IT-Sicherheit

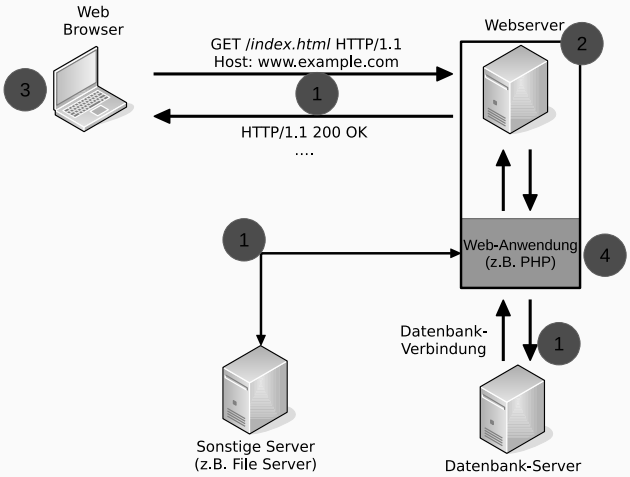
Sicherheit von Web-Anwendungen (Teil 2)

Prof. Dr. Dominik Merli, Prof. Dr. Lothar Braun

Sommersemester 2020

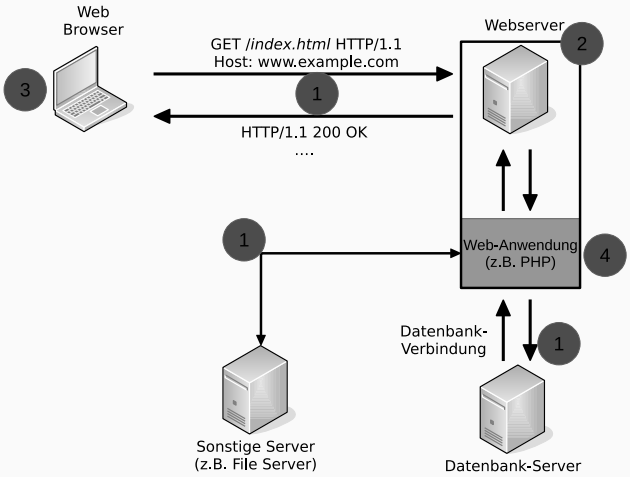
Hochschule Augsburg - Fakultät für Informatik

Recap: Typische Web-Anwendungen - Angriffsoberflächen



1) Kommunikation und Zugriffsschutz
2) Webserver-Software
3) Client-Sicherheit
4) Web-Anwendung

Recap: Typische Web-Anwendungen - Angriffsoberflächen



1) Kommunikation und Zugriffsschutz
2) Webserver-Software
3) Client-Sicherheit
4) Web-Anwendung

Schwachstellen von Web-Anwendungen

Bestandteile der Web-Anwendungen

- Standard-Webserver
  - Schnittstelle zum HTTP-Client, nimmt Anfragen entgegen und schickt Antworten
  - Liefert (in der Regel) statischen Inhalt wie Dateien aus
  - Leitet Anfragen für dynamische Inhalte an weitere Programme weiter

Bestandteile der Web-Anwendungen

- Standard-Webserver
  - Schnittstelle zum HTTP-Client, nimmt Anfragen entgegen und schickt Antworten
  - Liefert (in der Regel) statischen Inhalt wie Dateien aus
  - Leitet Anfragen für dynamische Inhalte an weitere Programme weiter

- Verarbeitung von Anfragen für dynamisch generierten Inhalt
  - Geschrieben in beliebiger Sprache, z.B. PHP, Python, Ruby, ...
  - Eingabe:
    - HTTP-Request von einem Web-Server über definierte Schnittstelle
  - Ausgabe:
    - Body der HTTP-Response an den Web-Server
    - Setzen einiger Antwort-Header möglich

OWASP Projekt

- OWASP = Open Web Application Security Project
  - Gegründet 2001
  - Gemeinnützige Organisation seit 2004
- Zweck
  - ”Be the thriving global community that drives visibility and evolution in the safety and security of the world’s software.”
- Online Präsenz
  - https://www.owasp.org

OWASP Top 10 2017

- Top 10 Risiken für Webanwendungen
  - A1 Injection
  - A2 Broken Authentication
  - A3 Sensitive Data Exposure
  - A4 XML External Entities (XXE)
  - A5 Broken Access Control
  - A6 Security Misconfiguration
  - A7 Cross-Site Scripting
  - A8 Insecure Deserialization
  - A9 Using Components with Known Vulnerabilities
  - A10 Insufficient Logging and Monitoring

- Link
  - https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

A1: Injection

- **Grundlage**
  - Manipulation von Eingabe-Daten
  - Ungeprüfte Übernahme der Eingabe-Daten in anfällige Funktionen
  - Viele unterschiedliche Injection-Typen

A1: Injection

- Grundlage
  - Manipulation von Eingabe-Daten
  - Ungeprüfte Übernahme der Eingabe-Daten in anfällige Funktionen
  - Viele unterschiedliche Injection-Typen

- Auswirkung
  - Unberechtigter Zugriff auf geschützte Daten
  - Manipulation oder Löschen von Daten
  - Ausführung von Befehlen oder eigenem Code

A1: Injection

- Grundlage
  - Manipulation von Eingabe-Daten
  - Ungeprüfte Übernahme der Eingabe-Daten in anfällige Funktionen
  - Viele unterschiedliche Injection-Typen

- Auswirkung
  - Unberechtigter Zugriff auf geschützte Daten
  - Manipulation oder Löschen von Daten
  - Ausführung von Befehlen oder eigenem Code

- Schutz
  - Sichere APIs verwenden, die keine Injection zulassen
  - Input überprüfen, z.B. mit Whitelists, Plausibilitätstest
  - Spezielle Zeichen durch Escape Syntax filtern

Injection - Beispiel: Command Injection

- Verwundbarer Code in PHP
  - `system("/bin/ls " . $_GET['directory'] . "'";`

Injection - Beispiel: Command Injection

- Verwundbarer Code in PHP
  - `system("/bin/ls " . $_GET['directory'] . "'";`
- Problem
  - Angreifer: `directory="/ ; /bin/rm -rf ."`
  - Ausführung beliebiger Kommandos durch den Angreifer

Injection - Beispiel: SQL Injection

- Verwundbarer Code in PHP
  - `mysql_query("SELECT * FROM secrets WHERE secID='" . $_GET['id'] . "'";`

Injection - Beispiel: SQL Injection

- Verwundbarer Code in PHP
  - `mysql_query("SELECT * FROM secrets WHERE secID='" . $_GET['id'] . "'";`
- Problem
  - Angreifer: `id="' or '1'='1"`
  - Rückgabe aller Geheimnisse in der Tabelle
  - Schlimmere Auswirkungen in manchen Szenarien möglich

Injection - Beispiel: Directory Traversal / Locale File Inclusion

- Verwundbarer Code in PHP
  - `fopen("dir/" . $_GET['path']);`
  - `include("dir/ . $_GET['path']);`

Injection - Beispiel: Directory Traversal / Locale File Inclusion

- Verwundbarer Code in PHP
  - `fopen("dir/" . $_GET['path']);`
  - `include("dir/ . $_GET['path']);`
- Problem
  - Angreifer: `path="../../../../../../../../etc/passwd"`
  - Directory Traversal: Unbefugtes Auslesen von Dateien
  - Local File Inclusion: Unbefugtes *Ausführen* von Dateien

Cross-Site Scripting (XSS)

- Grundlage
  - Angreifer manipuliert Daten,
    z.B. in URL (Reflected XSS) oder Datenbank (Stored XSS)
  - Web-Applikation bindet manipulierten Daten direkt ein,
    entweder serverseitig oder clientseitig

Cross-Site Scripting (XSS)

- **Grundlage**
  - Angreifer manipuliert Daten,
    z.B. in URL (Reflected XSS) oder Datenbank (Stored XSS)
  - Web-Applikation bindet manipulierten Daten direkt ein,
    entweder serverseitig oder clientseitig

- **Auswirkung**
  - Angreifer Skript-Code wird auf Server oder Client ausgeführt
  - Vertrauliche Daten werden offengelegt oder Daten manipuliert

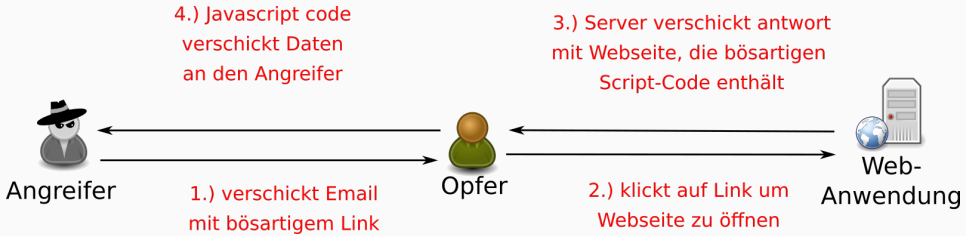
Cross-Site Scripting (XSS)

- Grundlage
  - Angreifer manipuliert Daten,
    z.B. in URL (Reflected XSS) oder Datenbank (Stored XSS)
  - Web-Applikation bindet manipulierten Daten direkt ein,
    entweder serverseitig oder clientseitig

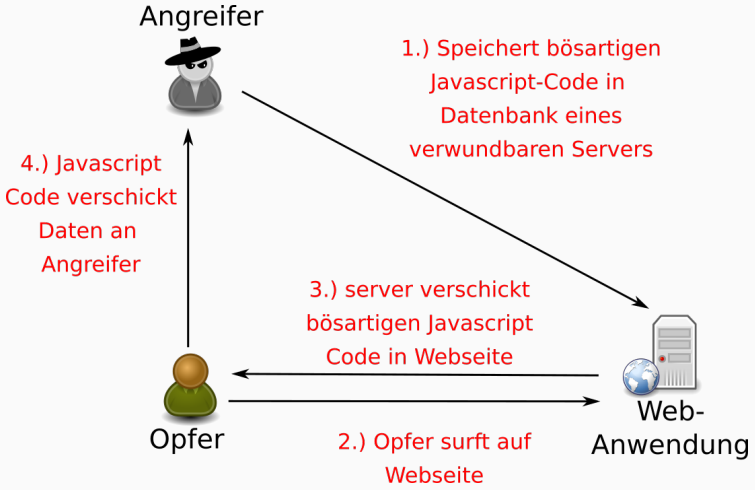
- Auswirkung
  - Angreifer Skript-Code wird auf Server oder Client ausgeführt
  - Vertrauliche Daten werden offengelegt oder Daten manipuliert

- Schutz
  - Daten aus nicht vertrauenswürdigen Quellen immer prüfen/filtern bzw. nur mit speziellen APIs verwenden

Cross-Site Scripting (XSS) - Ablauf Reflected XSS



Cross-Site Scripting (XSS) - Ablauf Stored XSS



Cross-Site Scripting (XSS) - Beispiel

- Verwundbarer PHP Code
  - Anfrage: `http://www.suchmaschine.org/?input=<Suchbegriffe>`
  - Code: `echo "Ihre Suchanfrage ist: " . $_GET['input'];`

Cross-Site Scripting (XSS) - Beispiel

- Verwundbarer PHP Code
  - Anfrage: `http://www.suchmaschine.org/?input=<Suchbegriffe>`
  - Code: `echo "Ihre Suchanfrage ist: " . $_GET['input'];`

- Problem
  - Angreifer: `input='<script type="text/javascript">alert("XSS ATTACK")</script>'`
  - Ausführung von fremdem Code im Browser des Benutzers

Cross-Site Request Forgery (CSRF)

- Grundlage
  - Angreifer präpariert manipulierten HTTP Request
  - Nutzer führt Request aus, z.B. via Bild, iframe, etc.
  - Funktioniert, wenn Nutzer autorisiert für Ziel-Applikation

Cross-Site Request Forgery (CSRF)

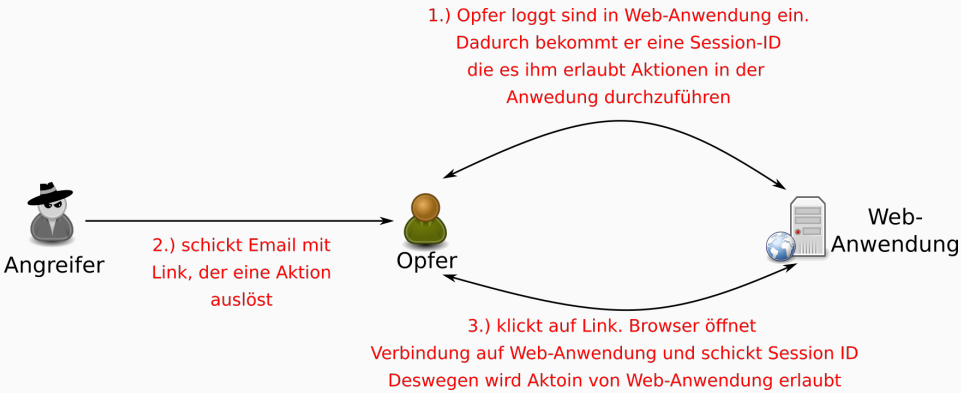
- **Grundlage**
  - Angreifer präpariert manipulierten HTTP Request
  - Nutzer führt Request aus, z.B. via Bild, iframe, etc.
  - Funktioniert, wenn Nutzer autorisiert für Ziel-Applikation

- **Auswirkung**
  - Nutzer führt Requests für Angreifer aus
  - Ungewollte Transaktionen/Kommandos
  - Weitergabe/Manipulation von sensiblen Daten

Cross-Site Request Forgery (CSRF)

- Grundlage
  - Angreifer präpariert manipulierten HTTP Request
  - Nutzer führt Request aus, z.B. via Bild, iframe, etc.
  - Funktioniert, wenn Nutzer autorisiert für Ziel-Applikation
- Auswirkung
  - Nutzer führt Requests für Angreifer aus
  - Ungewollte Transaktionen/Kommandos
  - Weitergabe/Manipulation von sensiblen Daten
- Schutz
  - Integration eines nicht vorhersehbaren Tokens, z.B. in verstecktem HTML Feld
  - Schutz gegen CSRFs in vielen Frameworks bereits integriert

Cross-Site Request Forgery (CSRF) - Ablauf



Probleme bei der Authentifizierung

- Grundlage
  - Falsches Verwalten von Authentifizierungsdaten, z.B. Speicherung von Passwörtern als Plaintext oder Übertragung über unsichere Kanäle
  - Session IDs können vorgegeben werden
  - Timeouts sind nicht vorgesehen oder falsch konfiguriert

Probleme bei der Authentifizierung

- Grundlage
  - Falsches Verwalten von Authentifizierungsdaten, z.B. Speicherung von Passwörtern als Plaintext oder Übertragung über unsichere Kanäle
  - Session IDs können vorgegeben werden
  - Timeouts sind nicht vorgesehen oder falsch konfiguriert
- Auswirkung
  - Zugänge können unberechtigterweise genutzt werden
  - Sessions werden durch Angreifer übernommen

Probleme bei der Authentifizierung

- Grundlage
  - Falsches Verwalten von Authentifizierungsdaten, z.B. Speicherung von Passwörtern als Plaintext oder Übertragung über unsichere Kanäle
  - Session IDs können vorgegeben werden
  - Timeouts sind nicht vorgesehen oder falsch konfiguriert
- Auswirkung
  - Zugänge können unberechtigterweise genutzt werden
  - Sessions werden durch Angreifer übernommen
- Schutz
  - Implementierung von Authentifizierung und Session Management anhand von Richtlinien, z.B. OWASP ASVS

Unvollständige Zugriffskontrolle

- Grundlage
  - Unbekannte oder autorisierte Nutzer manipulieren Zugriff
  - Anwendung überprüft nicht, ob Zugriff auf Objekt erlaubt

Unvollständige Zugriffskontrolle

- Grundlage
  - Unbekannte oder autorisierte Nutzer manipulieren Zugriff
  - Anwendung überprüft nicht, ob Zugriff auf Objekt erlaubt

- Auswirkung
  - Unbekannte erhalten Zugriff auf Daten legitimer Nutzer
  - Autorisierte Nutzer weiten ihren Zugriff aus, z.B. auf Admin

Unvollständige Zugriffskontrolle

- Grundlage
  - Unbekannte oder autorisierte Nutzer manipulieren Zugriff
  - Anwendung überprüft nicht, ob Zugriff auf Objekt erlaubt
- Auswirkung
  - Unbekannte erhalten Zugriff auf Daten legitimer Nutzer
  - Autorisierte Nutzer weiten ihren Zugriff aus, z.B. auf Admin
- Schutz
  - Direkten Zugriff auf Daten immer überprüfen
  - Wenn möglich, indirekte Datenreferenzen nutzen

Offenlegung sensibler Daten

- Grundlage
  - Sensible Daten wurden nicht ausreichend geschützt
  - z.B. Klartext-Speicherung/Übertragung von Passwörtern

Offenlegung sensibler Daten

- **Grundlage**
  - Sensible Daten wurden nicht ausreichend geschützt
  - z.B. Klartext-Speicherung/Übertragung von Passwörtern
- **Auswirkung**
  - Angreifer haben Zugriff auf kritische Daten

Offenlegung sensibler Daten

- Grundlage
  - Sensible Daten wurden nicht ausreichend geschützt
  - z.B. Klartext-Speicherung/Übertragung von Passwörtern
- Auswirkung
  - Angreifer haben Zugriff auf kritische Daten
- Schutz
  - Identifikation von sensiblen Daten
  - Verschlüsselung der Daten vor Speicherung/Versenden
  - Standardisierte Kryptographie und Passwortfunktionen

Ungeschützte APIs

- Grundlage
  - APIs sind komplex und oft schwer testbar
  - Tests decken nur einen kleinen Teil möglicher Angriffe ab

Ungeschützte APIs

- **Grundlage**
  - APIs sind komplex und oft schwer testbar
  - Tests decken nur einen kleinen Teil möglicher Angriffe ab

- **Auswirkung**
  - APIs können für Angriffe genutzt werden
  - API Reverse Engineering kann Schwachstellen aufdecken

Ungeschützte APIs

- Grundlage
  - APIs sind komplex und oft schwer testbar
  - Tests decken nur einen kleinen Teil möglicher Angriffe ab
- Auswirkung
  - APIs können für Angriffe genutzt werden
  - API Reverse Engineering kann Schwachstellen aufdecken
- Schutz
  - Entwurf von APIs nach Sicherheitskriterien
  - Ausgiebiges Testen von APIs bzgl. Sicherheit
  - z.B. Krypto, Authentifizierung, Zugriffskontrolle, etc.

Komponenten mit bekannten Schwachstellen

- Grundlage
  - Komponenten auf Server- oder Client-Seite verwundbar
  - Probleme öffentlich bekannt, aber noch nicht beseitigt

Komponenten mit bekannten Schwachstellen

- Grundlage
  - Komponenten auf Server- oder Client-Seite verwundbar
  - Probleme öffentlich bekannt, aber noch nicht beseitigt

- Auswirkung
  - Angreifer können bekannte Lücken evtl. ausnutzen
  - Angriffe einfacher weil Proof-of-Concept verfügbar

Komponenten mit bekannten Schwachstellen

- Grundlage
  - Komponenten auf Server- oder Client-Seite verwundbar
  - Probleme öffentlich bekannt, aber noch nicht beseitigt
- Auswirkung
  - Angreifer können bekannte Lücken evtl. ausnutzen
  - Angriffe einfacher weil Proof-of-Concept verfügbar
- Schutz
  - Katalogisierung der eigenen Versionsstände
  - Überwachung von Updates und Schwachstellen
  - Zeitnahes Upgrade oder ”virtueller” Patch (Filtern)

Falsche Sicherheitseinstellungen

- Grundlage
  - Fehler bei der Installation/Wartung des Servers
  - z.B. Directory Listing aktiv, Standard Konten nicht gelöscht

Falsche Sicherheitseinstellungen

- Grundlage
  - Fehler bei der Installation/Wartung des Servers
  - z.B. Directory Listing aktiv, Standard Konten nicht gelöscht
- Auswirkung
  - Angreifer können bekannte Lücken ausnutzen
  - Mehr Informationen zugänglich als nötig; hilfreich für Angriff

Falsche Sicherheitseinstellungen

- Grundlage
  - Fehler bei der Installation/Wartung des Servers
  - z.B. Directory Listing aktiv, Standard Konten nicht gelöscht
- Auswirkung
  - Angreifer können bekannte Lücken ausnutzen
  - Mehr Informationen zugänglich als nötig; hilfreich für Angriff
- Schutz
  - Gezielte Härtung des Webserver-Systems
  - Update Management Prozess

Unzureichende Schutzmaßnahmen

- Grundlage
  - Attacken werden nicht erkannt; es wird nicht reagiert
  - Abwehr gegen bekannte Angriffe ist nicht implementiert
  - Schnelle Reaktion durch Patches ist nicht vorgesehen

Unzureichende Schutzmaßnahmen

- Grundlage
  - Attacken werden nicht erkannt; es wird nicht reagiert
  - Abwehr gegen bekannte Angriffe ist nicht implementiert
  - Schnelle Reaktion durch Patches ist nicht vorgesehen

- Auswirkung
  - Angreifer können System scannen ohne erkannt zu werden
  - Angriffsvorbereitungen bleiben unerkannt
  - Im Angriffsfall kann nicht zeitnah reagiert werden

Unzureichende Schutzmaßnahmen

- Grundlage
  - Attacken werden nicht erkannt; es wird nicht reagiert
  - Abwehr gegen bekannte Angriffe ist nicht implementiert
  - Schnelle Reaktion durch Patches ist nicht vorgesehen
- Auswirkung
  - Angreifer können System scannen ohne erkannt zu werden
  - Angriffsvorbereitungen bleiben unerkannt
  - Im Angriffsfall kann nicht zeitnah reagiert werden
- Schutz
  - Logging und Analyse von Anfragen an die Web-App
  - Nutzung von Tools zum Schutz, z.B. OWASP AppSensor
  - Implementierung von schnellen Patch-Prozessen

Welche Maßnahmen sollten Sie beim Entwurf von Web-APIs beachten?

A) Sicherheitskriterien einfließen lassen und ausgiebig testen

B) Möglichst viele kryptographische Verfahren verwenden

C) Möglichst komplexe APIs erstellen, um Reverse Engineering zu erschweren

Durch Cross-Site Scripting könnte ein Angreifer eigenen Code im Browser seines Opfers ausführen. Richtig oder falsch?

A) Richtig

B) Falsch

Grundlegende Maßnahmen zur Sicherheit von Web-Anwendungen

- Jeder Teil einer Client-Anfrage *muss* überprüft werden
  - Jedes Feld in der Eingabe des Clients ist potentiell bösartig!
  - Niemals Überprüfung von Eingaben auf dem Client durchführen!!!

Grundlegende Maßnahmen zur Sicherheit von Web-Anwendungen

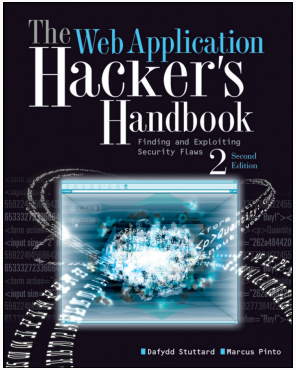
- Jeder Teil einer Client-Anfrage *muss* überprüft werden
  - Jedes Feld in der Eingabe des Clients ist potentiell bösartig!
  - Niemals Überprüfung von Eingaben auf dem Client durchführen!!!
- Verwendung von etablierten Web-Frameworks
  - Viele potentielle Schwachstellen sind seit Jahren bekannt
  - Frameworks bieten erprobte Methoden zur Vermeidung von Risiken
  - Methoden müssen benutzt werden!

Grundlegende Maßnahmen zur Sicherheit von Web-Anwendungen

- Jeder Teil einer Client-Anfrage *muss* überprüft werden
  - Jedes Feld in der Eingabe des Clients ist potentiell bösartig!
  - Niemals Überprüfung von Eingaben auf dem Client durchführen!!!
- Verwendung von etablierten Web-Frameworks
  - Viele potentielle Schwachstellen sind seit Jahren bekannt
  - Frameworks bieten erprobte Methoden zur Vermeidung von Risiken
  - Methoden müssen benutzt werden!
- Aktualisierung von eingesetzten 3rd-Party Komponenten
  - Schwachstellen in eingesetzten Komponenten können eigene Web-Anwendung kompromittieren
  - Regelmäßige Aktualisierung wichtig!

Interessante Unterlagen

Buch: Web Application Hacker's Handbook



- Verfügbar in der HSA Bibliothek

OWASP Projekt

- Online Präsenz
  - https://www.owasp.org
- Viele Materialien zu diversen Web-Security-Themen

Gibt es noch Fragen?