Informationssicherheit und IT-Forensik

University of Cologne

3-01 – Web-Security

Grundlagen

Lernziele dieser Einheit

- Die besondere Bedeutung von Webanwendungen für das heutige Leben erkennen und verstehen, welche Implikationen das für die Security dieser Anwendungen hat
- Grobkonzepte und das Prinzip der Parametermanipulation verstehen und erklären können
- Die hier gezeigten Angriffe verstehen und selbst anwenden können + Gegenmaßnahmen kennen und implementieren können

Grundüberlegungen

Geeignete Literatur

- Writing Secure Code, Second Edition, 0-7356-1722-8
- http://msdn.microsoft.com/en-us/library/ms994921 (Online und als PDF verfügbar, .NET-Security)
- Web Security Testing Cookbook
- Securing Web Services with WS-Security
- XSS Attacks, Syngress
- SQL Injection Attacks and Defense
- Hacking Exposed Web Applications 3
- Internet-Security aus Software-Sicht

- Sichere Webanwendungen

Grundüberlegungen

13-Jahres-Rückblick

- „Cross-Site-Scripting-Lücken ohne Ende“ (Heise.de, 16.06.2008)
- Zu den über 45.000 auf xssed.com gemeldeten XSS-Lücken gehörten lange Zeit auch etliche zu Verisign, McAfee und Symantec

https://www.tgdaily.com/security-features/50499-youtube-hackers-direct-bieber-fans-to-porn

Grundüberlegungen

Der erste große Web-Wurm

- Oktober 2005
- Samy Kamkar, 19, wollte seine Freundin und “hot chicks” beeindrucken
- Er wollte möglichst viele Freunde auf MySpace haben...
- Innerhalb von 20 Stunden hatten mehr als 1 Millionen Benutzer seinen selbst verbreitenden JavaScript-Wurm aufgerufen
- Funktion:
  - Nachricht auf eigenem Profil “Samy is my hero”
  - Freundschaftsanfrage an Samy
- Noch heute enthalten tausende Profile “Samy is my hero”
- MySpace war während des “Angriffs” für einige Zeit down...

Grundüberlegungen

DidYouWatchPorn? 1/2 (mittlerweile offline)

Grundüberlegungen

DidYouWatchPorn? 2/2 (mittlerweile offline)

Grundüberlegungen

Telefon-Flirts einer Dating-Plattform

- Über 5 Jahre vertrauliche Daten im Document-Root einer deutschen Partnervermittlung/Flirt-Plattform frei abrufbar.
- Es handelte sich dabei um Voice-Nachrichten von Flirtsuchenden.
- Diese Nachrichten waren eigentlich schon lange nicht mehr aktiv... (Datenschutz)
- Damalige Google-Suche:
  - -inurl:(htm|html|php) intitle:”index of” +”last modified” +”parent directory” +description +size +mp3 +voicefile
- Lieferte Tausende Sprachnachrichten im DocumentRoot.

Grundüberlegungen

Systematisches Information Disclosure auf Photobucket

- Die hier gezeigte Library ist seit 2011 online (Stand: 2020)!

Grundüberlegungen

Zusammenhang zwischen Web-Sec und IT-Sec allgemein: Überlick

- Webanwendungen auf Anwendungsschicht
- Alle anderen IT-Systeme auf darunter liegenden Schichten
- Webanwendungen sind damit von Problemen auf allen Schichten betroffen
- Web Application Security analog zu Application Security, bezogen auf Internet- und Web-Anwendungen

TCP/IP-Schicht	Beispiel
Anwendungen (Application)	HTTP, FTP, SMTP, IMAP
Transport (Transport)	TCP, UDP
Internet (Internet)	IP (IPv4 und IPv6), ICMP
Netzzugang (Network Access)	Ethernet, Token-Ring, FDDI

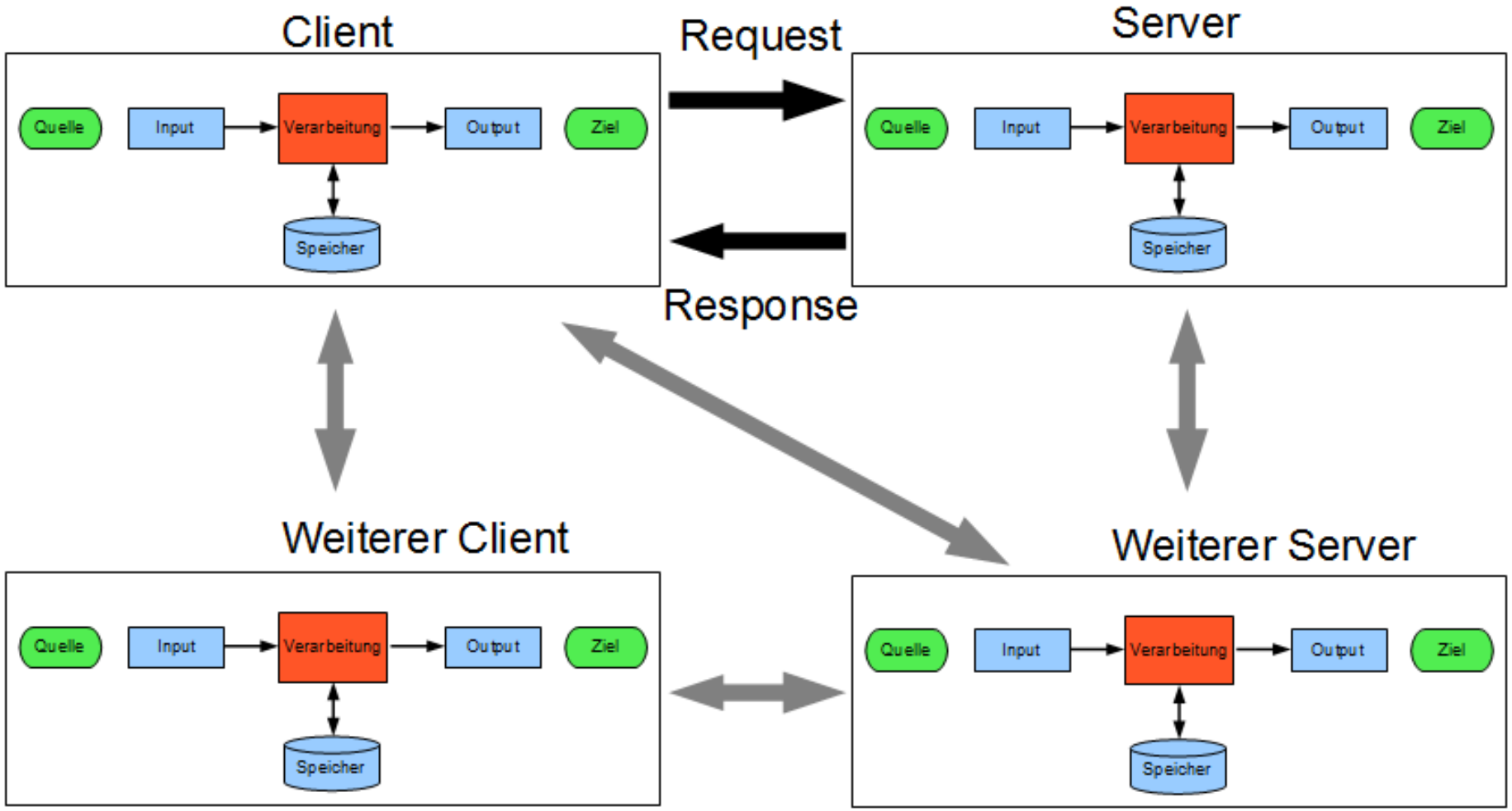
Grundüberlegungen

Zusammenhang zwischen Web-Sec und IT-Sec allgemein: Abstrakte Sicht auf den Web Technology Stack

	Windows	Linux/Unix
Applikation	VB.NET Applikation	Java EE Applikation
Middleware	.NET Runtime	J2EE Runtime
HTTP Server	Microsoft IIS	Tomcat
Betriebssystem	Windows 2016	Ubuntu Server 18.04
Netzwerkinfrastruktur (Firewall, Load Balancer, NAT, ...)		

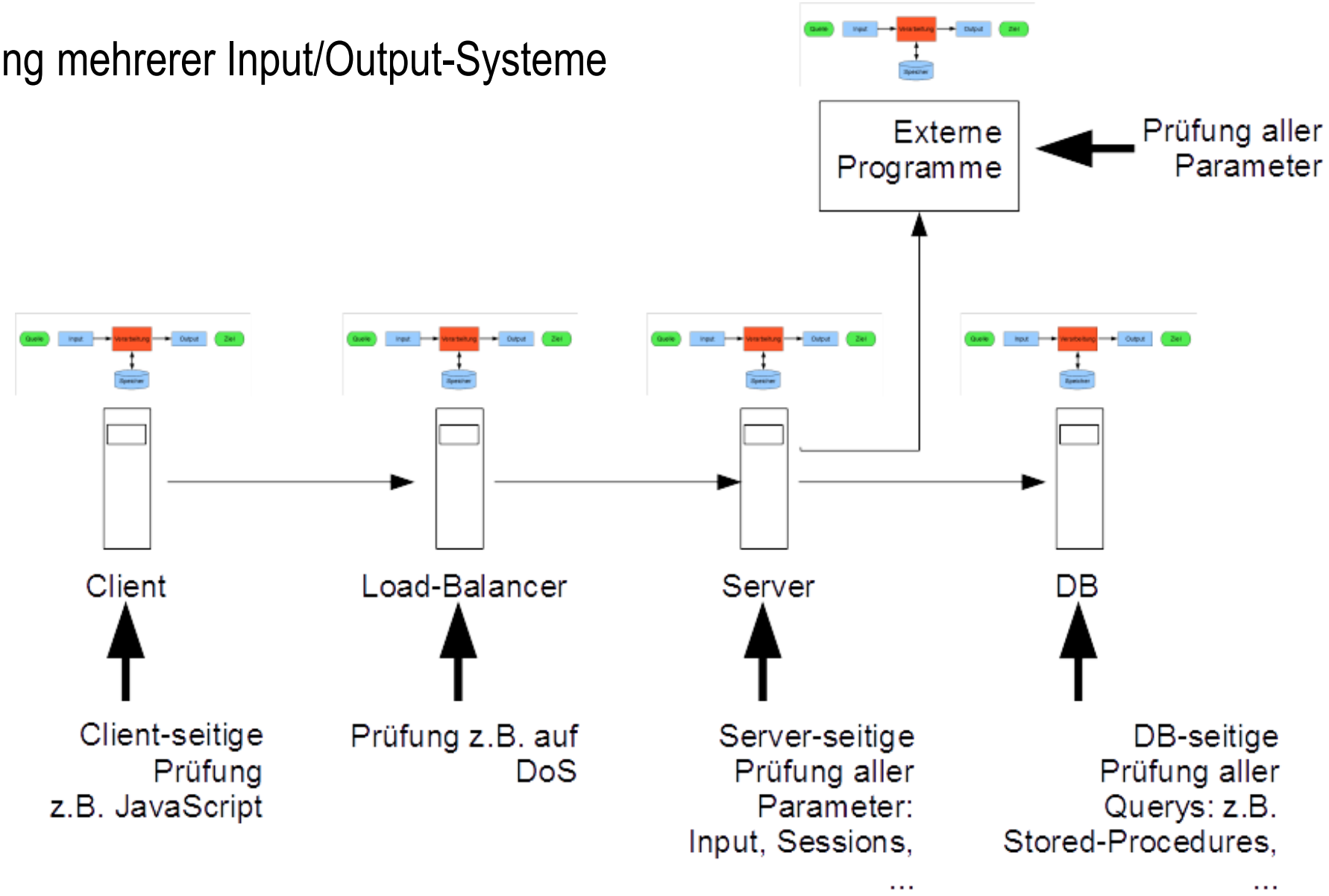
Grobkonzept Web und Grobkonzept Web-Sec

Client-Server-Interaktion: Wer hat wo den „Sündenbock“



Grobkonzept Web und Grobkonzept Web-Sec

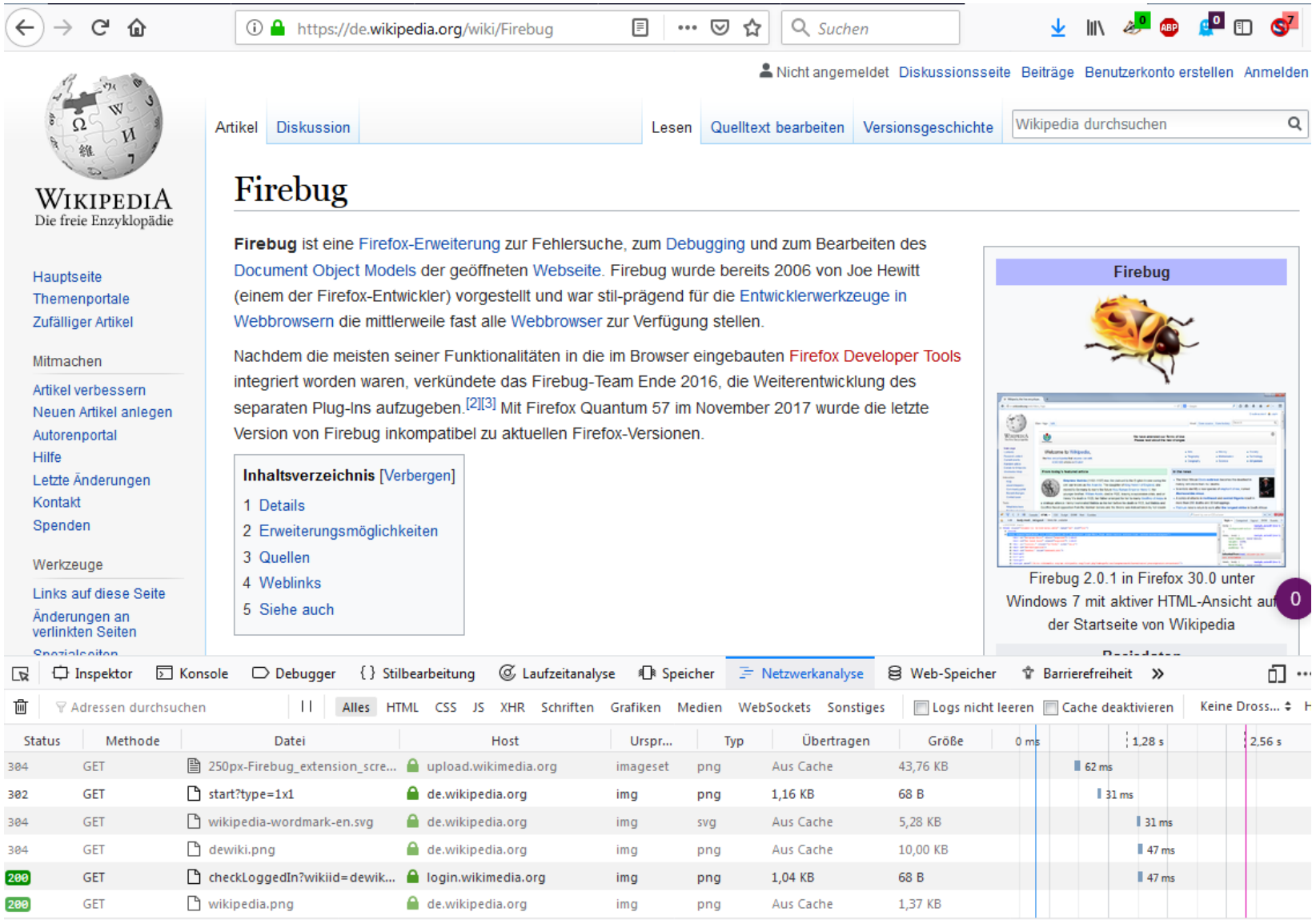
Verkettung mehrerer Input/Output-Systeme



Bewährte Mehrzwecktools kennenlernen und verwenden

z.B. Firefox Developer Tools

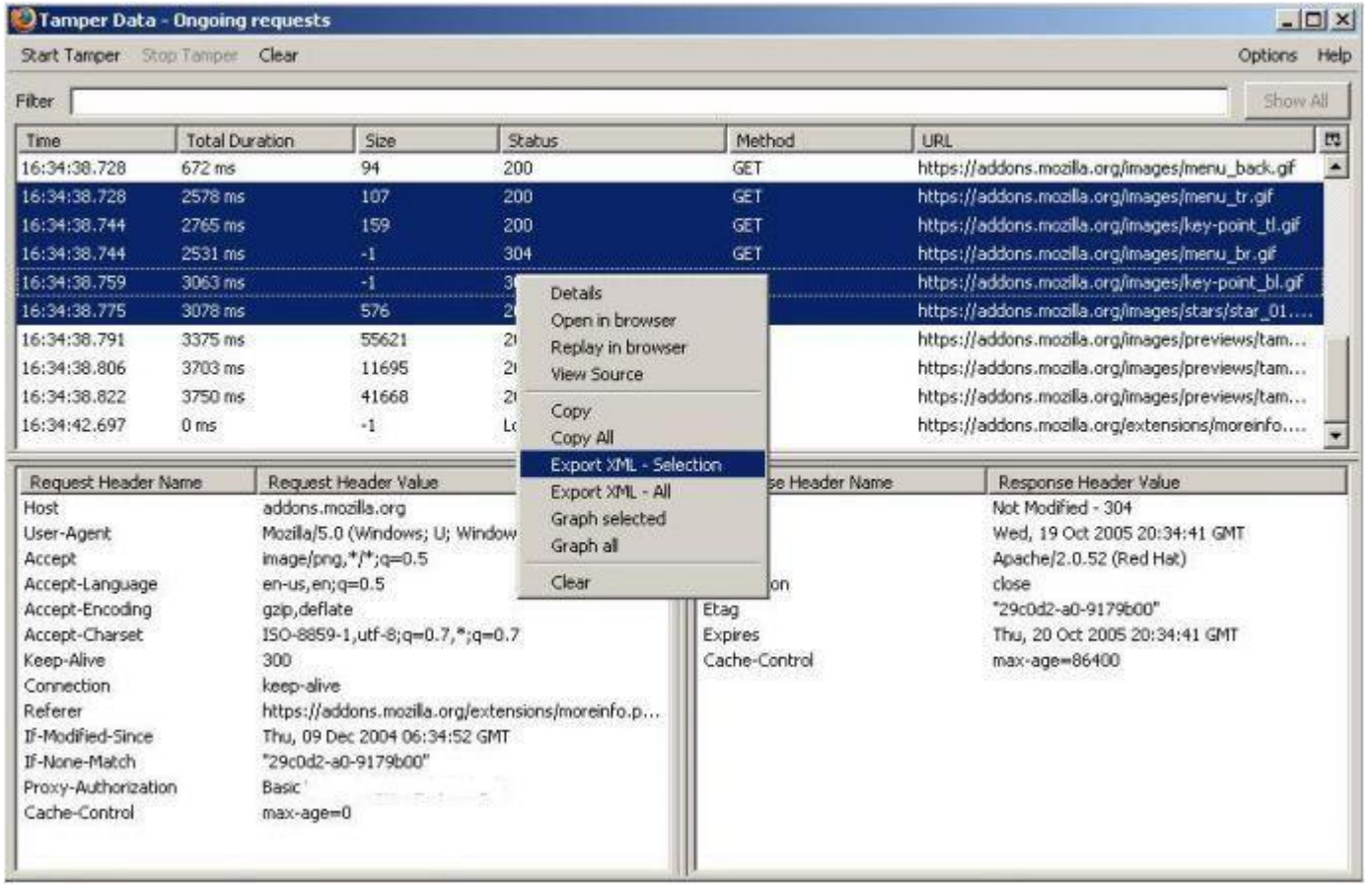
- Ursprung liegt in „Firebug“
- Funktionen z.B.
  - Netzwerkanalyse
  - Zugriff auf das Document Object Model
  - Zugriff auf JavaScript
  - Cookies und anderen lokale Speicher
  - …



Bewährte Mehrzwecktools kennenlernen und verwenden

Tamper Data

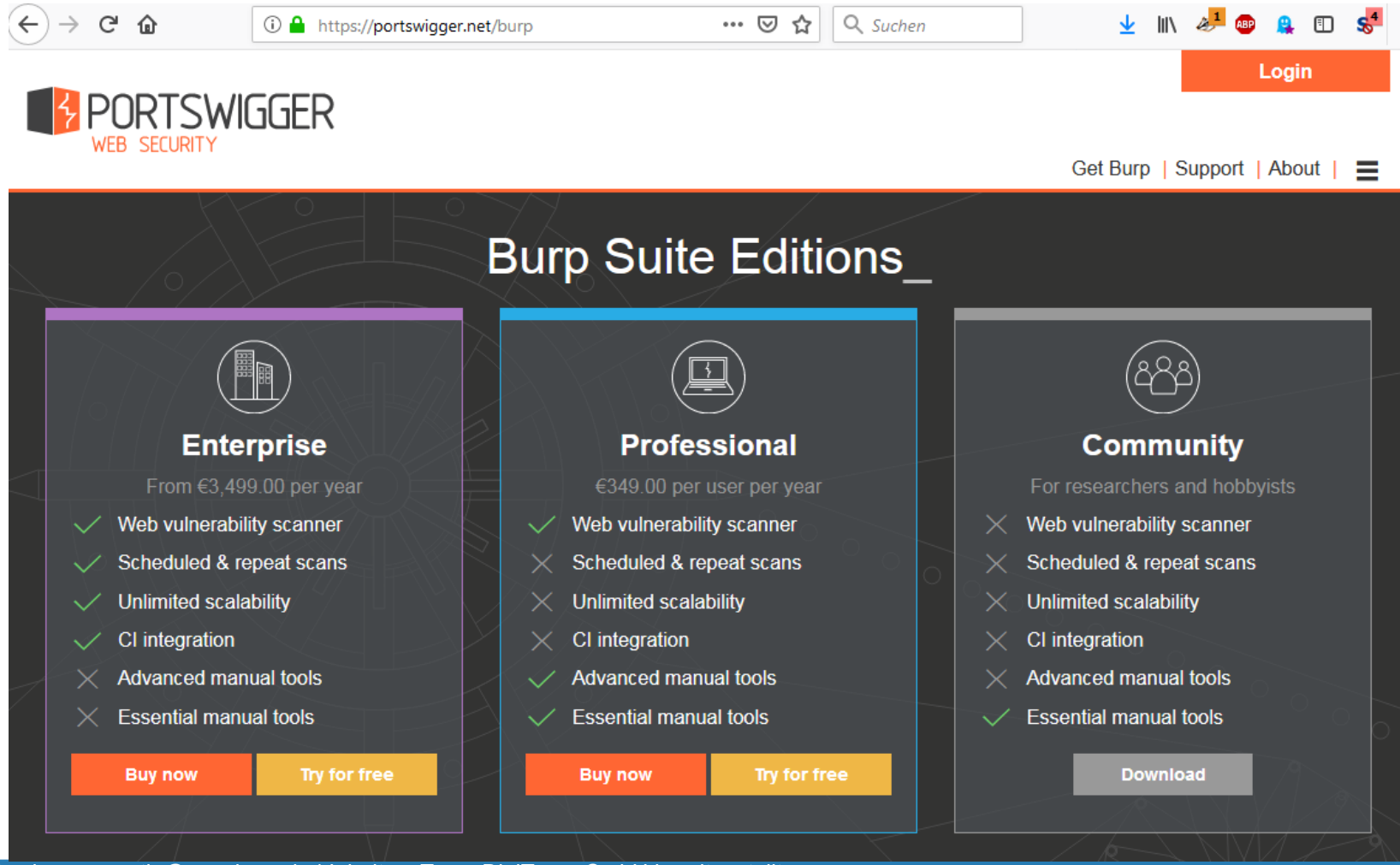
- Monitor live requests
- Edit headers on live requests
- Cancel live requests
- Redirect live requests
- https://addons.mozilla.org/de/firefox/addon/tamper-data-for-ff-quantum/



Bewährte Mehrzwecktools kennenlernen und verwenden

Burp Suite

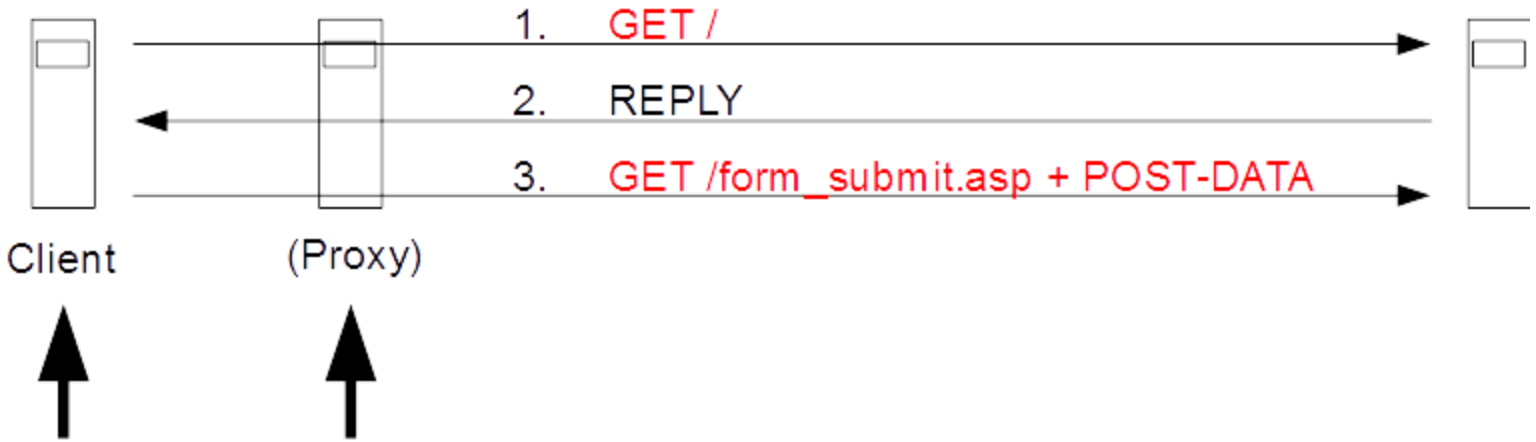
- Web-Proxy
- „Repeater“
- Kann auch SSL-/TLS-Man-In-The-Middle
- JavaScript Code-Analyse
- Kostenfreie Basisversion seit etlichen Jahren verfügbar



Bewährte Mehrzwecktools kennenlernen und verwenden

Parametermanipulation: GET und POST

- Beispiel: quantity=**3**&price=**500** -> quantity=**20**&price=**10**
- Beispiel: Header „User-Agent“, Header „Host“, ...
- Beispiel: Cookie-Daten (Session!)



Bewährte Mehrzwecktools kennenlernen und verwenden

Eine Bestellung im Webshop manipulieren

- Beispiel **http://www.jsishop.de/**
- **Achtung:** der hier gezeigte Angriff auf Elemente im Warenkorb erfolgt rein client-seitig im Browser und erfolgt daher ohne Manipulation von fremden Daten, ohne Leistungserschleichung innerhalb einer Demo-Plattform. Darüber hinausgehende Angriffe werden hier nicht gezeigt und können illegal sein – insbesondere, wenn es sich um Angriffe auf den Server und den serverseitigen Code handelt!

Cross-Site-Scripting / CSRF und DOM-Manipulation

Cross-Site-Scripting (XSS): Konzept

- XSS-Attacken gehören zu den häufigsten Angriffen, lassen sich oft leicht durchführen und werden oft unterschätzt
- Laufen im vertrauenswürdigen Kontext der betroffenen Webseite, jedoch nicht im Server sondern im Browser ab
- Angreifer übergibt dem Webserver Schadcode (meist JavaScript oder ActiveScript). Webserver übergibt diesen dem Browser oder einer anderen Komponente
- Reflektive und Persistente Angriffe

Cross-Site-Scripting / CSRF und DOM-Manipulation

Cross-Site-Scripting (XSS): Reflektiver Angriff

- **Reflektiver Angriff**: Ein Präparierter Link wird dem Opfer zugespielt, der Schadcode wird direkt nach Anklicken aktiv.

- **Servercode**:

```
$name = $_GET['name'];
print("Guten Tag Herr $name");
```

- **Angreifer**:

```
http://seite.de/skript.php?name=<script
language="javascript">top.location.href('http://www
.phishmydata.com');</script>
```

Cross-Site-Scripting / CSRF und DOM-Manipulation

Cross-Site-Scripting (XSS): Reflektiver Angriff

- Verschiedene **Varianten**, um Javascript-Code einzubinden:
  - <script>alert(„xss“)</script>
  - <script>myVar=/XSS/; alert(myVar.source)</script>
  - <s\0cript>alert('xss')</s\0cript>
  - In Bildern (IE): <img src=“javascript:alert('XSS')“>
  - https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

- URLEncode/URLDecode (Beispiel-Tools):
  - https://www.urlencoder.org/
  - https://gchq.github.io/CyberChef

Cross-Site-Scripting / CSRF und DOM-Manipulation

Cross-Site-Scripting (XSS): Persistenter Angriff

- **Persistenter Angriff**: Schadcode wird z.B. in die DB eingebracht und zu einem späteren Zeitpunkt z.B. im Browser des Administrators aktiv. Z.B. Header-XSS! Angriffe können so bis ins LAN reichen!

- **Servercode**:

```
$browser = $_SERVER['HTTP_USER_AGENT'];
$db->new_entry($datum, $browser);
```

- **Angreifer**:

```
HTML-Header: User-Agent=
<script language="javascript">
top.location.href('admin.php?action=DELETE_ALL');
</script>
```

Cross-Site-Scripting / CSRF und DOM-Manipulation

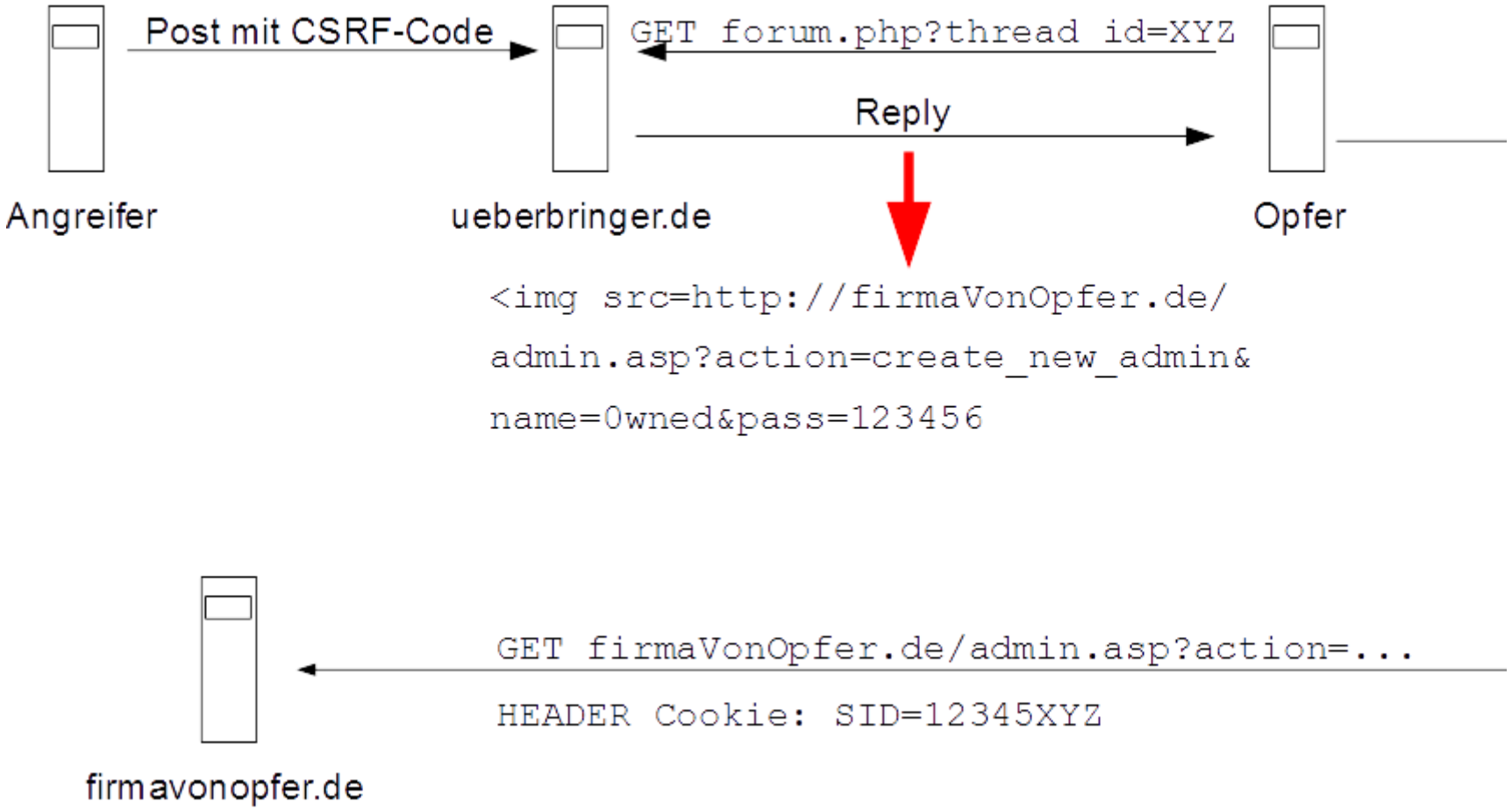
Cross-Site-Request-Forgery

- Angriffstechnik „jüngeren“ Datums (1998)
- CSRF veranlasst den Browser **ohne Wissen des Opfers Aktionen** auf einer **anderen Webseite** durchzuführen
- Dazu wird z.B. das zu der anderen Webseite gehörende Session-Cookie im Browser verwendet (Session Riding)

- Können, wie auch XSS-Angriffe, bis ins LAN vordringen, wenn Opfer-System im LAN ist und Ziel des Angriffs der LAN-Rechner ist!

Cross-Site-Scripting / CSRF und DOM-Manipulation

Cross-Site-Request-Forgery



Cross-Site-Scripting / CSRF und DOM-Manipulation

Gegenmaßnahmen

- Z.B. mit PHP
  - **strip_tags()** und **htmlentities()**
  - Berücksichtigen Sie bei htmlentities() die Option ENT_QUOTES

- **CSRF**
  - **Opfer-Plattform**
    Nur POST akzeptieren, Sicherheitsabfragen/CAPTCHA, ...
  - **Wirt-Plattform**
    z.B. getimagesize() als erster Schutz (Inhaltsprüfung)

SQL-Injection

Grundlagen

- Ausnutzung einer Sicherheitslücke in der Verbindung einer Applikation zu der dahinter liegenden Datenbank
- Mögliche Ziele dabei:
  - Auslesen von Daten
  - Verändern von Daten
  - Löschen von Daten
  - Ggf. sogar Einschleusen von Schadcode! (SELECT … INTO OUTFILE …)

SQL-Injection

Grundlagen

- Ungefilterte Weitergabe von Eingabedaten an SQL-Interpreter eröffnen die Möglichkeit eigene SQL-Abfragen zu modellieren
  - Etwa mit Hilfe des Semikolon ließe sich eine SQL-Abfrage abschließen und dahinter eine neue Abfrage modellieren
  - Ein WHERE könnte z.B. durch ein OR so manipuliert werden, dass eine andere oder gar jede Bedingung erfüllt wäre
- Angreifer suchen dabei gezielt nach anpassbaren Parametern und beobachten das Verhalten der Applikation.
- Interessant dabei sind vor allem deskriptive Fehlermeldungen, die von der Datenbank zurückgeben werden!

SQL-Injection

Einfaches Beispiel in PHP

```
$id = $_GET('id');
$result = mysql_query(„SELECT * FROM user WHERE id=$id“);
```

- Eine ID wie etwa **`1 OR 1=1`** würde in diesem Falle alle user ausgeben, da das zweite Statement der ODER-Verknüpfung immer gültig ist.
- Ganz andere Auswirkungen hätte etwa `1; DROP TABLE user;...`
  - Aber: z.B. MySQL bietet per Default keine Multi-Statements

SQL-Injection

Weiteres Beispiel 1/3

- ```
  $id = $_GET['id'];
  $query = „SELECT name, adresse FROM user
      WHERE id = $id“;
  $result = mysql_query($query);
  ```

- skript.php?id=4711 OR 1=1
- skript.php?id=3+1 → 3+1=4: mathematische Operation
- skript.php?id=4711'3434 → Fehlermeldung? → SQL-Inject, oder Blind SQL
  ```

SQL-Injection

Weiteres Beispiel 2/3

- ```
  $id = $_GET['id'];
  $query = „SELECT name, adresse FROM user
      WHERE id = $id“;
  $result = mysql_query($query);
  ```

- skript.php?id=**1 AND 2=3 UNION SELECT name, handy FROM user**
- SELECT name, adresse FROM user
  WHERE id = 1 AND 2=3
  UNION SELECT name, handy FROM user
  ```

SQL-Injection

Weiteres Beispiel 3/3

- ```
  SELECT name, adresse FROM user
       WHERE nachname LIKE  '$nachname';
  ```
- ```
  SELECT name, adresse FROM user
       WHERE nachname LIKE  '%'
       UNION SELECT benchmark(666,MD5('gemein')),
            null -- '
  ```

SQL-Injection

Besonderheit: Blind SQL-Injection

- Zur Erkennung, ob eine Webseite anfällig ist, dient häufig eine ausgegebene Fehlermeldung der SQL-Abfrage!

- Eine „Blind Injection“ hat diesen Vorteil nicht. Allerdings können erfahrene Angreifer anhand von Verhaltensweisen, etwa Ladezeiten, oder durch die Website generierte Fehlermeldungen darauf schließen, ob **und wie** eine Abfrage ausgeführt wurde oder nicht.

SQL-Injection

Durchführung

- Weitere denkbare Angriffe:
  ```
  id=30 or id=31
  id=30; drop table entries;
  id=30; GRANT ...
  SELECT * FROM auth WHERE Owner_ID=123 or 1=1
  ```
- **Oder**:
  select * from auth where username = '$username' and password = '$password‘

  $username = admin' /*
  => select * from auth where username = 'admin' /* ...

- Einfache Schutzmaßnahmen
  - z.B. $id = (int)$id

SQL-Injection

Gegenmaßnahmen

- Prüfen von Eingabedaten! Etwa durch Filtern oder Escapen von Metazeichen, wie ; oder '

- Effektiver: **Prepared Statements**. Vorkompillierte/vorbereitete Anweisungen für die Datenbank, die keine direkten Parameter, sondern nur Platzhalter enthalten
  - Parameter werden vom DBMS zuverlässig geprüft
- Zusätzlich Benutzerrechte der Webapplikation einschränken, so dass nur Privilegien vergeben werden, die tatsächlich gebraucht werden.

SQL-Injection

Gegenmaßnahmen – prepared Statements - Beispiel

```
<?php
$stmt = $dbh->prepare("SELECT user, password FROM tbl_user
WHERE (user=:user)");
$stmt->bindParam(':user', $user);
// eine Zeile abfragen
$user = 'Alice';
$stmt->execute();
// eine weitere Zeile mit anderen Werten abfragen
$user = 'Bob';
$stmt->execute();
?>
```

Sessions

Session Hijacking

- Hohes Schadenpotential, wenn Session in URL kodiert ist
  - Link wird per Mail verschickt
  - Link taucht in Suchmaschinen auf!
  - Link taucht in einem Serverlog im Referrer auf
  - => Besser per Cookie übermitteln
- Angriff durch XSS-Attacke, Session-Riding durch CSRF
- Problem, wenn bei Wechsel von HTTP<->HTTPS keine neue Session generiert wird -> **Sniffing**

Sessions

Session Fixation

1. Angreifer baut neue Session auf
2. Angreifer schickt Opfer Link mit Session-ID
3. Opfer verwendet daraufhin diese Session-ID und logged sich ein
4. Angreifer hat ebenfalls die ID und damit den Account übernommen

- Abhilfe:
  - Session-ID neu generieren vor Login
  - Benutzername/Passwort erneut abfragen vor kritischen Aktionen

Mail-Header Injection

Konzept

- **Mail-Header Injection** ist nach wie vor ein gefährliches Thema
- In Header-Felder, deren Werte aus User-Eingaben bestehen, werden weitere Header eingeschmuggelt
  - Beispiel: **Cc:** und **Bcc:** oder komplette weitere Mails
- Methode: **%0A** z.B. in „From“-Feld einfügen und dann neuen Header hinzufügen

Mail-Header Injection

Beispiel

```
From: gibt@es.nicht\r\n
To: spam@opfer.de\r\n
BCC: noch_mehr@opfer.de\r\n
Subject: Buy cheap Lolex watches\r\n
Buy genuine Lolex watches here:\r\n http://fakewatch.com\r\n
Only 20$ each!\r\n
.\r\n
To: support@unternehmen.de
Subject: Supportanfrage
...
```

File Inclusion

Local File Inclusion

```
<?php
  $datei = $_GET['datei'];
  if(isset($datei))   {
      include("content/$datei");
  }   else   {
      include("default.php");
  }
  ?>
```

- Ergänzend: Poison Null Byte-Exploit

```
$datei = $_GET['datei'];
require_once("/var/www/$datei.php");
```

File Inclusion

Remote File Inclusion

- Möglichkeit, Daten von einem externen Server in die verwundbare Applikation nachzuladen
  - z.B. mittels erlaubtem allow_url_fopen (PHP)

File Inclusion

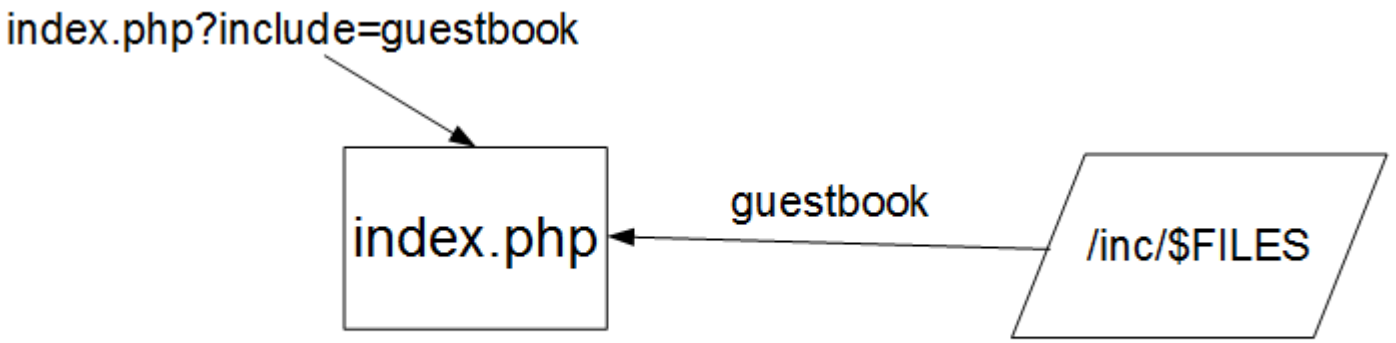
Gegenmaßnahmen

- Idealerweise nur White-List-Ansatz (evtl. in der Session eine Zuordnung Schlüssel->Wert: [1] = „/etc/datei“
- $datei = str_replace(chr(0), '', $string);

Remote Shell (Beispiel mit PHP)

Beschreibung für remote dynamisch nachgeladenen Code

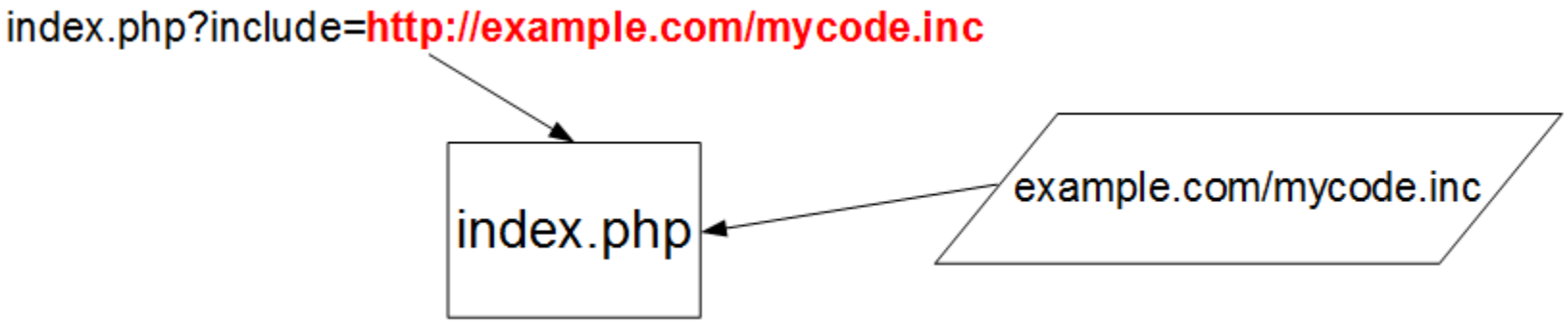
```
<?php
        $my_include = $_GET["include"];
        include($my_include);
?>
```



Remote Shell (Beispiel mit PHP)

Beschreibung für remote dynamisch nachgeladenen Code

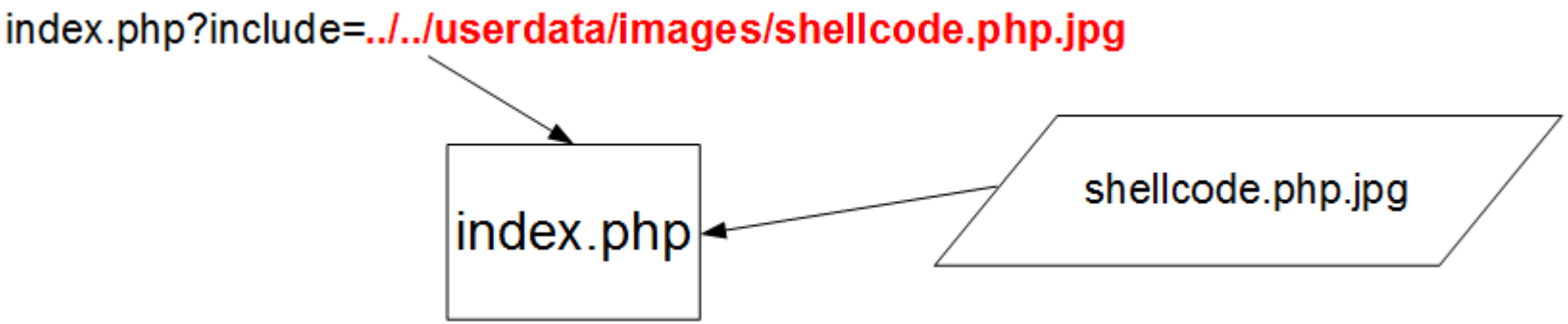
```
; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
allow_url_fopen = On

; Whether to allow include/require to open URLs (like http:// or ftp://) as files.
allow_url_include = On
```



Remote Shell (Beispiel mit PHP)

Beschreibung für zunächst hochgeladenen und dann lokal ausgeführten Code

1. Upload-Funktion einer Webanwendung nutzen
   z.B. shellcode.php.jpg als Avatar-Bild hochladen
2. Diesen Code dann in einer Local-File-Inclusion-Attacke nutzen



Gegenmaßnahmen

Überblick

- Schutzbedarf feststellen/festlegen
- Low Hanging Fruits
- Security by Obscurity
- 4-Augen-Prinzip
- Genug Zeit, Gründlichkeit
- Misstraue jedem Input!
- Externe Audits

Gegenmaßnahmen

Überblick

- Alles aus dem DocumentRoot rausnehmen, was nicht unbedingt dort sein muss
- Frameworks bauen oder fertige Frameworks verwenden (z.B. Symfony oder eins der dutzenden anderen guten und bewährten Frameworks)
- MVC-Konzept umsetzen (Model-View-Controller)
- Erwartete Inhalte festlegen (int? float? String? XY-Dimension? Datenvolumen? Stringlänge? ArrayCount?) und erzwingen:
  - White-List-Ansatz
  - Black-List-Ansatz
  - Reguläre Ausdrücke
  - „Heilen“ vs. Stoppen

Gegenmaßnahmen

Überblick

- Sonderzeichen erkennen und unschädlich machen
  - HTML-Entities, Entities
  - UTF-8/UTF-16
  - Nullbytes
  - ; ' # „ - \n \r < > …
  - Non-printable ASCII-Chars
- Nicht strippen/entfernen sondern ersetzen (ss*script*cript)
- Immer korrekte Encodings und MIME-Types setzen

Gegenmaßnahmen

Überblick

- Nur generische Fehlermeldungen anzeigen (intern jedoch loggen)
- Prepared Statements und Stored Procedures (Achtung! Auch dort SQL-Injection-Gefahr)
- CSRF-Tokens
- Kapselung einzelner Komponenten
- Auch intern gegen den Input anderer Komponenten prüfen
- Keine vertraulichen Daten unverschlüsselt an extern rausgeben (z.B. Credentials in Cookies nur kryptiert)

Gegenmaßnahmen

Überblick

- Missbräuchliche Verwendung verhindern (CAPTCHAs)
  - (Achtung: gute Systeme verwenden!)
- Missbräuchliche Verwendung erkennen und eskalieren (Mehr als X Requests pro Zeiteinheit? Durchschnitszahlen?)
- ~~Adobe Flash *absichern* nicht mehr verwenden~~
- Same Origin Policy berücksichtigen
- Auch an den Client denken! Nicht nur die Webanwendung und die DB sind gefährdet.
- Sicherheit im kompletten Lebenszyklus berücksichtigen und umsetzen

Gegenmaßnahmen

Überblick

- Jedem Input misstrauen
  - Auch Header, Cookies, ALLES kann manipuliert sein
- AJAX/XMLHttpRequests: Was passiert, wenn jemand XHRs direkt zur Anwendung schickt?
  - Auch hier immer und jederzeit dem Input misstrauen
- Sichere Hashing-Algorithmen inkl. Salting verwenden
- Daten nicht per GET und POST akzeptieren
- Schreibende Aktionen eigentlich nur per POST akzeptieren

Gegenmaßnahmen

Überblick

- Nicht PHP/JSP/... ist schlecht, sondern die jeweilige Programmierung → Die Verantwortung nicht jemandem anders zuschieben
- Auch auf System-Aufrufe achten (dort auch escapen/filtern)
- Immer nur so wenig Rechte vergeben, wie möglich und so viele, wie nötig
- Nur so wenig Informationen preisgeben, wie möglich und so viele, wie nötig
  - Keine fortlaufenden IDs, sondern Hashed-IDs
  - Fehler bei Login? → *„Benutzername ODER Passwort falsch“*

Gegenmaßnahmen

Überblick

- Das Backend härten!
- Auf Code achten, der von Dritten eingebunden wird (<script src=“http://extern“>)
- Keine unnötigen Versions- und Bannerinformationen preisgeben
- Bei Team-Meetings gemeinsam über Sicherheit und Sicherheitsfragen sprechen

Gegenmaßnahmen

Überblick

- **Data Provenance**
- Vor kritischen Aktionen rückfragen und evtl. sogar das Passwort erneut erfragen
- Sessions absichern (session_regenerate())
- Zufallszahlen richtig generieren (md5(time()) ist schlecht!)
- Bei hochgeladenen Dateien den DateiTYP wirklich überprüfen
- Offene Redirects verhindern
- Bei Redirects dem Benutzer eine Infoseite anzeigen, wo „die Reise hingeht“

Gegenmaßnahmen

Web/HTML als wichtiges Thema begreifen

- Bei Cookies die Flags „HTTPOnly“ und „Secure“ setzen
- Besonders aufpassen bei WYSIWIG-Editoren in Webseiten für User Generated Content
- Daran denken, dass HTML/HTTP auch in anderen Kontexten als Web eine Rolle spielen
  - Kaffeemaschinen
  - PKW-Navi-HeadUnits
  - Self-Service-Stationen (z.B. Fahrkartenautomaten)
  - Backend (Webshop mit angeschlossenem ERP im Backend, ebenfalls als Webanwendung)
  - …

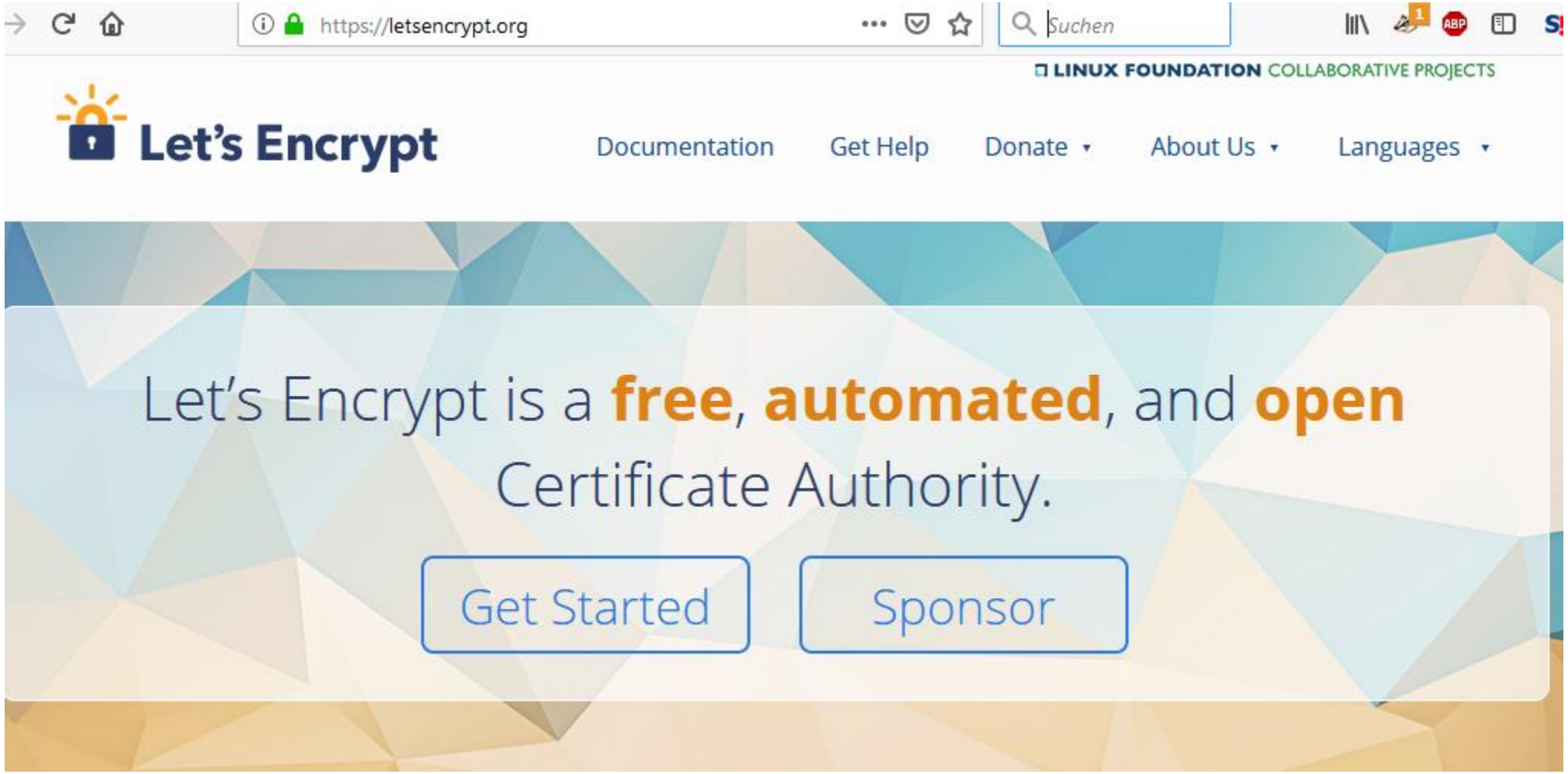
Gegenmaßnahmen

Allgemeine Design-Ansätze

- Welche Daten müssen wirklich gespeichert werden?
- Wo sollen die notwendigen Daten gespeichert werden?
  → evtl. nur verschlüsselt speichern?
  (an Nutzerpasswort gebunden)
- evtl. sogar zwei 2 getrennte Datenbanken, eine nicht über das Web zugreifbar

Gegenmaßnahmen

Traffic mit TLS verschlüsseln (z.B. mit LetsEncrypt)



Gegenmaßnahmen

Perfect Forward Secrecy

- Technik, die den Einsatz von kryptographischen Verfahren sicherer macht
- Problem: wenn jemand den verwendeten Schlüssel in Erfahrung bringen kann, kann er die damit verschlüsselten Daten entschlüsseln. Dies gilt auch rückwirkend für aufgezeichneten Datenverkehr.
- Idee: Schlüsselaustauschprotokolle so erweitern, dass diese zusätzlich zum Langzeitschlüssel (PKI) pro Sitzung einen individuellen Sitzungsschlüssel verwenden und diesen nach der Sitzung wieder „vergessen“
- So kann mitgeschnittener Traffic zukünftig auch dann nicht systematisch entschlüsselt werden, selbst wenn der Langzeitschlüssel dem Angreifer bekannt geworden ist.
- Wird z.B. von TLS/openssl unterstützt

Gegenmaßnahmen

HTTP Strict Transport Security (HSTS)

- HTTP response header „Strict-Transport-Security“
- Ein Server kann einem Client so mitteilen, dass dieser in Zukunft nur noch per HTTPS Verbindung mit dem Server aufbauen soll (bis der Wert „max-age“ erreicht ist)

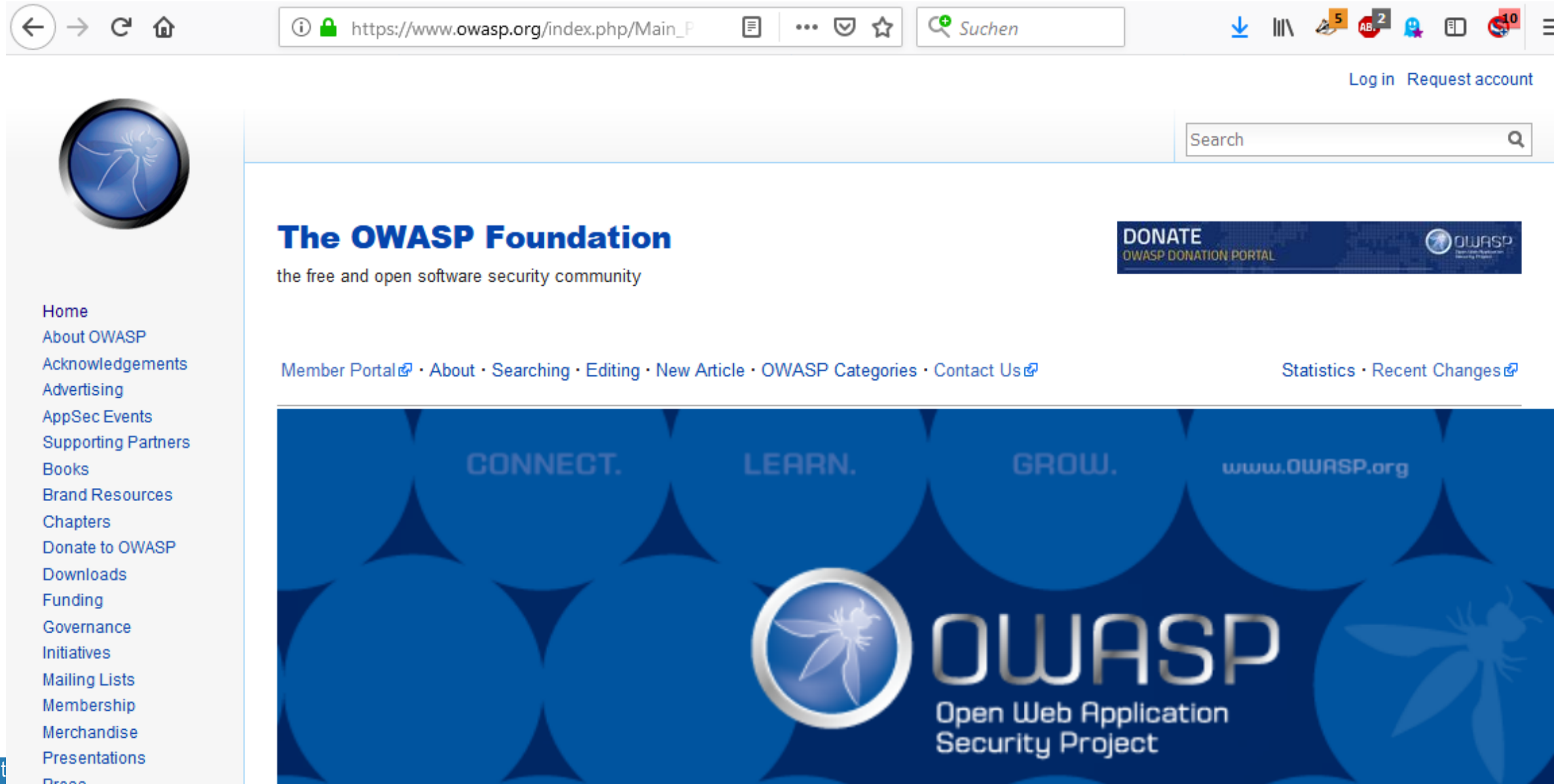
Gegenmaßnahmen

Content Security Policy (CSP)

- Headerinformationen, die Ressourcen einschränken
- Keine Inline-Skripte
  → Trennung führt zu sauberem, gut wartbarem Code
- (X-)Content-Security-Policy: script-src {Domains}

Gegenmaßnahmen

OWASP



Ein Auftrag an Sie: unser Musterfall

Wiederkehrende, vorlesungs- und übungsbegleitende Übungsaufgabe

- Überarbeiten Sie (als Hausaufgabe) Ihre Planung für die Errichtung einer „smarten“ Einbruchmeldeanlage 2.0 für das Handwerksunternehmen Ihrer Eltern
- Beantworten und begründen Sie unter eigenen Annahmen z.B. folgende Fragen:
  - Ändern Sie Ihre bisherige Planung?
  - Werden Sie eine Weboberfläche implementieren? Einen Web-Service für Anbindung an andere Dienste? Werden Sie andere, externe Dienste einbinden (z.B. Wettervorhersage)?
  - Wie wollen Sie Ihre EMA auf Web-Ebene absichern?
  - Skizzieren Sie mögliche Problemstellen. Wo könnten Probleme lauern?

Live-Hack

Mittels SQL-Injection einen Login-Schutz umgehen

- ACHTUNG:
  - SQLi ist grundsätzlich bei fremden Systemen illegal!