Web Hacking - Angriffe und Abwehr

UNIX-Stammtisch 31. Januar 2012

Frank Richter
Holger Trapp

TECHNISCHE UNIVERSITÄT CHEMNITZ

Technische Universität Chemnitz
Universitätsrechenzentrum

URZ

Motivation (1): Für uns

- Lehrveranstaltung: Techniken der IT-Sicherheit
- Arbeitsaufgaben: Webmaster, Beratung, Kurse
  - Sicherheit der eigenen Systeme prüfen
  - Angriffspotenzial verringern
  - „viel Elend“
- Anonyme Meldungen über Sicherheitsprobleme
  - „noch mehr Elend“

Motivation (2): Angreifer und deren Ziele

- **Underground Economy:**
  - Botnetze (Spam, DoS)
  - verschleierte Geld-Transaktionen
  - Spionage
- **Politische Organisationen**
  - Infrastruktur lahmlegen
- **„Kleinkriminelle“**
  - Geld (Konten, Kreditkarten, Betrug)
  - Verunglimpfung, Zerstörungswut, Erpressung ...
- **Gelegenheitstäter** – „Just for fun“

Szenario: Websurfen

DNS-Server

2. DNS

134.135.136.137

5. Interpretation, Anzeige

HTML
Bilder, CSS

JavaScript
Java, Flash

Internet

1. URL

```
http://www.domain.tld/doc
```

Webb r

Webserver

4. Verarbeitung

Dateien
(HTML, CSS, Bilder)

CGI, PHP, ASP, JSP

3. HTTP

```
A: GET /test.html HTTP/1.1

B:<html><head>
  <title>Tolle Seite</title>
```

Die brutale Art: Denial of Service

- DDOS: „Mit Bot-Netz / Cloud kriegst du alle tot.“

- Es geht auch ohne: **Hash collision DOS**
  - URL?name1=wert1&name2=wert2&...
    → `$_GET['name1'] = wert1`
    `$_GET['name2'] = wert2 ...`
  - Geschickte Wahl von ‘name‘
    → gleicher Hash → lineare Suche → CPU-intensiv
  - Alle Scriptsprachen – außer Perl
  - PHP: Fix mit `max_input_vars`

Remote / Local File Inclusion (RFI/LFI)

Ziele:

- Anzeige beliebiger (geschützter) Dateien
- Ausführen von fremdem Code

Schwachstelle:

- Ungenügende Prüfung von Parametern

Beispiel:

- www.../index.php?datei=name.html
- `<?php include($_GET['datei']); ?>`
- www.../index.php?datei=/etc/passwd

SQL Injection

Einschleusen von SQL-Befehlen in Webanwendungen

Ziele:

- Zugang zu Datenbanken: Lesen, Manipulation
- Ggf. Kode ausführen

Schwachstelle:

- Ungenügendes Maskieren von Parametern in SQL-Anweisungen

Beispiel:

- www.../db.php?id=42
- <?php mysql_query('SELECT ... WHERE id=' . $_GET['id']); ?>

Cross Site Scripting (XSS)

Unterbringen von HTML/JavaScript auf vertrauenswürdigen Webseiten

Ziele:

- Benutzerdaten - Identitätsdiebstahl
- Falsche Inhalte, Verunstaltung

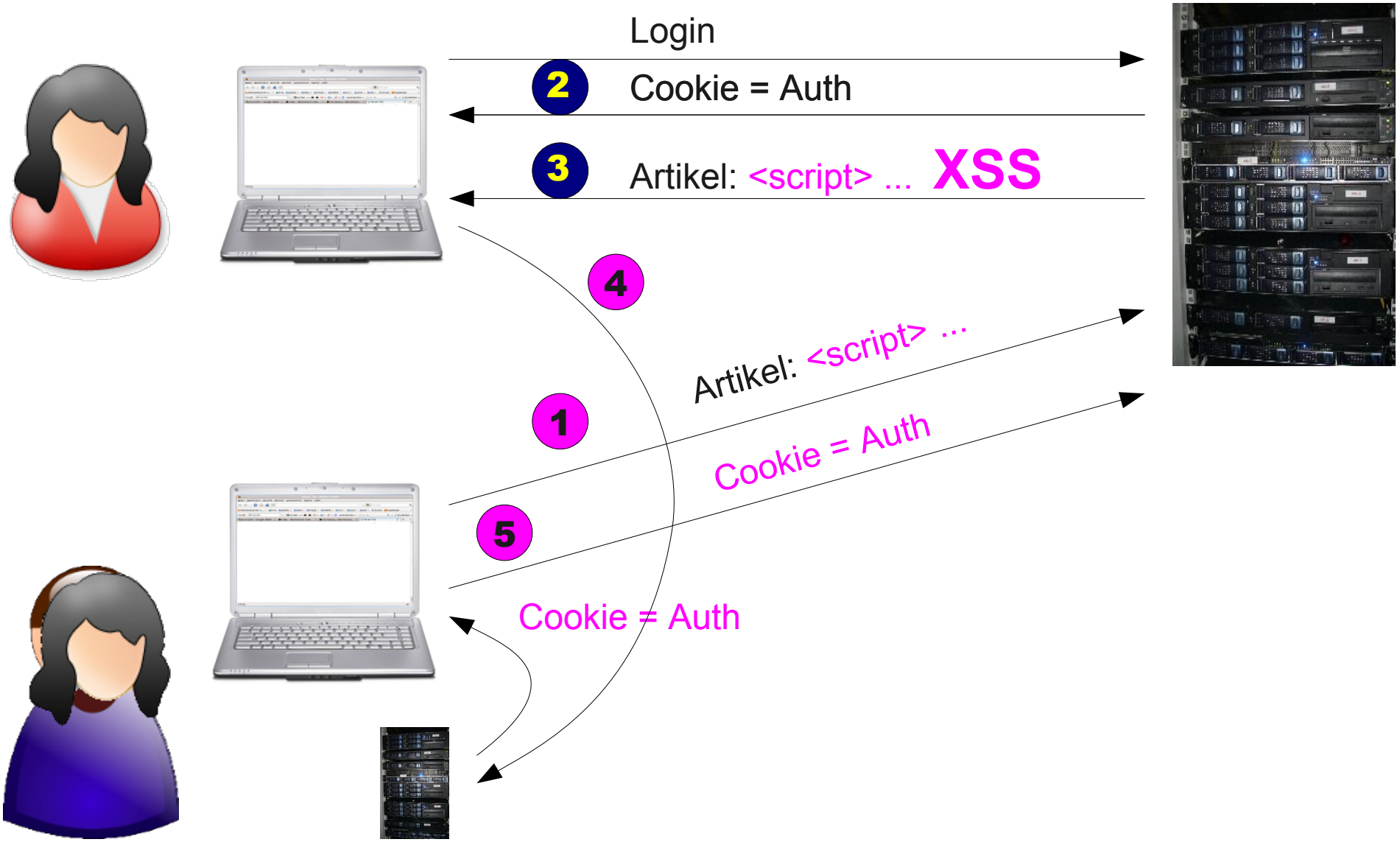
Schwachstelle:

- Ungenügendes Maskieren von Parametern in HTML-Ausgaben

Beispiel:

- www.../forum.php?text=Hallo
- <?php echo $_GET[‘text‘]; ?>

Identitätsdiebstahl mit XSS



Cross Site Request Forgery (CSRF)

Legitimen Nutzern präparierte URLs „unterschieben“

→ Transaktionen in Webanwendungen

Ziele:

- Verschleierte Aktionen ... Straftaten

Schwachstelle:

- Zustandsloses HTTP, XSS, Schadsoftware
- Mensch: Neugier, Arglosigkeit

Beispiele:

- webmail.../do?action=delete&id=42
- ebay.com/bid?auction=23&amount=1000000

Abwehr (1): Konfiguration, Programmierung

Produktionsserver:

- HTTP TRACE, Debug ausschalten
- Fehlerausgaben sehr sparsam
- Zugriff auf `.bak`, `.old` ... unterbinden

Misstraue allen Benutzereingaben/-daten:

- GET/POST, Cookies, URL-Parameter, Upload ...
- sauberes Quoting und Sanitizing:
  - Whitelists mit gültigen Werten, Syntaxkontrolle
  - etablierte Funktionen nutzen, z.B.
  - HTML-Ausgabe mit `htmlspecialchars()`
  - externe Programme via Shell: `escapeshellarg()`

Abwehr (2): Konfiguration, Programmierung

SQL

- `mysql_real_escape_string()`
- Prepared Statements
- Stored Procedures
- Beachte: String-Vergleiche sind normalerweise case-insensitive
- Geheimnisse (DB-Passwort) so sicher wie möglich ablegen

Abwehr (3): CSRF

Nutzer: „What can I do to protect myself as a user?

Nothing. The fact is as long as you visit websites and don't have control of the inner architecture of these applications you can't do a thing. The truth hurts doesn't it?“ cgisecurity.com

- Vorsicht vor URLs aus unsichere Quelle: E-Mails, Kurz-URLs
  - HTML-Mails: Laden von Bildern ... ausschalten
- Kritische Aktionen nicht parallel, abmelden, NoScript

Web-Anwendungen:

- Leichter Schutz: nur POST, kein GET
- Challenge Token pro Request zusätzlich zur Session – nicht nur Cookie

Abwehr (4): Programmierung, Administration

- Session IDs aktualisieren (gegen Session Fixation)
- HTML in (reflektierten) Eingaben unterbinden (XSS)
- aktuelle Tools/Frameworks einsetzen
- `mod_security` – Application Level Firewall

Informieren und handeln:

- Security News, neue Versionen installieren
- www.OWASP.org

Aufwändig und teuer:

- Ständige Pflege, Code Review, Penetration Tests
- Qualifiziertes Personal, Zeit

Tools und Links

- `telnet / nc`
- `wget / curl`
- Browser (mit Addons zur Manipulation des Datenverkehrs)
- Attack-/Audit-Tools, z.B. `w3af`
- Proxy, z.B. `WebScarab`, Proxy von `w3af`
- http://www.phpfreaks.com/tutorial/php-security
- Gruyere - Web Application Exploits and Defenses
  - http://google-gruyere.appspot.com