# FUSE: Finding File Upload Bugs via Penetration Testing

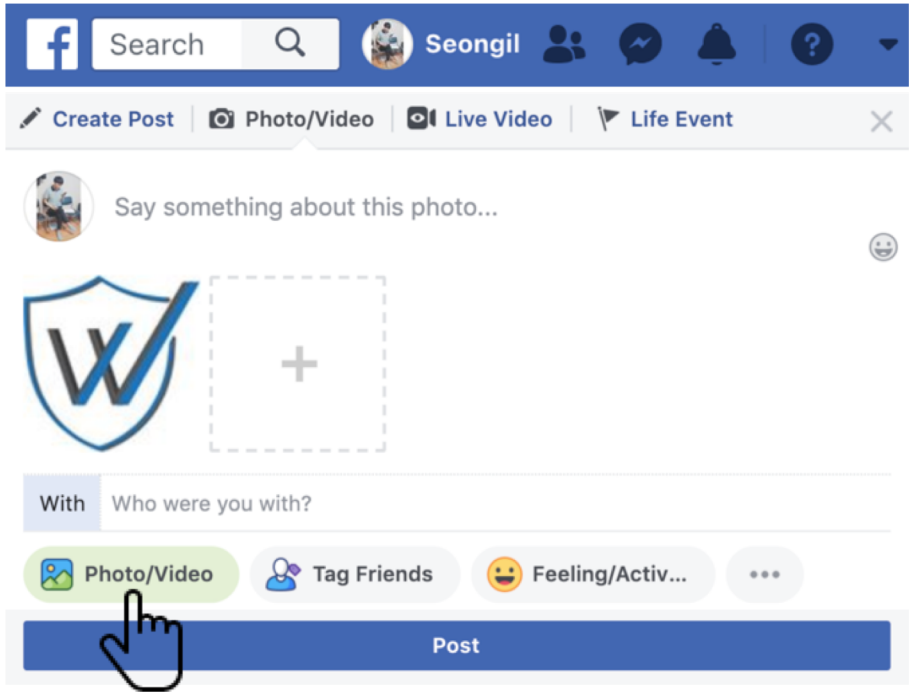
Taekjin Lee, **Seongil Wi**, Suyoung Lee, Sooel Son

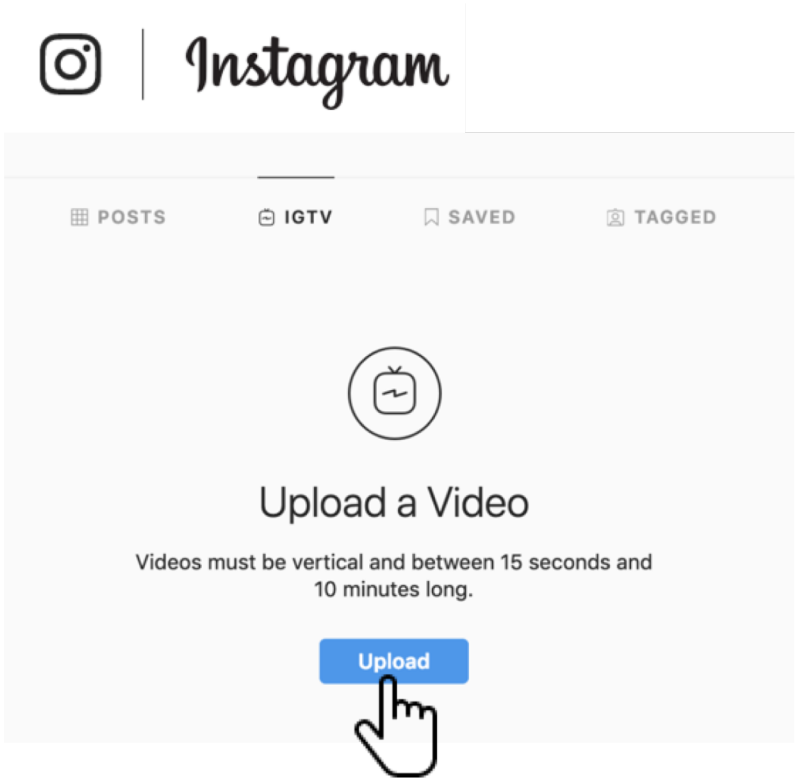
KAIST

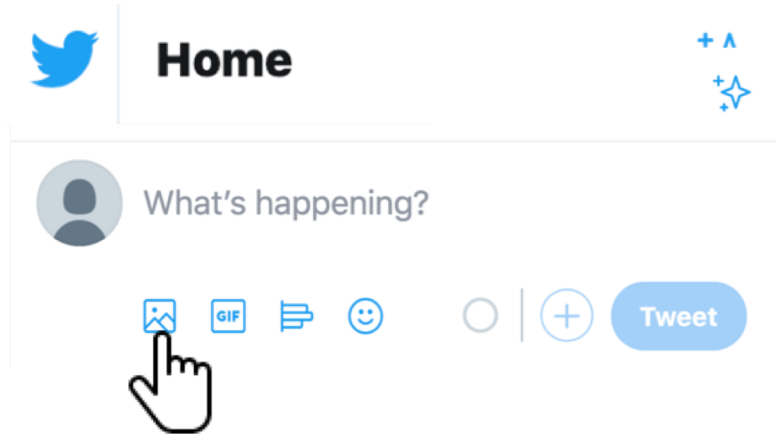
KAIST Web Security & Privacy Lab

# Upload Functionality

- Sharing user-provided content has become a *de facto* standard feature of modern web applications



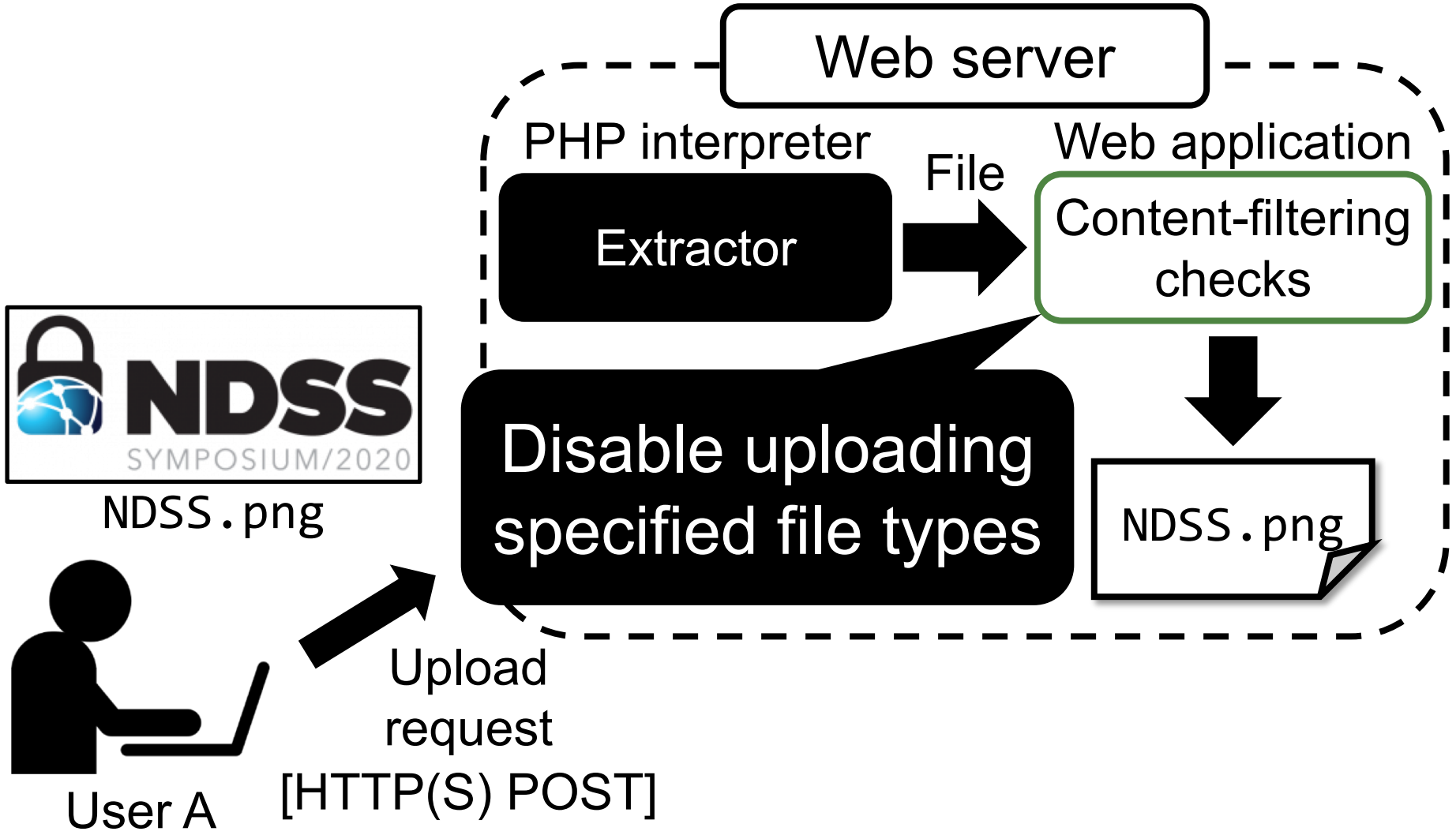



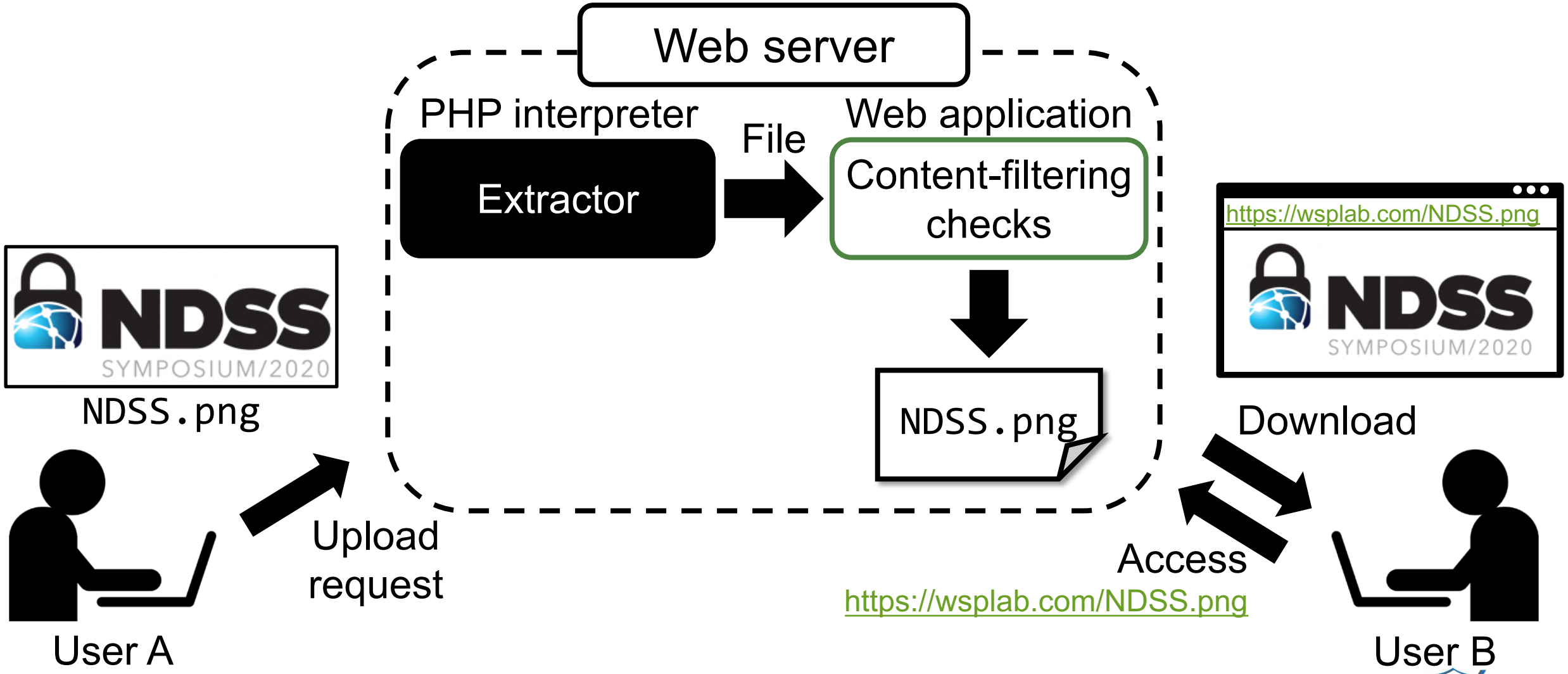



KAIST Web Security & Privacy Lab

# File Uploading Procedure



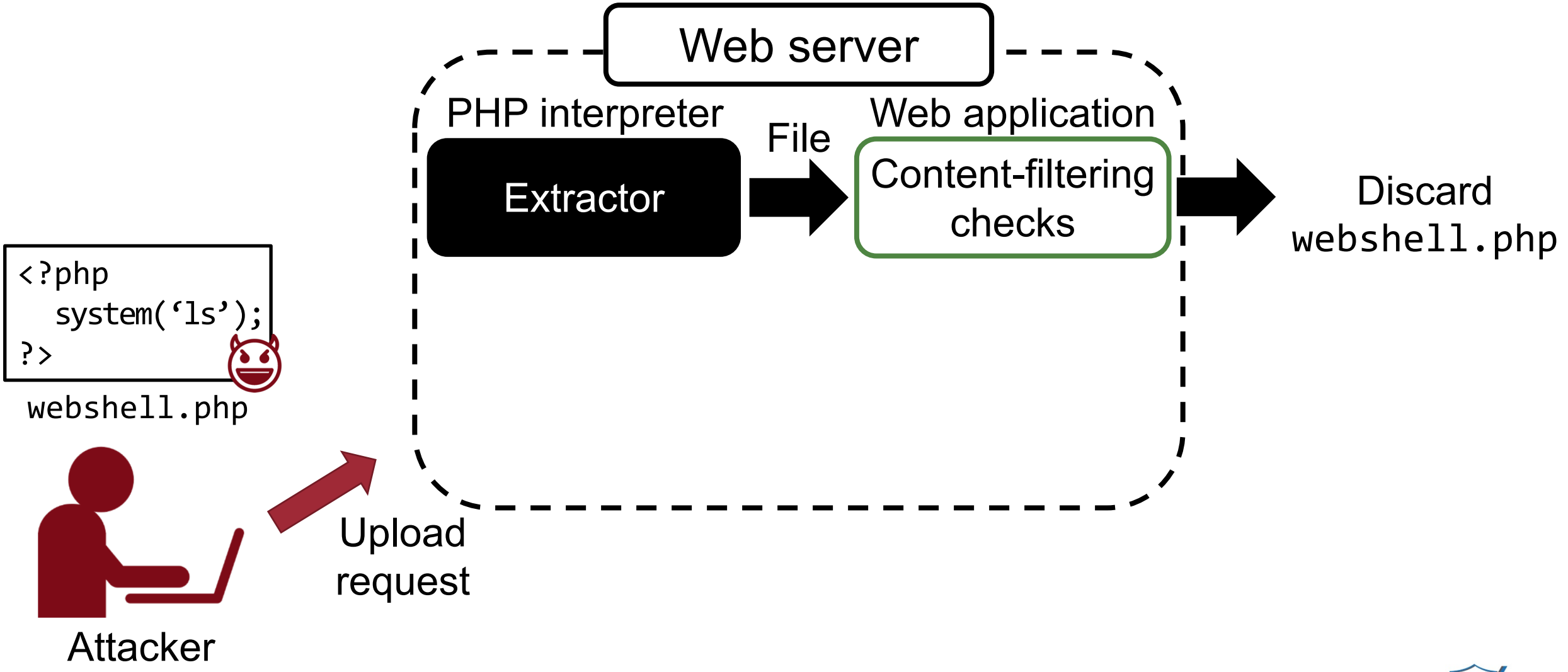
# File Uploading Procedure



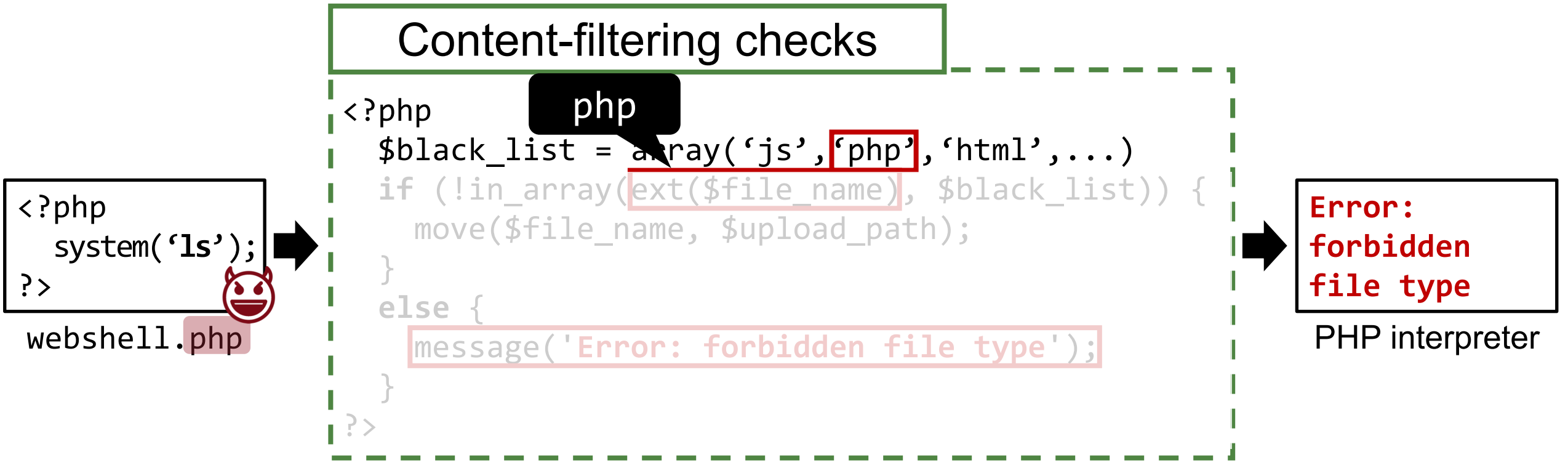



KAIST Web Security & Privacy Lab

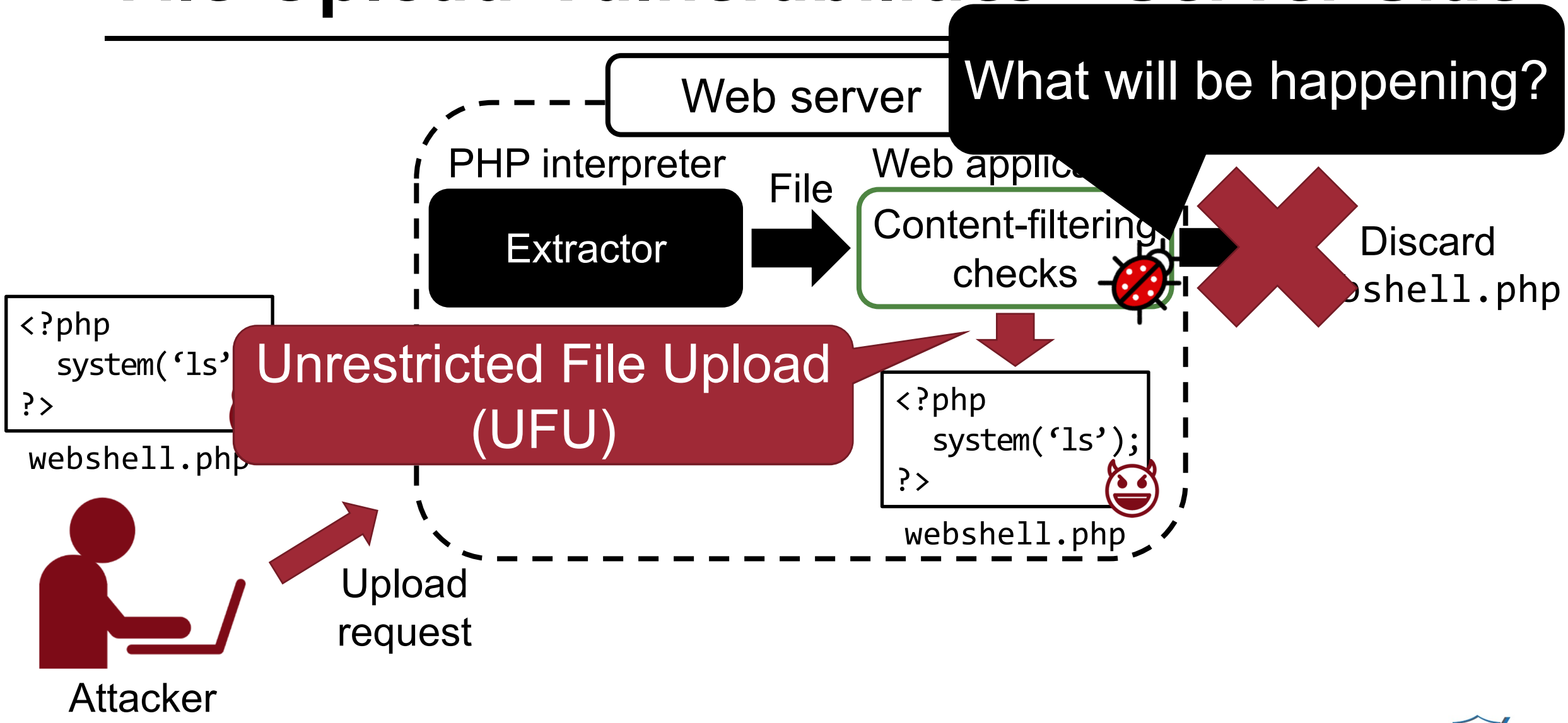
# Disable Uploading Malicious Files



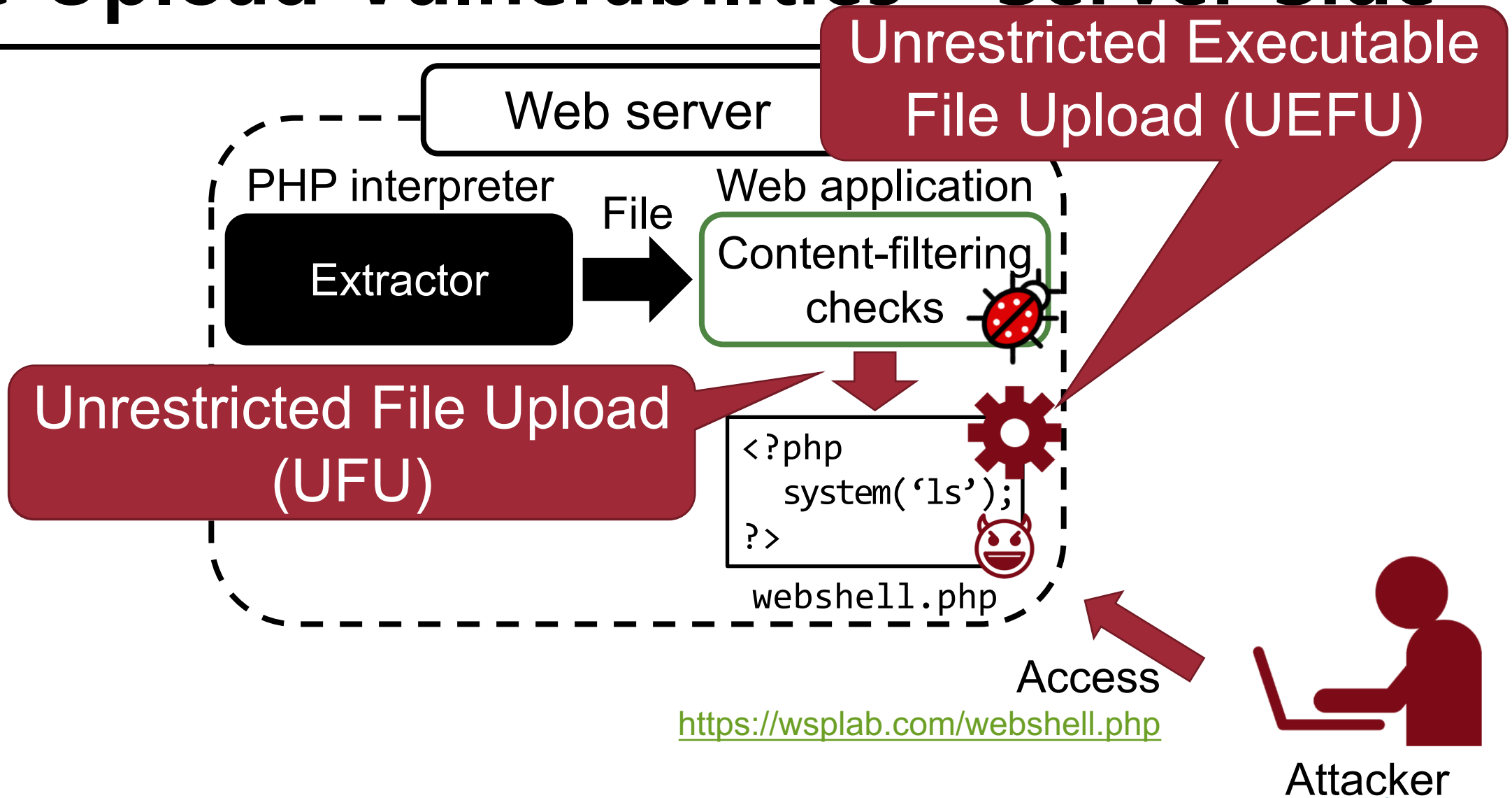
# Content-filtering Checks



# File Upload Vulnerabilities - Server Side



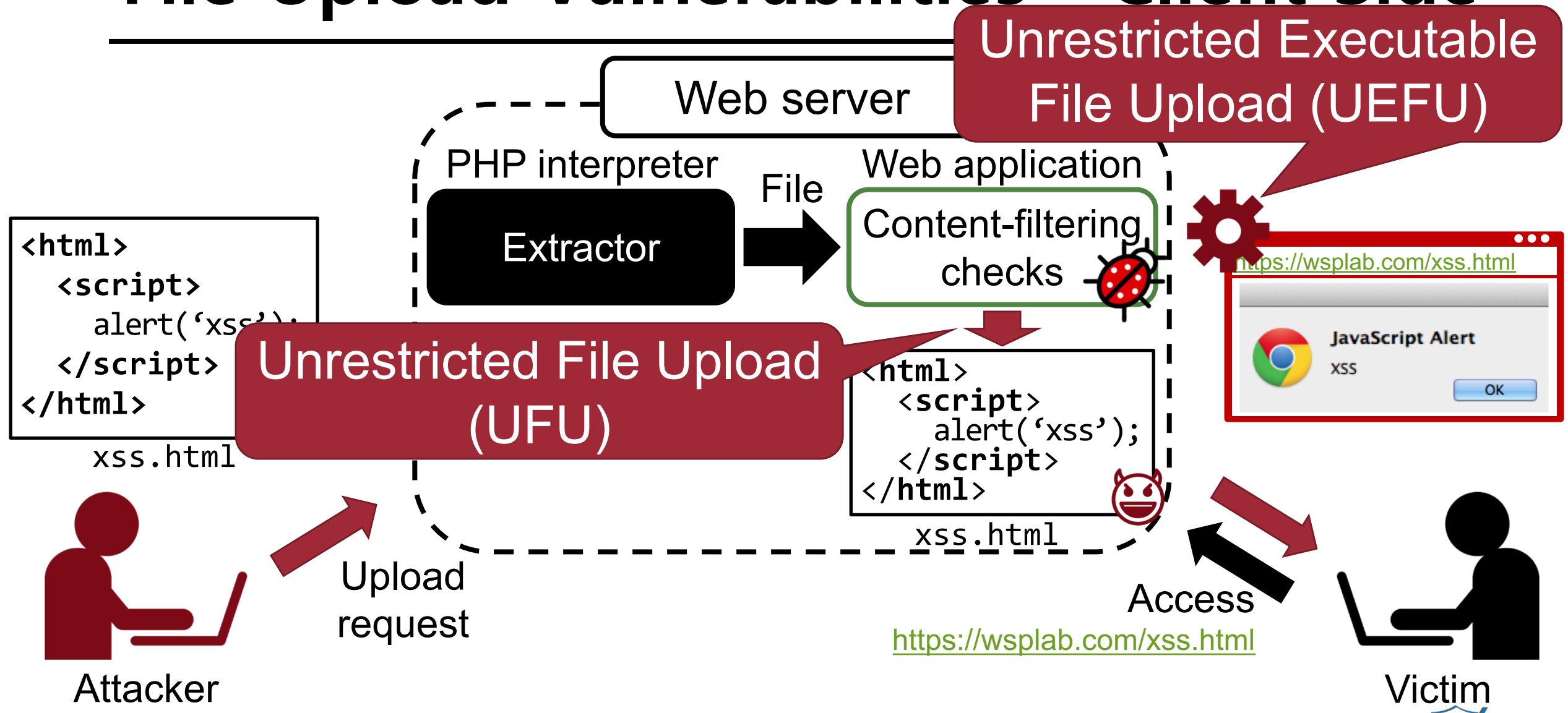
# File Upload Vulnerabilities - Server Side






KAIST Web Security & Privacy Lab

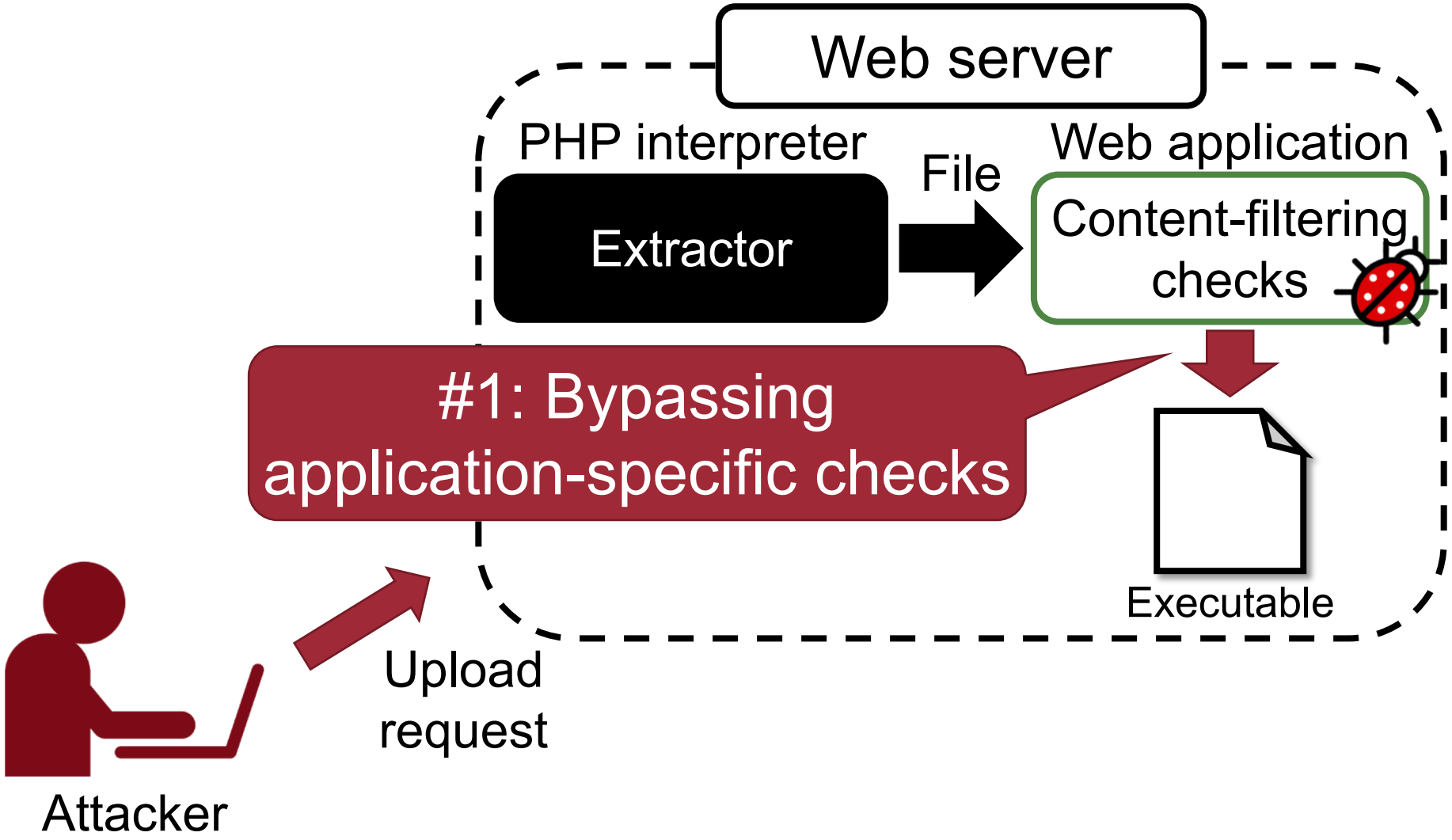
# File Upload Vulnerabilities - Client Side



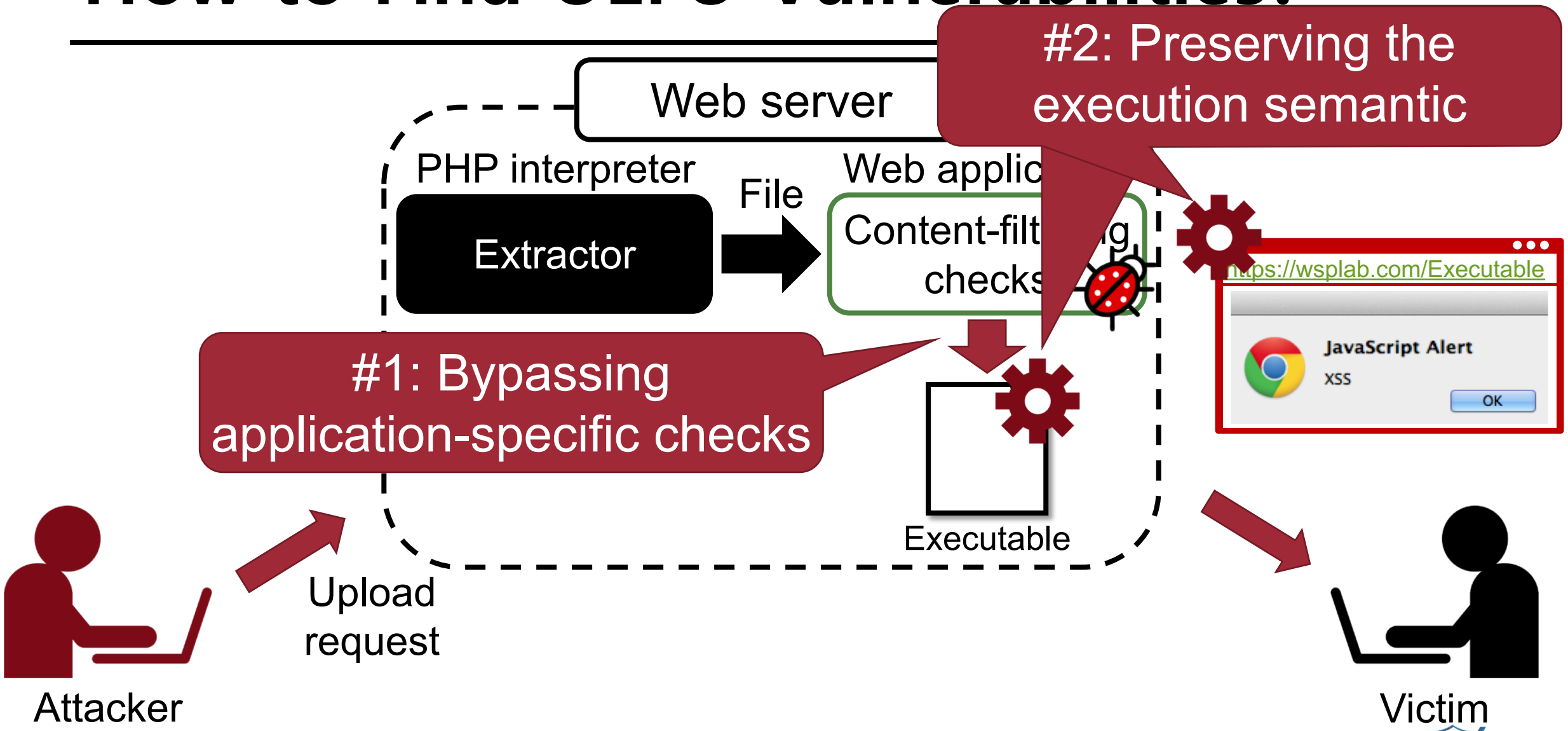



KAIST Web Security & Privacy Lab

# How to Find UEFU Vulnerabilities?



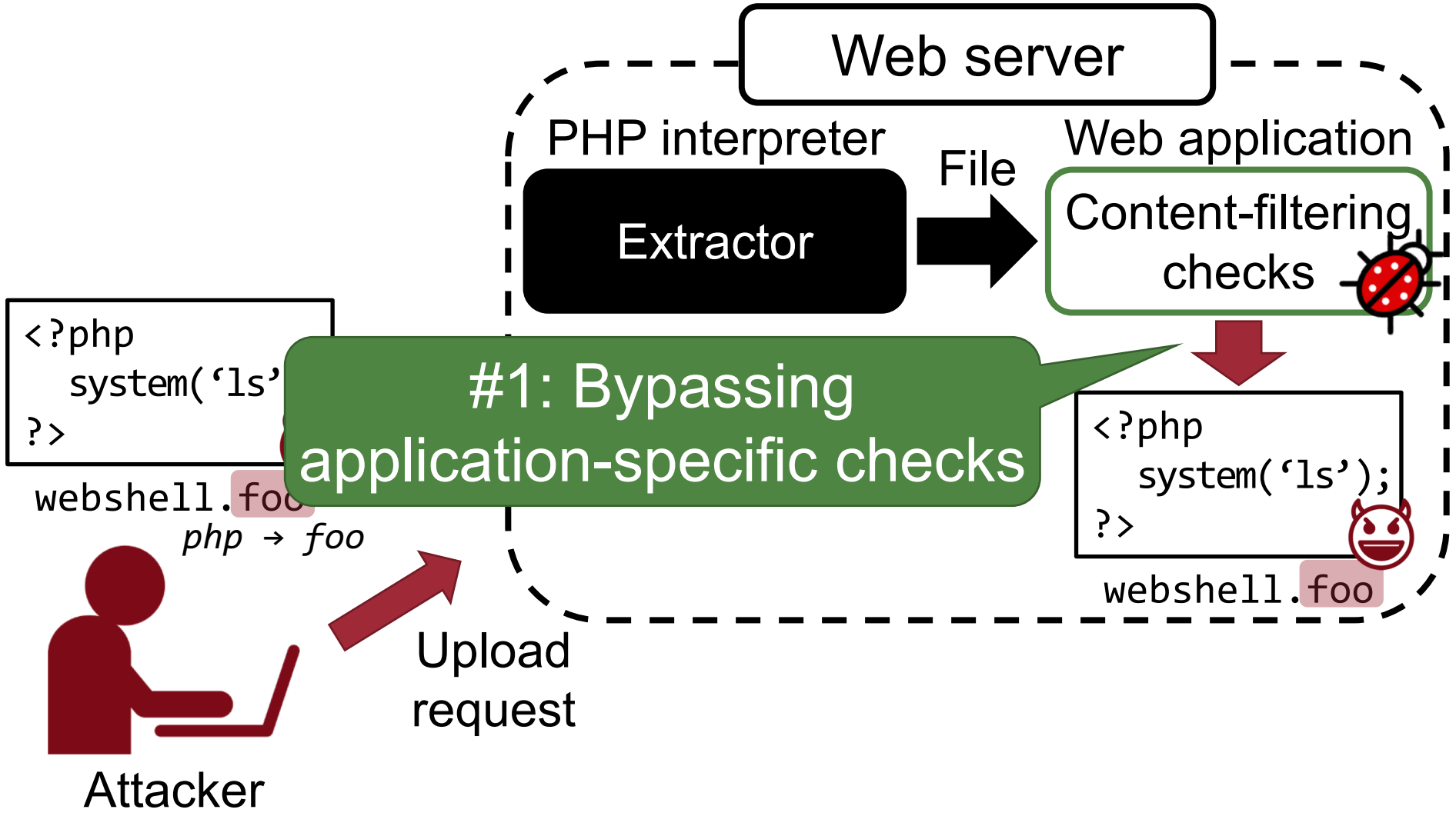
# How to Find UEFU Vulnerabilities?



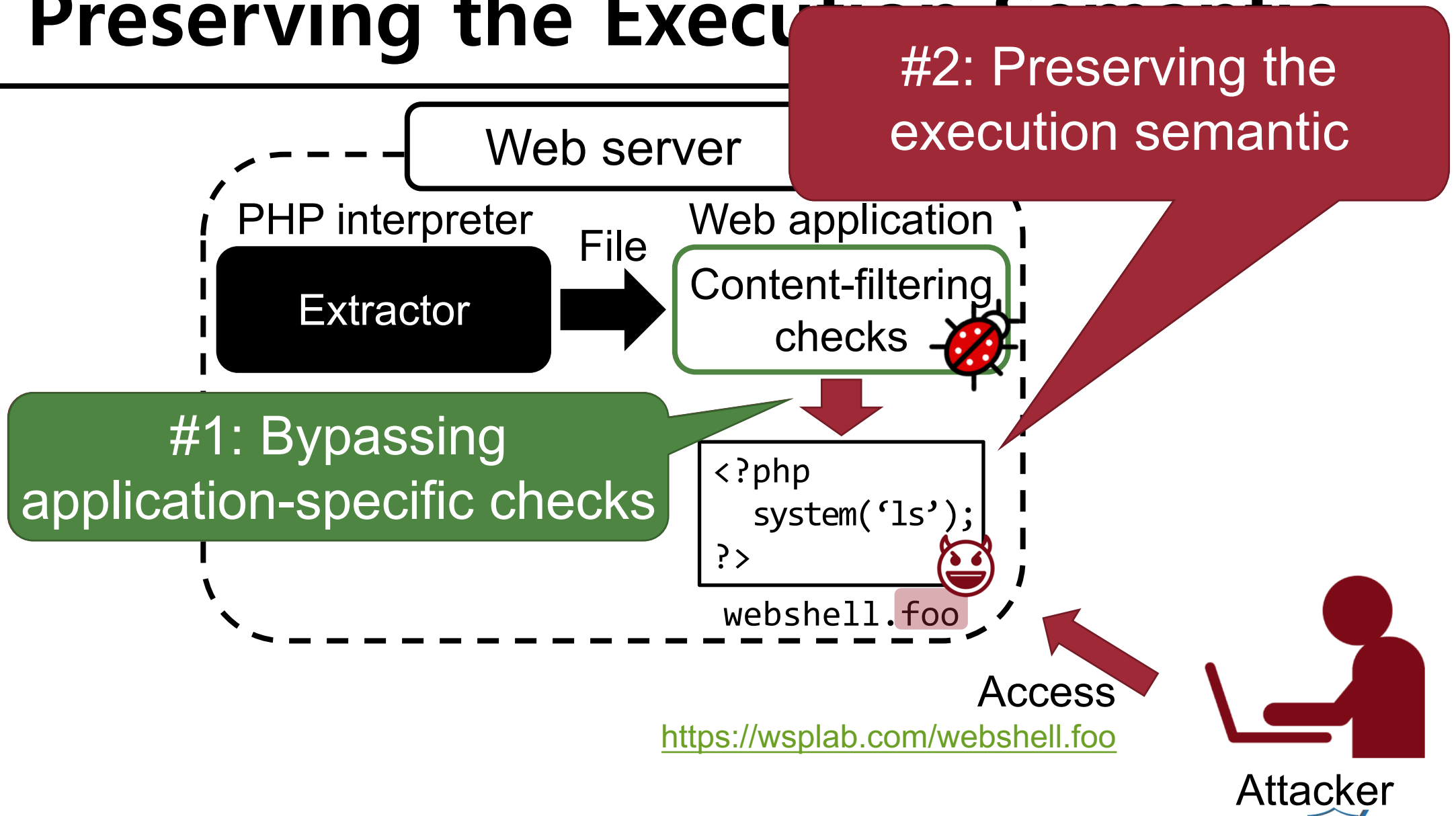



KAIST Web Security & Privacy Lab

# #2: Preserving the Execution Semantic



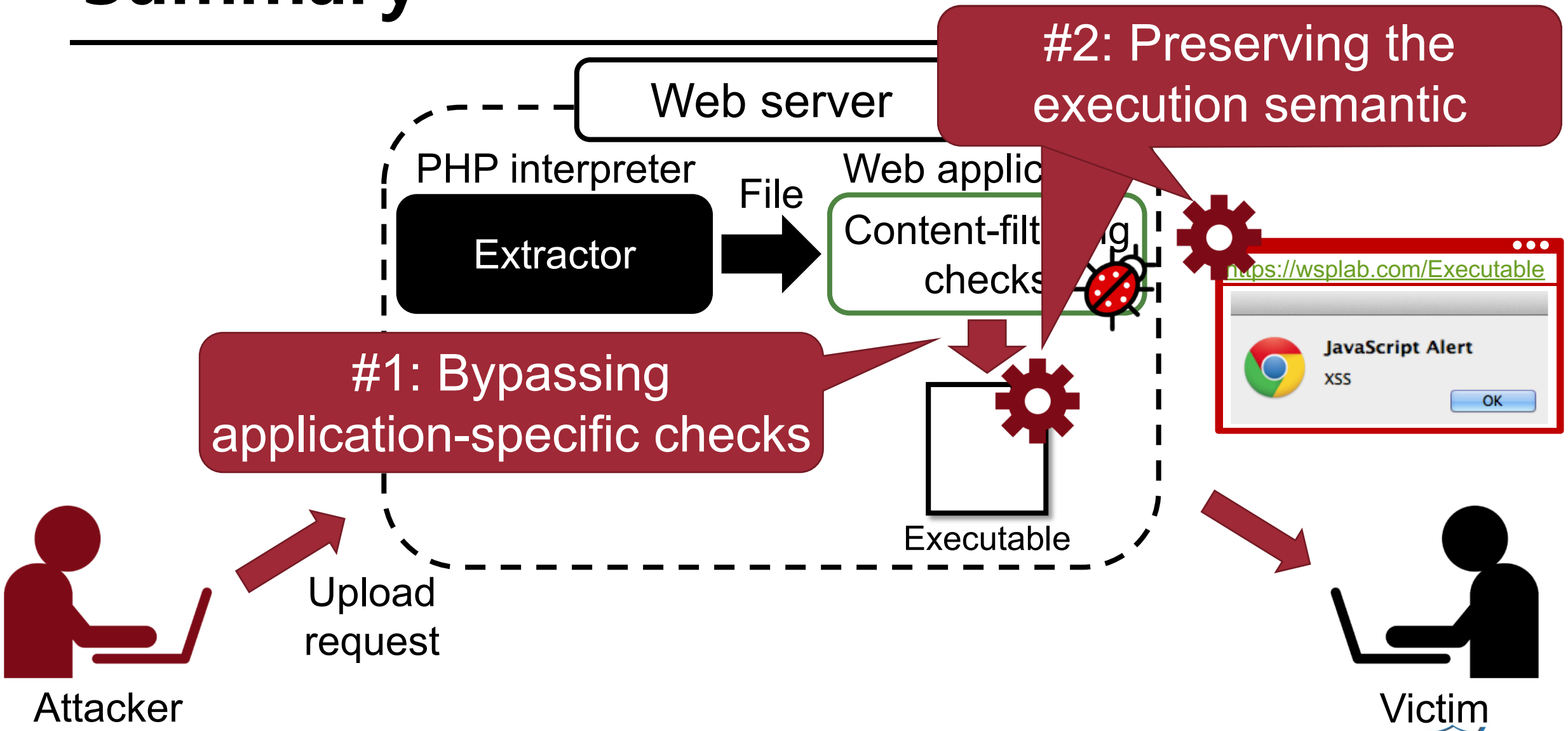
# #2: Preserving the Execu






KAIST Web Security & Privacy Lab

# Summary






KAIST Web Security & Privacy Lab

# Previous Studies

- Static analysis
  - Pixy, ***Oakland '06***
  - Merlin, ***PLDI '09***
- Dynamic analysis
  - Saner, ***Oakland '08***
  - Riding out DOMsday, ***NDSS '18***
- Symbolic execution
  - NAVEX, ***USENIX '18***
  - SAFERPHP, ***PLAS '11***

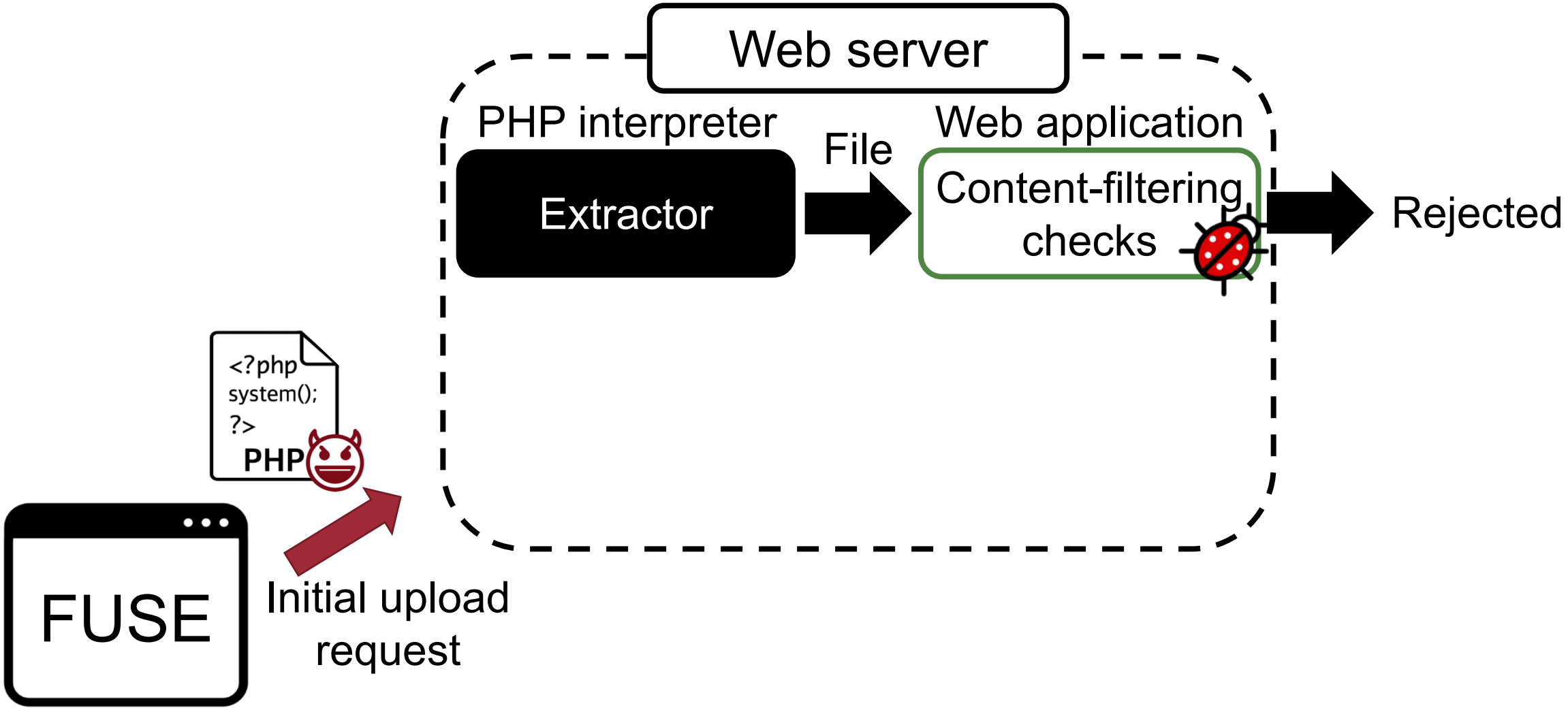
*Few studies have addressed finding **U(E)FU vulnerabilities**!*

# How we address all the challenges?

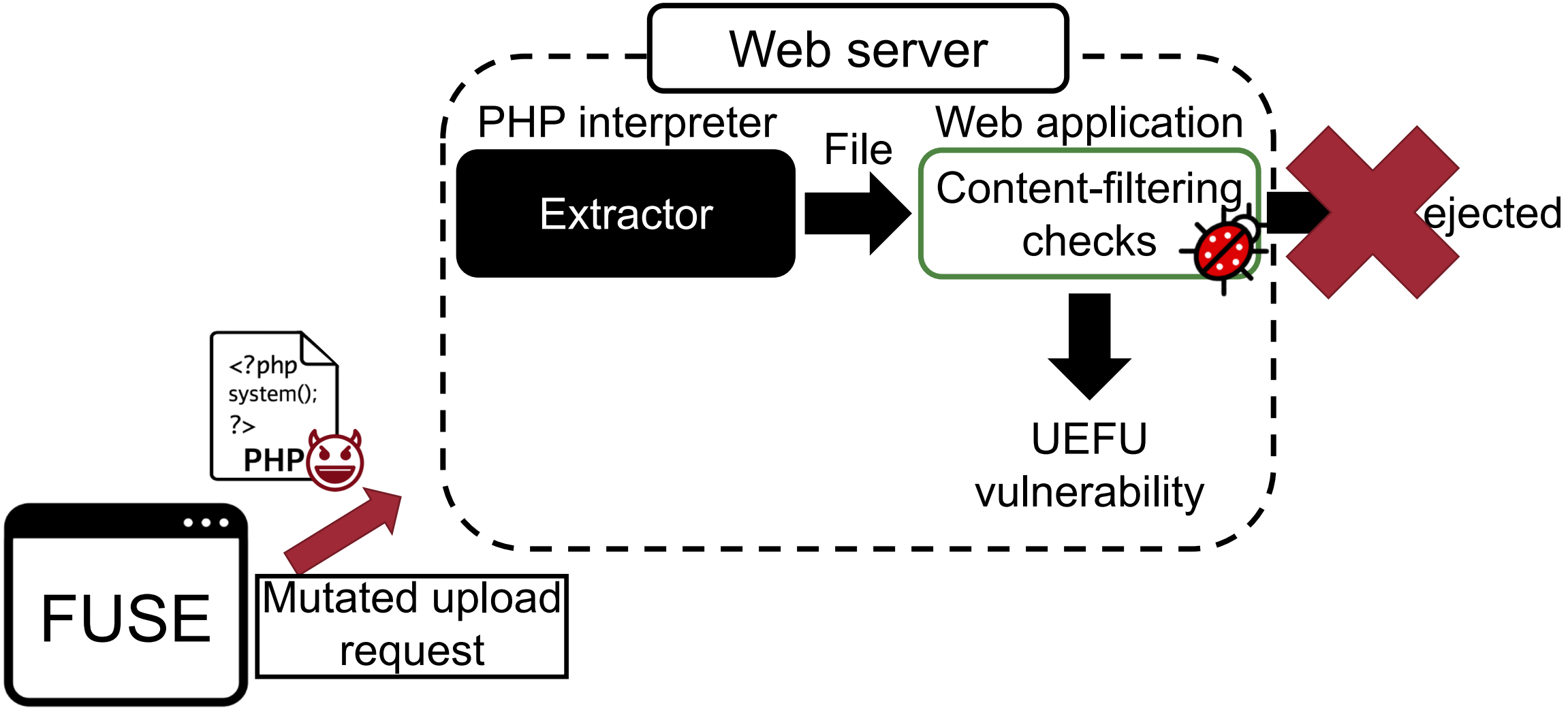
# We propose

# FUSE

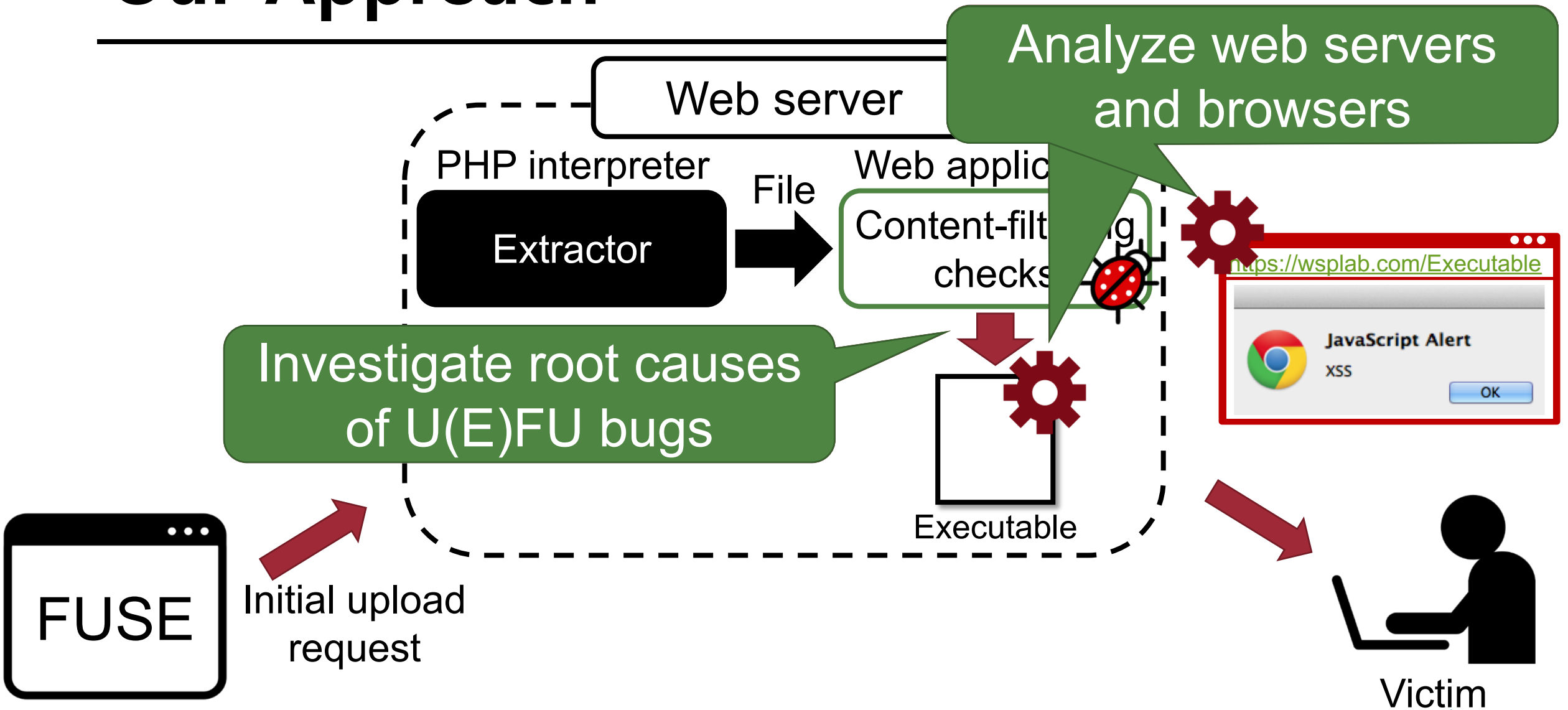
# Our Approach



# Our Approach - Mutate Upload Request



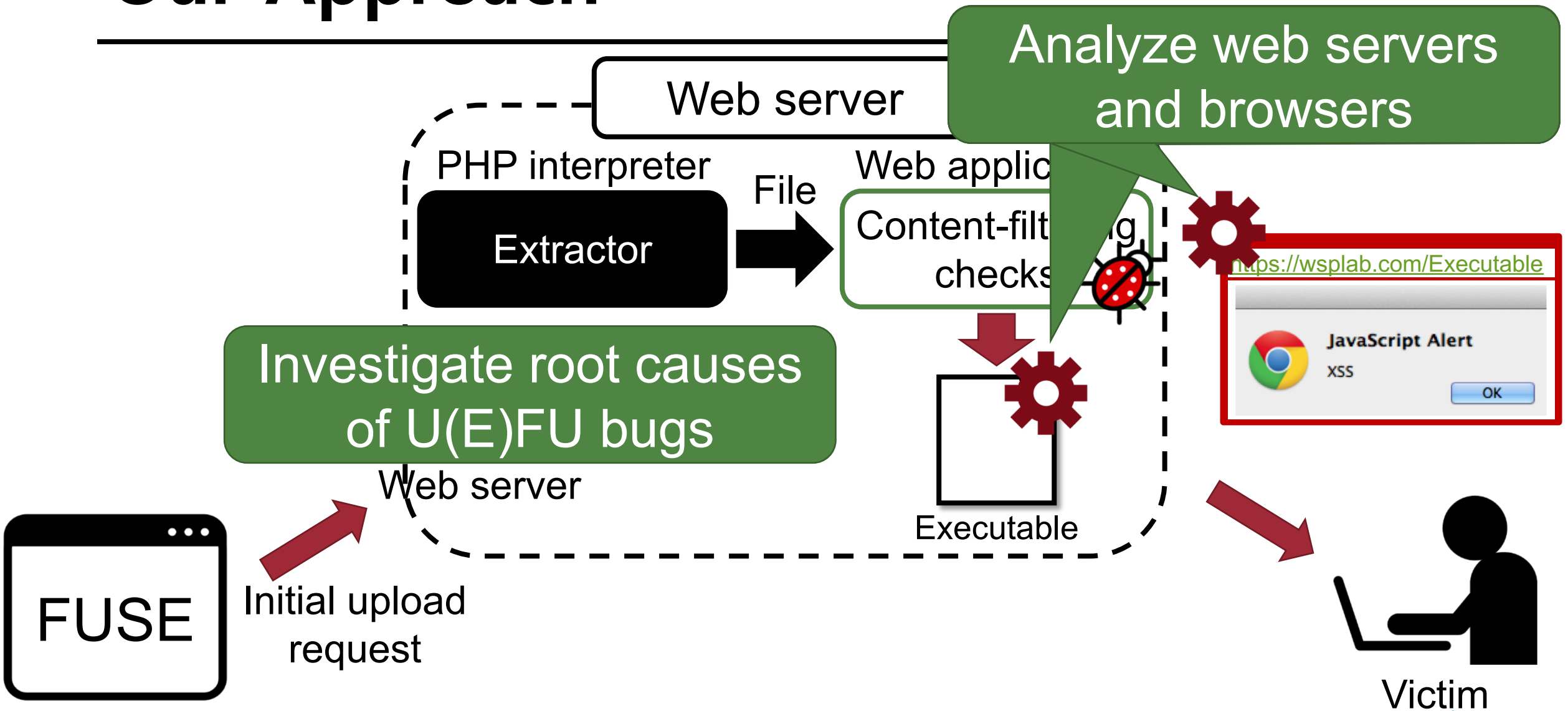
# Our Approach






KAIST Web Security & Privacy Lab

# Our Approach



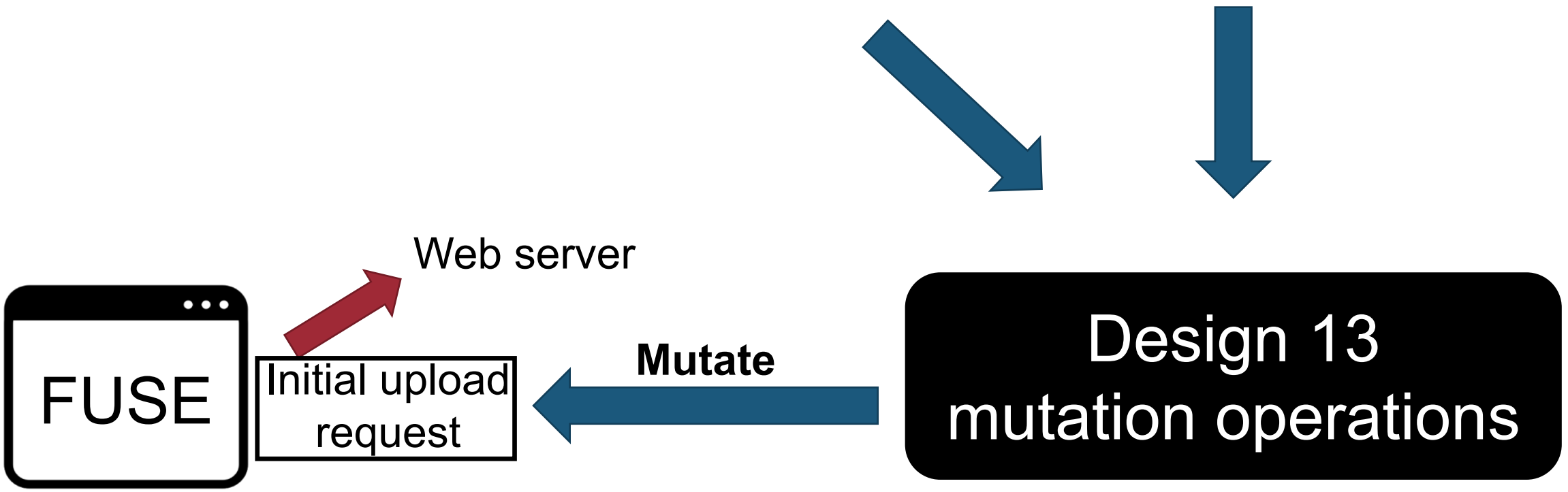



KAIST Web Security & Privacy Lab

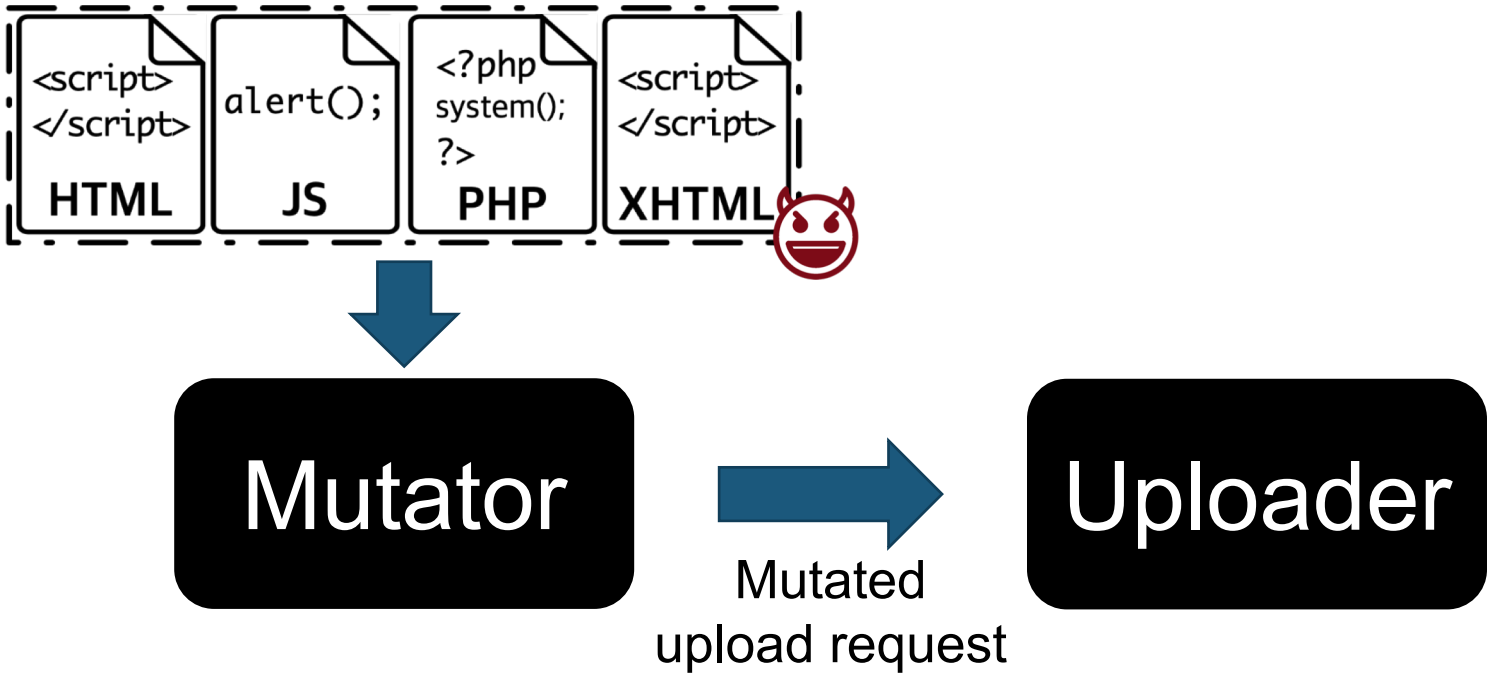
# Mutate Upload Request

Investigate root causes of U(E)FU bugs

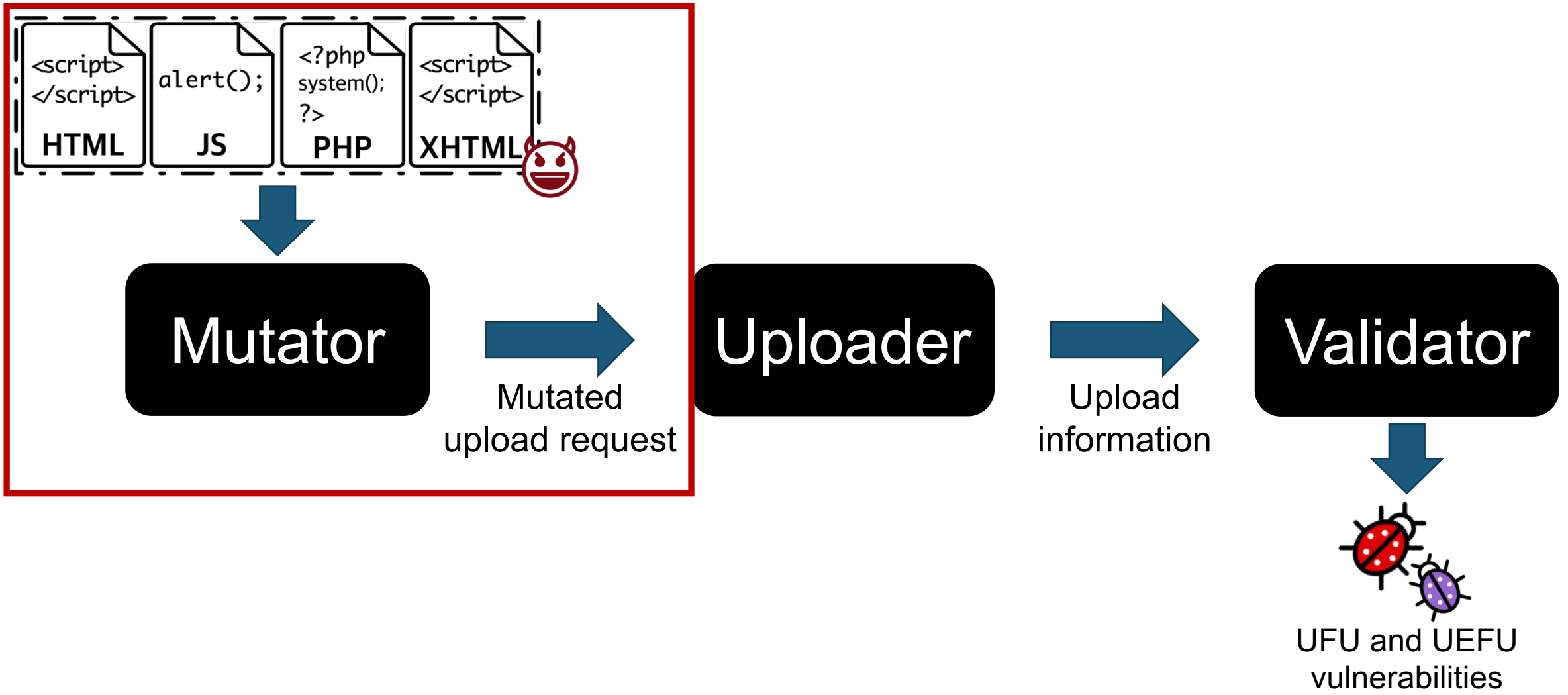
Analyze web servers and browsers



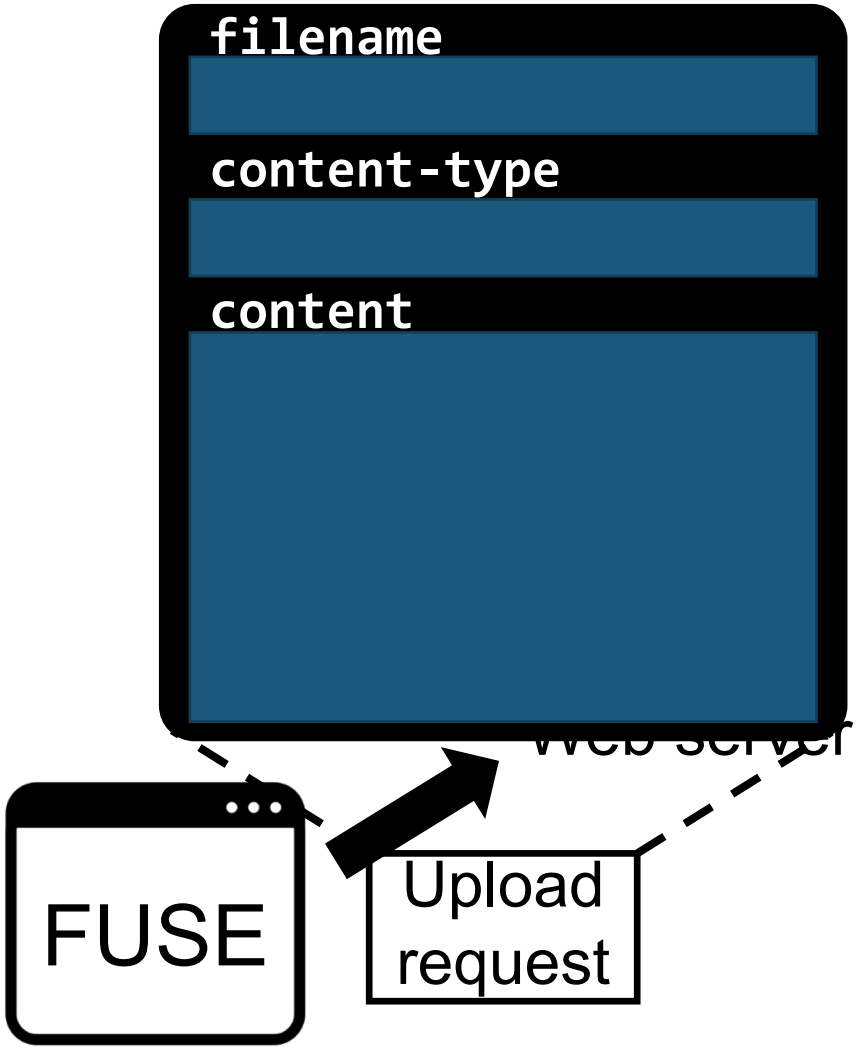
# Our Goal: Finding U(E)FU Bugs



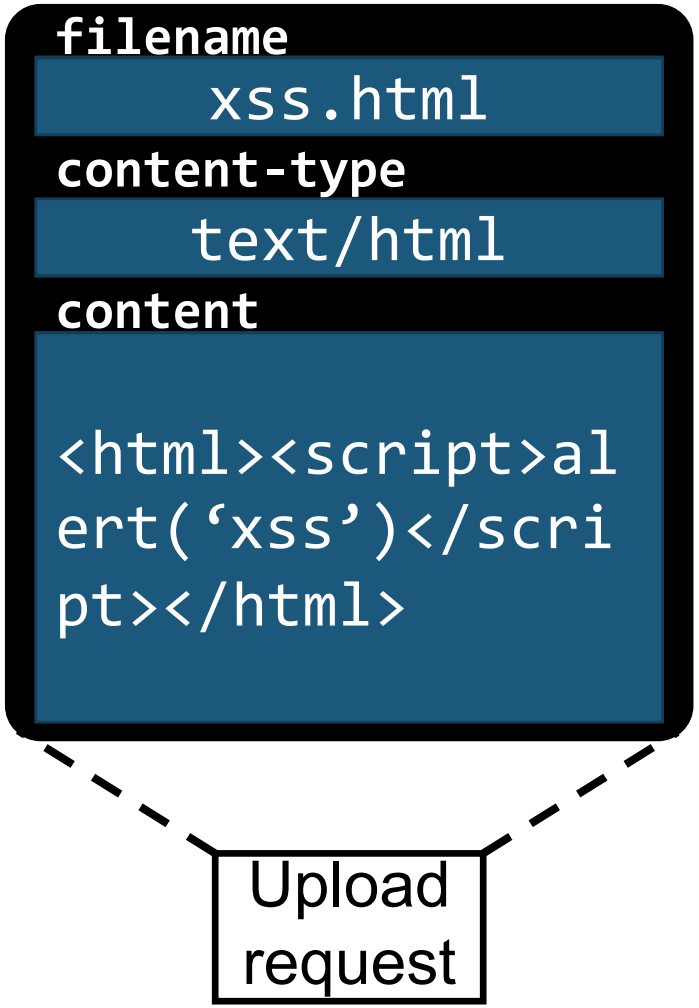
# Our Goal: Finding U(E)FU Bugs



# Upload Request



# Upload Request

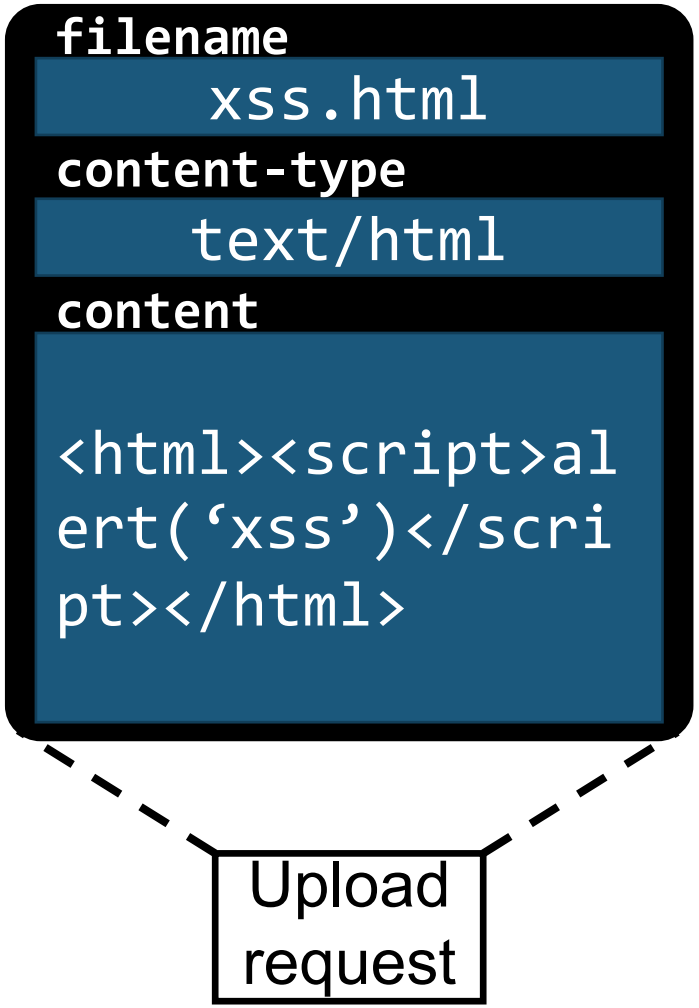



```
<html>
  <script>
    alert('xss');
  </script>
</html>
```

xss.html

# Mutation Objectives

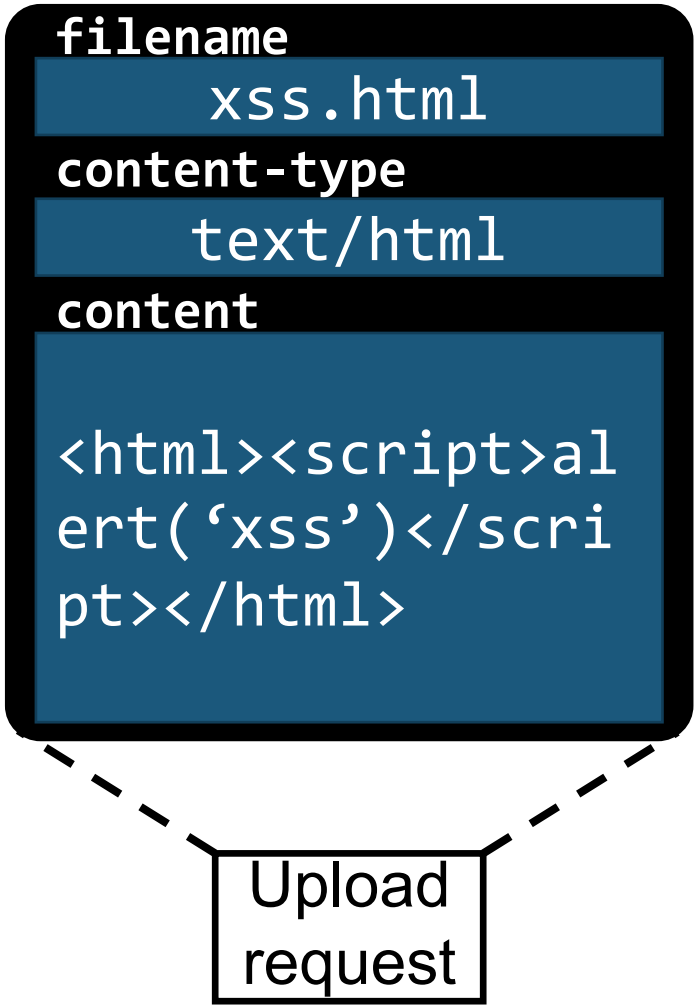
**Five** objectives that trigger common mistakes in implementing checks




Content-filtering checks

```
if (finfo_file(content) not in expected_type)
    reject(file);
if (ext(file_name) not in expected_ext)
    reject(file);
if (expected_keyword in content)
    reject(file);
if (content_type not in expected_type)
    reject(file);
accept(file)
```

# Mutation Objectives #1



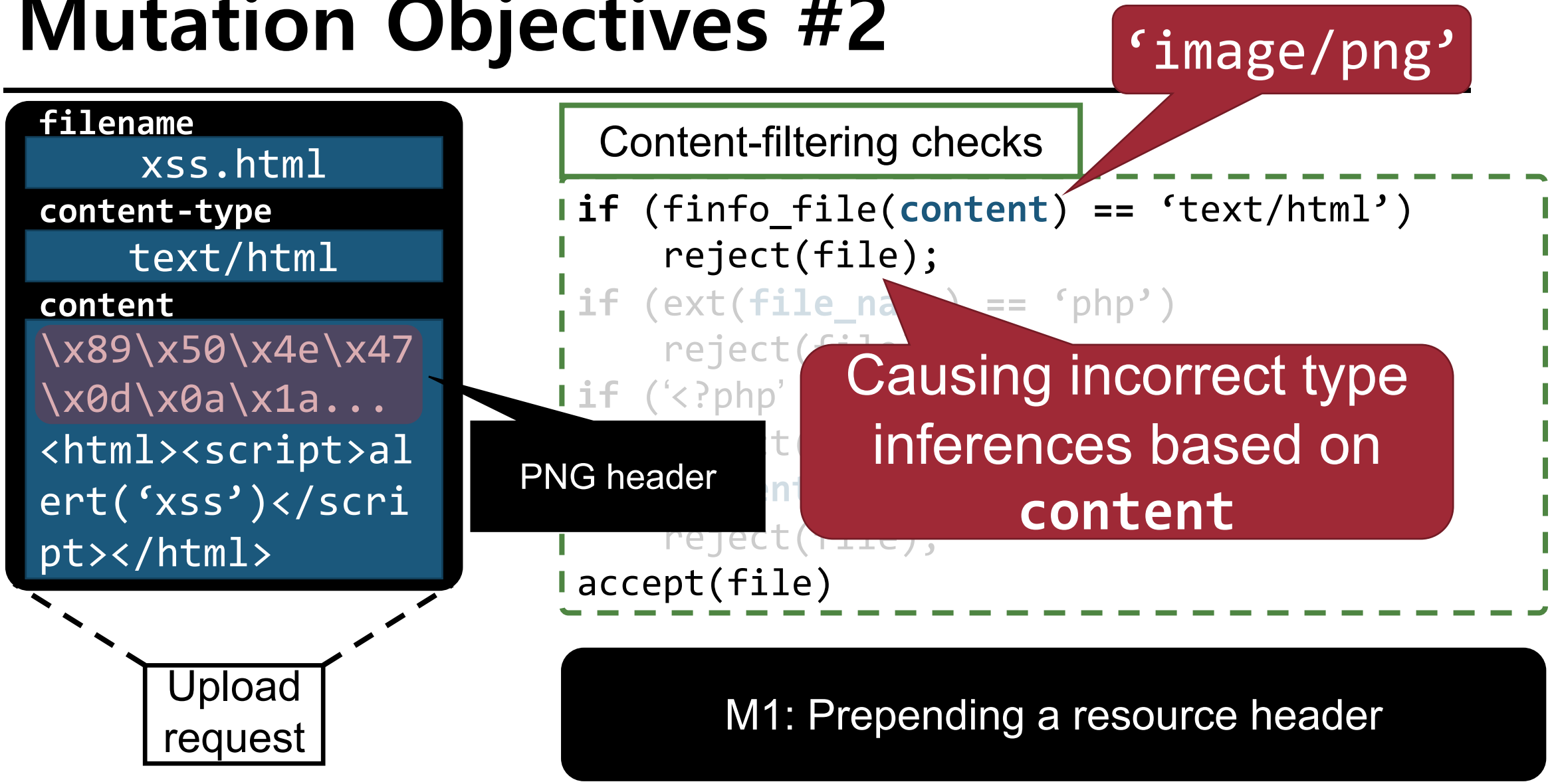

Content-filtering checks

```
if (finfo_file(content) == 'text/html')
    reje
if (ext(
    reje
if ('<?ph
    reject(fi
if (content_      == 'text/html')
    reject(     ile);
accept(file)
```

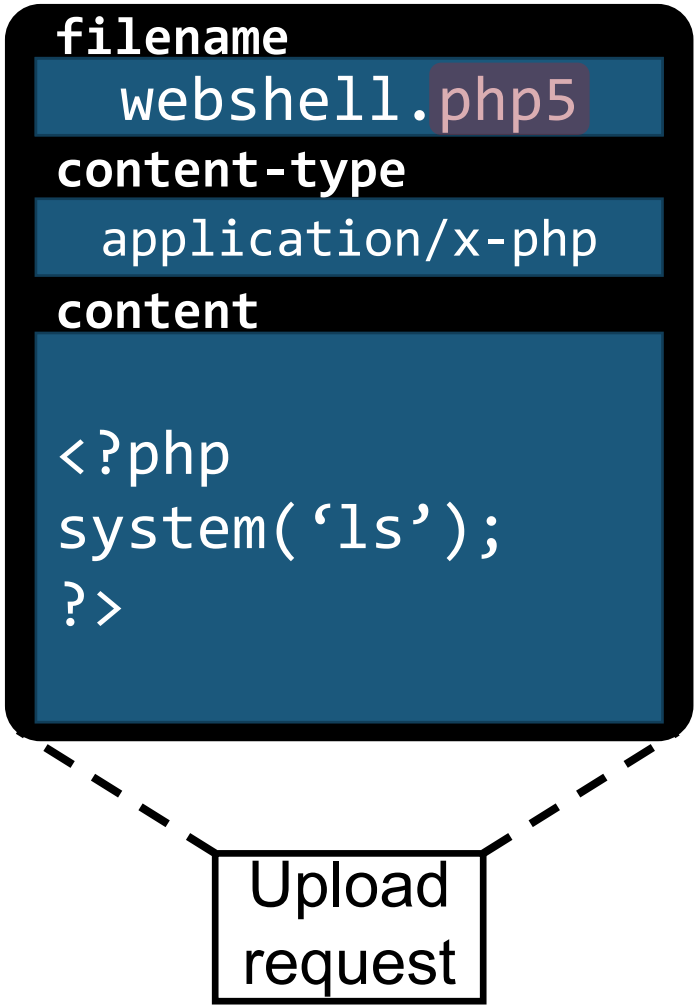
Exploiting the absence of content-filtering checks

No mutation

# Mutation Objectives #2



# Mutation Objectives #3




Content-filtering chec

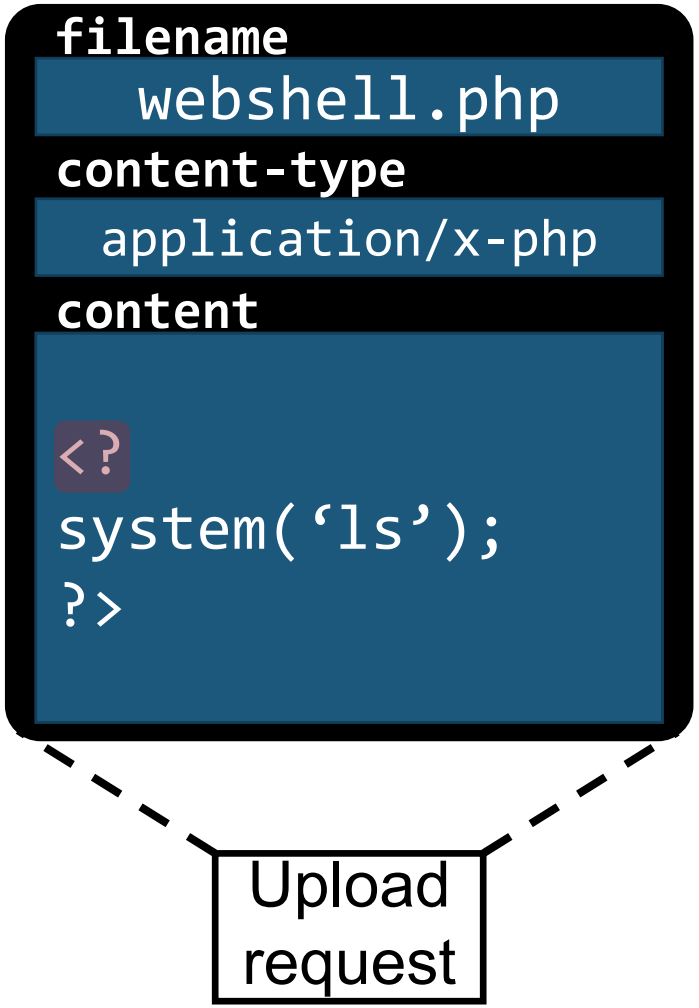
‘php5’

```
if (finfo_file(content == 'text/html')
    reject(file);
if (ext(file_name) == 'php')
    reject(file);
if ('<?php' in conte
    reject(fil
if (content_ty
    reject(fil
accept(file)
```

Exploiting incomplete blacklist based on **extension**

M4: Changing a file extension

# Mutation Objectives #4




Content-filteri

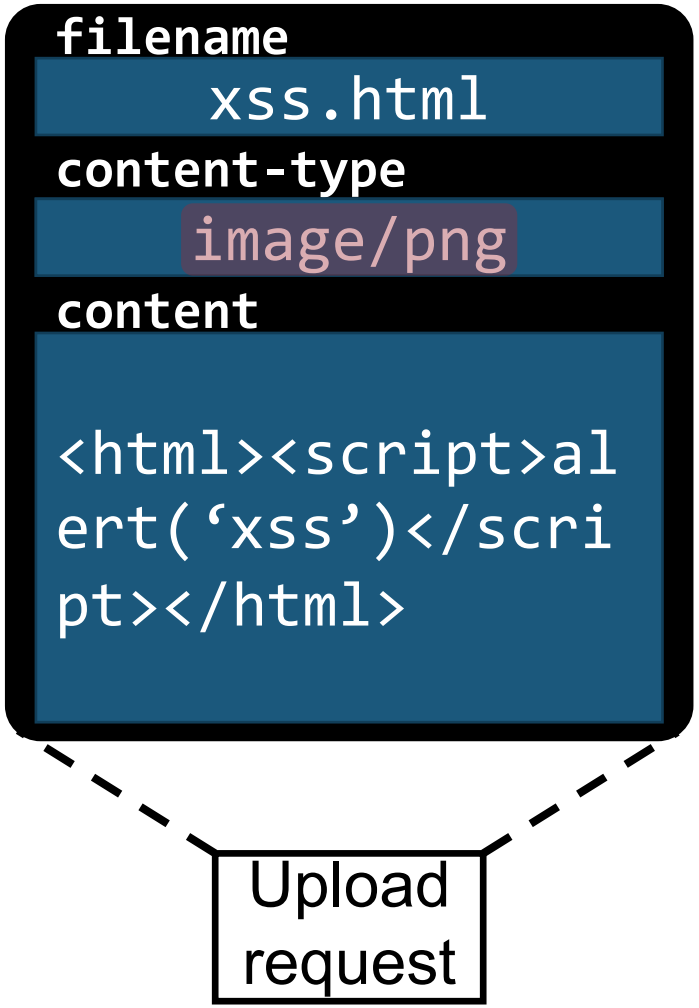
```
if (finfo_fil
    reject(fi
if (ext(file_name
    reject(file)
if ('<?php' in content)
    reject(file);
if (content_type ==
    reject(file);
accept(file)
```

Bypassing keyword checks based on **content**

'<?'

M5: Replace PHP tags with short tags

# Mutation Objectives #5




Content-filtering checks

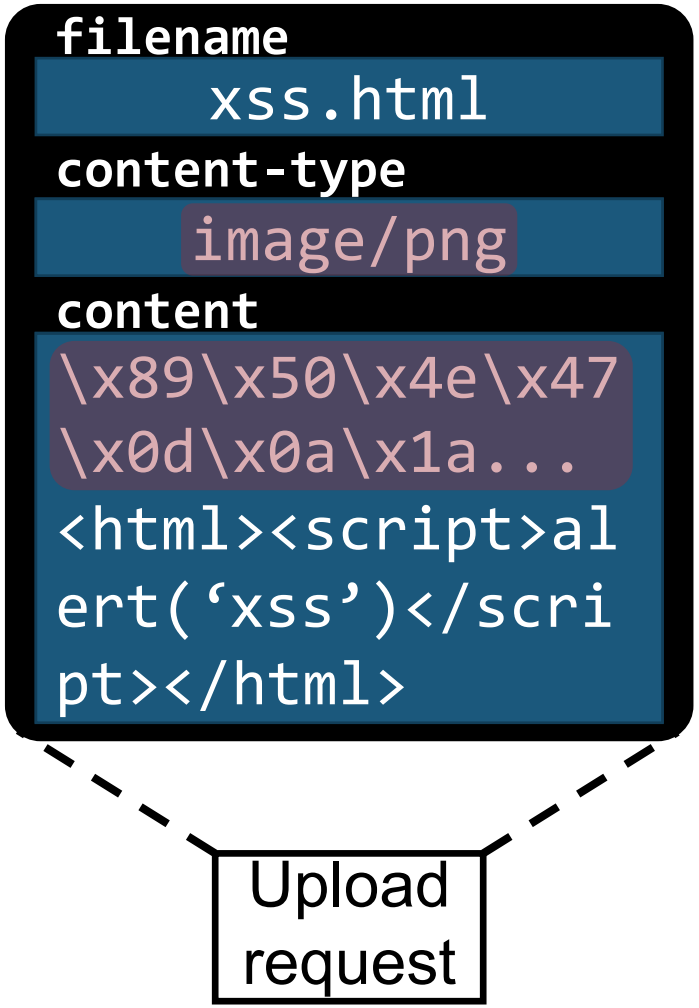
```
if (finfo_fil
    reject(fi
if (ext(file_
    reject(fi
    reject(file);
if (content_type == 'text/html')
    reject(file);
accept(file)
```

'image/png'

Bypassing filtering logic based on **content-type**

M3: Changing the content-type of an upload request

# Combinations of Mutation Operations




‘image/png’

Content-filtering checks

```
if (finfo_file(content) == ‘text/html’)
    reject(file);
if (ext(file_name) ==
    reject(file);
    n content)
    reject(file);
if (content_type == ‘text/html’)
    reject(file);
accept(file)
```

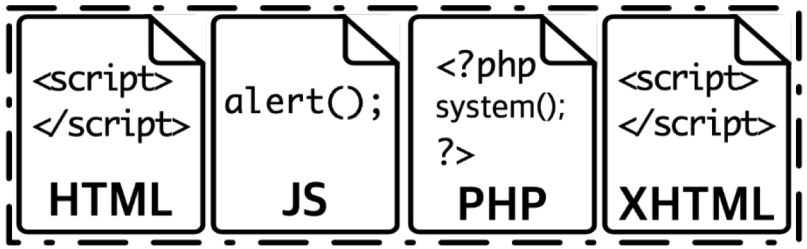
‘image/png’

M1: Prepending a resource header
+
M3: Changing the content-type of an upload request

# More in the Paper

- M2: Inserting a seed into metadata
- M6: Converting HTML into EML
- M7: Removing a file extension
- M8: Converting a file in SVG
- M9: Prepending an HTML comment
- M10: Changing a file extension to an arbitrary string
- M11: Converting a file extension to uppercase
- M12: Prepending a file extension
- M13: Appending a resource header

# Evaluation




Mutator → Mutated upload request → Uploader → Upload information → Validator

UFU and UEFU vulnerabilities

# Experimental Setup

- 33 popular PHP web applications

| | | | | | |
|---|---|---|---|---|---|
| WordPress | Joomla | Concrete5 | OsCommerce2 | Monstra | Drupal |
| ZenCart | Bludit | Textpattern | CMSMadeSimple | Pagekit | Backdrop |
| CMSimple | Composr | OctoberCMS | phpBB3 | Elgg | Microweber |
| XE | SilverStripe | ECCube3 | GetSimpleCMS | DotPlant2 | MyBB |
| HotCRP | Subrion | SymphonyCMS | AnchorCMS | WeBid | Collabtive |
| X2engine | ClipperCMS | Codiad | | | |

- Web server: Apache 2.4

- PHP engine: PHP 5.6, 7.0, 7.1

# Real-World UEFU Finding

- Found **30 UEFU vulnerabilities** in 23 applications with 176 distinct upload request
  - WordPress, Concrete5, OsCommerce2, ZenCart, …
- Reported all the vulnerabilities
  - **15 CVEs** from 9 applications
- 8 bugs have been patched
- 5 bugs are being patched

# Case Study - Microweber

**filename**

webshell.pht

**content-type**

application/x-php

**content**

```
<?php
system('ls');
?>
\xff\xd8\xff\xee
\x00\x10JF
```

8 bytes header of a JPG file

Upload request

‘application/octet-stream’

Content-filtering c

```
if (finfo_file(content) == 'application/x-php')
    reject(file);
if (ext(file_name) == 'php')
    reject(file);
'p' in conte
t(file);
if (content_type == 'text/html')
    reject(file);
accept(file)
```

‘pht’

M13: Appending a resource header
+
M4: Changing a file extension

# vs. State-of-the-Arts

- Fuxploider: **open-source** upload vulnerability scanning tool
- UploadScanner: an extension for Burp Suit Pro, a **commercial platform** for web application security testing
- Ran on the same benchmarks and counted vulnerabilities

| Vulnerability (Seed) | FUSE | Fuxploider | UploadScanner |
|---|---|---|---|
| UEFU (PHP) | 12 | 7 | 5 |
| UEFU (HTML) | 23 | N/A | 14 |
| UFU (JS) | 26 | N/A | 21 |

# Why FUSE found more bug than the others?

- Better `extension` coverage (`pht`, `php7`, …)
- Better mutation operation coverage
  - M9: Prepending an HTML comment
  - M13: Appending a resource header
  - Combination: M4+M13
  - …
- Implementational Issues
  - Retrieving URLs

| Vulnerability (Seed) | FUSE | Fuxploider | UploadScanner |
|---|---|---|---|
| UEFU (PHP) | 12 | 7 | 5 |
| UEFU (HTML) | 23 | N/A | 14 |
| UFU (JS) | 26 | N/A | 21 |

# Vulnerability Causes

Inferring upload file types based on user-provided extensions opens a door for further attacks

| Causes | | UFU + UEFU Bugs Found |
|---|---|---|
| #1 | Exploiting the absence of ch | 27 |
| #2 | Causing incorrect type inferences based on **`content`** | 5 |
| #3 | Exploiting incomplete blacklist based on **`extension`** | 35 |
| #4 | Bypassing keyword checks based on **`content`** | 6 |
| #5 | Bypassing checks based on **`content-type`** | 5 |
| #2+#3 | Combined Operation | 6 |
| #2+#3+#4 | Combined Operation | 1 |

# Limitation

- There may exist other mutation operations that we didn't consider

- Manually examined the execution constraints of browsers and PHP interpreters

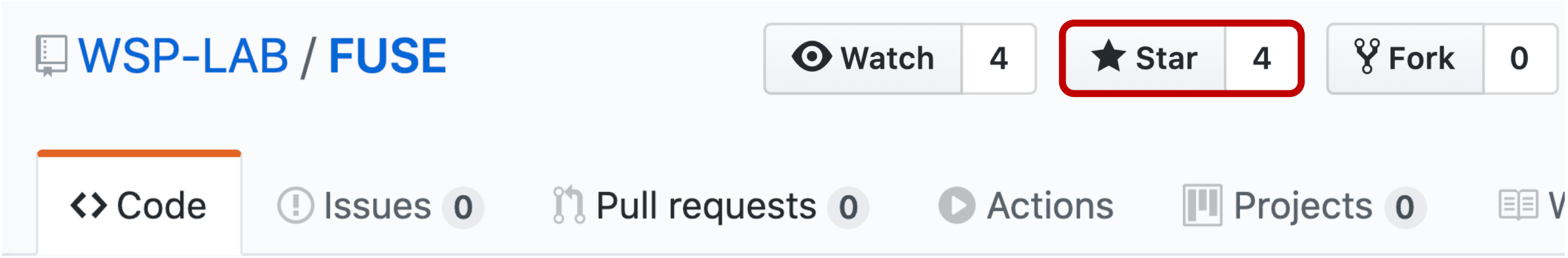
# Conclusion

- Propose FUSE, a penetration testing tool designed to find U(E)FU vulnerabilities

- Present 13 operations that mutate upload request to bypass content-filtering checks, but to remain executable in target execution environments

- Found 30 UEFU vulnerabilities including 15 CVEs from 33 PHP applications

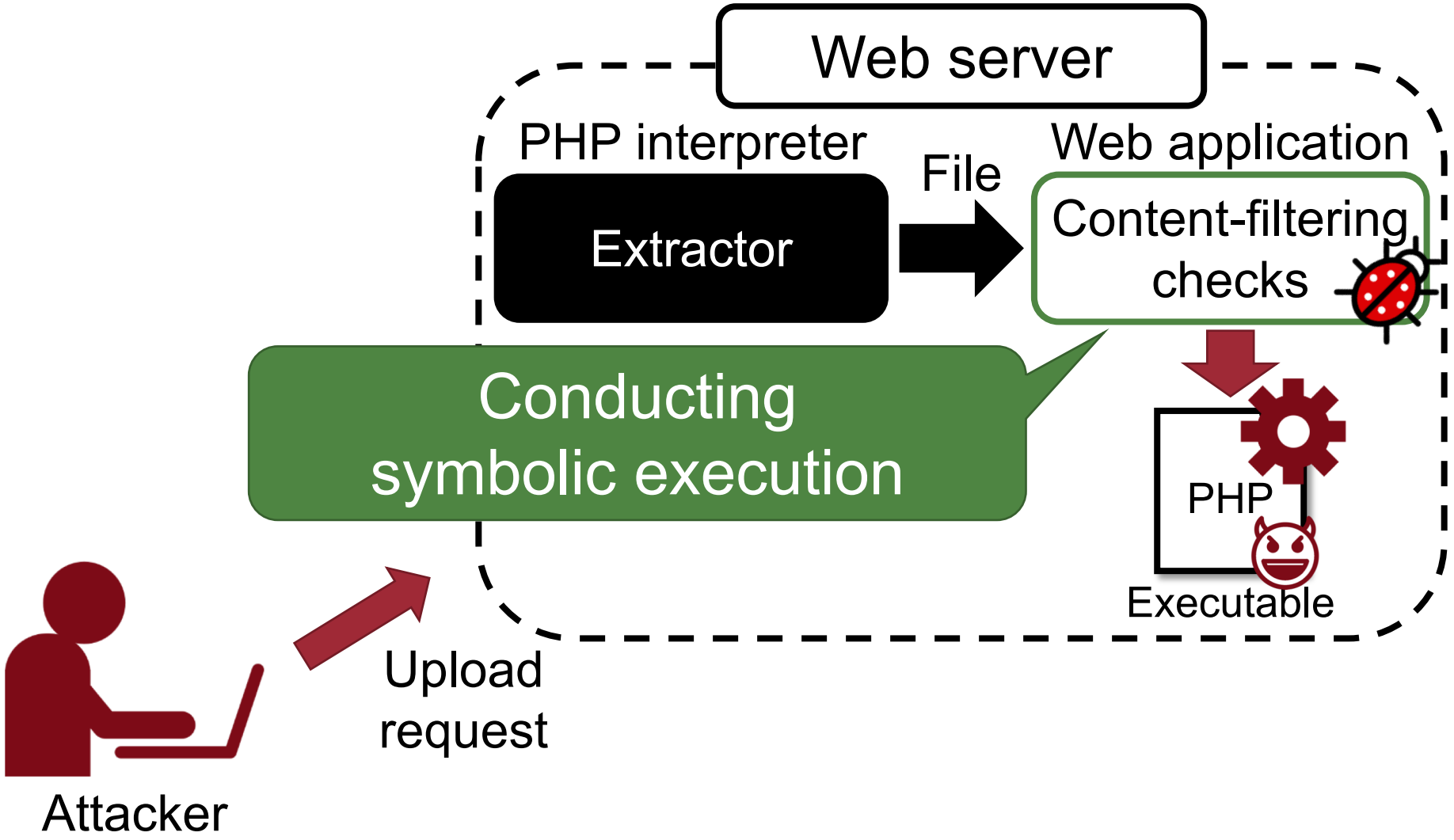
# Open Science



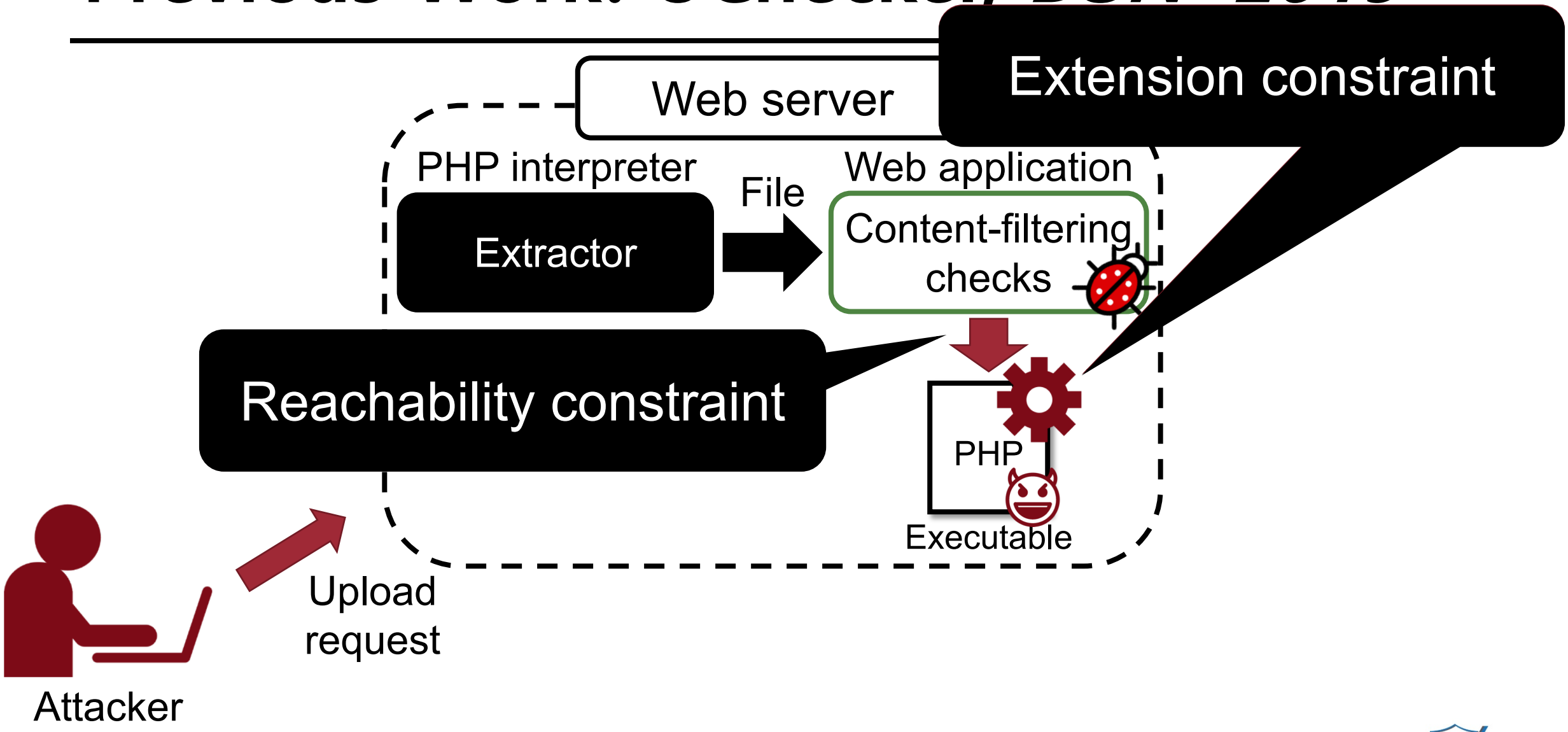

https://github.com/WSP-LAB/FUSE

# Question?

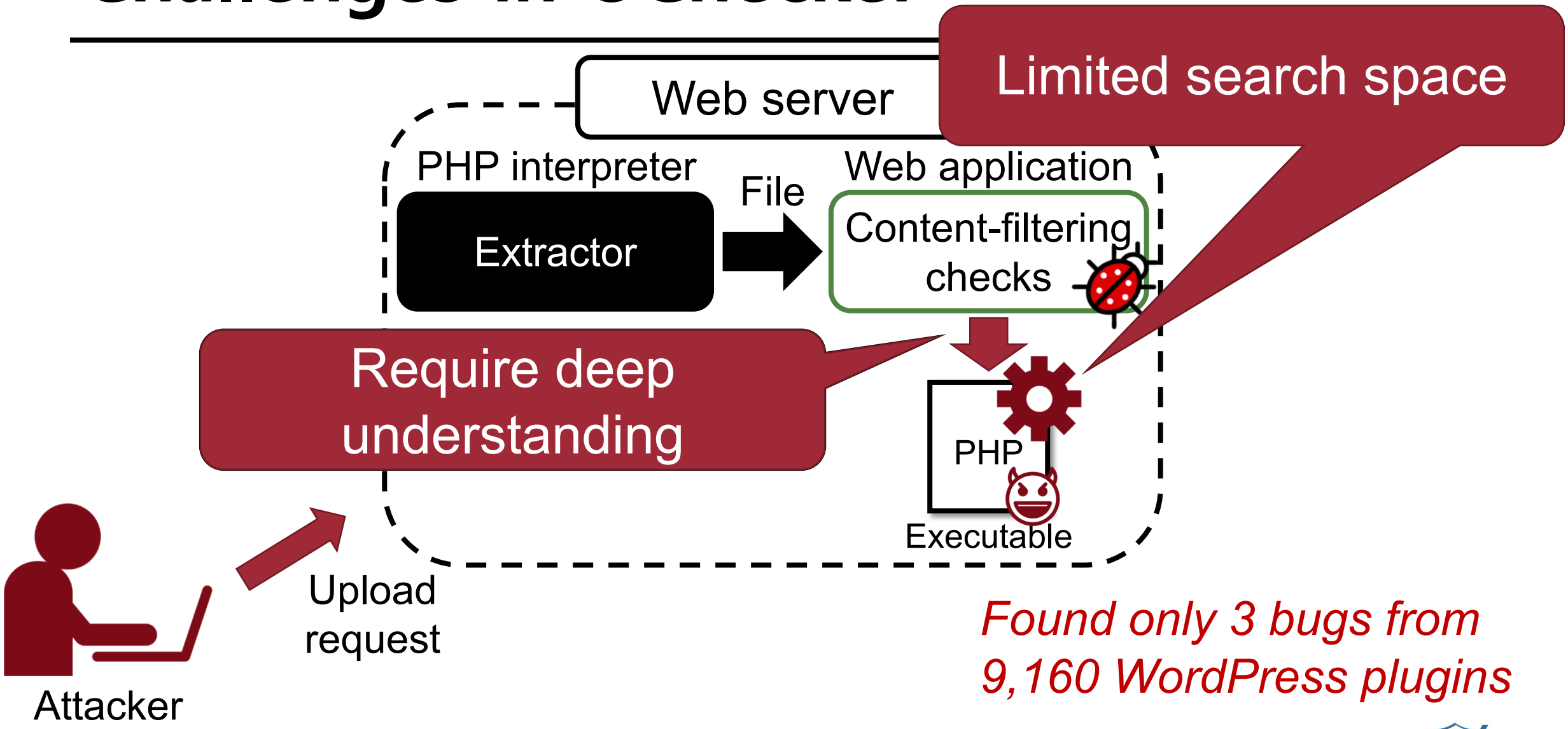
# Previous Work: UChecker, *DSN '2019*



# Previous Work: UChecker, *DSN '2019*



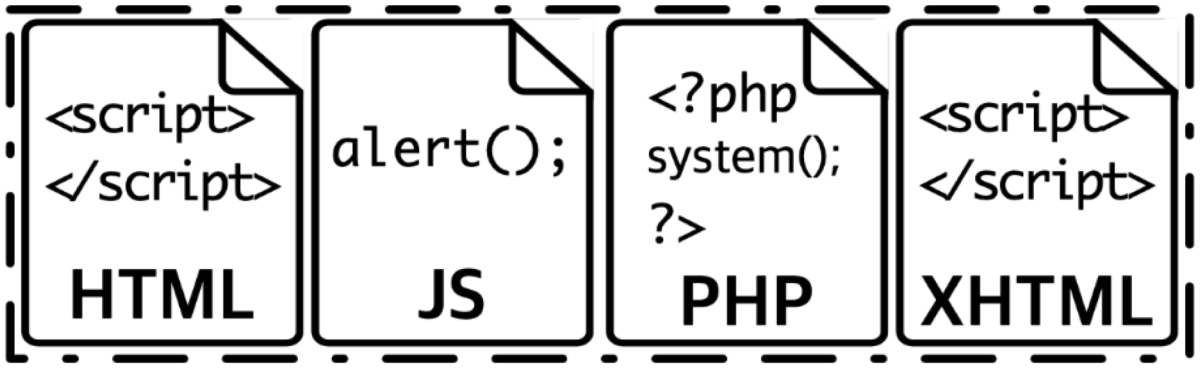
# Challenges in UChecker




*Found only 3 bugs from 9,160 WordPress plugins*

# Seed Files

- We selected these file types because they are directly involved in code executions (CEs) or potential code executions (PCEs) in Web execution environments.



# Why XHTML Seed File?

- More structural
- More strict for grammar – Mutation is different
- MIME type is different with html

# vs. State-of-the-Arts

- First attempts to find UEFU vulnerabilities by leveraging penetration testing
- Baseline for further research
- More comprehensive mutation operation
  - M5: Replacing PHP tags with short tags
  - M7: Removing a file extension
  - M9: Prepending an HTML comment
  - M10: Changing a file extension to an arbitrary string
  - M13: Appending a resource header
- Comprehensive combination of mutation operation
- File monitoring system

# vs. Symbolic Execution

- Modeling the relationships of the symbol for various PHP built-in function is different
- Hard to pinpoint reachable sink from the source
- Path explosion

- Penetration testing is more efficient

# Admin Required

- Among the 30 UEFU vulnerabilities, 14 bugs required an administrator-level privilege for their exploitation

- Web hosting administrator often separates application administrators from the host management

- CSRF…,

# Execution Constraints

- We manually analyzed the source code of Chrome 74, Firefox 68, eight different versions of Apache `mod_php` modules, and PHP 5.6 interpreter engines

# Execution Constraints

- A PHP interpreter executes a PHP file that contains the PHP start tag (i.e., `<?php` or `<?`)
- An Apache `mod_php` module requires an executable PHP file to have one of the seven PHP-style file extensions (e.g., php3, phar) for its execution via direct URL invocations
- In the Chrome and Firefox browsers, we also identified that an executable HTML file must start with pre-defined start tags within its first 512 bytes with subsequent valid HTML code
- An executable XHTML file shares the same constraints as the HTML case but requires the presence of xmlns tags
- …

| OP | Description | Seed File(s) | Objectives |
|---|---|---|---|
| M1 | Prepending a resource header | PHP, HTML | 2 |
| M2 | Inserting a seed into metadata | PHP, HTML, JS | 2 |
| M3 | Changing the content-type of a request | PHP, HTML, XHTML, JS | 5 |
| M4 | Changing a file extension | PHP, HTML, XHTML, JS | 3 |
| M5 | Replacing PHP tags with short tags | PHP | 4 |
| M6 | Converting HTML into EML | HTML, XHTML | 2, 3 |
| M7 | Removing a file extension | PHP, HTML, XHTML, JS | 3 |
| M8 | Converting a file in SVG | HTML | 3 |
| M9 | Prepending an HTML comment | HTML, XHTML | 2, 4 |
| M10 | Changing a file extension to an arbitrary string | PHP, HTML, XHTML, JS | 3 |
| M11 | Converting a file extension to uppercase | PHP, HTML, XHTML, JS | 3 |
| M12 | Prepending a file extension | PHP, HTML, XHTML, JS | 3 |
| M13 | Appending a resource header | PHP, HTML, XHTML, JS | 2 |

TABLE I: List of mutation operations for each seed file.

# Chain Length

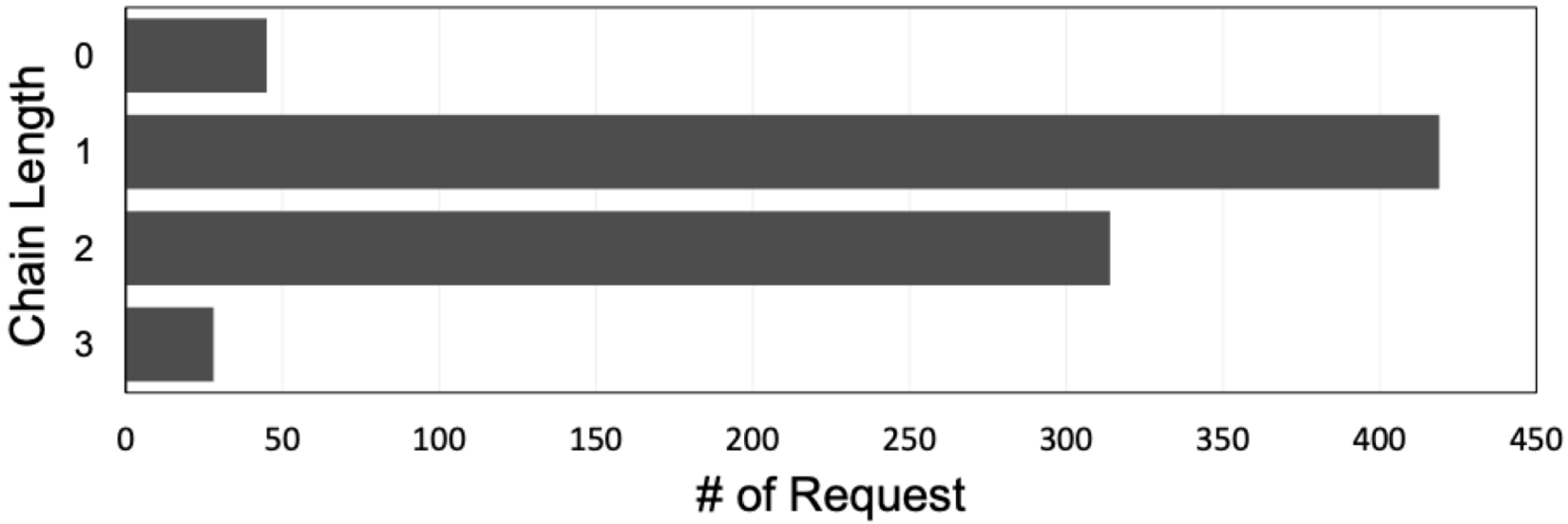



Fig. 8: The chain length frequency of successful chains.

# CVEs

| Idx | Application | Causes | CVEs | CVSS v3.0 Base Scores | Reporting Status |
|---|---|---|---|---|---|
| 1 | Textpattern | #1 | - | - | Waiting response |
| 2 | Composr | #1 | - | - | Working on patching |
| 3 | Elgg | #1 | CVE-XXXX-XXXX | R | Waiting response |
| 4 | ECCube3 | #1 | CVE-XXXX-XXXX | R | Declined |
| 5 | DotPlant2 | #1 | - | - | Waiting response |
| 6 | Codiad | #1 | - | - | Declined |
| 7 | Bludit | #3 | - | - | Waiting response |
| 8 | CMSMadeSimple | #1, #3 | CVE-XXXX-XXXX<br>CVE-XXXX-XXXX | R<br>R | Waiting response |
| 9 | Pagekit | #3 | - | - | Under discussion |
| 10 | CMSimple | #3 | CVE-XXXX-XXXX | R | Working on patching |
| 11 | Concrete5 | #3 | CVE-XXXX-XXXX | 4.8 Medium | Patched |
| 12 | OctoberCMS | #3 | - | - | Waiting response |
| 13 | SilverStripe | #3 | - | - | Under discussion |
| 14 | ZenCart | #3 | - | - | Waiting response |
| 15 | GetSimpleCMS | #1, #3 | CVE-XXXX-XXXX<br>CVE-XXXX-XXXX | 3.8 Low<br>3.8 Low | Patched |
| 16 | Subrion | #1, #3 | CVE-XXXX-XXXX | 7.2 High | Patched |
| 17 | SymphonyCMS | #1, #3 | - | - | Patched |
| 18 | OsCommerce2 | #1, #3 | CVE-XXXX-XXXX<br>CVE-XXXX-XXXX<br>CVE-XXXX-XXXX<br>CVE-XXXX-XXXX | 4.9 Medium<br>4.9 Medium<br>4.9 Medium<br>7.2 High | Waiting response |
| 19 | ClipperCMS | #1, #3 | - | - | Waiting response |
| 20 | Monstra | #1, #3 | CVE-XXXX-XXXX<br>CVE-XXXX-XXXX | 7.2 High<br>4.8 Medium | †Working on patching |
| 21 | XE | #4 | XEVE-XXXX-XXXX | - | Patched |
| 22 | WordPress | #2+#3 | - | - | Working on patching |
| 23 | Microweber | #2+#3 | - | - | Waiting response |

# Mutation Conflicts

- For a given operation (M1), we defined a conflicting mutation (M2) as when
  1. both M1 and M2 revise the same portion of a mutation vector, or
  2. M1 combined with M2 causes a CE failure, thus rendering M2 unnecessary.

| Application (Version) | Total # of Attempted Requests | CE | | | PCE | | .htaccess Uploaded | Monitor Enabled | Execution Time |
|---|---|---|---|---|---|---|---|---|---|
| | | PHP | HTML | XHTML | PHP | JS | | | |
| Bludit(3.8.1) | 117,267 | 0 | 1 | 0 | 3 | 0 | ✗ | ✓ | 37m 34s |
| Textpattern (4.7.3) | 11 | 1 | 1 | 1 | 0 | 1 | ✗ | ✗ | 0s |
| Joomla (3.9.3) | 121,117 | 0 | 0 | 0 | 28 | 2 | ✗ | ✓ | 47m 20s |
| Drupal (8.6.9) | 120,849 | 0 | 0 | 0 | 18 | 0 | ✗ | ✗ | 70m 39s |
| CMSMadeSimple (2.2.9.1) | 24,986 | 2 | 1 | 1 | 14 | 1 | ✗ | ✗ | 22m 53s |
| Pagekit (1.0.16) | 107,609 | 0 | 2 | 1 | 5 | 2 | ✗ | ✗ | 36m 59s |
| Backdrop (1.12.1) | 26,930 | 0 | 0 | 0 | 34 | 1 | ✗ | ✗ | 17m 16s |
| CMSimple (4.7.7) | 102,168 | 0 | 1 | 0 | 5 | 3 | ✗ | ✗ | 19m 3s |
| WordPress (5.0.3) | 98,730 | 0 | 4 | 4 | 43 | 8 | ✗ | ✗ | 15m 26s |
| Concrete5 (8.4.4) | 96,638 | 0 | 3 | 2 | 6 | 4 | ✗ | ✗ | 38m 59s |
| Composr (10.0.22) | 60 | 0 | 1 | 1 | 50 | 1 | ✗ | ✓ | 1s |
| OctoberCMS‡ (1.0.446) | 94,294 | 0 | 1 | 0 | 5 | 1 | ✗ | ✓ | 14m 39s |
| phpBB3 (3.2.5) | 119,796 | 0 | 0 | 0 | †21 (21) | 0 | ✗ | ✓ | 7m 42s |
| Elgg (2.3.10) | 11 | 1 | 1 | 1 | 0 | 1 | ✗ | ✓ | 0s |
| Microweber (1.1.2.1) | 47,419 | 26 | 39 | 17 | 156 | 13 | ✗ | ✗ | 25m 44s |
| XE (1.11.2) | 105,757 | 0 | †2 (1) | †2 (1) | 1 | 1 | ✗ | ✗ | 325m 51s |
| SilverStripe (4.3.0) | 87,312 | 0 | 2 | 2 | 8 | 5 | ✗ | ✗ | 100m 22s |
| ZenCart (1.5.6a) | 121,827 | 0 | 1 | 1 | 1 | 1 | ✗ | ✓ | 24m 34s |
| ECCube3 (3.0.17) | 5 | 1 | 1 | 1 | 0 | 1 | ✓ | ✗ | 1s |
| GetSimpleCMS (3.3.15) | 52,564 | 0 | 9 | 1 | 15 | 12 | ✗ | ✗ | 16m 26s |
| DotPlant2 (N/A) | 5 | 1 | 1 | 1 | 0 | 1 | ✓ | ✗ | 1s |
| MyBB (1.8.19) | 12,142 | 0 | †1 (1) | 0 | †33 (33) | †4 (4) | ✗ | ✓ | 2m 58s |
| HotCRP¶ (2.102) | 94,034 | 0 | 0 | 0 | †3 (3) | 0 | ✗ | ✗ | 257m 18s |
| Subrion (4.2.1) | 60 | 1 | 1 | 1 | 48 | 1 | ✗ | ✗ | 4s |
| SymphonyCMS (2.7.7) | 24,980 | 1 | 1 | 1 | 14 | 1 | ✓ | ✗ | 4m 18s |
| AnchorCMS (0.12.7) | 108,292 | 0 | 0 | 0 | 4 | 1 | ✗ | ✗ | 3m 28s |
| WeBid (1.2.2) | 85,317 | 0 | 0 | 0 | 6 | 0 | ✗ | ✗ | 19m 42s |
| Collabtive (3.1) | 102,097 | 0 | 0 | 0 | 1 | 1 | ✗ | ✗ | 184m 20s |
| OsCommerce2 (2.3.4.1) | 6,825 | 1 | 11 | 1 | 49 | 1 | ✓ | ✗ | 10m 31s |
| X2engine (6.9) | 71,021 | 0 | 0 | 0 | 14 | 0 | ✗ | ✓ | 71m 38s |
| ClipperCMS (1.3.3) | 63,259 | 0 | 1 | 1 | 7 | 1 | ✓ | ✗ | 18m 41s |
| Monstra (3.0.4) | 16,982 | 2 | 12 | 1 | 15 | 14 | ✗ | ✗ | 13m 56s |
| Codiad (2.8.4) | 5 | 1 | 1 | 1 | 0 | 1 | ✓ | ✗ | 0s |

# How Validator Works?

1. Check uploading
2. Extract URL
   - Common prefix of URLs
   - Upload response and summary webpage
   - File Monitor
3. Validate Bugs
   - PHP: Sting checking
   - HTML, JS, XHTML: Checks whether the Content-Type header in the response is among our selections of 10 MIME types