<!-- Slide number: 1 -->
# Living at 21 programmers’ st.Pitfalls in code review
Fady Mphamed Osman
Smartec-Group
Fady.mohamed.osman@gmail.com
12/4/2014

### Notes:

<!-- Slide number: 2 -->
About Me
Software Engineer @ smartec group.
Both embedded and desktop software.
Exploit Writer.
Metasploit Contributor.
Addicted to code reading.

![Smartec logo](Picture4.jpg)

<!-- Slide number: 3 -->
# Agenda
Abstract.
Pitfalls in technology
Third party libraries.
Platforms.
Programming Language, DBMS... etc.
Pitfalls in coding.
Trusting/Missing some inputs.
Unfamous bug classes.
Bad mitigation techniques.
Good Practices.
Exploiting opensource rocks.

<!-- Slide number: 4 -->
# Abstract
What this presentation is NOT about?
Not about SDLC code review.
This presentation shows only some examples.
Some other dangerous issues are not discussed here and you have to do some digging by yourself.
The code samples here provides the very simple case. Real life situations tend to be more complex.

<!-- Slide number: 5 -->
# Tech. Pitfalls : Third party libsTimthumb.php Zero Day
Zero day in timbthumb.php was discovered in 2011.
Many wordpress themes used the script and many of them was vulnerable.

<!-- Slide number: 6 -->
# Tech. Pitfalls : Third party libsuploadify.php
Uploadify.php is a script used to make file upload easier.
The script is vulnerable by nature just put it in your script and it's vulnerable. As mentioned in thier website.

A lot of opensource projects and thier plugins are using it.

![Uploadify](Picture4.jpg)

<!-- Slide number: 7 -->
# Tech. Pitfalls : Third party libsuploadify.php
You will be amazed by what you can find by using a code search engine to search for "uploadify.php"

<!-- Slide number: 8 -->
# Tech. Pitfalls : Platform
When writing a plugin for a certain platform you have to be carefull.
You can't take input from these platforms as trusted.
Some of the inputs provided by these platforms by the database or the api are not filtered.
See the following examples of wordpress database.

<!-- Slide number: 9 -->
# Tech. Pitfalls : PlatformWordpress Example 1
Wordpress doesn't filter comment agent and stores it as it in the data base.

![CommentAgent_xss](Picture4.jpg)

<!-- Slide number: 10 -->
# Tech. Pitfalls : PlatformWordpress Example 2
It also doesn't filter the metadata from images.
As a matter of fact the first plugin I tested was vulnerable to xss because of that.

![WpImageMeta_xss](Picture4.jpg)

<!-- Slide number: 11 -->
# Tech. Pitfalls : Language(php)
Language weakness.
Weak typing.
Object Injection.
Language Misconfigurations.
register_globals.
Some configurations causing info. leak.

<!-- Slide number: 12 -->
# Tech. Pitfalls : Language(php)Weakness : weak typing
PHP is weakly typed, which means that it will automatically convert data of an incorrect type into the expected type.
Imagine the following code.

![Type_Juggling](Picture4.jpg)

<!-- Slide number: 13 -->
# Tech. Pitfalls : Language(php)Weakness : weak typing

![Type_Juggling](Picture4.jpg)
According to php documentation strcasecmp returns < 0 if str1 is less than str2 > 0 if str1 is greater than str2, and 0 if they are equal.
So it looks it will only equal 0 if the pass variable is equal to "mypass"

<!-- Slide number: 14 -->
# Tech. Pitfalls : Language(php)Weakness : weak typing
So the url : http://127.0.0.1/fady/Owasp_Session/type_juggling.php?pass=mypass
Will work and any other password will not work unless..
You added two brackets after pass i.e. the url will be:
http://127.0.0.1/fady/Owasp_Session/type_juggling.php?pass[]=
But why??

<!-- Slide number: 15 -->
# Tech. Pitfalls : Language(php)Weakness : weak typing
The answer is type juggling.
By passing the pass variable as an array the function "strcasecmp" will fail because it accepts only strings and will return NULL.
Because of weak typing of php null is actually a zero so it will pass.
The fix is actually straight forward can you spot the difference betweem vulnerable and not vulnerable code in the following slide.

<!-- Slide number: 16 -->
# Tech. Pitfalls : Language(php)Weakness : weak typing
Vulnerable.

Fixed.

![Type_Juggling](Picture4.jpg)

![Type_Juggling_fix](Picture5.jpg)

<!-- Slide number: 17 -->
# Tech. Pitfalls : Language(php)Weakness : Object Injection
It allows the attacker to perform different kinds of malicious attacks, such as Code Injection, SQL Injection, Path Traversal and Application Denial of Service, depending on the context.
First lets see how serialize works and then we can continue from there to explain object injection.

<!-- Slide number: 18 -->
# Tech. Pitfalls : Language(php)Weakness : Object Injection
By serializing an object we mean converting that object to a string somehow to make it easier to save to a file or to transfer it through network.
By unserializing we convert that string back to that object.
The problem is that when unserializing an object one of the php magic functions is called may be __tostring() __wakeup() or __destruct() or another one according to the program flow.

<!-- Slide number: 19 -->
# Tech. Pitfalls : Language(php)Weakness : Object Injection
The following example (from owasp.org):

http://testsite.com/vuln.php?data=O:8:"Example1":1:{s:10:"cache_file";s:15:"../../index.php";}

![unserialize](Picture4.jpg)

<!-- Slide number: 20 -->
# Tech. Pitfalls : Language(php)MisConfig.:Register Globals
Register gloabals allows the user to set global variables through url.
Went from on to off from php 4.2.0 and deprecated as of PHP 5.3.0 and removed as of PHP 5.4.0.

Exploit : auth.php?authorized=1 since authorized variable wasn't initialized.

![register_globals](Picture4.jpg)

<!-- Slide number: 21 -->
# Tech. Pitfalls : Language(php)MisConfig.:Info leak
expose_php : adds the php signature to server header.
display_errors will display the errors to the user.
Session directory should not be set to world readable directory.
A very nice script to check php misconfigurations can be found at : http://phpsecinfo.com/

<!-- Slide number: 22 -->
# Coding Pitfalls: Inputs
User agent string.
Referrer.
Ip adress through HTTP_X_FORWARD_FOR or any headers that can be set by user.
Metadata from local files EXIF headers from images and so on.

<!-- Slide number: 23 -->
# Coding Pitfalls: Unfamous Vulnerabilits
Object injection (Discussed before).
Json hijacking.
Function injection.

<!-- Slide number: 24 -->
# Coding PitfallsBad Mitigation
Relying on some language defences for example "mysql_escape_string" and "mysql_real_escape_string which in some cases can be bypassed based on server configuration.
Bad filteration : forgetting about null injections and other bad chars.
Black listing in file uploads.

<!-- Slide number: 25 -->
# Good practices
Disable what you don't use for example if you're not allowing users to upload files disable it.
Disable configurations per folders when not needed for example disable script execution in the uploads folder using .htaccess file.

<!-- Slide number: 26 -->
# Why exploiting open source rocks??
You can add code to test your scenarios for example print something when a certain execution path is used.
You can reuse open source exploits by simply replacing some strings and paths here and there and get a working exploits.
You can easily examine closly the execution paths to find bugs.

<!-- Slide number: 27 -->
# Q & A