State of PHP Security

Ilia Alshanetsky

ZendCon 2007


DANGER
QUICKSAND
STAY AWAY

- Reactive rather then proactive
- Depends entirely on people reporting issues
- Little done to audit existing code for security concerns
- Limited test-suit and tools to identify issues
- Design with features and not security in mind

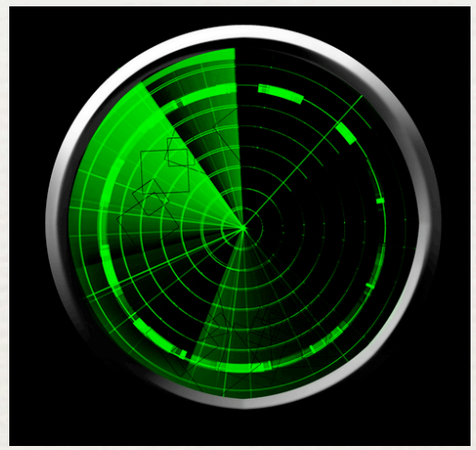
The Consequences?

- Frustrated security researchers
- Bug reporters prompt for 0-day rather then responsible disclosure
- Bugs being solved and similar ones are being introduced in other areas of the code
- Security regressions and/or incomplete fixes

Solutions

Automated Code Analysis

- Improve quality of Coverity automated scanning by working together with Coverity team
  - Fewer false positives
  - More meaningful results
  - Easy to identify real issues and deploy solutions
  - Close attention being paid to scan results

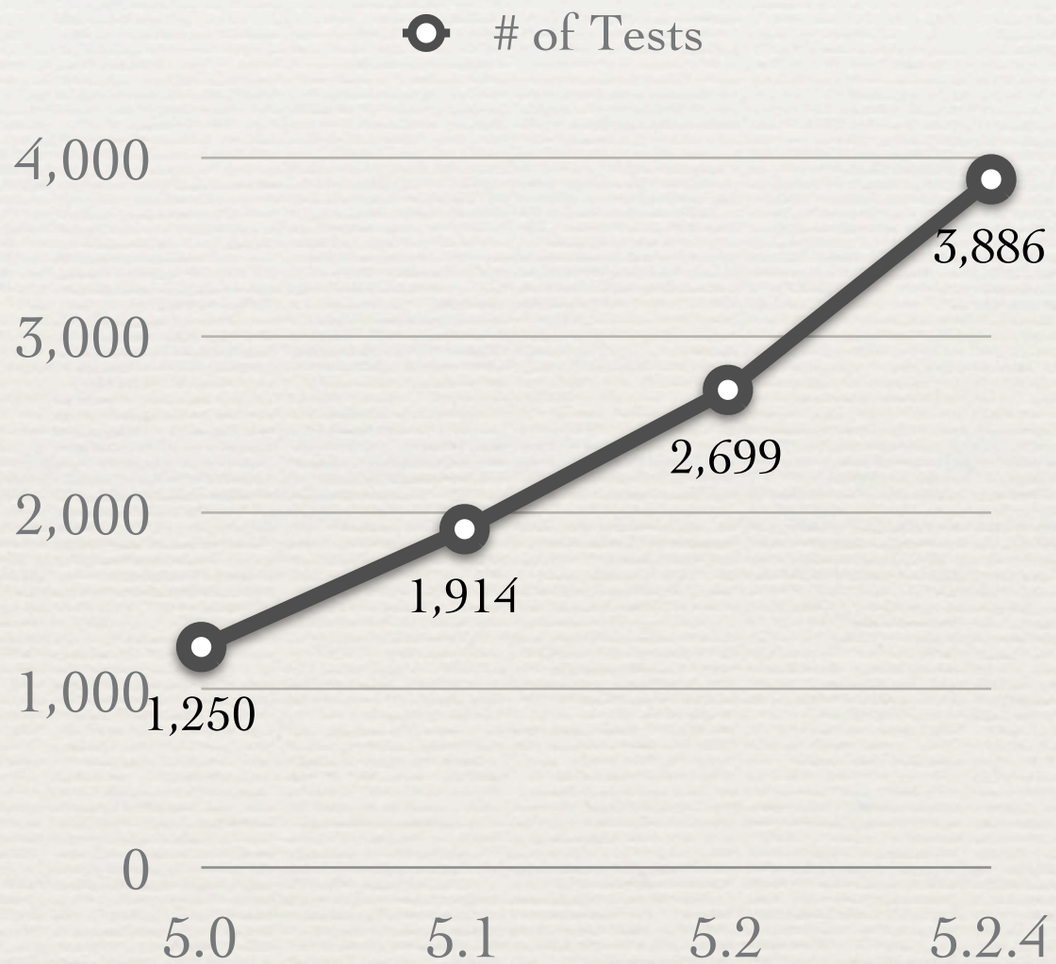


Until September 13th 2007 - No defects for 3 months

Of 37 new issues identified - 30 resolved or identified as false positives in less then 10 days.

Test Suite Expansion

- More new tests
- Write tests for resolved security bugs to prevent regressions
- Focus more on corner cases then common behavior
- Filter tests through valgrind
- Focus on improving code coverage



http://gcov.php.net/

Use Fuzzing to Find Bugs

- Develop a series of tools to try and identify issues by generating bogus inputs and passing them to functions
  - Overly long strings (1 MB and longer)
  - Strings with “strange” characters embedded in them. (0x00, 0x0A, \, etc…)
  - Big integers 2^24
  - Bogus resources

Give Credit Where It is Due

- Give credit in release announcements & ChangeLog to people discovering security bugs

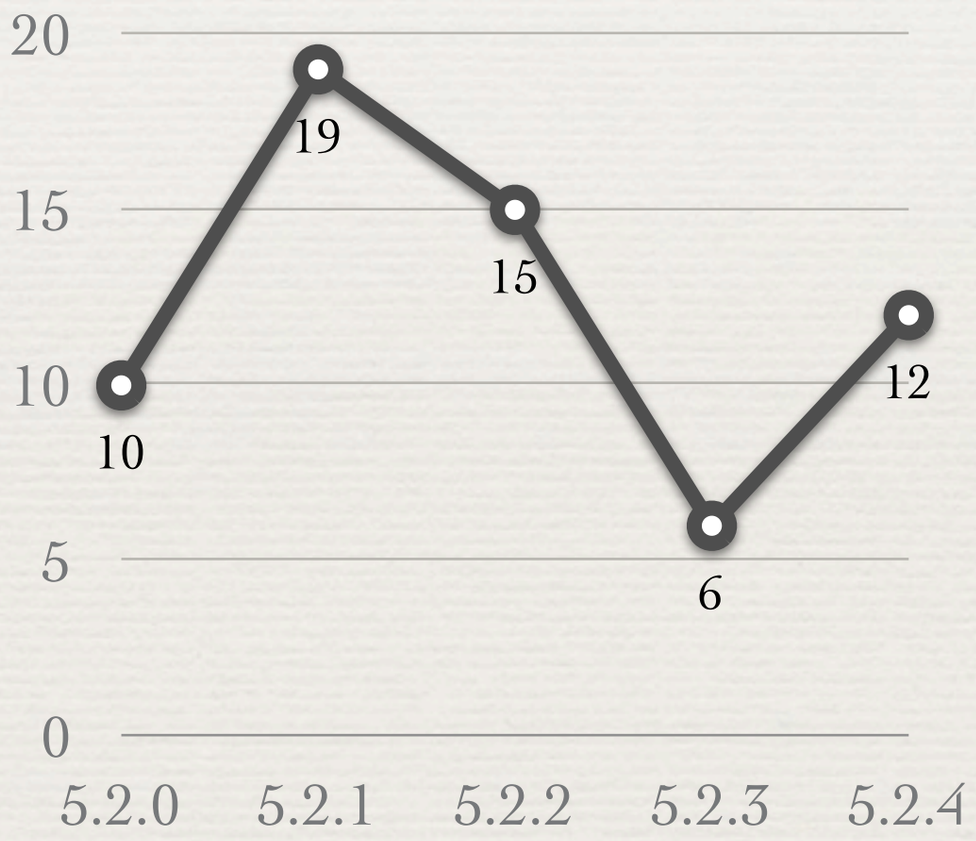
- Communicate more rapidly with people reporting security issues and be more open in terms of expected resolution

- Try to involve bug reporters in the resolution process

What did we fix?

Per-Release Security Fixes

- The big push in 5.2.0-5.2.1 was thanks to fuzzing performed by Stas, Tony and myself.
- 5.2.0 - 5.2.2 resolved many issues that were identified by Stefan Esser as part of M.O.P.B.
- Contributions by many different security researchers.



What were the problems?

- open_basedir and safe_mode bypasses
- A fair number of buffer and integer overflows
- Denial of service attacks by crashing PHP and subsequently a web-server thread
- Validator issues inside the newly added filter extension
- Several string format vulnerabilities

Exploitability

- Majority of the identified issues can only be triggered by a local user.

The ones that are remote, require certain, not trivial to match conditions.

- **However, it takes just one hostile local user on an shared server or a badly written script to turn a local exploit into a remote one!**

Most Affected Areas

- String functions inside ext/standard
- Session, GD, mbstring, interbase, imap, filter, zip and ODBC extensions.
- File operations inside ext/standard
- mail() function
- SOAP and XMLRPC extension (remote exploit)

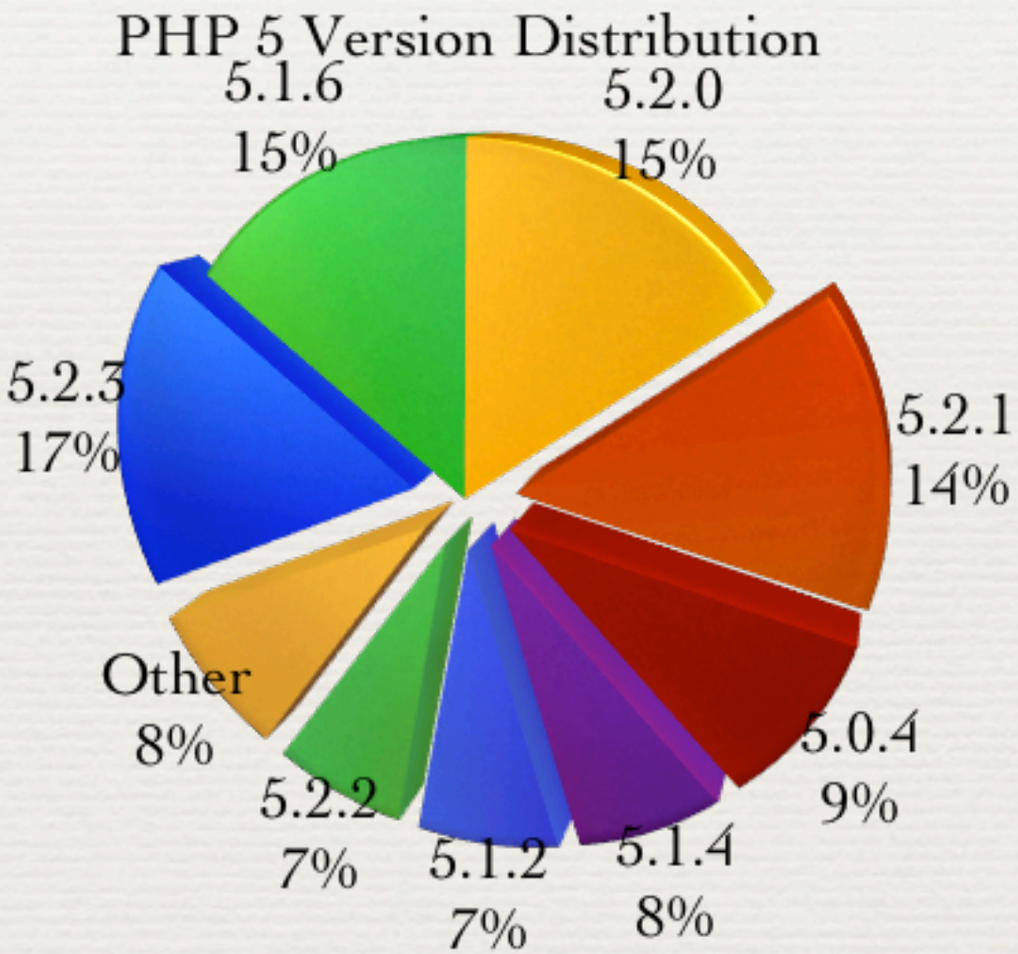
Security Enhancements

- Added internal heap protection to reduce consequences of buffer overflows
- Memory limit is now always enabled
- Filter extension was introduced and enabled by default
- Introduced **allow_url_include** setting that is disabled by default
- Added nesting limit on input arrays

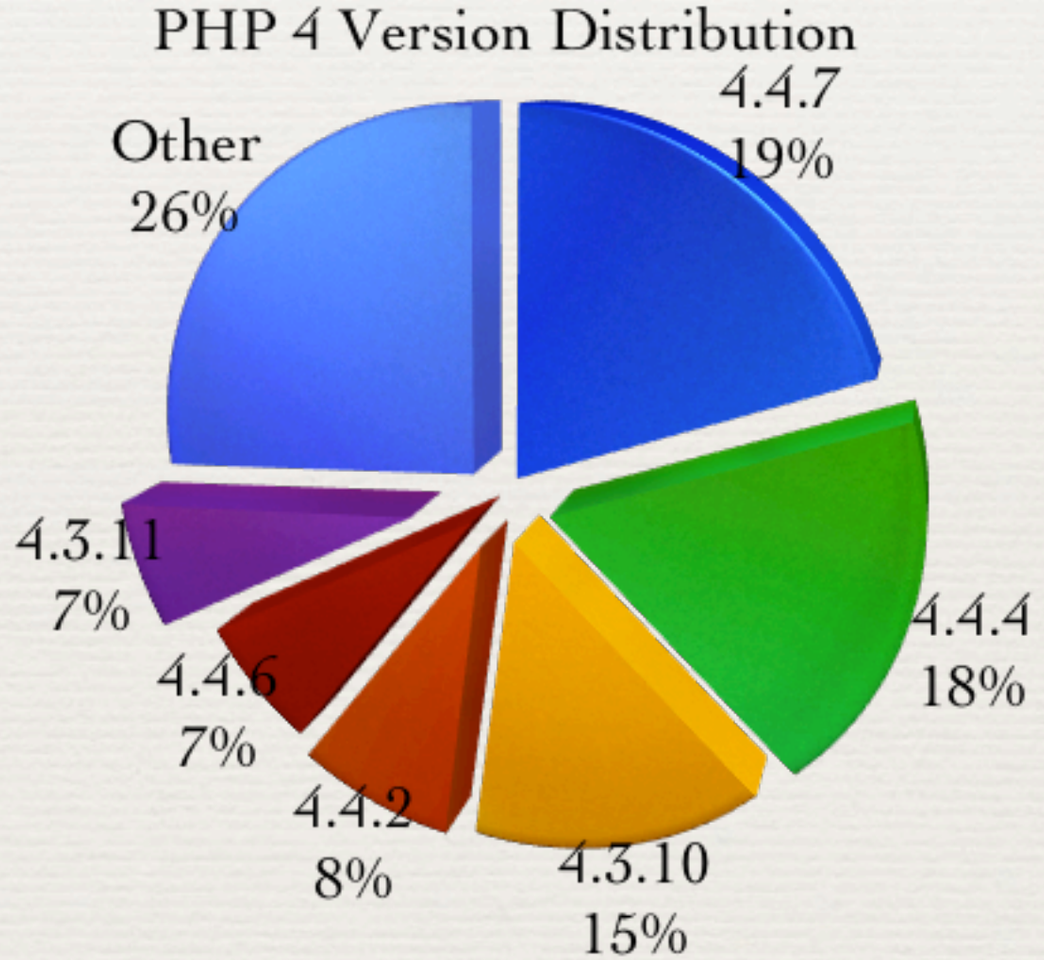
What else is needed?

Get People to Upgrade

5.2.3 5.1.6 5.2.0 5.2.1 5.0.4
5.1.4 5.1.2 5.2.2 Other



4.4.7 4.4.4 4.3.10 4.4.2
4.4.6 4.3.11 Other

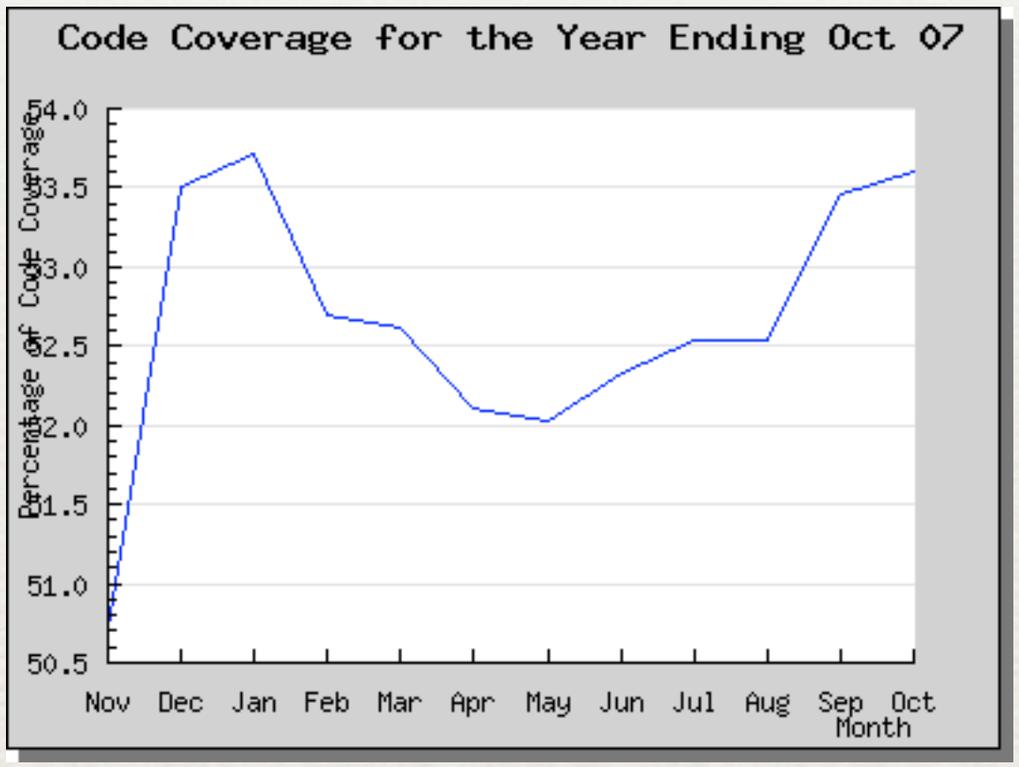


Overall PHP 4 still comprises 77.72% of all installations

Data provided by Damien Seguy

Improve Code Coverage

- As you saw from previous slides the number of tests had increased by nearly 200%.



- Conversely code coverage went up only by about **4%** in the last year

**The drop in coverage was caused by addition of new code through new extensions.

Manual Code Auditing

- More manual code auditing needs to be performed with a focus on security

- Newly added code (extensions, functions, etc…) needs to be examined for security as a criterial for inclusion

- Looks at all crash bugs as potential exploits until examination is done to prove otherwise

Better Communication

- Improve communication with Open Source distributions in respect to security fixes

- More informative news announcements in regard to security issues

- Better interaction with security researchers & encourage more people to look at PHP's security

Rapid Release Cycle

- Release early, release often
- Micro-security releases PHP-X.X.X-s[N]

More security features

- Look at common challenges developers have when trying to develop secure PHP applications and provide tools, not automation (Ex. magic_quotes) to make it simpler

Thank you for listening

Additional Resources:

- These slides - http://ilia.ws
- Test suite & code coverage results - http://gcov.php.net
- Writing new tests & testing snapshots - http://qa.php.net/
- PHP 5 ChangeLog - http://www.php.net/ChangeLog-5.php