Web-Engineering

Sicherheit

Hochschule Niederrhein
University of Applied Sciences
Elektrotechnik und Informatik
Faculty of Engineering and Computer Science

Vorbemerkung

- IT-Sicherheit:
  - Umfassendere Darstellung in einer eigenen Veranstaltung
  - Hier: spezielle Aspekte bei Web-Anwendungen

- OWASP (Open Web Application Security Project)
  - www.owasp.org
  - Top 10 Liste der Sicherheitsrisiken zu einzelnen Jahren

Sicherheitsprobleme (1)

- Aspekte des Thema "Sicherheit":
  - Schutz vor unbefugter Verwendung von Software
    - Autorisierung : wer darf was tun ?
  - Schutz vertrauenswürdiger Daten
    - Authenzität:
      - Originalität (ist es das ursprüngliche Dokument ?)
      - Rechtsgültigkeit
    - Personenbezogene Daten
  - Schutz vor Schädigung:
    - Ruf / Vertrauen zu einem Informationsanbieter
    - Gezielte Fehlinformationen
    - Beschädigung / Zerstörung von Informationsangeboten und Diensten

Sicherheitsprobleme (2)

- Speziell bei Web-Anwendungen:
  - Zerlegung in 2 unabhängige, einander nicht bekannte Systeme (Client-System, Server-System)
    - Gegenseitige Vertrauenswürdigkeit ist problematisch
  - Technologien sind nicht per se mit Schutzmechanismen ausgestattet:
    - Klartext-Übertragungen (HTTP)
    - Clientseitige dynamische Auslieferung / Installation von Software
    - Clientseitig Transparenz der verwendeten Datenstrukturen

Angriffsmöglichkeiten (Bsp.)

- Clientseitige Angriffe:
  - XSS: Cross-Site-Scripting
    - Injizieren von Schad-Code auf dem Client-System
  - Cross-Site Request Forgery:
    - Fälschen der Requests
  - Ausnutzen von Fehlern in den Webbrowsern
- Übertragungsweg:
  - Man-in-the-Middle
    - Abhören und Verfälschen des Nachrichtenaustauschs
- Serverseitige Angriffe
  - Injizieren von Schad-Code auf dem Server-System
  - Ausnutzen von Fehlern in den Web-Servern, Applikations- und Datenbankserver
  - Gezielte Überlastung (Denial of Service-, Distributed Denial of Service-Attack)

Merke

- Misstraue allen Nachrichten ! Seien es
  - Nachrichten vom Server
  - Nachrichten vom Client
  - Ressourcen aus verschiedenen Quellen (s.u.)
- Daraus folgt:
  - Nachrichteninhalte müssen beim Client und beim Server überprüft werden
    - Syntaktisch (Form)
    - Semantisch (Inhalt)

Clientseitige Angriffe

- Nutzen die Programmiermöglichkeiten beim Webbrowser aus
  - Zugriff / Veränderung des DOM
  - Dynamisches Nachladen / Erzeugen von Programmcode
  - Dynamische Erzeugung von Inhalten
    - Direkt serverseitig
    - Aufgrund gelieferter Daten (per AJAX)
- Auch bei statischen (X)HTML-Seiten, wenn auf Client z.B. javascript ausgeführt werden darf

XSS: Cross-Site-Scripting (1)

- "Same-Origin-Policy":
  - Wird eine Quelle durch den Aufruf einer Webseite als vertrauenswürdig angesehen, dann sind auch alle weiteren Informationen dieser Quelle vertrauenswürdig
  - Alle Daten einer Website (= Web-Standort) werden gemeinsam eingeschätzt
    - Anderen Quellen wird misstraut, d.h. die gelieferten Informationen werden nicht berücksichtigt
  - Also nur Unterscheidung in Pfad und Dateiname zulässig
- Beispiel: Website http://www.mustersoft.com
  - http://www.mustersoft.com/main.html $\Rightarrow$ vertrauenswürdig
  - http://www.anders.com/main.html $\Rightarrow$ nicht vertrauenswürdig

XSS: Cross-Site-Scripting (2)

- "Same-Origin-Policy": **Problem**
  - Wird nur bei Seiten-Requests und Zugriffe auf DOM-Bäume unterschiedlichen Ursprungs berücksichtigt
  - Wird **nicht** bei Requests berücksichtigt, die durch `src`-Attribute ausgelöst werden, z.B.

    Image-Element `<img src="http://fremdedomain.com/Bild">`

    Iframe-Element `<iframe src="http://fremdedomain.de/x.htm">`

    Script-Element `<script src="http://fremdedomain.de/x.js">`

XSS: Cross-Site-Scripting (3)

- Injection von Schad-Code:
  - Nutzt die Möglichkeit aus, z.B. Code-Abschnitte (<script>, i.d.R. javascript) an beliebigen Stellen einzufügen
  - Einfache Variante:
    - Direktes Speichern externer Eingaben
    - Erzeugung einer Ausgabe aufgrund dieser Eingaben
    - Mögliche Abhilfe: Meta-Zeichen so umwandeln, dass ihre besondere Bedeutung verloren geht
      - Spezielle Module der Web-Frameworks verwenden
      - Mit regulären Ausdrücken prüfen / bearbeiten

XSS: Cross-Site-Scripting (4)

- Injection von Schad-Code (Forts.):
  - Seiten mit präparierten Links
    - Direkt mit angehängten Script-Teilen
    - Sind ggf. in Statuszeilen sichtbar
  - Anstelle einfacher Links: präparierte src-Angaben, z.B. für Script-Bereiche, die generell unsichtbar bleiben
  - Veränderung von GET-Anfragen, wenn URLs von dort direkt übernommen werden
    - GET-Parameter sind i.d.R. sichtbar in URL-Eingabefeldern

XSS: Cross-Site-Scripting (5)

- Mögliche Schäden:
  - Auslesen von Cookies/clientseitig gespeicherten Daten, Übermittlung an Dritte
    - Insbesondere Session-ID
  - Auslesen von clienseitig intern gespeicherten Daten (z.B. Formulardaten), Übermittlung an Dritte
    - Insbesondere Passwörter (auch bei verschlüsselten Verbindungen!)

Man-in-the-Middle

- Zwischen die Kommunikationspartner Client und Server schaltet sich ein *Angreifer*:
  - Beobachtet den ursprünglichen Datenverkehr
  - Simuliert dann zu beiden Seiten hin den passenden Datenverkehr
    - Um sensible Informationen zu erlangen
    - Um die Kommunikation zu verändern
    - Ggf. sogar die Kommunikation weiter umzulenken
  - Abhilfe: Verschlüsselung der Kommunikation

Cross-Site Request Forgery (1)

- Requests fälschen:
  - Datenverkehr eines berechtigten Benutzers verwenden, um Schad-Code beim Webserver (und damit ggf. später bei den Web-Clients) zu injizieren
  - i.d.R. wird der Benutzer dazu veranlasst, einen manipulierten Link zu verwenden, der die gewünschte Wirkung hervorruft:
    - Z.B. in Cookies gespeicherte Sitzungsdaten benutzen
    - Z.B. mit XSS Seite des Benutzers passend verändern

Cross-Site Request Forgery (2)

- Mögliche Abhilfen:
  - Zunächst sicherstellen, dass kein XSS möglich ist
  - Bei Requests, die serverseitig Daten ändern sollen:
    - Ids auch auf anderem Weg übertragen
    - Einmalige Austausch-Ids verwenden, die dann serverseitig geprüft werden können

Serverseitige Angriffe

- Bekanntes Beispiel:
  - "SQL-Injection":
    - SQL-Anweisungen in Eingaben verstecken
    - Wenn die Eingabe direkt zum Aufbau einer SQL-Anweisung verwendet wird, kann der Schad-Code ausgeführt werden
    - Vertrauliche Inhalte können selektiert und zum Client übertragen werden
- Ausnutzen von Fehler in Interpretern / Laufzeitbibliotheken

Sicherheitstests

- Im Rahmen der Tests (siehe dort) auszuführen
- Überprüfen (Beispiele):
  - Welche Sicherheitsanforderungen bestehen?
  - Hardware / Sicherheitssoftware
  - Verschlüsselung
  - Passwortschutz
  - Benutzerführung
  - Session Handling
  - Protokollierung
  - Zugriffsrechte
- Penetrationstest durchführen: unerlaubter Systemzugriff möglich ?