

My tryst
with
Source Code Review

Anant Shrivastava
Information Security
Consultant

\$whoami

\$ Name : Anant Shrivastava

\$ Work : Information Security Consultant

\$ Work on : Web, Mobile, Linux

\$ Project Leader :

- * Android Tamer

- * Codevigilant

\$ Past life :

- * System and server Administrator

- * Developer (wp-filemanager >1L downloads)

Agenda

- My journey so far in world of bug finding via code review.
- And Yes I bluffed its not just about code review its also about associated automation and simple yet relevant techniques we used to identify all of that.
- Simplest form : idea is to showcase what and how I have done stuff and how others can also do it.

What not to expect

- Tools Release
- Highly Sophisticated Code
- Artificial Intelligence
- Discussion about SAST (Static Application Security Testing)

Disclaimer

- No commercial Source code review tools were harmed during the exercise.

Lets Read

With enough eyes all bugs are shallow

- Linus Torvalds

Let me re-write it

With enough **expert** eyes all bugs are shallow

WHY

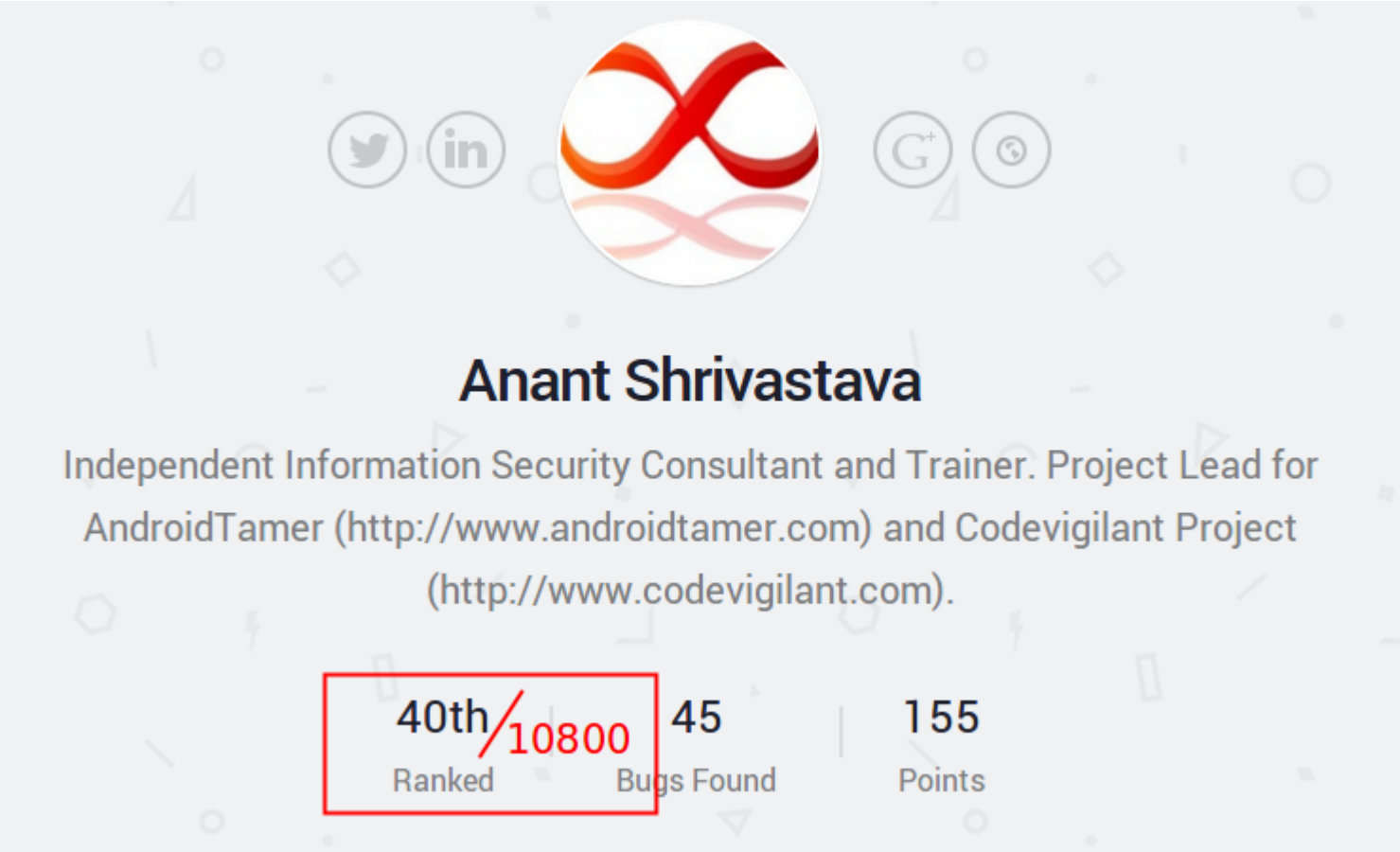
- Remember Last slide. Become “Expert Eye”
- Wanted to Learn and Experiment new stuff.

1yr back (2013) : Security Researcher
mainly Blackbox penetration tester and
tool’s author.

Why not Bug Bounties

- Invested time on Latest fad : Bugbounties
- Realized : mostly its about alert('XSS') and X-FRAME-OPTIONS or httpOnly for most of them.
- Very few actually do something good.
- Surface area is pretty small and its blackbox most of the time.
- Personal opinion and people may have different opinion and its perfectly fine.

Bug Bounty efforts



The profile card features a central circular avatar with a red infinity symbol. Above the avatar are social media icons for Twitter, LinkedIn, Google+, and GitHub. Below the avatar is the name 'Anant Shrivastava' in bold. Underneath is a bio: 'Independent Information Security Consultant and Trainer. Project Lead for AndroidTamer (<http://www.androidtamer.com>) and Codevigilant Project (<http://www.codevigilant.com>).'. At the bottom, there are three statistics: '40th / 10800' (with '10800' in red and a red box around the entire row), '45 Bugs Found', and '155 Points'.

Anant Shrivastava

Independent Information Security Consultant and Trainer. Project Lead for AndroidTamer (<http://www.androidtamer.com>) and Codevigilant Project (<http://www.codevigilant.com>).

40th / 10800	45	155
Ranked	Bugs Found	Points

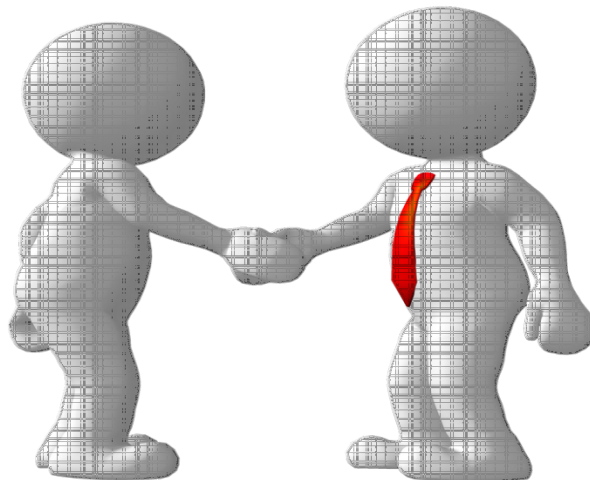
What Next

- Lets try Code review
- But I don't know code review
- So lets learn code review

- And off it goes into the ice box called pending things to learn.

Parallel efforts

- I am not the only person thinking this, met another fellow researcher “Prajal Kulkarni” who was also planning on something similar and was looking for collaboration
- We touched bases during #nullblr meet and off we-started with the project.



{Cv;}

code{vigilant}

What is codevigilant

- A community collaboration effort to make opensource software's secure.
- Finding bugs and responsibly disclosing them to respective author and preferable getting software updated.
- Responsible disclosure on website after sufficient interval

Target A EcoSystem

- We Picked WordPress Ecosystem which meant
 - WordPress Plugins (current focus)
 - WordPress Themes (current Focus)
 - WordPress Core (future check)
- Pick an ecosystem which you think is near and dear to you and the language which you can easily understand.

Lets Roll

- This is where things started to act funny.
- We started with
 - Lets download top 10 and analyze one by one.
 - Ended up getting frustrated in couple of days
 - Mind you we were just two pentesters fiddling around with source code. Whitebox was not exactly our forte.

Lets Re-Roll

- Lets automate and improvise
 - Download all plugins and Themes
 - Focus on vulnerability type and not on individual plugin
- Seems like a good plan : so lets roll

Lets count

WordPress 3.9 has been downloaded

47,841,560

times

32,943 PLUGINS **719,955,434** DOWNLOADS, AND COUNTING

2,656 THEMES, **109,464,666** DOWNLOADS, AND COUNTING

Automate Please

- Quick WordPress information extractor and downloader.
- Simple python script with grep / cut friendly output.

```
$python wordpress_plugin_info.py 1 wp-filemanager  
"wp-filemanager/","Published","2014-08-23 01:47:35.720368","2013-5-17","1.4.0","117156","3.5.2","3.2  
or higher","anantshri:J","http://downloads.wordpress.org/plugin/wp-filemanager.1.4.0.zip"  
$
```

wp-FileManager

FileManager for WordPress allows you to easily change, delete, organize and upload files.

[Download Version 1.4.0](#)

[Description](#) [Installation](#) [FAQ](#) [Screenshots](#) [Changelog](#) [Stats](#) [Support](#) [Reviews](#) [Developers](#)

WP-Filemanager is your one stop solution for all file management work right from the wordpress admin page.

Following features are present as of now.

- Create File, Folder
- Upload ,Download file
- View, Edit files
- rename an delete files
- Configuration menu inside WP-admin panel

More features to be added soon.

- Code editor for script fles
- WYSIWYG editors for html files
- Image editor for image files

Tags: [change](#), [delete](#), [file](#), [management](#), [organize](#), [upload](#)

Requires: 3.2 or higher

Compatible up to: 3.5.2

Last Updated: 2013-5-17

Downloads: 117,155

Ratings



4.3 out of 5 stars

5 stars 27

4 stars 5

3 stars 1

2 stars 1

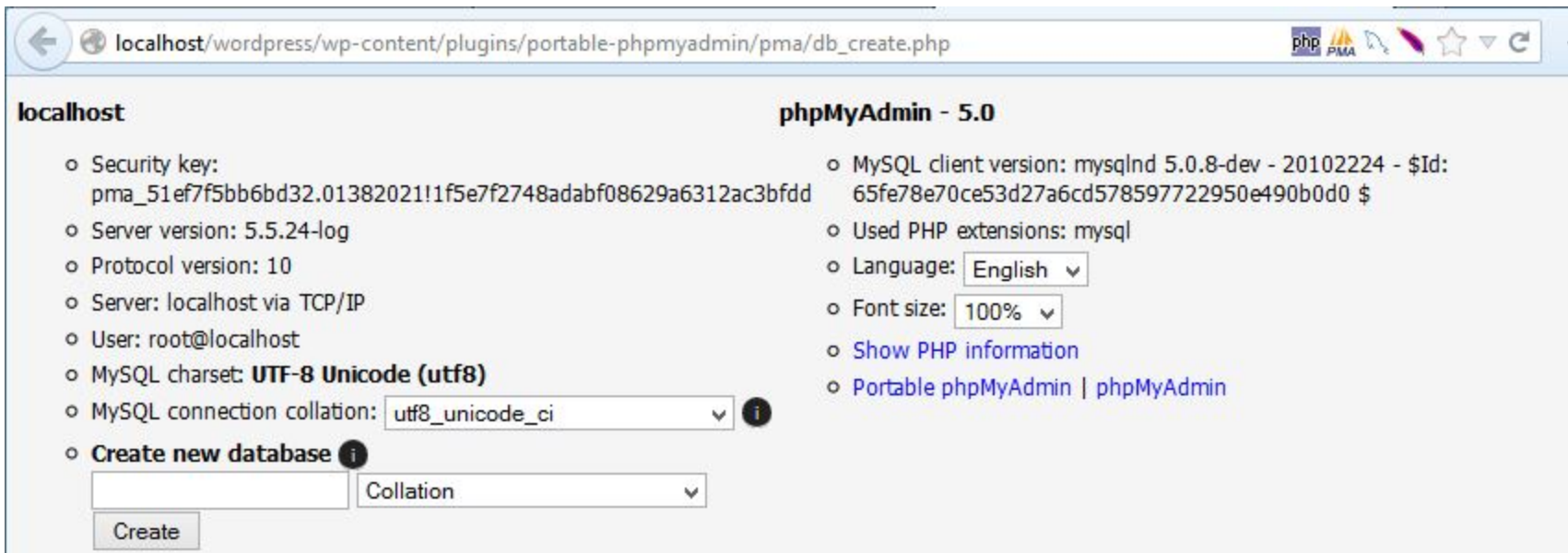
1 stars 4

Authors

anantshri
2 plugins

Lazy Me

- Lets start with some easy stuff
- How about looking at direct / unauthorized Access



The screenshot shows the phpMyAdmin 5.0 interface in a browser window. The address bar displays the URL: localhost/wordpress/wp-content/plugins/portable-phpmyadmin/pma/db_create.php. The interface is divided into two main sections: 'localhost' and 'phpMyAdmin - 5.0'.

localhost

- Security key: pma_51ef7f5bb6bd32.01382021!1f5e7f2748adabf08629a6312ac3bfdd
- Server version: 5.5.24-log
- Protocol version: 10
- Server: localhost via TCP/IP
- User: root@localhost
- MySQL charset: **UTF-8 Unicode (utf8)**
- MySQL connection collation: utf8_unicode_ci
- **Create new database**

phpMyAdmin - 5.0

- MySQL client version: mysqlnd 5.0.8-dev - 20102224 - \$Id: 65fe78e70ce53d27a6cd578597722950e490b0d0 \$
- Used PHP extensions: mysql
- Language: English
- Font size: 100%
- [Show PHP information](#)
- [Portable phpMyAdmin](#) | [phpMyAdmin](#)

The 'Create new database' section includes a text input field for the database name, a dropdown menu for the collation (currently set to utf8_unicode_ci), and a 'Create' button.

Any tool available

- Inspathx works just fine but I never got it to work for me.
- So wrote a simple python script.

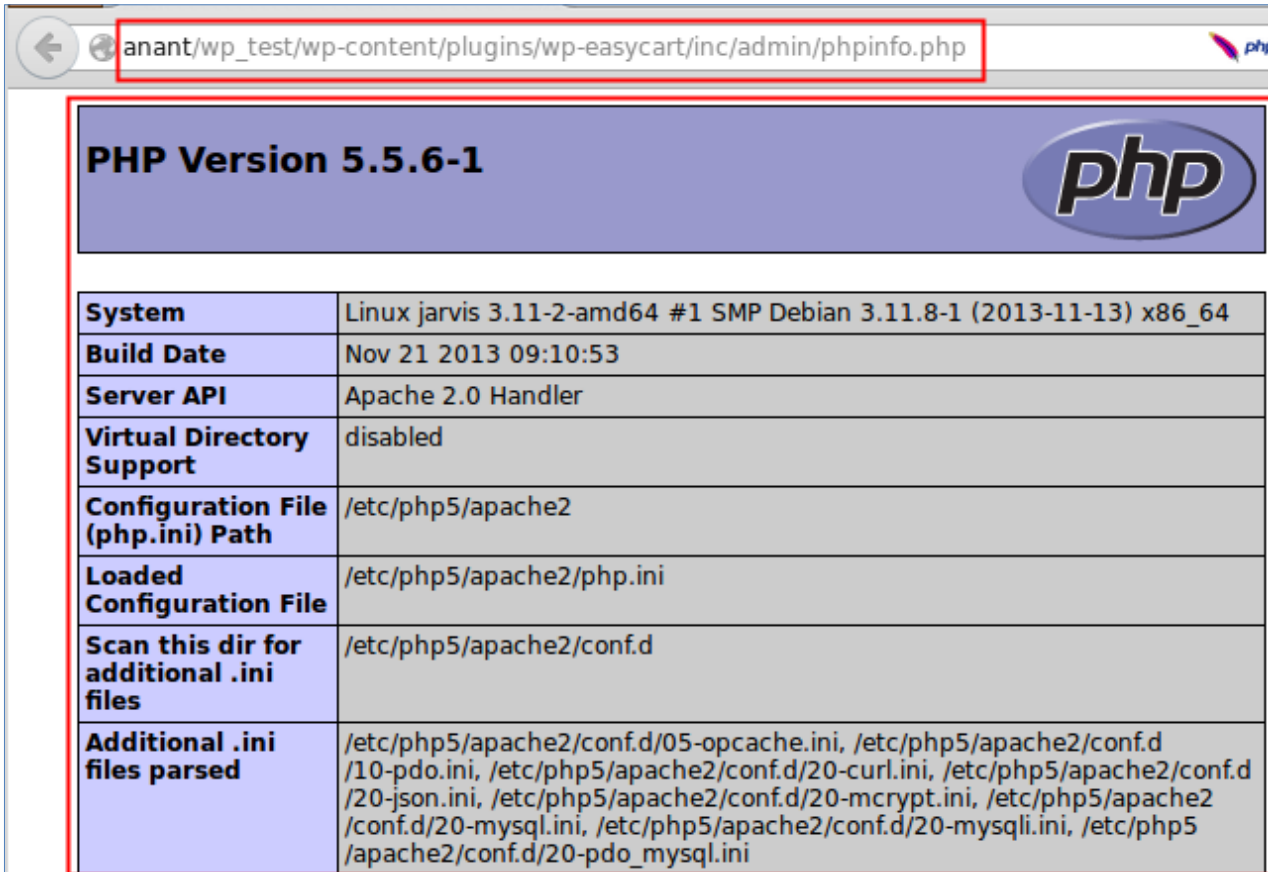
Tool release

- Well I Lied let me release some tools for you
- Error_finder release

https://github.com/Codevigilant/error_finder

Output

- Massive amount of Full Path Disclosure
- Few direct access issues



PHP Version 5.5.6-1

System	Linux jarvis 3.11-2-amd64 #1 SMP Debian 3.11.8-1 (2013-11-13) x86_64
Build Date	Nov 21 2013 09:10:53
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-curl.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-mcrypt.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini

Full Path Disclosure

- WordPress stand of FPD is clear so no point reporting it.

Why are there path disclosures when directly loading certain files?

This is considered a server configuration problem. Never enable `display_errors` on a production site.

WordPress Ecosystem

- Lets step back and understand ecosystem a bit more
 - WordPress is a CMS
 - Various User Roles
 - Super Administrator
 - Administrator
 - Editor
 - Author
 - Contributor
 - Subscriber

WordPress ecosystem

- Various plugin bind to various roles
- Issues without authentication are prime concern.
- Issues affecting subscriber and contributor hot 2nd.
- Editor and Admins have unescaped HTML access

What Next

- With this understanding in place we wanted to focus on unauthenticated issues first.
- Lets start with A3-Cross Site Scripting

XSS

- We thought its as simple as getting a `<script>alert('XSS')</script>` back
- Alas that should have been the case.
- How to find it via Source code review.
 - `Echo $_GET['input'];`
- Grep “`echo \$_GET`” should work

Did we missed something

- How could we not miss obvious stuff

```
<?php
$x=$_GET['input'];
$y=120+420;
echo $x;
?>
```

- Attempt 2
 - Either build a lexical parser tokenize whole source code or play intelligent
 - Extract all GET/POST/REQUEST parameters and access url with those parameters in place.

A3 XSS cont'ed

- We did find massive amount on entries and then realized we have again screwed up.
- If Content type is text/html XSS works
- But if content type is
 - Json
 - Xml
 - Javascript
- XSS failed

Automated more

- Wrote another set of scripts which gives proper response types also.

```
XSS FOUND : text/html : http://anant/wordpress_research/wp-content/plugins/spotlightyour/library/includes/payment/paypalexpress/DoDirectPayment.php?paymentType=paymentType'><script>alert(document.cookie)</script>&
XSS FOUND : text/html : http://anant/wordpress_research/wp-content/plugins/wp-social-invitations/test.php?url=url'><script>alert(document.cookie)</script>&xhrurl=xhrurl'><script>alert(document.cookie)</script>&fsock=fsock'><script>alert(document.cookie)</script>&
XSS FOUND : text/html : http://anant/wordpress_research/wp-content/plugins/rezgo/book_ajax.php?response=response'><script>alert(document.cookie)</script>&
XSS FOUND : text/html : http://anant/wordpress_research/wp-content/plugins/rezgo/templates/default/index.php?end_date=end_date'><script>alert(document.cookie)</script>&cid=cid'><script>alert(document.cookie)</script>&tags=tags'><script>alert(document.cookie)</script>&search_for=search_for'><script>alert(document.cookie)</script>&pg=pg'><script>alert(document.cookie)</script>&search_in=search_in'><script>alert(document.cookie)</script>&start_date=start_date'><script>alert(document.cookie)</script>&
XSS FOUND : text/html : http://anant/wordpress_research/wp-content/plugins/swipe-hq-checkout-for-eshop/test-plugin.php?api_key=api_key'><script>alert(document.cookie)</script>&payment_page_url=payment_page_url'><script>alert(document.cookie)</script>&merchant_id=merchant_id'><script>alert(document.cookie)</script>&api_url=api_url'><script>alert(document.cookie)</script>&currency=currency'><script>alert(document.cookie)</script>&
XSS FOUND : text/html : http://anant/wordpress_research/wp-content/plugins/rootspress/pgv/treenav.php?locale=locale'><script>alert(document.cookie)</script>&rootid=rootid'><script>alert(document.cookie)</script>&jsname=jsname'><script>alert(document.cookie)</script>&zoom=zoom'><script>alert(document.cookie)</script>&gedid=gedid'><script>alert(document.cookie)</script>&
XSS FOUND : text/html; charset=utf-8 : http://anant/wordpress_research/wp-content/plugins/gdeslon-affiliate-shop/go.php?url=url'><script>alert(document.cookie)</script>&
XSS FOUND : text/html : http://anant/wordpress_research/wp-content/plugins/wp-ttisbdir/forms/sub_cat.php?<?php if ($result['enabled=<?php if ($result['enabled'><script>alert(document.cookie)</script>&cat_id=cat_id'><script>alert(document.cookie)</script>&edit_sub=edit_sub'><script>alert(document.cookie)</script>&sub_cat_id=sub_cat_id'><script>alert(document.cookie)</script>&
```

A9 - Known Vulnerable components

- We also focused on this issues category and identified multiple issues here also.
- Mainly those were concerned with outdated SWF binaries used or old library files used.

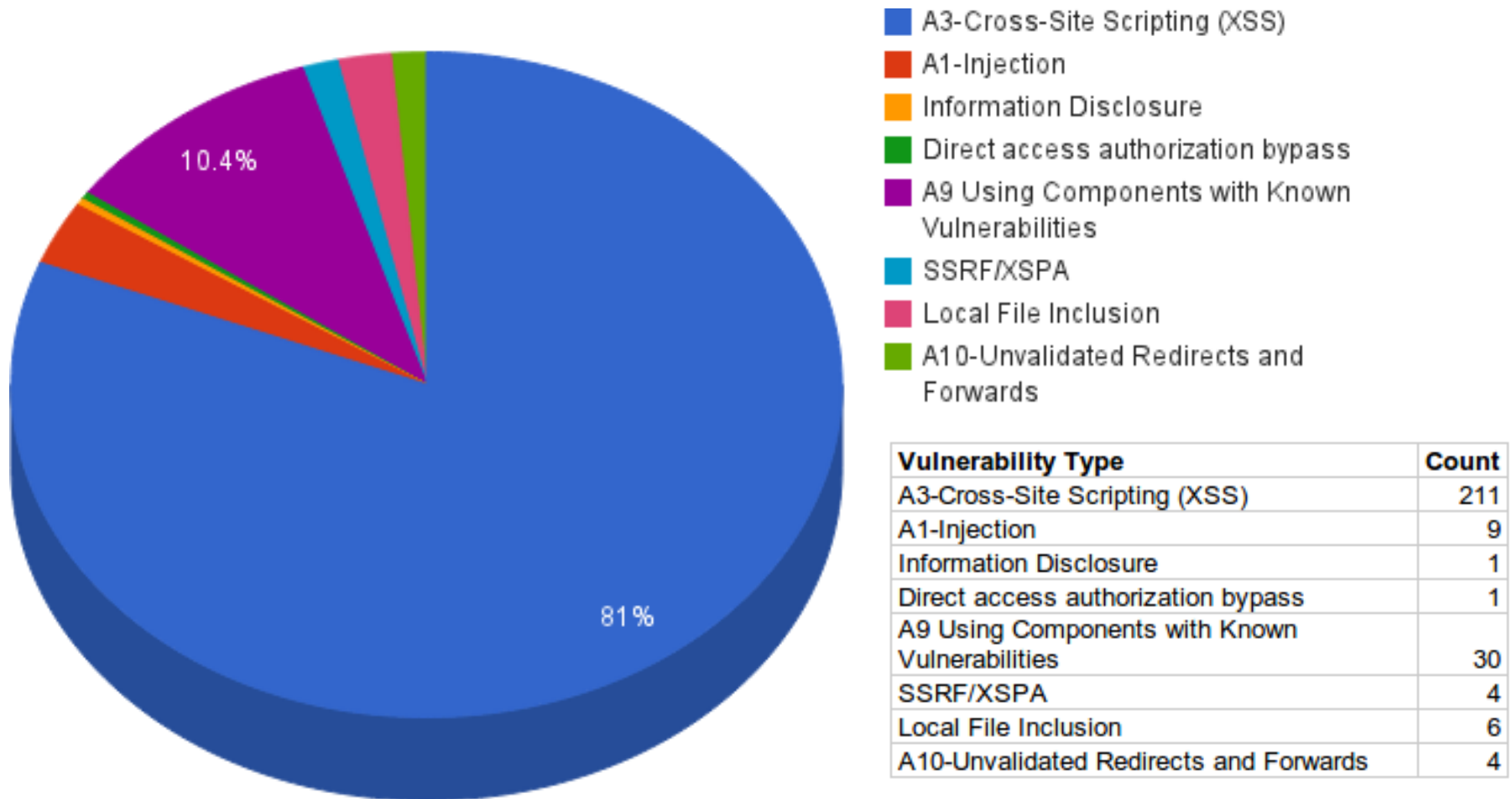
By-Product

- Error messages extracted via inspathx code yielded to multiple other issues like Directory traversal

End Result

- Although this was like a rookie attempts to finding I don't know what I am finding but we ended up with 250 plus issues in various WordPress plugins
- We Decided to call this Phase 1

Phase 1 Statistics



Phase 2

- So What's next
- Authenticated issues
 - SQL Injection
 - Stored/Reflected Cross Site scripting in Admin console
 - CSRF
 - And more

Phase 2 Hurdles

- We realized that authenticated flaws are prioritized based on user access.
- We need to map all 30K Plugins with each type of access.
 - Script in progress to do exactly that.

Team Expansion

- Started with me and Prajal we are now 4 people strong team
 - Anant Shrivastava
 - Prajal Kulkarni
 - Chaitu
 - Madhu Akula

What Next

- We are seeking for more volunteers to come forward and help us make opensource softwares a more secure platform.

What's in this for audience

- Simple list of vulnerable functions in PHP you can look for in your own codebases

User Controls

`$GLOBALS`
`$_SERVER`
`$_GET`
`$_POST`
`$_FILES`
`$_COOKIE`
`$_SESSION`
`$_REQUEST`
`$_ENV`

Cross Site Scripting (XSS)

`echo()`
`print()`
`printf()`

File Tainted

`file()`
`fopen()`
`popen()`
`file_get_contents()`
`fread()`
`fscanf()`

Database tainted

`mysql_fetch_array()`
`mysql_fetch_assoc()`
`mysql_fetch_field()`
`mysql_fetch_object()`
`mysql_fetch_row()`

File Inclusion

`include()`
`require()`
`require_once()`
`include_once()`

Command Execution

`exec()`
`shell_exec()`
`system()`
`proc_open()`

SQL Injection

`mysql_query()`
`pg_query()`

What's in this for audience

- Appeal to use codevigilant platform
- You find flaws
 - Either join our team and do continuous contribution
 - You get an author's page at codevigilant
 - If you get any bounty for the bug you keep it.
 - Send details as one off cases of finding
 - We will do co-ordination with third party
 - We will **try** to get it patched or remove it from internet if not patched.
 - We will publish advisory on website with yours and co-ordinator's name in advisory.

What's in this for audience

- If you want a open source product tested contact us and we will see what we can do about it.
- If you want quick test's you can think about donating to the project.

Simple Checklist

- Look for Obvious flaws in unauthenticated Code
 - Reflected XSS
 - SQL Injection
 - Direct access / information disclosure
 - Directory Traversal
- Understand Application Architecture
 - Language specific checks
 - List of language specific vulnerable functions
 - List of User Roles with impact of confidentiality
- Attack Authenticated section
 - Stored XSS
 - CSRF
 - XSPA
 - SQL Injection
 - Direct URL access

CodeVigilant

- <http://www.codevigilant.com>
- <https://github.com/Codevigilant>
- <https://facebook.com/Codevigilant>
- <https://twitter.com/Codevigilant>

Questions?

Why not automated scanners

- They are either good at black or whitebox.
- We wanted to confirm from both sides.
- They have a workflow which should be followed.

Open Source automation Tools

- Tested rats and couple of other tools only rips worked marginally good.
- But rips workflow demanded we enter url in webview everytime and web view keeps getting hanged from time to time.

Commercial scanners

- No motivation to use them (we will be processing result not learning from it)
- No money to spend on them
- We did get one generous offer and tried one product

Commercial scanner

- I don't play name shame game hence no names here.
- Commercial product was cloud hosted app where we need to upload code for review.
- Software missed simple XSS and SQLi but so did open source tools also.

Why scanners missed

- WordPress or Other CMS have there own functions to handle stuff
- Example
 - Mysql query
 - WordPress query
- These scanners don't know about it and failes to detect it.

Why scanners missed

- Or it could have been a simple case of misconfiguration at our end.
- But after sharing results with Tool Dev they kind of vanished and didn't responded back.