Securing PHP

Survey of the solutions

Stanislav Malyshev

stas@zend.com

Most code is extremely buggy…

Can we help?

Input filtering

- Unauthorized code (remote include)
- Unauthorized DB access (SQL Injection)
- Client subversion (XSS, XSRF)

Let’s protect all data

Magic quotes:

☺ a.php?data=1’2 -> $data == “1\’2” can be inside quotes

☹ Optional

☹ No support for context

Let’s restrict the user

Safe mode:

☺ Allow access only to own files

☺ Allow only “safe” actions

☹ No OS support

☹ Too many modules not controlled

☹ Too hard to find out all “unsafe” ones and not kill apps

Let’s filter

☺ $var = filter_input(INPUT_GET, 'var');

☺ Standard filters for standard use-cases

☹ No time machine

☹ Voluntary

Let’s watch the data

Data tainting

☺ No unfiltered data in sensitive contexts

☹ How do I know the filtering was right?

☹ Complex implementation – contexts

☹ Performance

Static vs. Dynamic

Static

☺ Can be as slow as it needs to

☺ False positive OK

☺ External engine

☹ $$foo = $$bar

☹ $foo->$bar($baz)

☹ eval($foo.$bar)

Dynamic

☺ Real code, real data

☺ Can prevent attack

☹ Need for speed

☹ Engine modification

☹ Breaks applications

Let’s watch the data - II

CSSE

☺ Track each character of data

☺ Ensure the data is safely

☹ Safety is context-dependant

☹ Modification for all operations

☹ Performance?

Let’s watch the input & learn

Runtime detection

☺ No need to study application

☺ No need to study context

☹ Complex heuristics

☹ Needs data collection

?