

http://www.fishnetsecurity.com



Black Hat Europe 2005

Building Zero-Day Self-Defending Web Applications:

Enforcing Authoritative Action to Defend against State, Session and Authorization Attacks



Black Hat Europe 2005

Self-Defending Web Applications Simple Version:

Session and Authorization Strength



Who

Arian J. Evans

Researcher, Teacher, Coder, Breaker Senior Security Engineer FishNet Security, Application Security Services

FishNet Security

Security Consulting Company with Focus on AppSec They pay me to test and fix things (blah, blah)

(In Europe you may have heard of the company <u>Secure Passage</u> that makes Firemon, a spin-off of FishNet's Application Group)



What this subject is about:

- Lack of understanding of session weaknesses and techniques to defend against session and authorization attacks.
- Limited documentation on the real threats to weak sessions and authorization attacks.
- Limited documentation on proper use of dynamic tokens in web applications.
- Limited documentation on use of newer session-oriented technologies.



Limited Background

- Started with: "Cross Site Request Forgeries" http://www.securityfocus.com/archive/1/191390
- "Session Fixation" by Mitja Kolsek http://www.acrossecurity.com/papers/session fixation.pdf
- "Web App Session Strength" by Michael Schema at BH Vegas 04 http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-shema-up.pdf and Schema's February 05 RSA presentation on same explores session token weaknesses, both excellent, but fails to address newer session and authorization automation attacks.
- "Session Riding" by Thomas Schreiber http://www.securenet.de/papers/Session Riding.pdf provides first public documented insight into automated session and authorization attacks; contradicts "Session Fixation" suggested practices/defenses.
- Fake vulnerability disclosure on session attacks by Anachronic (myself) and follow-up discussion demonstrated surprising lack of understanding or interest in these issues in the webappsec and webapp development communities.
- "Host Naming & URL Conventions" by Gunter Ollman http://www.ngssoftware.com kindof sortof partially addresses defenses to URL-based session/authorization attacks.



Why do you care about Session and Authorization Security?

Everyone is doing the same thing; how bad can it be?



Why do you care?

::High Risk::

Session weaknesses are real, easy to target

::You are Targeted::

Incident Response on the Rise; Real Money is finally being made from web hacking

::You are Accountable::

Regulations, Management finger-pointing, (anecdote of the once-unaccountable now outsourced dev team)



The Application must defend itself

First and most important application security principle:

The Application Must Defend Itself



Presentation Overview

- I. Introduction (done)
- II. Chapter One: In the Beginning was the Wibbly Wobbly Web
- III. Chapter Two: Stuck to the Web of the Now
- IV. Chapter Three: Subverting Session & State, and Automating Authorization Attacks
- V. Chapter Four: Self-Defending Dynamic Session Security: DATs, DSTs, and DFFs
- VI. Interlude: Interactive Session with the Devil
- VII. Chapter Five: Spinning Security in the Web's Future
- VIII. Chapter Six: Q&A (Shoot the Messenger)



Chapter I

In the Beginning was the Wibbly Wobbly Web



Lightweight and Stateless by Design

- I. HTTP and "Session Management" designed to be:
 - Lightweight and Stateless like my ex-girlfriend
 - Fast: 9600 baud was the pride of the neighborhood
 - Minimize Overhead vs. Secure: RFC 2109 and 2956

II. Race to First Place:

- Competition to be *First*
- Little real loss; ask Randy Moss: \$10k USD "ain't ****"

III. Perception Problems:

- Not me targeted (who would target me? why?)
- Not me accountable
- Whatever, sandbagger, we're making \$\$\$



Chapter II

Stuck to the Web of the Now



Security Vendor Marketing Hype

I. The Problem of Particulars

- XSS, XST, XDS, XDF, XBS: why particulars can hurt more than they help (reference: Issues slide)
- Selling managers/auditors bullet-point friendly products
- Classification of Threat/Risk/Issue: Lack of distinction between Category/Class/Particular—Does anyone take formal logic classes anymore?

II. Immature Metrics

- What is *really* being exploited? (This is not what the OWASP Top-10 or WASC taxonomy addresses.)
- *What* exploits result in real loss? (Not what the threads on webappsec@sf are talking about.)
- No reward in securing without demonstrable risk.



Issues: How many Issues here?

- **⇒**Parameter Tampering
- Cookie Tampering
- **Ocross-site Scripting**
- **SQL** Injection
- **Script Injection**
- **Command Injection**
- **Description** Encoding Attacks
- **⇒**Buffer Overflows
- Format-string attacks

- **⊃**Harvesting User IDs
- **⇒**Brute-forcing Accounts
- **Path Truncation Attacks**
- Hidden Path Discovery
- ⇒Application Directory and File Mapping
- **⇒**Forcible Browsing
- Source Code Disclosure
- Web server vulnerability exploitation



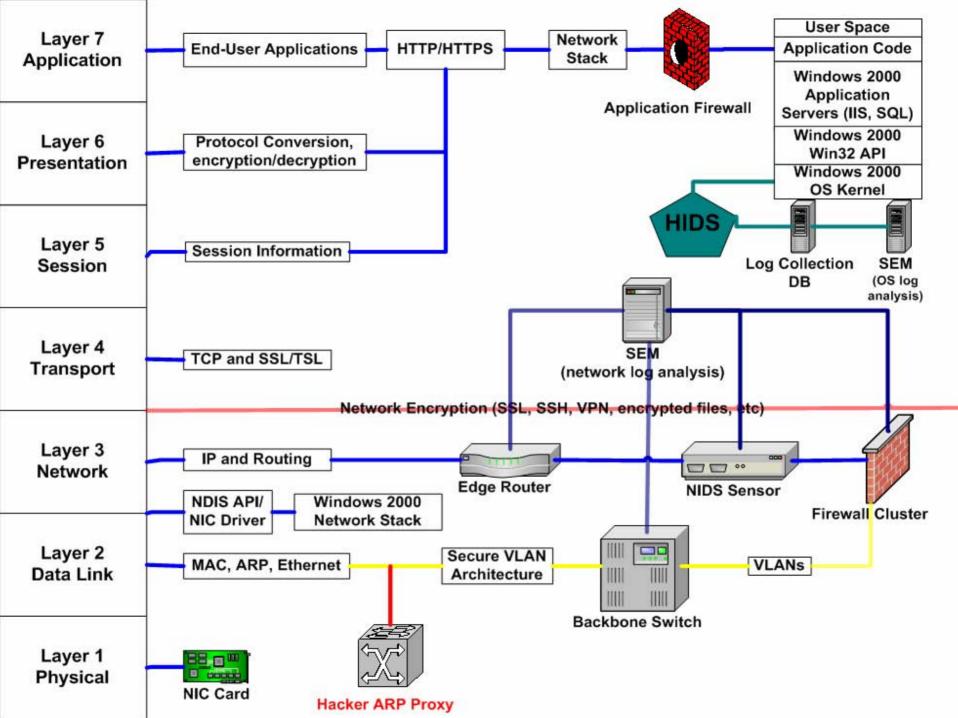
Beckoning to Babylon

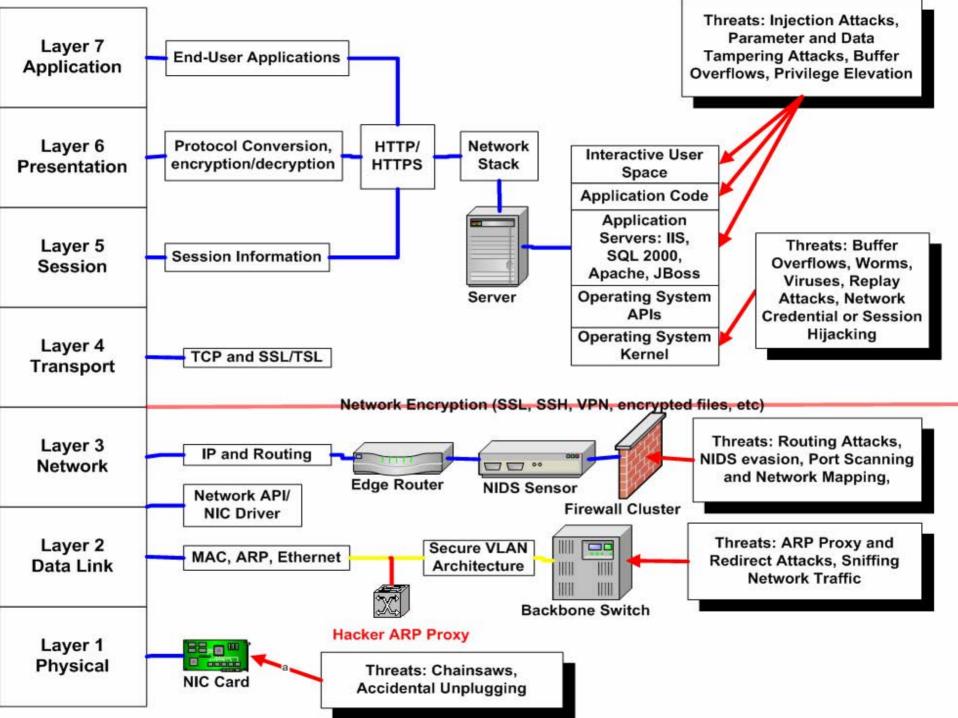
III. Cryptographic Confusion

- encrypted "Session Tunnels" (SSL).
- encrypted cookies (so what if your cookie is encrypted?).
- encrypted "storage" (swell; you have DB password hashes).
- client-side certificates (now your clients can securely authenticate before performing authorization attacks).
- Where are you encrypting anyway? (refer to diagram)

IV. Client Constraints & Injecting Enemas (IE)

- Stateless Clients
- Script-Happy Client Spews All
- What do we do about clients that are fundamentally broken?







Chapter III

Subverting Session & State, and Automating Authorization Attacks



OWASP Top-10 Vulnerabilities

- 1. Unvalidated Input
- 2. Broken Access Controls
- 3. Broken Account and Session Management
- 4. Cross-site Scripting (XSS) Flaws
- 5. Buffer Overflows
- 6. Injection Flaws
- 7. Improper Error Handling
- 8. Insecure Storage (Improper use of Cryptographic Controls)
- 9. Denial of Service
- 10. Insecure Configuration Management



Arian's Most Common Vulnerabilities

- 1. XSS (Cross-site Scripting)
- 2. Weak or Broken Session Handling
- 3. User ID and/or Password Harvesting by Brute Force
- 4. Parameter Tampering
- 5. Cookie Tampering (often resulting in #2)

Several of the above fall into the "weak use of cryptography" category, including cookies and other session tokens...



I. State Subversion

- SRC Spoofing
- MitM (Monkey in the Middle)
- Hijacking
- Redirection (via DNS, Routing, BGP, etc.)
- Lack of State in Business Logic components
- Concurrency and ASMQ issues
- Issues with using native "session management" components to address load-balancing issues (see session example)



II. Session Subversion Part I

- Session Fixation (set & forget with persistence).
- Session Castling (break & remake, proxy swap, HTTP Response Splitting).
- Session Recording, Cloning, and Replay.
- Session "Riding".
- Session "Double Riding".
- XSS (Cross Site Scripting) Session Attacks.
- XST (Cross Site Tracing) Session Attacks, and why XST will never happen (if the browser is that broken you might as well be phishing or worse).



II. Session Subversion Part II—Advanced XSS:

- XSS URL Parameter Exploitation.
- XSS Header Field Exploitation.
- XSS Body-based Parameter Exploitation.
- XSS and other Script Injection Breakdown:



- II. Session Subversion Part III—Taxonomy of Script Injection Attack Vectors:
 - Client Direct <--! (wireless, L2)
 - Client Reflected, external Application (email)
 - Client Direct or Reflected, Persistent
 (XSS Proxy, Persistent Form Fields, Injected Persistent Frames, etc.)



- II. Session Subversion Part IV—Taxonomy of Script Injection Attack Vectors:
 - App-to-Client Bounce (Spoofed Request?)
 - App-to-Client Embedded (online forum, DB fields)
 - App-to-Secondary App, Embedded (log readers, CRM, "2nd Order Code Injection Attacks")
 - Which of these can trivially used to launch authorization attacks against valid sessions?



III. Authentication: Entity Authentication

- Authentication in Web Applications
- Dual-Factor Authentication (Bank of America)
- Client-Side Certificates
- Are you authenticating the client or the browser?
- Why is this a problem?
- Authenticating the user...



IV. Ignored Insecurities: Authorization Attacks

- Auto-Authentication, or: Asleep at the Wheel
- Will the real "Session Riding" please stand up?
- Advanced XSS for Authorization Attacks: Hugo Fortier's research & Anton Rager's XSS proxy:

http://sourceforge.net/projects/xss-proxy



Session Riding & Advanced XSS

EXAMPLES



V. Advanced Authorization Attacks

- Direct Authorization Attacks: wireless or L2 injected localized HTTP requests (through VPN?)
- Reflected Authorization Attacks: remotely crafted localized HTTP requests
- Embedded Authorization Attacks:
 - to presumptive strings executing URL-based parameter actions
 - brute-forcing URL-based session tokens
 - + 2nd to n-order code injection (Nokia exploits)



Advanced Authorization Attacks

EXAMPLES



Chapter IV

Self-Defending Dynamic Session Security: DATs, DSTs, and DFFs



The Application must defend itself

First and most important application security principle:

The Application Must Defend Itself



Strong State Strategies

I. Strong State Management

- Multi-tier State + Authentication
- State Protection through Secure Sessions
- New session-oriented technologies
- Flash 7.0 (it's the future of Window's interface as well)
- Yes, I said Flash, and with a straight face.
 - Flash grows up: http://www.ena.lu/mce.cfm



Strong Session Strategies

II. Strong Session Management

- Strong Session Affinity Tokens
- Dynamic Session Tokens (DSTs)
- Keep them out of your web logs (URL-based tokens stored in web logs by default)
- Consider the Cookie Carefully
- Dynamic Session Rotation via DSTs: compare to IPSEC or SSL VPNs



Enforcing Authoritative Action

III. Strong Authorization Enforcement Part I

- Dynamic Authorization Tokens (DATs)
- Dynamic Form Fields (DFFs)
- Think One Time PINs (OTPs)
- NOT the session cookie or default DOM-abused objects
- Ditch your URL-based parameters
- NOT in the URL or your web logs



Enforcing Authoritative Action

III. Strong Authorization Enforcement Part II

- In the URL if you insist on URL-params, else
- Dynamic Form Fields for Session Token Handling
- Dynamic Authorization Tokens as a form-value
- Dynamic Form Fields stop some types of harvesting and workflow-bypass attacks
- DATs break trivially-exploitable URL-based XSS
- DATs mitigate XST action or workflow attacks



Enforcing Authoritative Action

IV. Design by Number

- 1. Use DATs and DFFs to protect sensitive functions
- 2. Enforce DATs and DFFs at and post authentication
- 3. Define application function Entry-Points
- 4. Define application function Exit-Points
- 5. Enforce function workflow from Entry-Point by:
 - a. Validating DSTs at each workflow step
 - b. Requiring unique DATs for each workflow step
- 6. Ensure tokens are destroyed at function Exit-Point



Enforcing Authoritative Action

EXAMPLES

Defeating Advanced XSS and Authorization Attacks with DSTs and DATs



Interlude with the Devil

Authorization weaknesses that we can't do much about today



Interlude with the Devil

- My Mother: Yes! (she clicks on anything)
- My Father: No! (he won't even install the patch)
- My 0wn3d Windowz Boxen: Predictable File Names, Locations & Sun's JVM
 - would this have happened if my OS had dynamic workflow-based authorization tokens? (random cache directories used for downloads)
- The Losing Battle? (IE)
 - Belittling the Broken Browser
- Enforcing Stateful Workflow (workflow handout)
- Catholics vote for "Dynamic Authorization Tokens"



Chapter V

Spinning Security in the Web's Future



Spinning Security in the Web's Future

- I. Towards a new OWASP Top-10 Taxonomy
 - Distinction between Category, Class, and Particular
 - Distinction between Risk, Threat, and Vulnerability
 - Metrics distinguishing exploitability

II. Context Criticality

- Threat Modeling
- Strategic Solutions (Focus on reusable frameworks, not new widgets)

III. Retiring RFC's

- Session Management RFCs from 1997?
- Isn't it about time?
- Stop perpetuating this mess.



Spinning Security in the Web's Future

IV. Future Shock

- Flash and Session-based technologies
- Wait until you see what we are doing with Flash...

V. DSTs, DATs, DFFs

- Use Dynamic Session Tokens
- Use Dynamic Authorization Tokens for sensitive functions
- Use Dynamic Form Fields to limit workflow bypasses
- Define Entry Points
- Define Exit Points
- Destroy all tokens at Exit and Absolute Time



Chapter VI

Q&A: Shoot the Messenger



Additional Notes



Highest Risk Web App Vulnerabilities

- 1. What does your application do?
- 2. What sensitivity of information does your application handle?
- 3. What type of functionality does your application provide?
- 4. What are the business requirements your customers demand of your application for it to exist?
- 5. Can you quantify the impact of a compromise?

There is no such thing as an arbitrary risk or threat in your application, regardless of what a tool or vendor tells you.



References

- OWASP http://www.owasp.org
- SecureCoding http://www.securecoding.org
- STRIDE/DREAD Threat Models <u>http://www.microsoft.com</u>
- D. Schwabe and G. Rossi, "An Object-Oriented Approach to Web-Based Application Design," *Theory* and Practice of Object Systems (TAPOS), special issue on the Internet, vol. 4, no. 4, 1998, pp. 207-225.
- "Measuring Relative Attack Surfaces" http://www-2.cs.cmu.edu/~wing/



References

- Started with "Cross Site Request Forgeries" http://www.securityfocus.com/archive/1/191390
- "Session Fixation" by Mitja Kolsek http://www.acrossecurity.com/papers/session fixation.pdf
- "Web App Session Strength" by Michael Schema at BH
 Vegas 04 http://www.blackhat.com/presentations/bh-usa-04/bh-usa-04/bh-usa-04-shema-up.pdf (and Schema's February 05 RSA presentation on same explores)
- "Session Riding" by Thomas Schreiber http://www.securenet.de/papers/Session Riding.pdf
- "Host Naming & URL Conventions" by Gunter Ollman http://www.ngssoftware.com
- Unpublished work by myself and Hugo Fortier on session and authorization attacks via XSS, automation, and session fixation to be released this year (2005) on the OWASP portal.



Arian J. Evans
Senior Security Engineer
FishNet Security
816.701.2045
arian.evans@fishnetsecurity.com

OWASP Chapter Leader, Kansas City