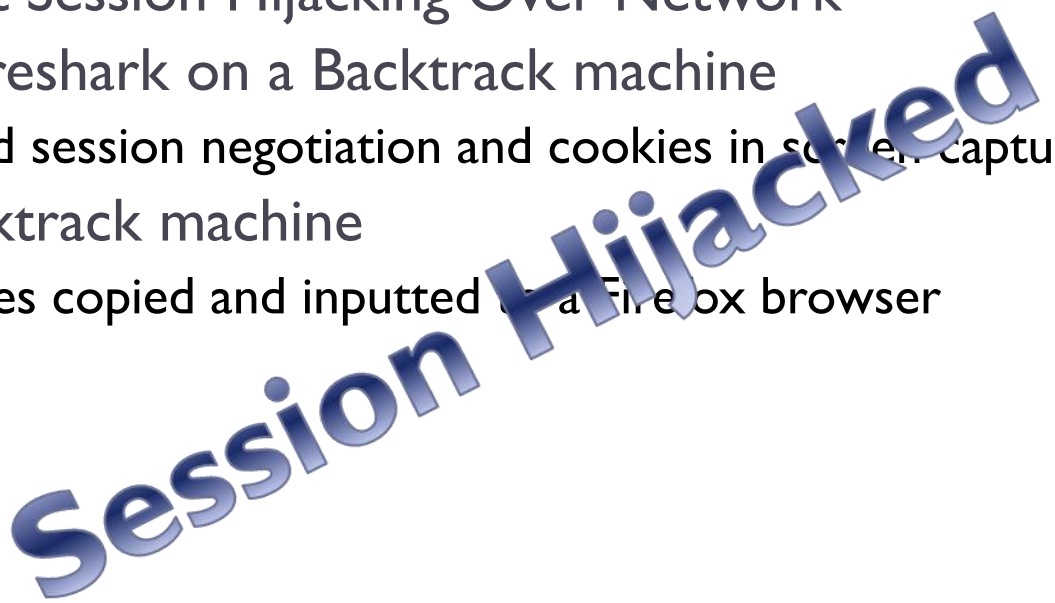
Session Hijacking

- Test Case 1: On Same Machine
  - Google Chrome used to authenticate session
  - Firefox used to navigate to homepage and session was refused.
  - Using Google Development Tools, the cookie values were copied and used to update Firefox.



Session Hijacking

- Test Case 2:
  - Attempt Session Hijacking Over Network
  - Use Wireshark on a Backtrack machine
    - Viewed session negotiation and cookies in screen capture
  - On Backtrack machine
    - Cookies copied and inputted in a Firefox browser



  - And to top it off, moved cookies to another machine with a different IP address, still hijacked!

Test Case Three

- Live Server Test:
  - Authentication done via smart cards
  - Sniffer run on local machine
  - Cookie Visible

Session Hijacked

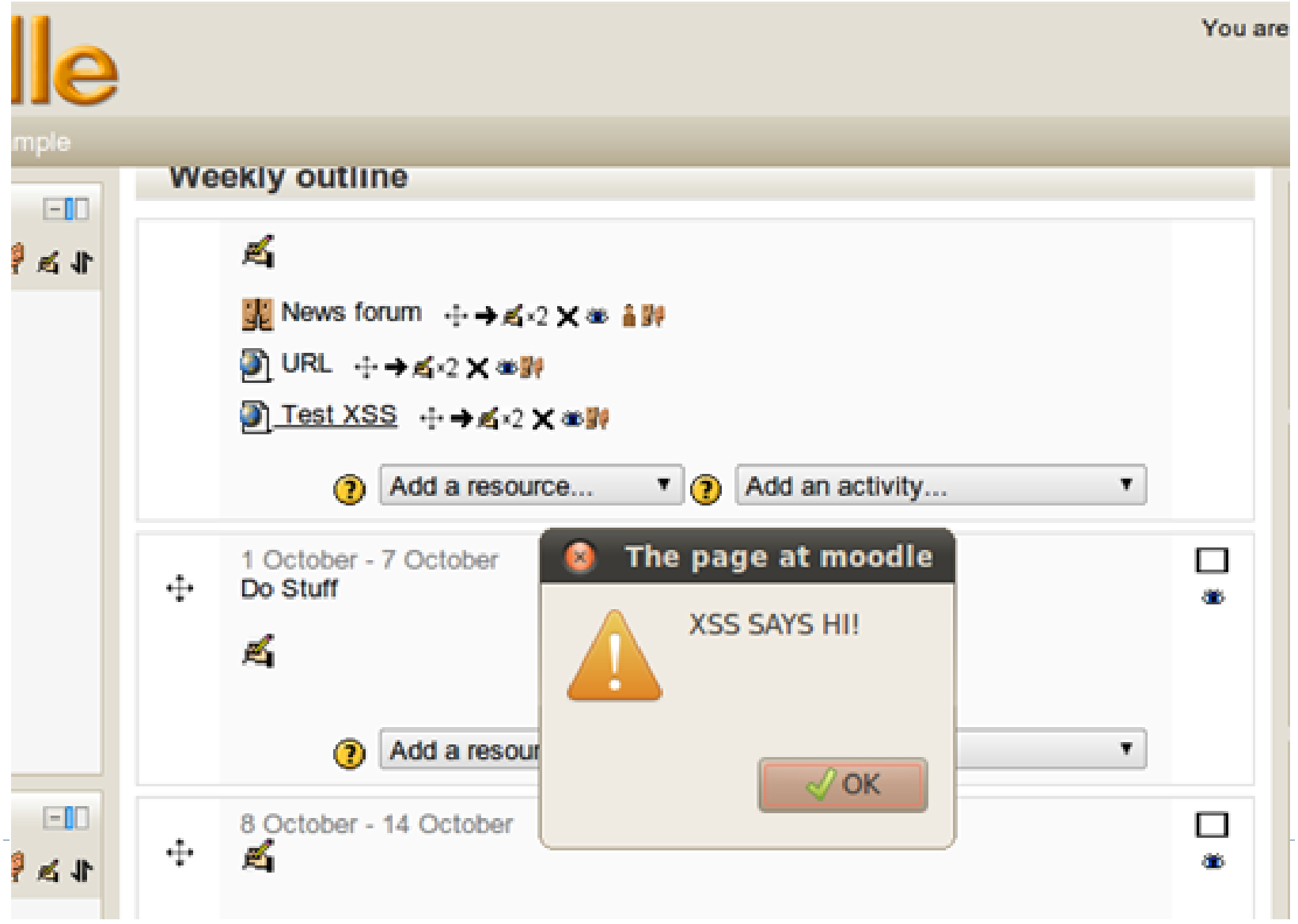
Some other issues that were tested

- SQL Injection
  - Input was well tested on common teacher and student forms: No issues.
- Cookie Best Practices
  - No predictable patterns were seen in cookie generation (good) but cookie sessions were not encrypted and could be accessed by client side scripts (bad)

Some other issues that were tested

- SQL Injection
  - Input was well tested on common teacher and student forms No issues
- Cookie Best Practices
  - No predictable patterns were seen in cookie generation (***good***) but cookie sessions were not encrypted and could be accessed by client side scripts (***bad***)
- XSS Injection
  - Overall: Good (with one minor vulnerability found on administrator accounts which can cause information to be submitted in a resource configuration page and processed as HTML code)

XXS Example Expliotation



Other Vulnerabilities

- In addition to previous session management flaws, *account lock out due to password guesses is tracked at the client-side cookie.*

- Quiz Engine Flaws:
  - Disabling JavaScript on client side disables tracking of time on quizzes.

Recommendations

- Moodle Architecture Changes
  - Additional checks on sessions
  - Options to block simultaneous logins are helpful
  - In non-SSL environments, HttpOnly cookies should be set by default.
  - For SSL environments, HttpOnly and Secure cookies should be set by default.
  - Moodle Quiz engine should ensure that timer works with JavaScript off.

Recommendations

- Moodle Implementation Recommendations
  - Run completely under SSL when possible
  - At a minimum, run logins under SSL to prevent stealing of plain text passwords.
  - Cookies should be marked as HttpOnly and Secure.

Questions?

HTTP-Only

- On a supported browser, an HttpOnly session cookie will be used only when transmitting HTTP (or HTTPS) requests, thus restricting access from other, non-HTTP APIs (such as JavaScript). This restriction mitigates but does not eliminate the threat of session cookie theft via cross-site scripting (XSS).[15] This feature applies only to session-management cookies, and not other browser cookies. <Thank you Wikipedia>

<return>

HTTP Secure

- A secure cookie has the secure attribute enabled and is only used via HTTPS, ensuring that the cookie is always encrypted when transmitting from client to server. This makes the cookie less likely to be exposed to cookie theft via eavesdropping. <Thank you Wikipedia!>

<return>