

## Top 10 der Applications Attacks

Spätestens seit dem Auftauchen des Begriffes Web 2.0 möchten Unternehmen vermehrt Applikationen ins Web portieren oder diese gleich dort aufbauen. Ein Applikationsentwickler für webbasierte Anwendungen sollte deshalb mindestens ein Grundwissen in Bezug auf Web-Applikations-Sicherheit besitzen. Dies sind die OWASP Top 10.



### Sven Vetsch

Leader des OWASP Switzerland Local Chapters  
[www.owasp.org](http://www.owasp.org)

Das Open Web Application Security Project (OWASP - [www.owasp.org](http://www.owasp.org)) betreut neben vielen anderen Projekten auch die sogenannten OWASP Top 10, welche die zehn akutesten Probleme in Bezug auf Webapplikationen wiedergeben. Diese sind international anerkannt und werden zum Beispiel bei Sun Microsystems, IBM, Swiss Federal Institute of Technology, British Telecom und vielen anderen Unternehmen bei der Applikationsentwicklung sowie beim Testen eben solcher Applikationen eingesetzt. Auch haben zum Beispiel die Michigan State University (MSU) und die University of California at San Diego (UCSD) die OWASP Top 10 in den Lehrplan aufgenommen, und für den Payment Card Industry (PCI) Data Security Standard gelten diese ab 2008 gar als offizielles Requirement für jegliche Security Code Reviews.

### A1 – Cross Site Scripting (XSS)

Cross Site Scripting, besser bekannt unter der Abkürzung XSS, ist die momentan am weitesten verbreitete Verwundbarkeit bei Webapplikationen. XSS sind dann vorhanden, wenn Daten, die von einem User eingegeben

werden, in ungeprüfter / ungefilterter Form von Seiten des Servers zurückgesendet und im Webbrowser ausgegeben werden.

security! zone **Kongress**  
PLATTFORM FÜR INFORMATIONSSICHERHEIT

---

Mittwoch, 19. Sept. 07  
16.00-17.00 Uhr

**Web Applications under Attack !**

Sven Vetsch & H.P. Waldegger,  
OWASP

Hacking-Demo und Schutzmassnahmen

---

**kostenlose Anmeldung hier ...**

XSS erlaubt einem Angreifer das Ausführen von Scriptcode im Webbrowser des Opfers, wodurch das Übernehmen der User Session, Defacen einer Website,

Einfügen eigener Contents, das Durchführen von Phishing Attacks und der Missbrauch des Webbrowsers zur Ausführung von scriptbasierter Malware möglich wird. Dieser schadhafte Scriptcode wird meist mit JavaScript geschrieben; jedoch ist dies mit jeder Scriptsprache möglich, welche vom Webbrowser des Opfers unterstützt wird.

### **A2 – Injection Flaws**

Injection Flaws, hauptsächlich SQL Injections, sind eine weit verbreitete Verwundbarkeit in Webapplikationen. Solche Injections erfolgen dann, wenn an eine Applikation gesendete Daten von einem Interpreter auf Seiten des Servers als Command, Query oder als Teil eines solchen verstanden und ausgeführt werden. Injection flaws können es einem Angreifer erlauben, beliebige Daten und/oder Dateien zu lesen, zu schreiben, zu erweitern oder sogar auszuführen. Im schlimmsten Fall kann eine solche Verwundbarkeit dem Angreifer erlauben, die ganze Applikation oder gar den ganzen Server zu kompromittieren.

### **A3 – Insecure Remote File Include**

Auch Insecure Remote File Includes können in vielen Webapplikationen gefunden werden; dies weil Developers dem Input eines Users vertrauen und die von diesem gesendeten Daten in einer File-, Include- oder Stream-Funktion ungeprüft und/oder ungefiltert verwenden. Auf vielen Plattformen erlauben

Frameworks das Einbinden von externen Objekten, wie z.B. Websites oder auch lokalen Dateien. Wenn solche Daten ungeprüft eingelesen werden, kann dies dazu führen, dass die inkludeten Daten einen Code enthalten, der beim Verarbeiten durch den Interpreter zur Ausführung kommt.

### **A4 – Insecure Direct Object Reference**

Als Insecure Direct Object Reference bezeichnet man den Verweis auf ein Objekt, wie zum Beispiel Files, Ordner, Datenbank-Einträge oder Keys, durch welchen diese intern eingebunden werden.

Sofern kein Access Control Check implementiert wurde, kann ein Angreifer durch die Manipulation der Parameter, welche auf solche Files usw. verweisen, auf andere Ressourcen zugreifen, ohne sich authentifizieren zu müssen.

### **A5 – Cross Site Request Forgery (CSRF)**

Cross Site Request Forgery ist keine neue Verwundbarkeitsart in Bezug auf Webapplikationen, dennoch ist sie simpel, und gleichzeitig kann dadurch grosser Schaden entstehen. Ein CSRF-Angriff richtet sich gegen den im Webbrowser eingeloggten User einer Applikation, wobei versucht wird, aus dem Browser einen Request an die verwundbare Webapplikation zu senden, wo dann die gewünschte Aktion mit den Berechtigungen des Benutzers ausgeführt wird. Diese Verwundbarkeit besteht in fast jeder Webapplikati-

on im Netz, da jede Applikation, die die Legitimität eines Requests nur anhand von automatisch übertragenen Credentials durch den Browser beurteilt, potentiell verwundbar ist.

Diese Verwundbarkeit ist auch unter verschiedenen anderen Namen bekannt, wie etwa Session Riding, One-Click Attacks, Cross Site Reference Forgery, Hostile Linking und Automation Attack. Teilweise wird auch die Abkürzung XSRF verwendet. OWASP sowie MITRE haben sich auf die Bezeichnung Cross Site Request Forgery mit der dazugehörigen Abkürzung CSRF festgelegt.

### **A6 – Information Leakage and Improper Error Handling**

Applikationen können unbeabsichtigter Weise durch verschiedenste Fehler Informationen über Konfigurationen, interne Abläufe oder Personen preisgeben. Sie können auch Informationen über die Dauer der Verarbeitung von gewissen Operationen oder auch über die verschiedenen Antworten auf unterschiedliche Anfragen preisgeben, so zum Beispiel den selben Error Text jedoch mit anderer Error Nummer. Webapplikationen geben oft sensible Informationen in Form von teilweise sehr detaillierten (Debug-) Error Messages preis.

### **A7 – Broken Authentication and Session Management**

Ein korrekt funktionierendes Session Management und eine saubere Authentifizierung sind zwei kritische Faktoren im Be-

reich der Web-Applikations-Sicherheit. Oft bestehen Fehler in diesem Bereich darin, dass Credentials und/oder Session Tokens während der Dauer ihrer Gültigkeit nicht ausreichend geschützt werden. Diese Schwächen können es einem Angreifer ermöglichen, Userdaten zu stehlen oder die aktuelle Session eines Users zu übernehmen. Weiter können auch Autorisierungsprozesse oder Accountability Controls umgangen werden sowie Datenschutzverletzungen auftreten.

### A8 – Insecure Cryptographic Storage

Webapplikationen nutzen sehr oft Kryptografie zum Schutz sensiti-

ver Daten, häufig werden die dazu nötigen Funktionen jedoch falsch implementiert oder genutzt. Sensitive Daten einfach unverschlüsselt zu lassen, ist die wohl grösste Gefahr in diesem Bereich, jedoch ist es nicht selten der Fall, dass entweder schwache kryptografische Algorithmen verwendet werden oder aber, dass starke Verschlüsselungen falsch angewendet werden, wodurch diese Ihre Wirkung verfehlen. Insecure Cryptographic Storage kann dazu führen, dass sensitive Daten einsehbar sind oder auch, dass gegen gewisse Compliance-Vorgaben verstossen wird.

**Für den Payment Card Industry (PCI) Standard gelten die OWASP Top 10 ab 2008 als offizielles Requirement.**

### A9 – Insecure Communications

Oftmals wird vergessen, Netzwerk Traffic durch die Applikation zu verschlüsseln, und zwar dort, wo dies zum Schutz von sensitiven Daten nötig wäre. Verschlüsselung (meist SSL) muss für jede Verbindung verwendet werden, die nicht sowieso jedem zugänglich ist. Besonders zu beachten ist dies bei Webseiten, welche vom Internet aus erreichbar sind. Doch sollten Backend Verbindungen genauso geschützt werden, denn sonst können darüber sensible Daten wie Authentication und/oder Session Tokens preisgegeben werden. Weiter sollte Verschlüsselung auch immer dann verwendet werden, wenn

sensible Daten wie Informationen von Kreditkarten oder medizinische Daten übertragen werden. Es ist auch zu beachten, dass die Applikation nicht dazu gebracht werden kann, die Verschlüsselung auszuschalten oder auf einen veralteten und unsicheren Algorithmus auszuweichen, da dies durch einen Angreifer leicht ausgenutzt werden kann.

### A10 – Failure to Restrict URL

Immer wieder kommt es vor, dass bestimmte Bereiche innerhalb einer Webapplikation nicht durch einen Login-Mechanismus oder Ähnliches geschützt sind, sondern lediglich durch das Aufrufen der korrekten URL Daten eingesehen oder gar administra-

tive Funktionen verwendet werden können. Ein motivierter und kompetenter Angreifer, der auch noch ein wenig Glück hat, kann zum Beispiel durch blosses Ausprobieren Zugang zu diesen versteckten Seiten erhalten. In der Vergangenheit hat es sich immer wieder gezeigt, dass „Security by Obscurity“ keine geeignete Schutzmassnahme ist und somit davon abgesehen werden sollte. Es sollten immer Access Control Checks implementiert werden, wo Funktionen angeboten werden, welche nicht öffentlich zugänglich sein sollten. Damit wird nicht autorisierten Benutzern dieser Zugang verweigert.