



**leviathan**  
security group

# Web Session Management

## Wait, who are you again?

Jon McClintock  
Leviathan Security Group

# Agenda

---

- ▶ Introduction
- ▶ Takeaways
- ▶ Background
- ▶ Platform Implementations
- ▶ Common Vulnerabilities
- ▶ Takeaways
- ▶ Questions?



# Introduction

---

My name is Jon McClintock

I am an Application Security Consultant

I work at Leviathan Security Group

I get paid to find security problems in software

Web applications are fun

They're always easy to break

and

they always break

---



# Takeaways

---

- ▶ The stateless nature of HTTP makes session management difficult
- ▶ It is very difficult to build it yourself correctly
- ▶ Current platform implementations still have flaws
- ▶ You need to protect your users against:
  - ▶ Session enumeration
  - ▶ Session fixation
  - ▶ Session tampering
- ▶ Make sure that your session management works
  - ▶ Session expiration
  - ▶ Logout



# Agenda

---

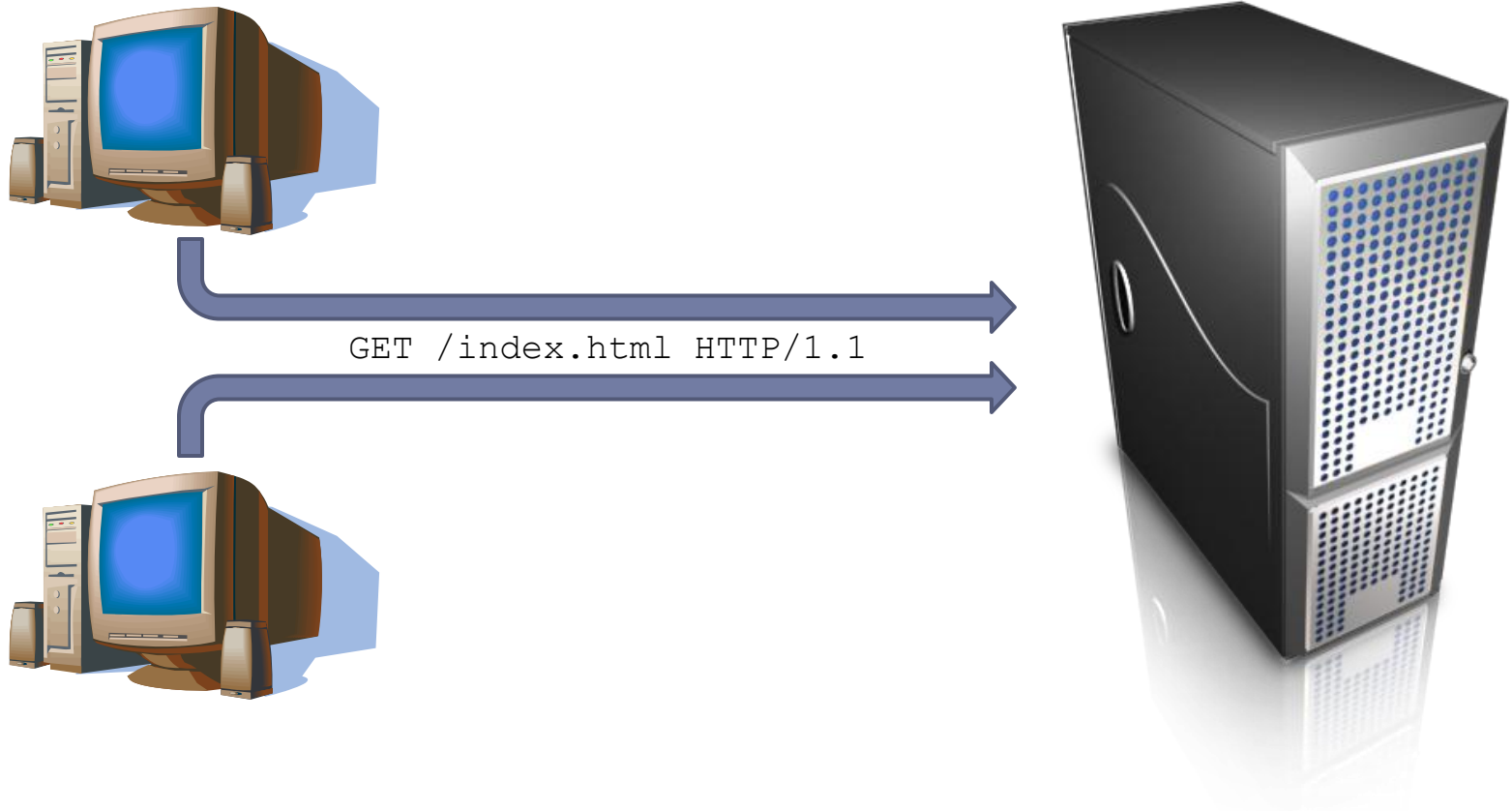
- ▶ Introduction
- ▶ Takeaways
- ▶ Background
- ▶ Platform Implementations
- ▶ Common Vulnerabilities
- ▶ Takeaways
- ▶ Questions?



# HTTP Primer

---

- ▶ HTTP is stateless – there's no built-in mechanism to tie any one request to another



# State is Good

---

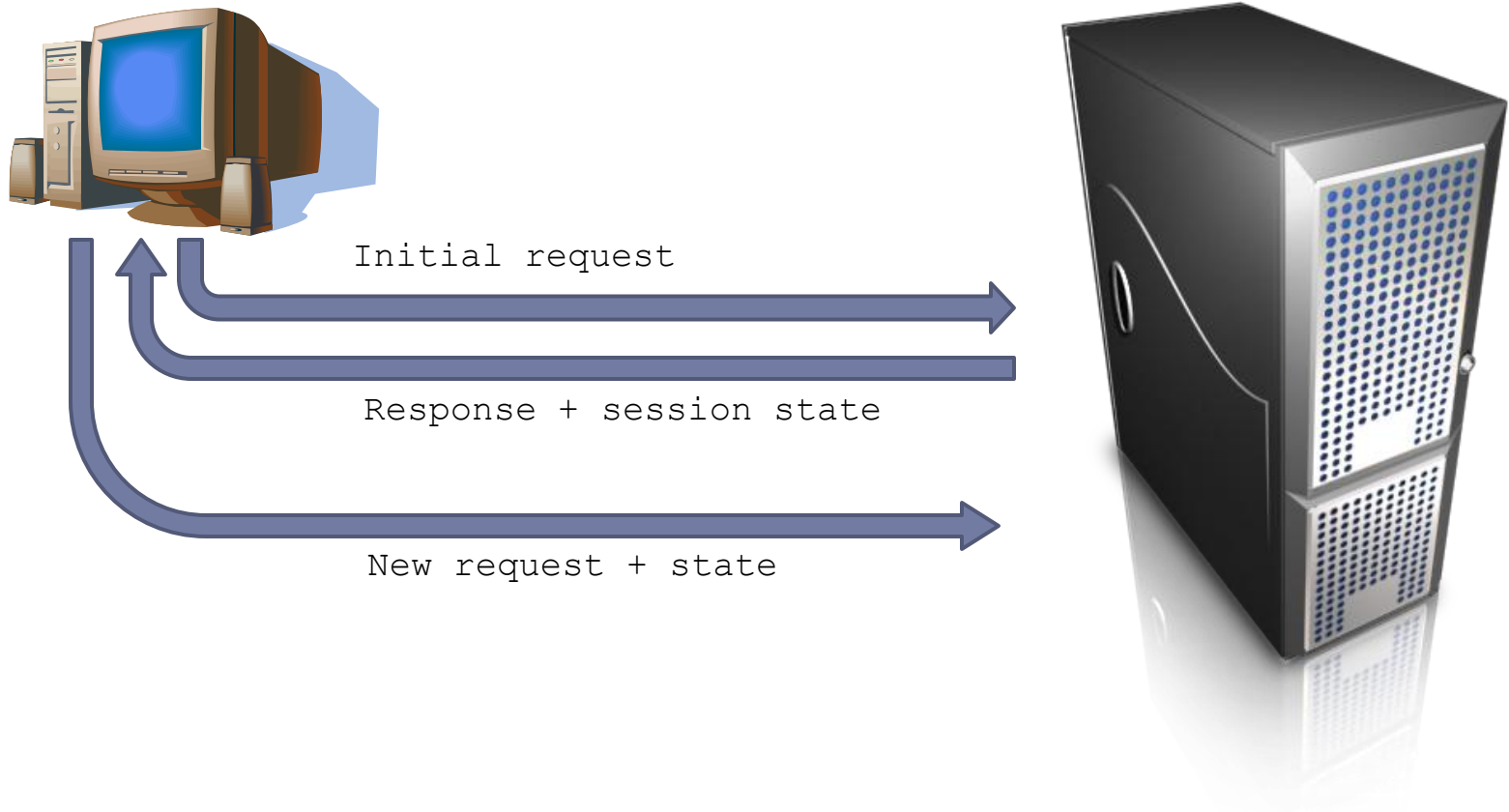
- ▶ **Stateful web applications are usable web applications**
  - ▶ User tracking
  - ▶ Personalization
  - ▶ Shopping carts
  - ▶ Authentication
  - ▶ Authorization



# Implementing State

---

- ▶ Send the client information
- ▶ The client sends it back with later requests



# Moving State

---

- ▶ Three options for moving state between the client and the server:
  - ▶ Query parameters in the URL
    - ▶ `http://example.com/index.html?state=<state>`
  - ▶ Hidden form fields
    - ▶ But then everything must be a form
  - ▶ Cookies
    - ▶ But not everyone loves cookies
- ▶ Of course, the client has full control over anything you give them
- ▶ So give them as little as possible
  - ▶ Often, just an ID into your session database



# Agenda

---

- ▶ Introduction
- ▶ Takeaways
- ▶ Background
- ▶ **Platform Implementations**
- ▶ **Common Vulnerabilities**
- ▶ **Takeaways**
- ▶ **Questions?**



# How It's Done

---

- ▶ Let's take a look at how some common web frameworks implement session management
  - ▶ Java Servlets
  - ▶ Ruby on Rails
  - ▶ PHP



# Java Servlets

---

- ▶ JSESSIONID identifies the client session
- ▶ Session ID is tracked in:
  - ▶ JSESSIONID cookie – session lifetime
  - ▶ URL – `http://example.com/index.jsp;jsessionid=<...>`
  - ▶ SSL state mechanism
- ▶ Sessions timeout on the server after 30 minutes of inactivity

See section 7 of the Java Servlet specification for details:

<http://jcp.org/aboutjava/communityprocess/final/jsr154/index.html>

---



# Ruby on Rails

---

- ▶ Session ID is generated as an MD5 of the current time, a pseudorandom string, and the server PID
- ▶ Tracked in the `_session_id` session cookie
- ▶ Sessions are persisted on filesystem by default
- ▶ No session expiration mechanism until Rails 2

Details at: <http://wiki.rubyonrails.org/rails/pages/UnderstandingHowStateIsMaintained>

---



# PHP

---

- ▶ PHPSESSID identifies the user's session
- ▶ Session ID is tracked in:
  - ▶ Cookie – PHPSESSID, session lifetime
  - ▶ URL – `http://example.com/index.php?PHPSESSID=<...>`
  - ▶ Hidden form parameters
- ▶ Session expiration is determined by `session.gc_maxlifetime` configuration setting, with a default of 1440 seconds (24 minutes)

Details at: <http://devzone.zend.com/node/view/id/1312>

---



# Agenda

---

- ▶ Introduction
- ▶ Takeaways
- ▶ Background
- ▶ Platform Implementations
- ▶ Common Vulnerabilities
- ▶ Takeaways
- ▶ Questions?



# Common Vulnerabilities

---

- ▶ **Failure to protect the session data:**
  - ▶ Session Manipulation
  - ▶ Session Enumeration
  - ▶ Session Fixation
- ▶ **Failure to clean up when the user is done:**
  - ▶ Session Expiration
  - ▶ Broken Logout
- ▶ **Failure to tie the session to requests**
  - ▶ Cross Site Request Forgery



# Session Manipulation

---

- ▶ Very simple, straightforward attack
- ▶ Anything you give to the client, they can modify
- ▶ Protect the data with cryptography
  - ▶ HMAC
  - ▶ Encryption
- ▶ Or, send as little as possible to the client – just a session ID



# Session Enumeration

---

- ▶ Made possible when session IDs are easy to guess
  - ▶ Sequential
  - ▶ Easily derived
- ▶ If you can figure out a valid session ID, you can find and impersonate other users
- ▶ Make session IDs difficult to guess
  - ▶ Don't just hash your database sequence numbers
  - ▶ Use an HMAC, or a UUID library



# Session Fixation

---

- ▶ Instead of guessing a user's session ID, make them use one of your choosing
  - ▶ Very common problem, especially with URL based session tracking
  - ▶ Send the user a URL on the target site, and wait for them to log in
- ▶ Always issue a new session ID when the user changes authentication state
- ▶ Never accept a session ID not created by your application



# Session Lifetime

---

- ▶ Not a vulnerability per se, but compounds other session flaws
- ▶ If your sessions are valid forever, you're making the window of attack larger
  - ▶ More valid sessions to guess or enumerate
  - ▶ Longer window to exploit session fixation attacks
- ▶ Deauthenticate sessions after a short period of inactivity
- ▶ Force reauthentication for sensitive operations
- ▶ Destroy sessions after a long period of inactivity
- ▶ Reissue cookies to ensure that lifetime is respected



# Broken Logout

---

- ▶ Especially problematic on public terminals
- ▶ Deleting the session cookie is not sufficient
- ▶ Neither is asking the user to close the browser window
- ▶ Always invalidate the session on logout



# Cross Site Request Forgery

---

- ▶ **Not really a session management flaw**
  - ▶ Even if your session management is implemented perfectly, you're likely vulnerable
- ▶ **Validate the request as well as the session**
  - ▶ Requiring that requests include a valid, matching session IDs is sufficient



# Takeaways

---

- ▶ The stateless nature of HTTP makes session management difficult
- ▶ It is very difficult to build it yourself correctly
- ▶ Current platform implementations still have flaws
- ▶ You need to protect your users against:
  - ▶ Session enumeration
  - ▶ Session fixation
  - ▶ Session tampering
- ▶ Make sure that your session management works
  - ▶ Session expiration
  - ▶ Logout



# Questions?

---

Jon McClintock  
[jon@leviathansecurity.com](mailto:jon@leviathansecurity.com)



**leviathan**  
security group

