Browser Security

Same Origin Policy (SOP):

- Origin A can access origin B's DOM if match on (protocol, domain, port)
- isoliert Webseiten voneinander
- es ist sicher 2 verschiedene Seiten zu besuchen
- erlaubt die sichere Einbettung von anderen Webseiten (sichere Delegierung)

- ABER: von anderen Servern geladene Scripte haben die Privilegien der importierenden Seite und nicht des Ursprungsservers

Frame Security:

Wenn man auf seiner Seite fremde Inhalte wie zum Beispiel Werbebanner einbindet, dann möchte man nicht, dass diese auf das DOM der eigenen Seite zugreifen kann.

Descendant Policy:

Ein Frame kann nur seine direkten Kinder und Kindeskinder steuern, aber nicht irgendwelche Frames im gleichen oder in anderen Fenster.

Opener Restriction: Ein aktiver Frame kann nur dann einen Ziel-Top-Level-Frame steuern, wenn er eine JavaScript-Referenz zum Zielframe besitzt oder sein "Öffner" ist.

Origin Propagation: Der Browser sollte einen aktiven Frame nur dann einen Zielframe steuern lassen, wenn der Frame den gleichen Ursprung (same policy) hat oder der Zielframe von diesem abstammt.

Frame Communication (Cross-origin Interaction):

Häufig wollen Seiten miteinander kommunizieren. Wenn Seite A ein Script laden möchte, dann geht dies zum Beispiel so: <script src=//siteB.com/script.js>
Script B läuft nun in mit dem Ursprung von A und hat somit vollen Zugriff auf das DOM von A.

Eine Nachricht wird mit "*targetWindow.postMessage(message, targetOrigin);*" an einen "untergeordneten" Frame versendet. Es ist wichtig *targetOrigin* anzugeben, weil ansonsten eine schädliche Seite die Location des Windows ändern könnte und die Nachricht abfangen könnte.

Cookie Security:

Löschen:

um einen Cookie zu löschen setzt man "expires" auf ein Datum in der Vergangenheit

Setzen:

- als Domäne kann man jede Nachsilbe des eigenen URL-Hostname wählen, außer die Top-Level-Domain (TLD)
- Beispiel: mit Hostname "login.site.com" kann man nur Cookies für "login.site.com" und ".site.com setzen"
- als Pfad kann man einen beliebigen Pfad angeben

Lesen:

- der Server sendet alle Cookies, die im Scope der aufgerufenen Seite sind
- Cookie-Domäne muss Nachsilbe von URL-Domäne sein
- Cookie-Pfad muss Vorsilbe des URL-Pfades sein
- wenn "secure" gesetzt ist, muss das Protokoll https sein (ansonsten egal)
- Beispiel: im Scope von ".site.com" befinden sich alle URL's, die mit ".site.com" enden

HttpOnly:

- unsichtbar für JavaScript-Code (Schützt den Cookie vor XSS-Attacken)

weitere Eigenschaften:

- der Server bekommt nur "Cookie: NAME=VALUE" zugeschickt, aber sieht nicht Attribute (z.B. "secure") => Plausibilitätsüberprüfungen schwer
- der Server sieht nicht welche Domäne den Cookie gesetzt hat
- keine Verschlüsselung oder Integritätsüberprüfungen

Tracking:

Evercookie-Framework:

Dieses Framework dient dazu einen Nutzer (zum Beispiel aus Marketinggründen) zu Tracken. Das Löschen eines Cookies soll umgangen werden, indem der Cookie mit einer Unique-ID an verschiedenen Orten abgespeichert wird und wenn einer der Cookies gelöscht wird, kann dieser wiederhergestellt werden. Des Weiteren werden die Cookies nicht nur an verschiedenen Orten abgespeichert, sondern auch in verschiedenen Formen (Standard HTTP Cookie, Bilddatei, ...). Das Framework soll also das Datenschutzbedürfnis des Benutzers umgehen.

Private Browsing Mode:

- geöffnete Webseiten und gedownloadete Dateien verändern nicht die Browser- oder Download-History
- alle Cookies werden gelöscht, nachdem alle Inkognito-Fenster geschlossen wurden

- ABER: Besuchte Webseiten können speichern, dass sie von einem besucht wurden
- gespeicherte Dateien werden nicht gelöscht