# do_action(‘hack_me’)

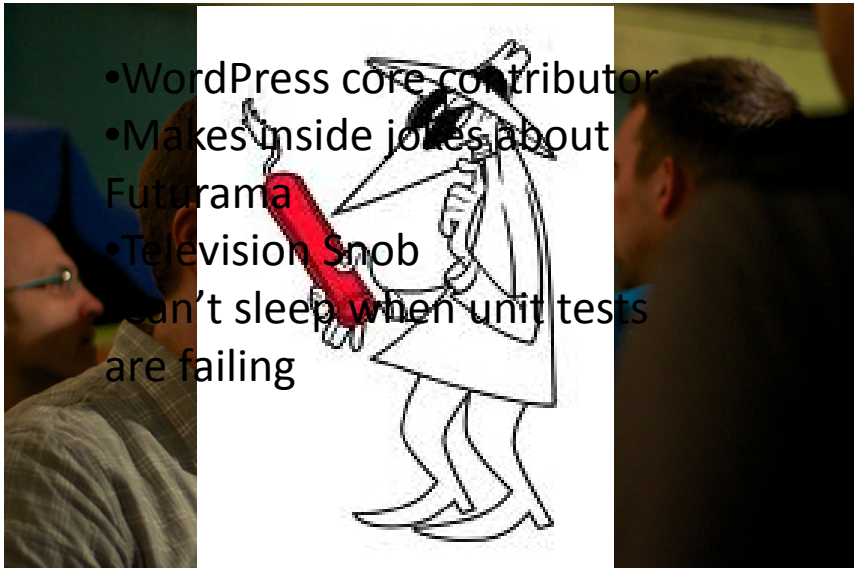
Who are we and why are we talking about this?

# ABOUT US

## Joshua Hansen

- This is my first WordCamp
- Security and Open Source enthusiast
- Prefers British television
- Finding vulnerabilities is fun

## Kurt Payne



- WordPress core contributor
- Makes inside jokes about Futurama
- Television Snob
- Can't sleep when unit tests are failing

Reasons your site gets hacked

# WHY ME?

# Money

- Phishing Site
- Malware Downloads
- Warez/Piracy
- SEO
- Spam
- DDOS

What are they exploiting?

# COMMON VULNERABILITIES

# OWASP Top Ten

- Injection
- Cross-Site Scripting
- Broken Authentication and Session Management
- Insecure Direct Object References
- Cross-Site Request Forgery
- Security Misconfiguration
- Insecure Cryptographic Storage
- Failure to Restrict URL Access
- Insufficient Transport Layer Protection
- Unvalidated Redirects and Forwards

Explanation and Example

# REMOTE FILE INCLUSION


DEMO

Prevention

# REMOTE FILE INCLUSION

- Never trust user supplied input!
- Sanitize your data - http://codex.wordpress.org/Data_Validation
- Verify length
- Verify type
- Test against regex
- Compare against whitelist
- Validate inputs

Explanation and Example

# CROSS-SITE SCRIPTING (XSS)


DEMO

Prevention

# CROSS-SITE SCRIPTING (XSS)

- Never trust user supplied input!
- Sanitize your data - http://codex.wordpress.org/Data_Validation
- Escape based on context
- Use the methods provided by WordPress

Explanation and Example

# SQL INJECTION


DEMO

Prevention

# SQL INJECTION

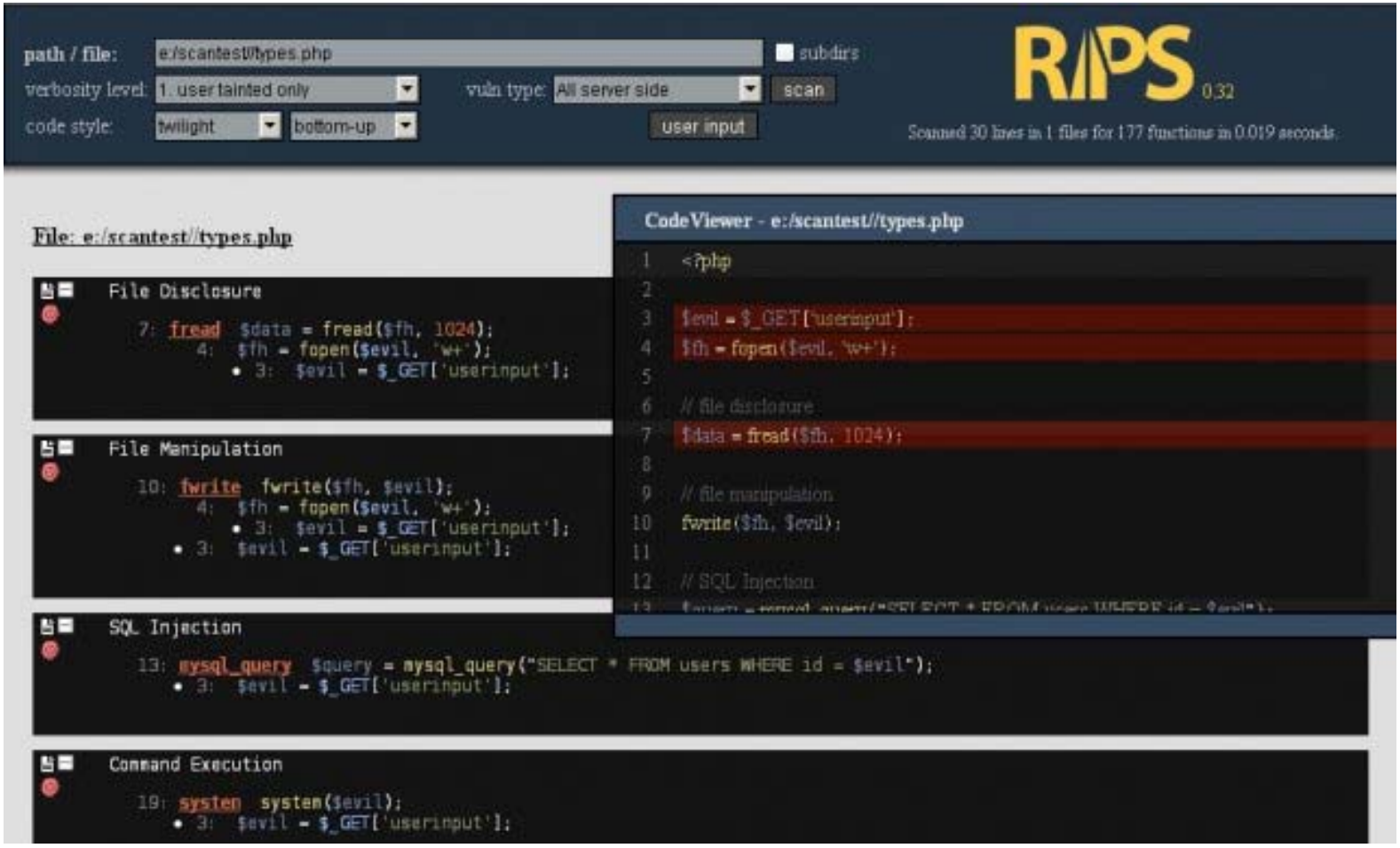
- Never trust user supplied input!
- Sanitize your data - http://codex.wordpress.org/Data_Validation
- Use Prepared Statements – wpdb::prepare()
- Escape user supplied input
- Least Privilege
- Whitelist input validation

Tools and Resources

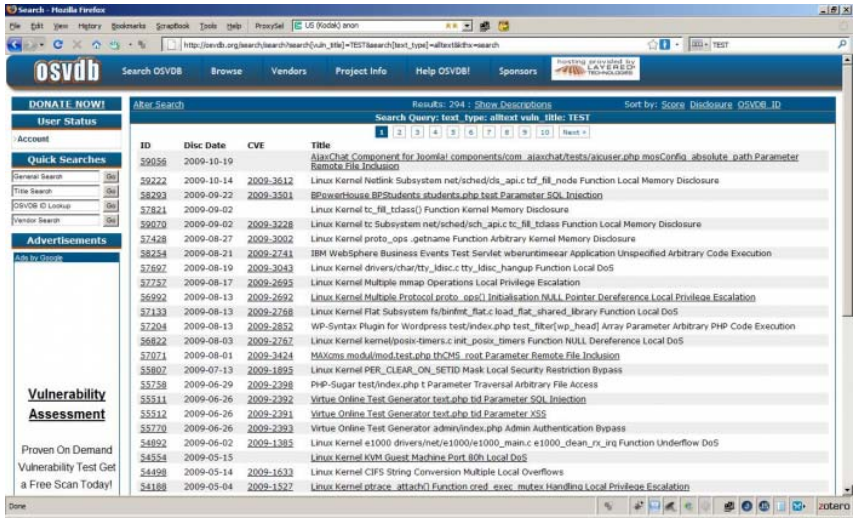
# WHAT CAN I DO?

# W3AF

# RIPS


path / file: e:/scantest/types.php
subdirs
verbosity level: 1: user tainted only
vuln type: All server side
scan
code style: twilight
bottom-up
user input
RIPS 0.32
Scanned 30 lines in 1 files for 177 functions in 0.019 seconds
File: e:/scantest//types.php
File Disclosure
7: fread $data = fread($fh, 1024);
4: $fh = fopen($evil, 'w+');
• 3: $evil = $_GET['userinput'];
File Manipulation
10: fwrite fwrite($fh, $evil);
4: $fh = fopen($evil, 'w+');
• 3: $evil = $_GET['userinput'];
• 3: $evil = $_GET['userinput'];
SQL Injection
13: mysql_query $query = mysql_query("SELECT * FROM users WHERE id = $evil");
• 3: $evil = $_GET['userinput'];
Command Execution
19: system system($evil);
• 3: $evil = $_GET['userinput'];
CodeViewer - e:/scantest//types.php
1 <?php
2
3 $evil = $_GET['userinput'];
4 $fh = fopen($evil, 'w+');
5
6 // file disclosure
7 $data = fread($fh, 1024);
8
9 // file manipulation
10 fwrite($fh, $evil);
11
12 // SQL Injection

# Vulnerability Databases




# OWASP

# Links

- W3AF - http://w3af.sourceforge.net/
- RIPS - http://rips-scanner.sourceforge.net/
- NVD - http://nvd.nist.gov/
- OSVDB - http://osvdb.org/
- Exploit-DB - http://www.exploit-db.com/
- OWASP - https://www.owasp.org
- Hardening WordPress - http://codex.wordpress.org/Hardening_WordPress
- WordPress SQLi - http://codex.wordpress.org/Class_Reference/wpdb#Protect_Queries_Against_SQL_Injection_Attacks
- WordPress Data Validation – http://codex.wordpress.org/Data_Validation