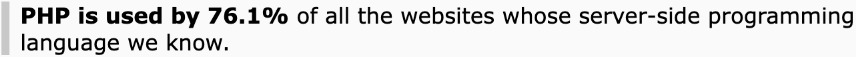
Atropos: Effective Fuzzing of Web Applications for Server-Side Vulnerabilities

Emre Güler[*], Sergej Schumilo[*], Moritz Schloegel[†], Nils Bars[†], Philipp Görz[†], Xinyi Xu[†], Cemal Kaygusuz[*], and Thorsten Holz[†]

[*] Chair for Systems Security, Ruhr University Bochum
[†] CISPA Helmholtz Center for Information Security

Motivation

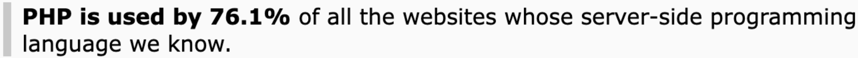
Why (PHP) Web Applications?



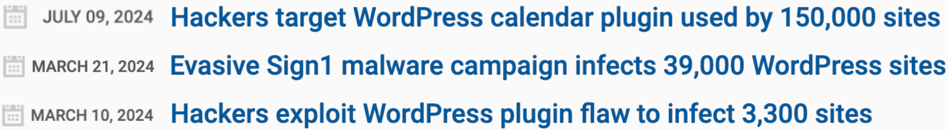
> **PHP is used by 76.1%** of all the websites whose server-side programming language we know.

Source: Usage statistics of PHP for websites

Why (PHP) Web Applications?



> **PHP is used by 76.1%** of all the websites whose server-side programming language we know.



JULY 09, 2024 Hackers target WordPress calendar plugin used by 150,000 sites

MARCH 21, 2024 Evasive Sign1 malware campaign infects 39,000 WordPress sites

MARCH 10, 2024 Hackers exploit WordPress plugin flaw to infect 3,300 sites

Source: Usage statistics of PHP for websites

Source: BleepingComputer - Latest WordPress news

State-of-the-art

- Static analysis usually suffers from FPs

State-of-the-art

- Static analysis usually suffers from FPs
- ... and provides no test cases for debugging

State-of-the-art

- Static analysis usually suffers from FPs
- … and provides no test cases for debugging
- Fuzzing has been applied across diverse applications

Fundamentals of Fuzzing

✎ Mutation

Fundamentals of Fuzzing

✎ Mutation

▶ Execution

Fundamentals of Fuzzing

- Mutation
- Execution
- Bug Oracle

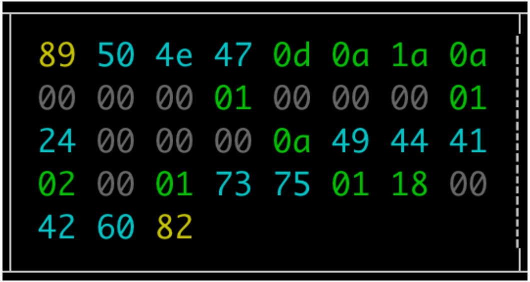
Fundamentals of Fuzzing

- Mutation
- Execution
- Bug Oracle
- Feedback

Fuzzing of PHP Applications

Input Space

Binary



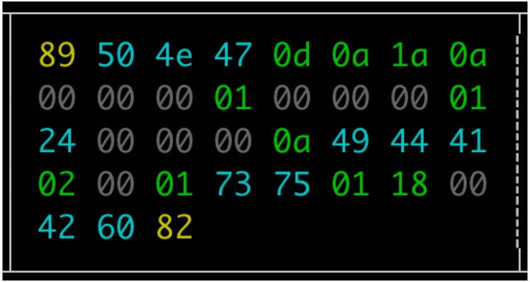
```
89 50 4e 47 0d 0a 1a 0a
00 00 00 01 00 00 00 01
24 00 00 00 0a 49 44 41
02 00 01 73 75 01 18 00
42 60 82
```

PHP

/index.php?page=login

Input Space

Binary



```
89 50 4e 47 0d 0a 1a 0a
00 00 00 01 00 00 00 01
24 00 00 00 0a 49 44 41
02 00 01 73 75 01 18 00
42 60 82
```

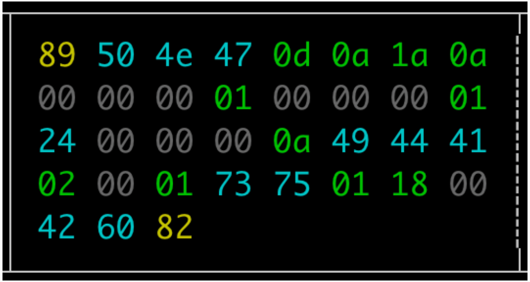
PHP

target

/ index.php ?page=login

Input Space

Binary



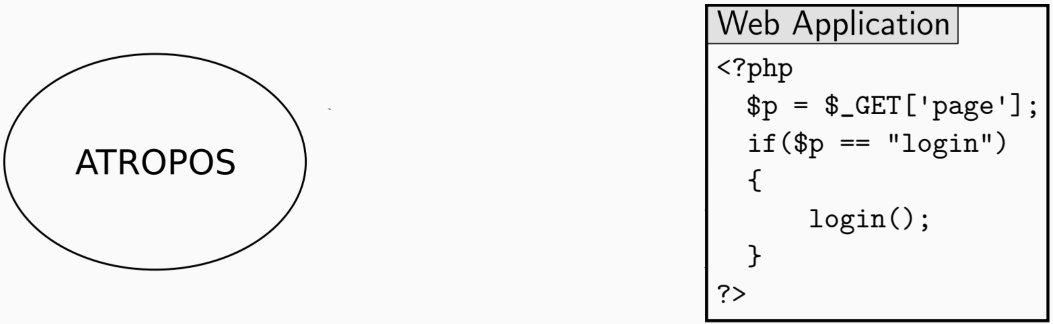
```
89 50 4e 47 0d 0a 1a 0a
00 00 00 01 00 00 00 01
24 00 00 00 0a 49 44 41
02 00 01 73 75 01 18 00
42 60 82
```

PHP

target key value

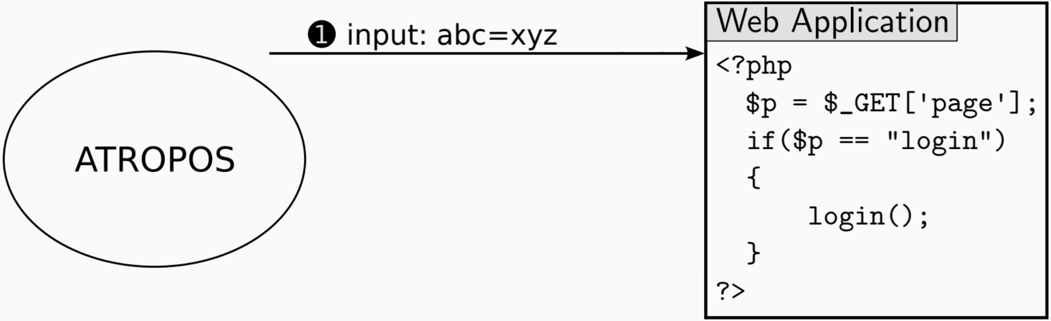
/index.php?page=login

Key-Value Pair Feedback



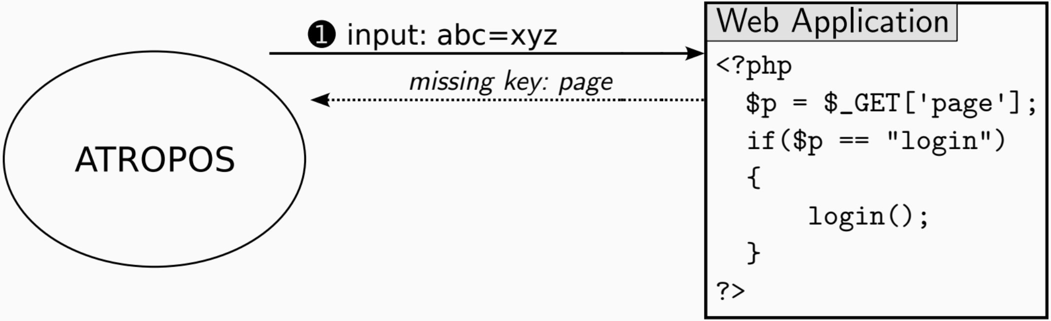
```
<?php
  $p = $_GET['page'];
  if($p == "login")
  {
      login();
  }
?>
```

Key-Value Pair Feedback



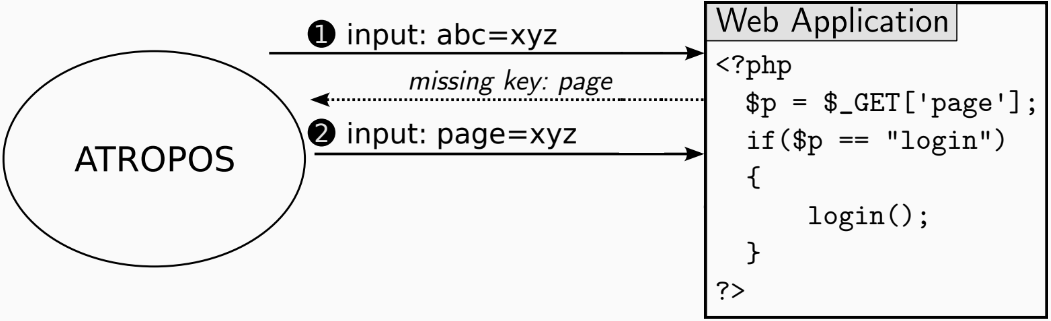
```
<?php
  $p = $_GET['page'];
  if($p == "login")
  {
      login();
  }
?>
```

Key-Value Pair Feedback



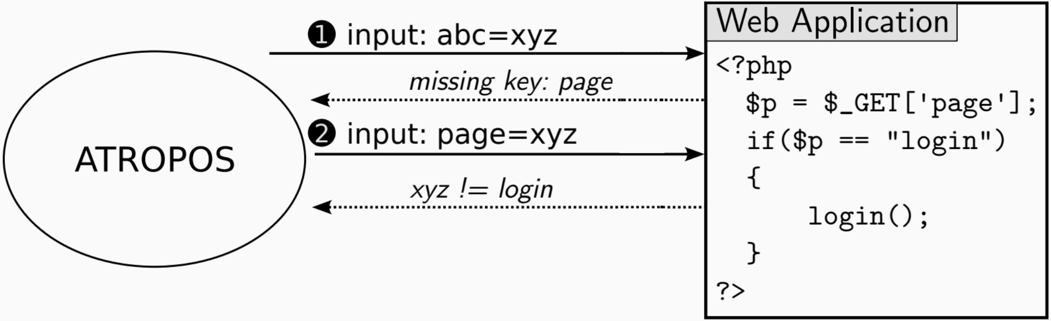
```
<?php
  $p = $_GET['page'];
  if($p == "login")
  {
      login();
  }
?>
```

Key-Value Pair Feedback



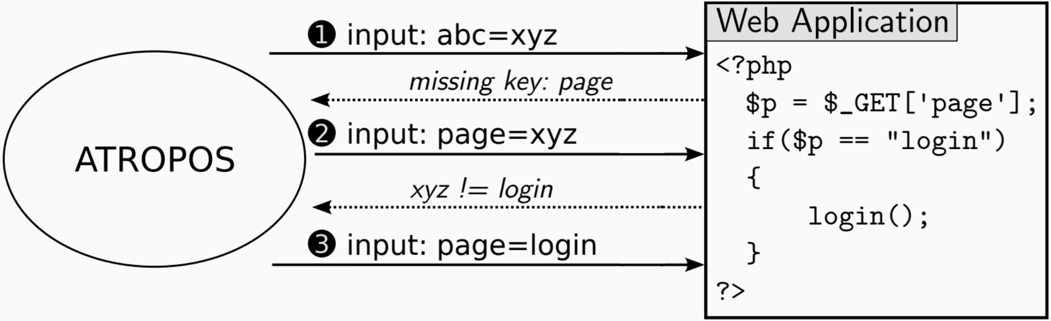
```
<?php
  $p = $_GET['page'];
  if($p == "login")
  {
      login();
  }
?>
```

Key-Value Pair Feedback



```
<?php
  $p = $_GET['page'];
  if($p == "login")
  {
      login();
  }
?>
```

Key-Value Pair Feedback



```
<?php
  $p = $_GET['page'];
  if($p == "login")
  {
      login();
  }
?>
```

Custom Bug Oracles

Custom Bug Oracle

- Many sensitive sinks requires structured inputs
  - mysqli_query
  - unserialize
  - eval
  - shell_exec

Custom Bug Oracle

- Many sensitive sinks requires structured inputs
  - mysqli_query
  - unserialize
  - eval
  - shell_exec

- Random inputs are unlikely to adhere to this syntax

Custom Bug Oracle

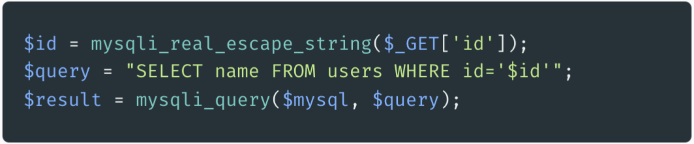
- Many sensitive sinks requires structured inputs
  - mysqli_query
  - unserialize
  - eval
  - shell_exec

- Random inputs are unlikely to adhere to this syntax

⚠ Fuzzer causes syntax errors if input is unsanitized

SQL Injection Example

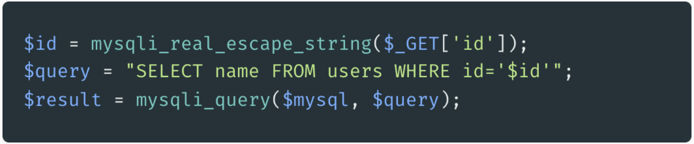
(a)



```php
$id = mysqli_real_escape_string($_GET['id']);
$query = "SELECT name FROM users WHERE id='$id'";
$result = mysqli_query($mysql, $query);
```

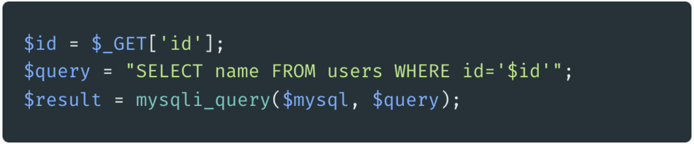
SQL Injection Example

(a)



```php
$id = mysqli_real_escape_string($_GET['id']);
$query = "SELECT name FROM users WHERE id='$id'";
$result = mysqli_query($mysql, $query);
```

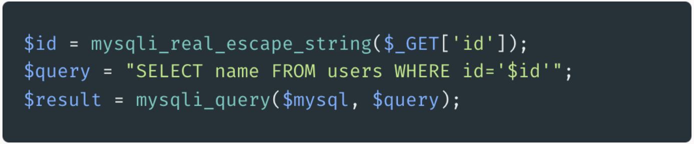
(b)



```php
$id = $_GET['id'];
$query = "SELECT name FROM users WHERE id='$id'";
$result = mysqli_query($mysql, $query);
```

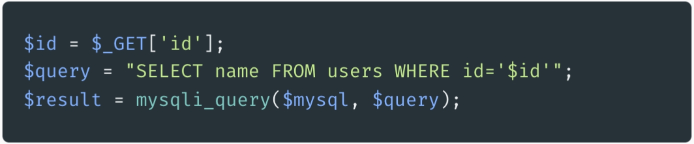
SQL Injection Example

(a)



```php
$id = mysqli_real_escape_string($_GET['id']);
$query = "SELECT name FROM users WHERE id='$id'";
$result = mysqli_query($mysql, $query);
```

(b)

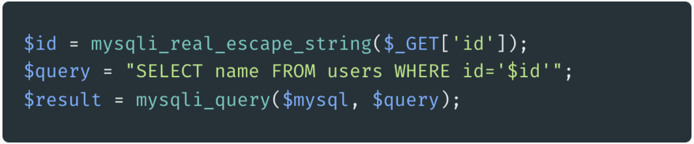


```php
$id = $_GET['id'];
$query = "SELECT name FROM users WHERE id='$id'";
$result = mysqli_query($mysql, $query);
```

input: a§!q5’2c

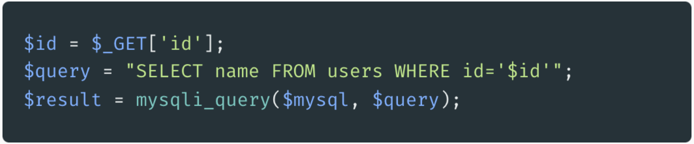
SQL Injection Example

(a)



```php
$id = mysqli_real_escape_string($_GET['id']);
$query = "SELECT name FROM users WHERE id='$id'";
$result = mysqli_query($mysql, $query);
```

(b)



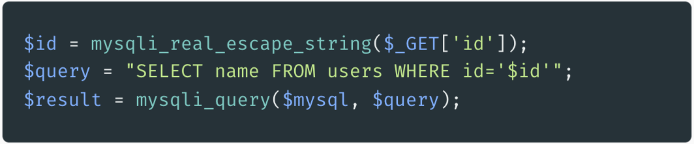
```php
$id = $_GET['id'];
$query = "SELECT name FROM users WHERE id='$id'";
$result = mysqli_query($mysql, $query);
```

input: a§!q5’2c

(a) no error

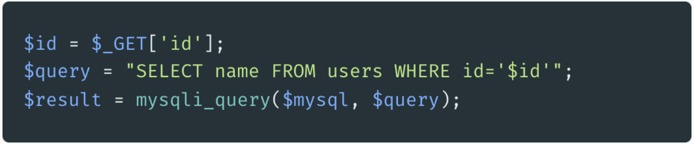
SQL Injection Example

(a)



```php
$id = mysqli_real_escape_string($_GET['id']);
$query = "SELECT name FROM users WHERE id='$id'";
$result = mysqli_query($mysql, $query);
```

(b)



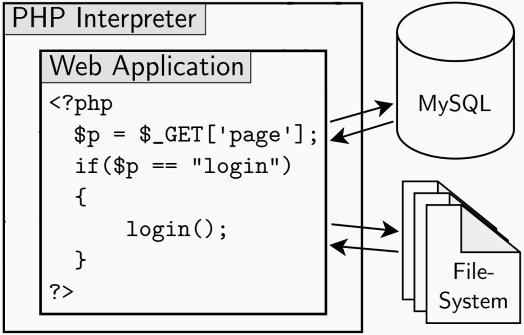
```php
$id = $_GET['id'];
$query = "SELECT name FROM users WHERE id='$id'";
$result = mysqli_query($mysql, $query);
```

input: a§!q5’2c

(a) no error

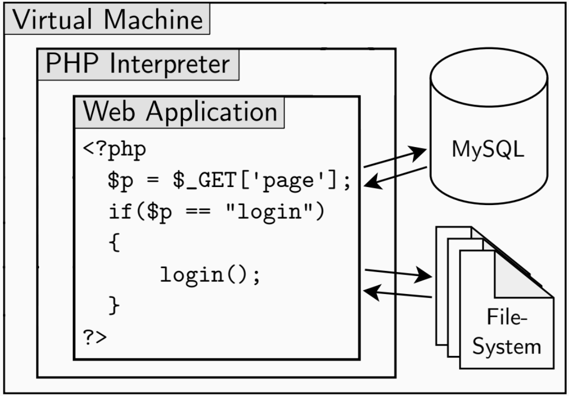
(b) “You have an error in your SQL syntax;”

Design



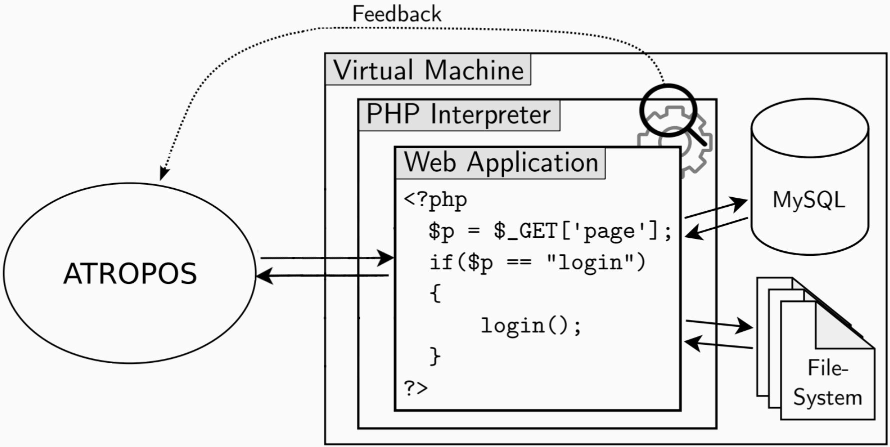
```
<?php
  $p = $_GET['page'];
  if($p == "login")
  {
      login();
  }
?>
```

Design



```
<?php
  $p = $_GET['page'];
  if($p == "login")
  {
        login();
  }
?>
```

Design



```
<?php
  $p = $_GET['page'];
  if($p == "login")
  {
      login();
  }
?>
```

Evaluation

Bug Finding (DVWA, XVWA, bWAPP)

Tool	Bugs found	False Positives
Atropos (40 cores)	**94% (49 / 52)**	**0**
Atropos (1 core)	75% (39 / 52)	**0**
SonarQube	56% (29 / 52)	**0**
Progpilot	65% (34 / 52)	7
Psalm	67% (35 / 52)	6
PHPCS-Security-Audit	71% (37 / 52)	59

There were 52 bugs in total in scope for our paper.

Code Coverage & Real-World Bugs

- Code coverage: +46% coverage vs. second-best tool
- Real-world bugs: seven new vulnerabilities discovered

Vulnerability	Web App	
PHP Object Injection	AltoCMS	
PHP Object Injection	MaxSite	
PHP Object Injection	phpwcms	
Server-Side Request-Forgery	InvoiceNinja	
Server-Side Request-Forgery	Iubenda	
Server-Side Request-Forgery	NextCloud	CVE-2022-31132
Remote Code Execution	lodel	
SQL Injection	lodel	

Questions?

Contact me

- *emrexgueler*
- ✉ *emre.gueler@rub.de*
- 🌐 *www.emre.re*

Paper

Icons by FontAwesome