Bringing UFUs Back into the Air With FUEL: A Framework for Evaluating the Effectiveness of Unrestricted File Upload Vulnerability Scanners

DIMVA 2024 - 18.07.2024 - Lausanne, Switzerland

Sebastian Neef & Maath Oudeh

Security in Telecommunications
TU Berlin, Germany

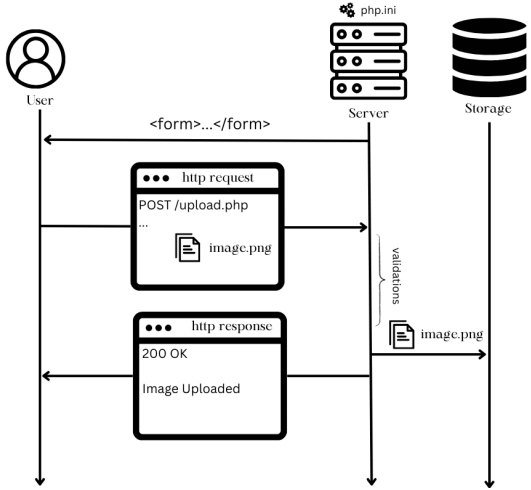
Important: UFO != UFU

Unidentified Flying Object (UFO) vs. Unrestricted File Upload (UFU)

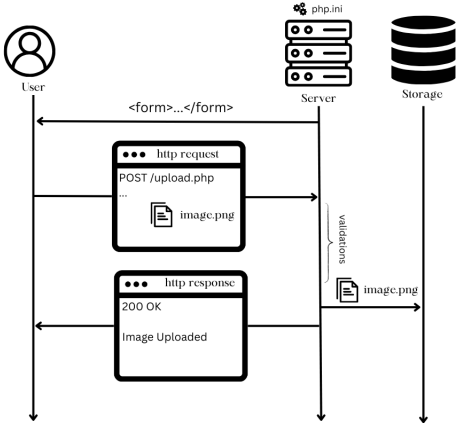
 [1]

[1] Generated by Microsoft Copilot/Dall-E

Recap: ~~Unrestricted~~ File Upload

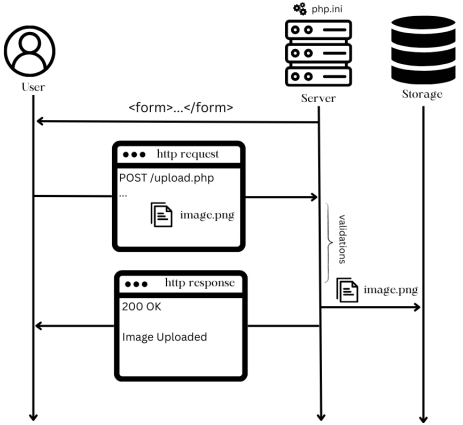


Recap: ~~Unrestricted~~ File Upload



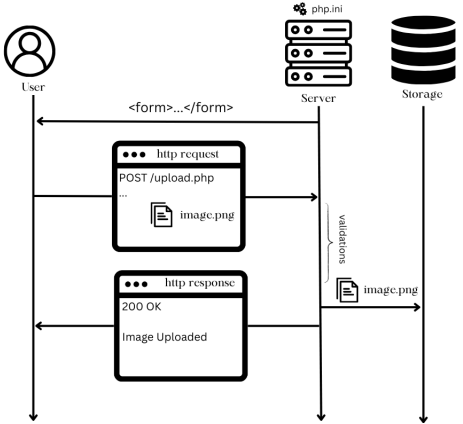

```
POST /upload.php HTTP/1.1
Host: localhost:10001
Content-Length: 848
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarysgK7MyQ8G
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) [...]
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,[...]
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close

------WebKitFormBoundarysgK7MyQ8GyWSeA2e
Content-Disposition: form-data; name="file"; filename="image.png"
Content-Type: image/png

<(Binary) file contents>

------WebKitFormBoundarysgK7MyQ8GyWSeA2e
Content-Disposition: form-data; name="submit"

submit
------WebKitFormBoundarysgK7MyQ8GyWSeA2e--
```

Recap: ~~Unrestricted~~ File Upload




```
POST /upload.php HTTP/1.1
Host: localhost:10001
Content-Length: 848
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarysgK7MyQ8G
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) [...]
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,[...]
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close

------WebKitFormBoundarysgK7MyQ8GyWSeA2e
Content-Disposition: form-data; name="file"; filename="image.png"
Content-Type: image/png

<(Binary) file contents>

------WebKitFormBoundarysgK7MyQ8GyWSeA2e
Content-Disposition: form-data; name="submit"

submit
------WebKitFormBoundarysgK7MyQ8GyWSeA2e--
```

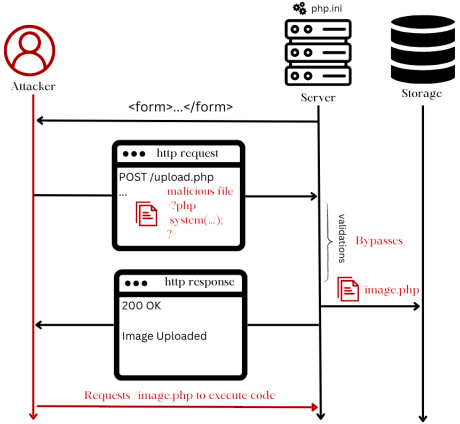
Recap: ~~Unrestricted~~ File Upload




```
POST /upload.php HTTP/1.1
Host: localhost:10001
Content-Length: 848
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarysgK7MyQ8G
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) [...]
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,[...]
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close

------WebKitFormBoundarysgK7MyQ8GyWSeA2e
Content-Disposition: form-data; name="file"; filename="image.png"
Content-Type: image/png

<(Binary) file contents>

------WebKitFormBoundarysgK7MyQ8GyWSeA2e
Content-Disposition: form-data; name="submit"

submit
------WebKitFormBoundarysgK7MyQ8GyWSeA2e--
```

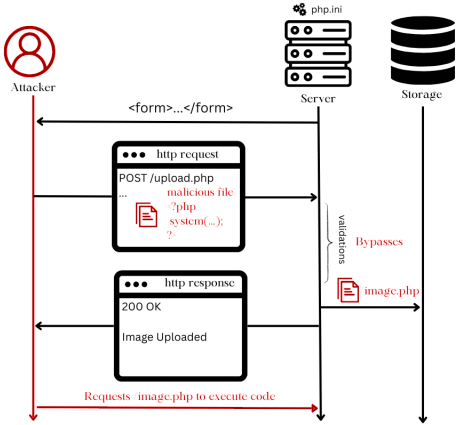
- $\Rightarrow$ Web applications have to validate the uploaded file's properties!

Recap: **Unrestricted** File Upload




```
POST /upload.php HTTP/1.1
Host: localhost:10001
Content-Length: 848
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarysgK7MyQ8G
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) [...]
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,[...]
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close

------WebKitFormBoundarysgK7MyQ8GyWSeA2e
Content-Disposition: form-data; name="file"; filename="image.php"
Content-Type: image/png

<(Binary) file contents>

------WebKitFormBoundarysgK7MyQ8GyWSeA2e
Content-Disposition: form-data; name="submit"

submit
------WebKitFormBoundarysgK7MyQ8GyWSeA2e--
```

Recap: **Unrestricted** File Upload




```
POST /upload.php HTTP/1.1
Host: localhost:10001
Content-Length: 848
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarysgK7MyQ8G
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) [...]
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,[...]
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close

------WebKitFormBoundarysgK7MyQ8GyWSeA2e
Content-Disposition: form-data; name="file"; filename="image.php"
Content-Type: image/png

<(Binary) file contents>

------WebKitFormBoundarysgK7MyQ8GyWSeA2e
Content-Disposition: form-data; name="submit"

submit
------WebKitFormBoundarysgK7MyQ8GyWSeA2e--
```

- $\Rightarrow$ Improper validation of uploaded files can lead to UFUs!
- $\Rightarrow$ UFUs can lead to a web application's compromise (e.g., RCE, XSS)

Related Work

- 2009/2011: First mentions of UFU vulnerabilities by Barth et al. and Barua et al.
- 2014: 6 PHP projects used by Dahse et al.
- 2016: Pooj et al. documented 16 different UFU variants
- 2016: osCommerce used by Riadi et al.
- 2017: DVWA used by De Meo et al.
- 2019: 9,000 WordPress plugins used by Huang et al. for *UChecker*
- 2020: 33 real-world PHP applications used by Lee et al. for *FUSE*
- 2021: WordPress plugins used (again) by Huang et al. for *UFuzzer*
- 2022: 2 custom web applications used by Yenduri et al.
- 2022: 4 CMSes used by Wichmann et al. for *FileUploadChecker*
- 2023: 18 PHP web applications used by Chen et al. for *URadar*

Related Work

- 2009/2011: First mentions of UFU vulnerabilities by Barth et al. and Barua et al.
- 2014: 6 PHP projects used by Dahse et al.
- 2016: Pooj et al. documented 16 different UFU variants
- 2016: osCommerce used by Riadi et al.
- 2017: DVWA used by De Meo et al.
- 2019: 9,000 WordPress plugins used by Huang et al. for *UChecker*
- 2020: 33 real-world PHP applications used by Lee et al. for *FUSE*
- 2021: WordPress plugins used (again) by Huang et al. for *UFuzzer*
- 2022: 2 custom web applications used by Yenduri et al.
- 2022: 4 CMSes used by Wichmann et al. for *FileUploadChecker*
- 2023: 18 PHP web applications used by Chen et al. for *URadar*

$\rightarrow$ Active research on UFU vulnerabilities

$\Rightarrow$ But: Different sets of web applications used for evaluation

Related Work

- 2009/2011: First mentions of UFU vulnerabilities by Barth et al. and Barua et al.
- 2014: 6 PHP projects used by Dahse et al.
- **2016: Pooj et al. documented 16 different UFU variants**
- 2016: osCommerce used by Riadi et al.
- 2017: DVWA used by De Meo et al.
- 2019: 9,000 WordPress plugins used by Huang et al. for *UChecker*
- 2020: 33 real-world PHP applications used by Lee et al. for *FUSE*
- 2021: WordPress plugins used (again) by Huang et al. for *UFuzzer*
- 2022: 2 custom web applications used by Yenduri et al.
- 2022: 4 CMSes used by Wichmann et al. for *FileUploadChecker*
- 2023: 18 PHP web applications used by Chen et al. for *URadar*

$\rightarrow$ Active research on UFU vulnerabilities

$\Rightarrow$ But: Different sets of web applications used for evaluation

Motivation

- How can we make the (new) approaches/tools comparable and the results reproducible?

Motivation

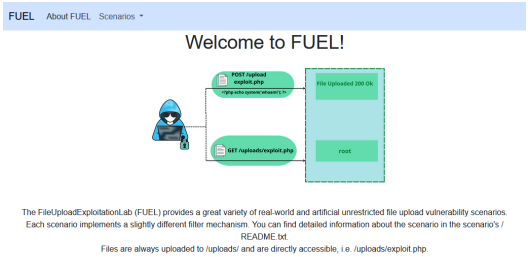
- How can we make the (new) approaches/tools comparable and the results reproducible?

- How can we ensure that all UFU variants are being considered?

Motivation

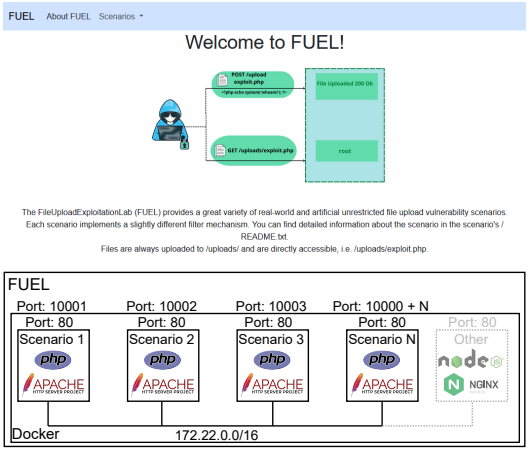
- How can we make the (new) approaches/tools comparable and the results reproducible?
- How can we ensure that all UFU variants are being considered?
- How do SOTA vulnerability scanners perform in detecting UFUs?

FUEL - File Upload Exploitation Lab



- Goals:
  - Common ground for UFU evaluations
  - Simple, extensive & extensible framework
  - Open source @ `github.com/FUEL-Project`

FUEL - File Upload Exploitation Lab



- Goals:
  - Common ground for UFU evaluations
  - Simple, extensive & extensible framework
  - Open source @ `github.com/FUEL-Project`
- 15 PHP-based UFU scenarios related to
  - File extension
  - File content
  - File name
  - Server configuration
- Each scenario has its own Docker container
  - Language & configuration agnostic

FUEL: 15 UFU Scenarios

- Baseline
  - S1: No validation
  - S2: Client-side validation

FUEL: 15 UFU Scenarios

- Baseline
  - S1: No validation
  - S2: Client-side validation

- File extension
  - S4: Alternatives (`.php4`)
  - S5: Capitalization (`.pHp`)
  - S6: Consecutive (`.php.png`)
  - S8: Nesting (`.p.phphp`)

FUEL: 15 UFU Scenarios

- Baseline
  - S1: No validation
  - S2: Client-side validation
- File extension
  - S4: Alternatives (`.php4`)
  - S5: Capitalization (`.pHp`)
  - S6: Consecutive (`.php.png`)
  - S8: Nesting (`.p.phphp`)
- File content
  - S9: Magic numbers (`GIF87a<?php ...`)
  - S10: Code in image (`EXIF metadata`)

FUEL: 15 UFU Scenarios

- Baseline
  - S1: No validation
  - S2: Client-side validation
- File extension
  - S4: Alternatives (`.php4`)
  - S5: Capitalization (`.pHp`)
  - S6: Consecutive (`.php.png`)
  - S8: Nesting (`.p.phphp`)
- File content
  - S9: Magic numbers (`GIF87a<?php ...`)
  - S10: Code in image (`EXIF metadata`)
- File name
  - S11: Path traversal (`../file.php`)
  - S12: Special chars (`file.php%00.png`)
  - S13: XSS (`<img src=x onerror=alert(1)>.png`)

FUEL: 15 UFU Scenarios

- Baseline
  - S1: No validation
  - S2: Client-side validation
- File extension
  - S4: Alternatives (`.php4`)
  - S5: Capitalization (`.pHp`)
  - S6: Consecutive (`.php.png`)
  - S8: Nesting (`.p.phphp`)
- File content
  - S9: Magic numbers (`GIF87a<?php ...`)
  - S10: Code in image (`EXIF metadata`)
- File name
  - S11: Path traversal (`../file.php`)
  - S12: Special chars (`file.php%00.png`)
  - S13: XSS (`<img src=x onerror=alert(1)>.png`)
- Others
  - S3: Mime-type (`Content-Type: image/png`)
  - S7: Dot-files (`.htaccess`)
  - S14: Race condition
  - S15: Request method (`PUT`)

Evaluation: Vulnerability scanners

- Non-academic:
  - BurpSuite (Upload Scanner-Plugin)
  - OWASP ZAP (FileUpload-Addon)
  - Fuxploider

Evaluation: Vulnerability scanners

- Non-academic:
  - BurpSuite (Upload Scanner-Plugin)
  - OWASP ZAP (FileUpload-Addon)
  - Fuxploider
- Academic:
  - FUSE
  - (UChecker)
  - (UFuzzer)
  - (URadar)

Evaluation: Vulnerability scanners

- Non-academic:
  - BurpSuite (Upload Scanner-Plugin)
  - OWASP ZAP (FileUpload-Addon)
  - Fuxploider
- Academic:
  - FUSE
  - (UChecker)
  - (UFuzzer)
  - (URadar)
- Some graphical, some CLI-based
- Minimal (manual) configuration
  - Endpoint and parameters
  - File-upload HTTP request

Evaluation: Results

Table 1. Results from running each UFU scanner against `FUEL`'s scenarios.

Scanner	**FUSE**			**Fuxploider**			**ZAP**			**BurpSuite**		
Type	iFUB	CE	XSS	iFUB	CE	XSS	iFUB	CE	XSS	iFUB	CE	XSS
Total	8	7	11	8	8	-	9	8	11	9	8	12
S1	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S2	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S3	✓	✓	✓	✓	✓	-	✓[1]	✓[1]	✓[1]	✓	✓	✓[1]
S4	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S5	✓	✓	✓	✓	✓	-	✓	✓	✓	×	×	✓
S6	×	×	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S7	✓	×	✓	✓	✓	-	×	×	✓	×	×	✓
S8	×	×	✓	×	×	-	×	×	✓	×	×	✓
S9	✓	✓	✓	×	×	-	✓	✓	×	✓	✓	✓
S10	✓	✓	✓	×	×	-	×	×	×	✓	✓	✓
S11	×	×	✓	×	×	-	×	×	✓	×	×	✓
S12	×	×	×	✓	✓	-	✓[1]	✓[1]	✓[1]	✓	✓	×
S13	×	-	×	-	-	-	✓	-	✓	✓	-	✓
S14	T	T	T	×	×	-	T	T	T	T	T	T
S15	×	×	×	×	×	-	×	×	×	×	×	×

✓: Found, ✓[1]: Only for `fuel.png`, ×: Not found, T: Timeout after 300s

- Terminology
  - iFUB - intended File Upload Bypass
  - CE - Code Execution
  - XSS - Cross-Site Scripting

Evaluation: Results

Table 1. Results from running each UFU scanner against `FUEL`'s scenarios.

Scanner	FUSE			Fuxploider			ZAP			BurpSuite		
Type	iFUB	CE	XSS	iFUB	CE	XSS	iFUB	CE	XSS	iFUB	CE	XSS
Total	8	7	11	8	8	-	9	8	11	9	8	12
S1	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S2	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S3	✓	✓	✓	✓	✓	-	✓[1]	✓[1]	✓[1]	✓	✓	✓[1]
S4	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S5	✓	✓	✓	✓	✓	-	✓	✓	✓	×	×	✓
S6	×	×	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S7	✓	×	✓	✓	✓	-	×	×	✓	×	×	✓
S8	×	×	✓	×	×	-	×	×	✓	×	×	✓
S9	✓	✓	✓	×	×	-	✓	✓	×	✓	✓	✓
S10	✓	✓	✓	×	×	-	×	×	×	✓	✓	✓
S11	×	×	✓	×	×	-	×	×	✓	×	×	✓
S12	×	×	×	✓	✓	-	✓[1]	✓[1]	✓[1]	✓	✓	×
S13	×	-	×	-	-	-	✓	-	✓	✓	-	✓
S14	T	T	T	×	×	-	T	T	T	T	T	T
S15	×	×	×	×	×	-	×	×	×	×	×	×

✓: Found, ✓[1]: Only for `fuel.png`, ×: Not found, T: Timeout after 300s

- Terminology
  - iFUB - intended File Upload Bypass
  - CE - Code Execution
  - XSS - Cross-Site Scripting
- Results
  - No single scanner discovered all UFUs
  - Best coverage with at least 2 scanners!
  - BurpSuite and ZAP find most UFUs and XSS

Evaluation: Unidentified UFUs

Table 1. Results from running each UFU scanner against `FUEL`'s scenarios.

Scanner	FUSE			Fuxploider			ZAP			BurpSuite		
Type	iFUB	CE	XSS	iFUB	CE	XSS	iFUB	CE	XSS	iFUB	CE	XSS
Total	8	7	11	8	8	-	9	8	11	9	8	12
S1	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S2	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S3	✓	✓	✓	✓	✓	-	✓[1]	✓[1]	✓[1]	✓	✓	✓[1]
S4	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S5	✓	✓	✓	✓	✓	-	✓	✓	✓	×	×	✓
S6	×	×	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S7	✓	×	✓	✓	✓	-	×	×	✓	×	×	✓
S8	×	×	✓	×	×	-	×	×	✓	×	×	✓
S9	✓	✓	✓	×	×	-	✓	✓	×	✓	✓	✓
S10	✓	✓	✓	×	×	-	×	×	×	✓	✓	✓
S11	×	×	✓	×	×	-	×	×	✓	×	×	✓
S12	×	×	×	✓	✓	-	✓[1]	✓[1]	✓[1]	✓	✓	×
S13	×	-	×	-	-	-	✓	-	✓	✓	-	✓
S14	T	T	T	×	×	-	T	T	T	T	T	T
S15	×	×	×	×	×	-	×	×	×	×	×	×

✓: Found, ✓[1]: Only for `fuel.png`, ×: Not found, T: Timeout after 300s

- Unidentified scenarios:
  - S8 → Nested file extension (`.phPHPp`)
  - S11 → Filename path traversal
  - S14 → Race condition
  - S15 → PUT-based file upload

Evaluation: Interesting cases

Table 1. Results from running each UFU scanner against `FUEL`'s scenarios.

Scanner	FUSE			Fuxploider			ZAP			BurpSuite		
Type	iFUB	CE	XSS	iFUB	CE	XSS	iFUB	CE	XSS	iFUB	CE	XSS
Total	8	7	11	8	8	-	9	8	11	9	8	12
S1	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S2	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S3	✓	✓	✓	✓	✓	-	✓[1]	✓[1]	✓[1]	✓	✓	✓[1]
S4	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S5	✓	✓	✓	✓	✓	-	✓	✓	✓	×	×	✓
S6	×	×	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
S7	✓	×	✓	✓	✓	-	×	×	✓	×	×	✓
S8	×	×	✓	×	×	-	×	×	✓	×	×	✓
S9	✓	✓	✓	×	×	-	✓	✓	×	✓	✓	✓
S10	✓	✓	✓	×	×	-	×	×	×	✓	✓	✓
S11	×	×	✓	×	×	-	×	×	✓	×	×	✓
S12	×	×	×	✓	✓	-	✓[1]	✓[1]	✓[1]	✓	✓	×
S13	×	-	×	-	-	-	✓	-	✓	✓	-	✓
S14	T	T	T	×	×	-	T	T	T	T	T	T
S15	×	×	×	×	×	-	×	×	×	×	×	×

✓: Found, ✓[1]: Only for `fuel.png`, ×: Not found, T: Timeout after 300s

- Interesting cases:
  - S3 → Content-Type header manipulation
  - S12 → Nullbytes in filenames

Evaluation: Fuxploider-NG

Fuxploider		
iFUB	CE	XSS
8	8	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
×	×	-
×	×	-
×	×	-
×	×	-
✓	✓	-
-	-	-
×	×	-
×	×	-

- Can we improve the detection rates?

Evaluation: Fuxploider-NG

Fuxploider		
iFUB	CE	XSS
8	8	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
×	×	-
×	×	-
×	×	-
×	×	-
✓	✓	-
-	-	-
×	×	-
×	×	-

- We extended Fuxploider's UFU detection capabilities
- Additionally, we implemented basic XSS detection

Evaluation: Fuxploider-NG

Fuxploider		
iFUB	CE	XSS
8	8	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
✓	✓	-
×	×	-
×	×	-
×	×	-
×	×	-
✓	✓	-
-	-	-
×	×	-
×	×	-

- We extended Fuxploider's UFU detection capabilities
- Additionally, we implemented basic XSS detection

Table 2. Our improved *Fuxploider-NG* scanner successfully exploits scenarios 1 - 14.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
iFUB	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×
CE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	×
XSS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×

- Scenario 15 would have required a major rewrite

Limitations & Future work

- FUEL implements only PHP-based UFU scenarios
  - → Implement UFU scenarios in other languages

Limitations & Future work

- FUEL implements only PHP-based UFU scenarios
  - → Implement UFU scenarios in other languages

- FUEL implements *basic* UFU scenarios
  - → Combinations of basic scenarios/filters
  - → Implement UFU scenarios based on real-world applications

Limitations & Future work

- FUEL implements only PHP-based UFU scenarios
  - → Implement UFU scenarios in other languages

- FUEL implements *basic* UFU scenarios
  - → Combinations of basic scenarios/filters
  - → Implement UFU scenarios based on real-world applications

- Scanner performance analysis (speed vs. accuracy)
  - → Further evaluations with FUEL required

Conclusion

- We contribute:
  - FUEL - Extensible File Upload Exploitation Lab as a benchmark for vulnerability scanners
  - Fuxploider-NG - Updated scanner supporting 14/15 UFU variants
- We show:
  - Existing vulnerability scanners do not support all UFU variants
  - Users would need to use at least 2 SOTA scanners
- Let's strive for more reproducible and comparable science :)

Sebastian Neef
neef@tu-berlin.de

https://github.com/FUEL-Project