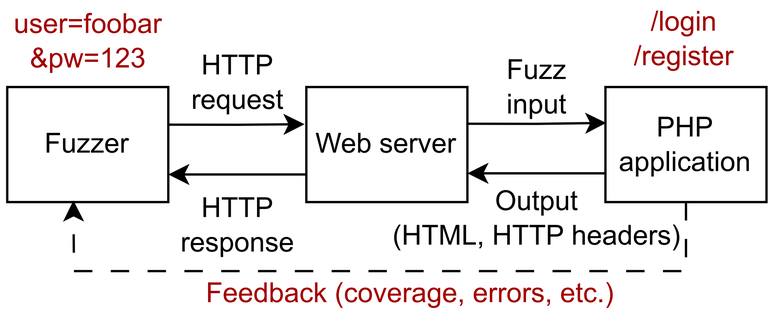
What All the PHUZZ Is About:

A Coverage-guided Fuzzer for Finding Vulnerabilities in PHP Web Applications

Sebastian Neef, Lorenz Kleissner, Jean-Pierre Seifert
SecT, Technische Universität Berlin, Germany

@ CSAW'24 - Grenoble INP/ESISAR, Valence, France - 08.11.2024

Motivation



- Fuzzing ≙ Automated software testing
- Extensively researched for binary software
- Little research on fuzzing web applications

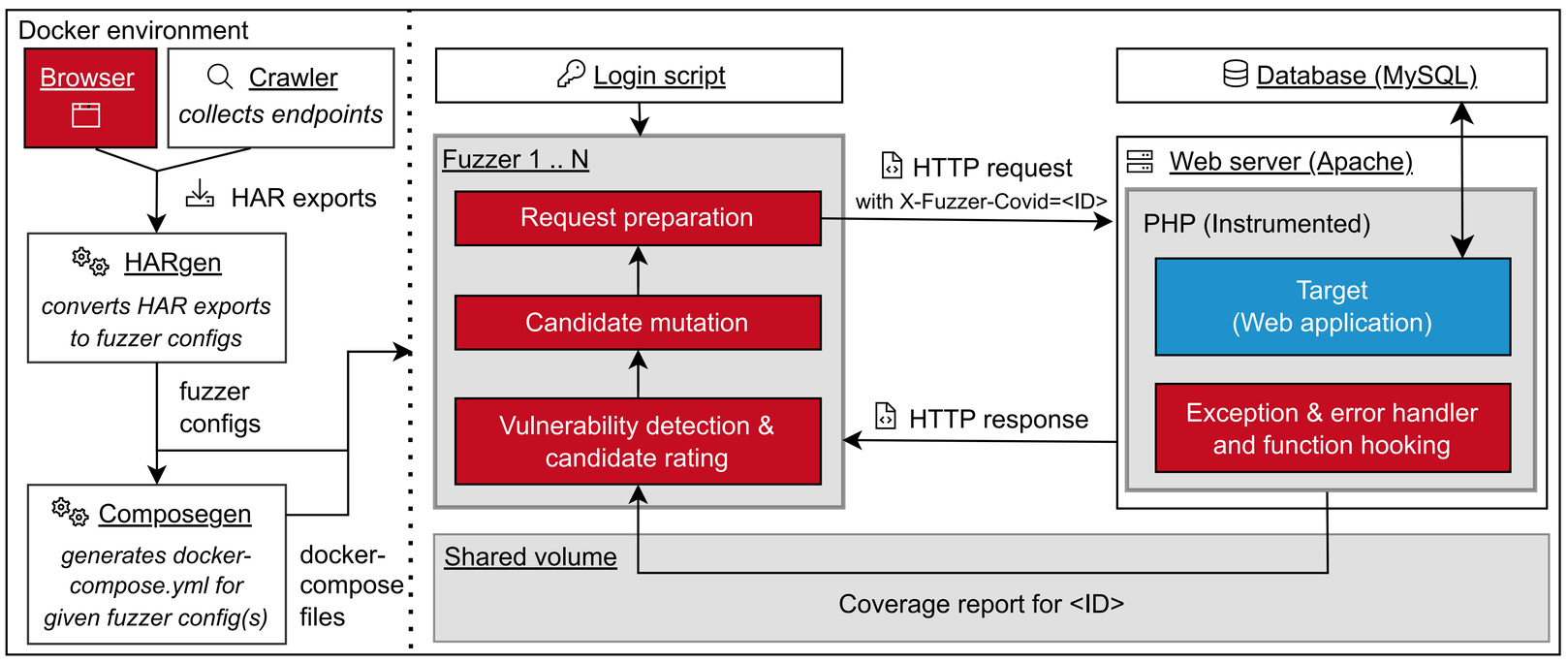
Challenges

1. **How to find endpoints and parameters?**
   - Comprehensive fuzzing of the application?
2. **How to mutate the input?**
   - Adherence to HTTP protocol?
   - Correct parameter to endpoint mapping?
3. **How to collect coverage & feedback?**
   - Instrumentation of high-level PHP code?
   - Coverage without shared memory?
4. **How to detect web vulnerabilities?**
   - Alternatives to missing *"crash"* indicator?

Contributions

- **PHUZZ** framework for fuzzing PHP web apps
- Browser-based approach for more control over fuzzed endpoints and parameters
- Simplified instrumentation without code changes to fuzz target or PHP interpreter
- Vulnerability detection that supports SQLi*, RCE*, XXE*, insecure deserialization*, path traversal*, open redirection†, XSS†
- Support for parallel fuzzing

Implementation & Components of PHUZZ



- Dockerized & modular design for enhanced usability and extensibility

Instrumentation

```php
uopz_set_return('func', function($args) {
 try {
  $ret = func($args);
 } catch($e) { /* Exception handling */ }
 // Log function, args, errors and exceptions
 return $ret;
}, true);
```

Listing 2: Instrumenting (non-)native PHP functions with UOPZ.

- Transparent function hooking of (non-)native PHP functions using UOPZ extension
- Transparent coverage collection using pcov or Xdebug extensions
- Observe dangerous functions, triggered errors and exceptions
- Increase coverage by overriding and disabling authentication and authorization functions

PHUZZ Outperforms Other Vulnerability Scanners

Table 1: Evaluation of PHUZZ and other vulnerability scanners against web applications with known vulnerabilities

Web application	#vuln.	PHUZZ	BurpSuite	ZAP	Wapiti	WFuzz
bWAPP	30	100%	93%	77%	70%	0%
DVWA	18	100%	83%	67%	78%	0%
XVWA	10	100%	60%	50%	50%	0%
WackoPicko	7	100%	71%	86%	71%	0%
WP Plugins	22	95%	50%	36%	41%	0%
Total	87	99%	75%	62%	62%	0%
***: Server-side**	48	98%	58%	52%	44%	0%
†: Client-side	39	100%	95%	74%	85%	0%

PHUZZ Discovers Zero-day Vulnerabilities

Table 3: Findings generated by fuzzing popular WordPress plugins for unknown (0-day) vulnerabilities

Vuln. class	PHUZZ (Plugins/APIs/Valid)	BurpSuite Pro (Plugins/APIs/Valid)
XSS	14 / 24 / 7	9 / 16 / 8
PaTr	20 (47) / 37 (110) / 16	1 / 1 / 1
SQLi	6 / 9 / 0	0 / 0 / 0
OpRe	1 / 1 / 1	1 / 1 / 1

- **Proof of concepts for two exploitable vulnerabilities**:
  - → CVE-2023-6294: popup-builder SSRF & Arbitrary File Read
  - → CVE-2023-6295: so-widgets-bundle Local File Inclusion

Conclusions

- Coverage-guided fuzzing presents a powerful method for uncovering web vulnerabilities.
- The PHUZZ framework effectively identifies seven distinct vulnerability classes, all without altering the web application's code, PHP interpreter, or external components.
- PHUZZ has proven its real-world effectiveness by uncovering previously unknown vulnerabilities.
- To support ongoing research, we have made all related code and data publicly available.
- We encourage future research to build upon and expand PHUZZ, advancing this field further.

More Information

Link to full paper

Link to repository

Contact: neef@tu-berlin.de