



Black Hat Europe 2006

Paraegis Project Round 2:

Using Razorwire HTTP proxy to
strengthen webapp session
handling and reduce attack surface



Project Paraegis is:

Currently Hosted at:

www.anachronic.com/paraegis

and also may be found at

www.ionize.net/paraegis

and

www.paraegis.com

We are unsure yet were final source releases will be hosted, but links to Code Project, Sourceforge, and various code locations will be tracked above.



Project Paraegis is:

- **A software experimentation project focused on web application security.**
- **Working closely with OWASP.**
- **Trying to create the Snort of webappsec.**
- **Providing open source and free solutions.**
- **Based on belief that simple XSS attack surface issues don't need multi-\$100 thousand dollar (euro) appliances to fix.**
- **Privately sponsored by the members.**



Who

Arian J. Evans

Researcher, Teacher, Breaker, Coding Faker

FishNet Security <http://www.fishnetsecurity.com>

Dan Thompson

Coder, Researcher, Web stuffs

Secure Passage <http://www.securepassage.com>

Mark Belles

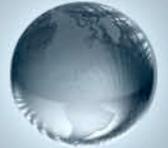
Coder, Researcher, Framework

Secure Passage <http://www.securepassage.com>



What this presentation is about:

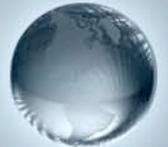
- The increasing complexity of script-injection attacks.
- Controlling browser sessions through framing and session riding to do Bad Things.
- Scanners cannot show you this
- WAFs (and good coding) can protect initial attack vectors to launch these attacks.
- Show similar example as Jeremiah's Phishing with Super-Bait (refinement inspired by that work)
- Demonstrate our evolving freeware proxy and how it can help you stop these issues.



The Application must defend itself

First and most important application security principle:

- **The Application Must Defend Itself**



The Application must defend itself

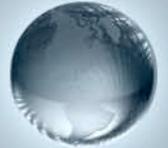
Second principle we are being forced to consider frequently:

- **Defense in depth because new coders always make old mistakes**



Presentation Overview

- I. Introduction (done)
- II. Reminders concerning the Wibbly Wobbly Web
- III. Stuck to the Web of the Now:
 - I. Still a HUGE *Problem of Particulars*
 - II. Taxonomy of Threat, Attack, Weakness, Vulnerability
- IV. Script Injection, XSS, CSRF, Web Trojans, 'Session Riding', and Super-Bait review
- V. Paraegis defensive concepts
- VI. Razorwire Proxy Demo, Super-bait frame injection type attack demo, simple XSS variants
- VII. Q&A (Please feed the Messengers)



Chapter Two

Reminders in case
you forgot about the
Wibbly Wobbly Web



Lightweight and Stateless by Design

- I. HTTP and “Session Management” designed to be:
 - Lightweight and Stateless
 - Focus was on minimizing overhead, not security: reference RFC 2109 and 2956 on HTTP state management mechanism

- II. Perception Problems (until recently):
 - Not me targeted (who would target me? why?)
 - Buy more firewalls
 - Not me accountable
 - You can’t do anything with an XSS

(we have prescribed medication and shut down a call management system with XSS, amongst other funs)



Chapter Three

Stuck to the Web of the Now: Of Particulars and Taxonomies



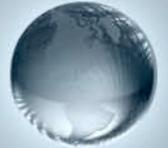
Webappsec Vendor Marketing Hype

I. The Problem of Particulars

- XSS, XST, XDS, XDF, CDC (!), XBS: why particulars can hurt more than they help
- Misses the many-to-one mapping between threats, attacks, and weaknesses
- We all know the game of selling managers/auditors bullet-point friendly products

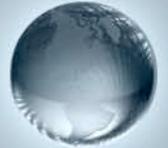
II. Classification Systems

- What is *really* being exploited?
- *What* exploits result in real loss? (Not what the threads on webappsec@sf are talking about.)
- Confusing terminology for developers, management, security folks, etc. blurs threat, attack, weakness



Chapter Four

Script Injection, XSS, CSRF, Web
Trojans, 'Session Riding', and
'Super-Bait'



Basic Script-Injection/XSS Categories

I. Short Sweet Categorization/Taxonomy:

1. URL-based attacks
2. Non-URL based attacks
 - a) Header value
 - b) body data
 - c) local DOM

scanners and FAQs seem to find these equal,
but they are clearly significantly unequal in attack
surface for reflected attacks



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 2004
<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 fail to address newer session and authorization automation attacks.
- **Innocent Code by Sverre Huseby** introduces automated session attacks under the name “Web Trojans” with a nice discussion of using nounces.
- **“Session Riding” by Thomas Schreiber**
http://www.securenet.de/papers/Session_Riding.pdf provides second public documented insight into automated session and authorization attacks; contradicts “Session Fixation” suggested practices/defenses.
- **Attack Tools** listed in our whitepaper on this subject linked below, include very nice efforts like Anton Rager’s XSS Proxy: <http://sourceforge.net/projects/xss-proxy> and our C# XSS controller based off a combination of our work and Jeremiah Grossman.
- **A whole list of info** on session attacks in a paper you can download off of <http://www.anachronic.com/paraegis> since the list got too long for this slide. There are a number of people who have discussed this subject and come to very different conclusions about the solution, and all are worth exploring.



Attack Demo Time:

**Obviously a demo
might be in
order here**



Chapter Five

Paraegis defensive concepts

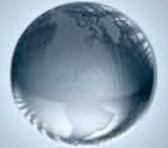


The Application must defend itself

First and most important application security principle:

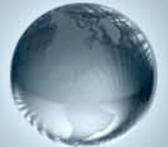
- **The Application Must Defend Itself**

This implies that throwing widgets on your network won't help you solve the real problem. Especially if you are an ISV, unless you are trying to sell protection widgets too...



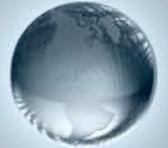
Essential Web App Self-Defense

- I. Input Validation
- II. Output Encoding
- III. Strong Session-Handling Techniques
- IV. Authorization Enforcement
- V. Human Detection
- VI. Anti-Automation



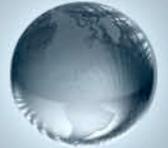
Chapter Six

Okay, no more talked of
DATs, nounces, and
other boring stuffs.



Chapter Six

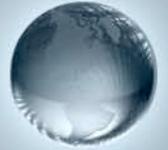
Razorwire Proxy Demo:
Super-bait frame injection type attack
Simple XSS variants
Razorwire token and rewrite modes



Enforcing Authoritative Action

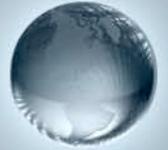
IV. Design by Number

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



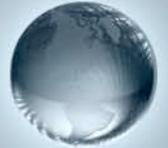
Project Paraegis

Code



Project Paraegis

Execution



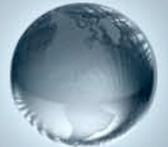
Project Paraegis

Trivial Attacks Thwarted



Project Paraegis

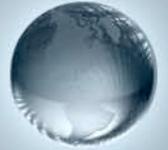
How you configure/use Razorwire



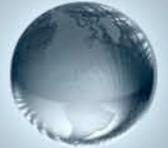
Chapter VII

Q&A:

Shoot the Messengers

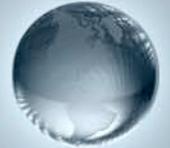


Additional Notes



References

- Paraegis <http://www.anachronic.com/paraegis>
- OWASP <http://www.owasp.org>
- WASC <http://www.webappsec.org>
- SecureCoding <http://www.securecoding.org>
- STRIDE Threat Models <http://www.microsoft.com>
- “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, D. Schwabe and G. Rossi.
- “Measuring Relative Attack Surfaces” <http://www-2.cs.cmu.edu/~wing/>



Arian J. Evans

<http://www.anachronic.com>

913.888.6524 (H)

913.710.7085 (M)

arian (at) anachronic{dot}com

Daniel Thompson

<http://www.ionize.net>

816.421.1901 (O)

816.509.7998 (M)

daniel (at) ionize{dot}net

Mark Belles

<http://mrbelles.brinkster.net/>

816.421.1901 (O)

913.555.1212 (M)

markbelles (at) gmail{dot}com