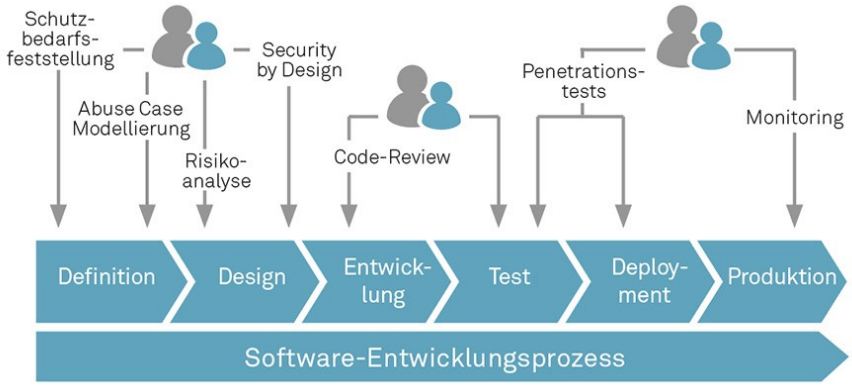
INFORMATIONS-TECHNIK
Ausstellungsstrasse 70 Telefon 044 446 96 60
CH-8090 Zürich Telefax 044 446 96 66
Homepage www.tbz.ch E-Mail admin.it@tbz.zh.ch



Modul 183

Applikationssicherheit implementieren

Version 4.2
22. August 2021

Inhaltsverzeichnis

1 Einleitung

1.1 Rahmenbedingungen

Alle auf den folgenden Seiten gegebenen Informationen sollen ausschliesslich dazu dienen das Bewusstsein über die Gefahren und Risiken zu fördern, denen sich ein "normaler" Internetnutzer beim täglichen Gebrauch des Internets aussetzt. Insbesondere das sorglose Vertrauen und Benutzen von sogenannten Webapplikationen stehen dabei im Fokus.

Dieses Skript soll in keiner Weise eine "Hacker-Anleitung" sein, oder dazu beitragen sich auf illegalem Wege Daten zu beschaffen. Damit das Verständnis für sichere Applikationen gefördert werden kann, ist es allerdings nötig, entsprechende Angriffsszenarios auch praktisch zu behandeln und zu verstehen wie "Hacker" ihre Opfer aussuchen und diese dann gezielt angreifen.

Haftungsausschluss

Die Inhalte dieses Skriptes wurden mit grösstmöglicher Sorgfalt recherchiert und implementiert. Fehler im Bearbeitungsvorgang sind dennoch nicht auszuschliessen. Hinweise und Korrekturen senden Sie bitte direkt an den Autor.

Alle während diesem Kurs durchgeführten Übungen werden nur in einer abgesicherten Umgebung durchgeführt. Es ist strengsten verboten, irgendwelche Angriffe im öffentlichen Netz durchzuführen. Der Autor lehnt jede Haftung ab und verweist auf Art. 322 Strafgesetzbuch (Schweiz)StGB.

Eine Haftung für die Richtigkeit, Vollständigkeit und Aktualität dieses Skript kann trotz sorgfältiger Prüfung nicht übernommen werden. Insbesondere wird keinerlei Haftung für die Inhalte externer Links. Für den Inhalt der verlinkten Seiten sind ausschliesslich deren Betreiber verantwortlich.

1.2 Modulidentifikation

ICT Berufsbildung
Formation professionnelle
Formazione professionale

Modulnummer	**183**
Titel	Applikationssicherheit implementieren
Kompetenz	Applikationen sicher planen, entwickeln und in Betrieb nehmen.
Handlungsziele	1 Aktuelle Bedrohungen erkennen und erläutern können. Aktuelle Informationen zum Thema (Erkennung und Gegenmassnahmen) beschaffen und mögliche Auswirkungen aufzeigen und erklären können. 2 Sicherheitslücken und ihre Ursachen in einer Applikation erkennen können. Gegenmassnahmen vorschlagen und implementieren können. 3 Mechanismen für die Authentifizierung und Autorisierung umsetzen können. 4 Sicherheitsrelevante Aspekte bei Entwurf, Implementierung und Inbetriebnahme berücksichtigen. 5 Informationen für Auditing und Logging generieren. Auswertungen und Alarme definieren und implementieren.
Kompetenzfeld	Application Engineering
Objekt	Web-Applikation mit Datenanbindung, Webservice
Niveau	4
Voraussetzungen	Erfahrung im Entwickeln von Applikationen (Multi-User, Web, verteilte Applikationen)
Anzahl Lektionen	40
Anerkennung	Eidg. Fähigkeitszeugnis
Modulversion	3.00

Quelle: www.ict-berufsbildung.ch [www.i-ch.ch]

1.3 Überblick

Es vergeht heutzutage kaum eine Woche, in welcher die Medien nicht über neue Unzulänglichkeiten in der Sicherheit von Softwaresystemen im Allgemeinen und von Internet Applikationen im Speziellen informieren. Seien dies Meldungen über neue **Malware**, über bekannt gewordene *Phishing Attacken* oder entdeckte Sicherheitslöcher in Betriebssystemen. Mit der wachsenden Publizität steigt auch die Sensibilität für Sicherheitsaspekte. Viele Gefahrenquellen sind aber immer noch unbeachtet in heutigen komplexen Systemen. Ein bisher sehr vernachlässigtes Gefahrenpotenzial besteht in den Applikationen und ihren Daten selbst. Jede Form von Datenzugriffen, auch durch authentisierte Benutzer, stellt eine Herausforderung an das Sicherheitssystem dar. Bestehende Technologiestandards wie J2EE und .NET bieten für sichere Applikationen nur unzureichende Schutzmechanismen.

Unterschied zwischen Viren, Würmen und Trojaner

Ein **Virus** hängt sich an ein Programm oder eine Datei an, sodass er sich von einem Computer auf den nächsten verbreiten und diese Geräte dabei infizieren kann. Ähnlich wie bei menschlichen Viren können Computerviren nach der Schwere der Infektion unterschieden werden. Einige Viren verursachen nur geringfügige störende Effekte, während andere Ihre Hardware, Software oder Dateien beschädigen können. Fast alle Viren sind an eine ausführbare Datei angehängt. Das bedeutet, dass der Virus möglicherweise auf Ihrem Computer vorhanden ist, ihn jedoch erst infizieren kann, wenn Sie das bösartige Programm ausführen oder öffnen. Ein wichtiger Punkt ist, dass sich ein Virus nicht ohne menschliches Zutun verbreiten kann (z. B. durch Ausführen eines infizierten Programms, um ihn zu aktivieren). Nutzer verbreiten – in den meisten Fällen unwissentlich – Computerviren weiter, indem sie infizierte Dateien teilen oder E-Mails mit Viren als E-Mail-Anhang versenden

Ein **Wurm** ähnelt in seinem Aufbau einem **Virus** und wird als Unterklasse eines Virus angesehen. Würmer verbreiten sich von einem Computer auf einen anderen, doch anders als Viren können sie ohne Zutun einer Person übertragen werden. Ein Wurm nutzt auf Ihrem System vorhandene Datei- oder Datenübertragungsfunktionen, was es ihm ermöglicht, sich ohne Hilfe fortzubewegen. Die grösste Gefahr bei einem Wurm ist seine Fähigkeit, sich auf Ihrem System zu replizieren. Das bedeutet, dass von Ihrem Computer anstatt nur eines Wurms Hunderte oder Tausende Kopien desselben Wurms versendet werden könnten – mit potenziell katastrophalen Folgen. Beispielsweise könnte ein Wurm eine Kopie von sich an alle Personen in Ihrem E-Mail-Adressbuch senden. Anschliessend repliziert sich der Wurm und überträgt sich an alle Personen im Adressbuch jedes Empfängers. Dieser Vorgang wird anschliessend immer weiter fortgesetzt. Die Kopierfähigkeit eines Wurms und seine Fähigkeit, sich über Netzwerke zu übertragen, führt letztendlich in den meisten Fällen dazu, dass der Wurm zu viel Systemspeicher verbraucht. Die Folge ist, dass Webserver, Netzwerkserver und einzelne Computer nicht mehr reagieren. Bei neueren Wurmangriffen, wie etwa dem vielbeachteten *Blaster-Wurm*, wurde der Wurm so programmiert, dass er eine Hintertür auf Ihrem System öffnet, über die bösartigen Benutzer per Fernsteuerung die Kontrolle über Ihren Computer übernehmen können.

Ein Trojaner ist kein Virus, sondern ein destruktives Programm, dessen implementierte *Ist-Funktionalität* nicht mit der angegebenen Sollfunktionalität übereinstimmt. Im Gegensatz zu Viren reproduzieren Trojaner sich nicht selbst. Können jedoch ebenso grossen Schaden anrichten. Heutige Trojaner bieten den Angreifern eine Vielzahl von Möglichkeiten, den befallenen Rechner zu kontrollieren, gespeicherte Daten auszuspähen, oder aber auch Tastatureingaben, wie z.B. Passwörter aufzuzeichnen und über eine Netzwerkverbindung zu senden. Mit Hilfe des Trojaners lässt sich dann der Computer fernsteuern. Meistens installieren Sie sich Trojaner selbst, wenn Sie "irgendwelche" Software installieren. Es kann aber auch beim Surfen über das Internet, oder durch E-Mails passieren.

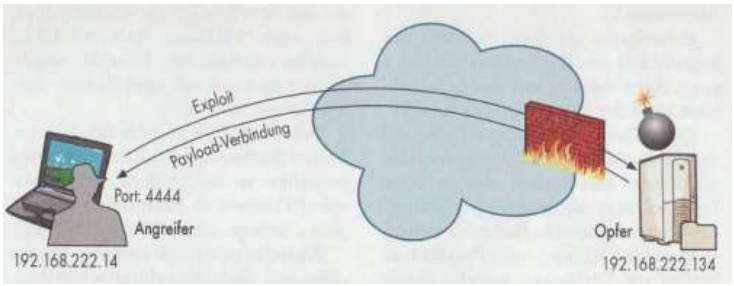


Abbildung 1: Angreifer nutzt die Schwachstelle des Opfers

Trojaner werden danach klassifiziert, wie sie Systeme infizieren und welche Schäden sie verursachen. Es gibt sieben Haupttypen von Trojanern:

- Trojaner, die Fernzugriff ermöglichen
- Trojaner, die Daten senden
- Destruktive Trojaner
- Proxy-Trojaner
- FTP-Trojaner
- Trojaner, die Security-Software deaktivieren
- Trojaner, die ***Denial-of-Service***-Angriffe ausführen

Mehr Umsatz mit Datendiebstahl als mit Drogen

Gemäss Gunnar Porada[1] kann heute vor allem ein Anstieg bei den Trojanern festgestellt werden. Täglich kommt eine grosse Menge neu hinzu. Dabei geht es, um das Ausspionieren von mobilen Endgeräten an die Daten heranzukommen. Zurzeit wird mit Datendiebstahl mehr Umsatz erzielt als mit Drogen.

Heute hat praktisch jeder die Möglichkeit sich einen Trojaner "zusammenstellen" zu lassen. Einschlägige Webseiten bieten Service- und Dienstleistungen bereits ab CHF 1500 für einen funktionierenden Trojaner an. (S.a. Darknet)

Aussage von Gunar Porada: "Kein Virenscanner ist in der Lage 100% aller Viren zu erkennen!"

[1] Gunnar Porada, Ex-Hacker, CEO innSec GmbH

1.4 Informationssicherheit

Informationen eines Unternehmens zur richtigen Zeit in der richtigen Qualität am richtigen Ort der richtigen Person zur Verfügung stellen, bildet den Hauptzweck der Informationssicherheit. Dies wird erreicht, indem die Verfügbarkeit, die Integrität, die Vertraulichkeit und die Nicht-Abstreitbarkeit bei Informationssystemen sichergestellt sind.

Verfügbarkeit

Verfügbarkeit wird umschrieben mit der Eigenschaft eines Systems, sämtliche Daten und Funktionen zu einem bestimmten Zeitpunkt zur Verfügung stellen zu können. *Denial of Service Attacken* führen zu Verlusten der Verfügbarkeit, da die Kommunikationsinfrastrukturen dadurch nicht mehr für die Abwicklung des Tagesgeschäftes zur Verfügung stehen.

Integrität

Die Integrität wird dann verletzt, wenn ein System unbefugte oder unbeabsichtigte Veränderungen an Daten oder der Software zulässt. Es kann somit nicht mehr garantiert werden, dass alle sicherheitsrelevanten Objekte vollständig, unverfälscht und korrekt sind. Viren können Daten und Programme derart verändern, dass die Integrität verletzt wird.

Vertraulichkeit

Vertraulichkeit bedeutet, dass nur bestimmte Personen auf Daten und Systeme zugreifen können oder dürfen. Soll die Vertraulichkeit gewahrt werden, müssen die Daten so gesichert sein, dass ein Zugriff nur denjenigen Nutzern möglich ist, welche durch Zugriffsrechte die Erlaubnis erhalten. Vertraulichkeit kann z.B. mit Verschlüsselung der Daten bei ihrer Übertragung oder Speicherung gewahrt bleiben.

Nicht-Abstreitbarkeit (Verbindlichkeit)

Nicht-Abstreitbarkeit bedeutet, dass die Transaktionssicherheit soweit gewährleistet sein muss, dass mit Sicherheit die Identität des Senders und Empfängers und weitere Identifikationsmerkmale von einer Transaktion (z.B. Bestellvorgang) erfasst werden. Ist die Nicht-Abstreitbarkeit nicht gewährleistet, so kann es vorkommen, dass ein Kunde im e-Commerce behauptet, dass er die gelieferte Ware nie bestellt hat.

Authentizität

Echtheit, Glaubwürdigkeit eines Benutzers

Privatsphäre

Keine unautorisierte Profilbildung, Privatsphäre

1.5 Applikationssicherheit

Der Begriff "Applikationssicherheit" umfasst alle Tätigkeiten und Prozesse sowie die Infrastruktur, welche für die Spezifikation, Evaluation, Entwicklung, Integration, Ausbreitung, Betrieb, Support, Ausbau und Ablösung von Applikationen sowie für die zugehörige Rechte- und Rollenverwaltung, Datenbewirtschaftung, Schnittstellen und Middleware-Komponenten benötigt werden. Die Applikationssicherheit setzt voraus, dass die darunterliegenden Schichten vor Angriffen und Missbrauch genügend geschützt sind.

Die technische IT-Sicherheit umfasst die Host- und Netzwerksicherheit. In vielen grösseren Unternehmen ist dafür ein entsprechender Sicherheitsprozess bereits standardmässig umgesetzt oder zumindest eingeführt oder schrittweise verbessert worden

Im Gegensatz dazu ist die Applikationssicherheit in vielen Unternehmen derzeit noch eine schlecht mit der IT-Sicherheit koordinierte Aufgabe gemäss Hannes P. Lubich[2]

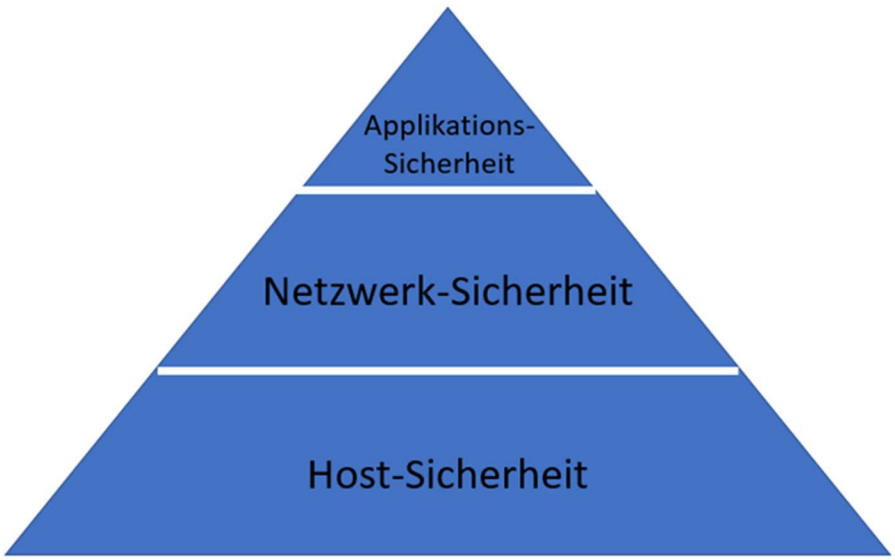


Abbildung 2: Applikationssicherheit basiert auf der Sicherheit der darunterliegenden Systeme

Fazit: Eine Kette ist immer so stark wie ihr schwächstes Glied.

[2] Prof. Dr. Hannes P. Lubich, Privatdozent ETHZ

OWASP 5-Ebenen-Modell

Um Sicherheit und Anwendungen zu verbessern, wurde das Community-Projekt "Open Web Application Security Project (OWASP)" ins Leben gerufen. Die Verwundbarkeit von Webanwendungen lässt sich in fünf Ebenen gliedern.

	Ebene	Inhalt	Beispiele:
5	Semantik	**Schutz vor Täuschung und Betrug**	Phishing-Schutz Informationspreisgabe
4	Logik	**Absicherung von Prozessen und Workflows als Ganzes**	"Passwort vergessen"-Fkt Benutzer Lock-Out
3	Implementierung	**Vermeiden von Programmierfehlern, die zu Schwachstellen führen**	Cross-Site Scripting SQL-Injection
2	Technologie	**Richtige Wahl und sicherer Einsatz von Technologie**	Verschlüsselung Authentisierung
1	System	**Absicherung der auf der Systemplattform eingesetzten Software**	Known Vulnerabilities Konfigurationsfehler
0	Netzwerk & Host	**Absicherung von Host und Netzwerk**	

Abbildung 3: OWASP Foundation, Kickstart für sichere Webanwendungen

Ebene 0 –Netzwerk und Host

Die Ebene von Netzwerk, Server-Hardware und darauf laufendem Betriebssystem wird hier nicht direkt der Sicherheit der Webanwendung zugeordnet. Diese Ebene schliesst sich vielmehr nach unten an. Die Umsetzung zwingender Sicherheitsmassnahmen auf dieser Ebene wird gleichwohl als zwingende Voraussetzung für sichere Webanwendungen betrachtet.
Verantwortliche Organisationseinheit: *Betrieb IT-Sicherheit*
Fachkenntnisse: *Netzwerk- und Systemadministration*

Ebene 1 – Systemebene

Hier werden all jene Programme betrachtet, die für das Funktionieren der gesamten Webanwendung benötigt werden. Dazu gehören Webserver und der Applikationsserver, aber auch Datenbank- und Backend-Systeme. Diese Komponenten müssen bei der Sicherheitsbetrachtung einer Webanwendung mit einbezogen werden.
Verantwortliche Organisationseinheit: *Betrieb*
Fachkenntnisse: *Netzwerk- und Systemadministration*

Ebene 2 – Technologie

Diese Ebene betrifft die Verwendung der für den jeweiligen Einsatzzweck und Schutzbedarf richtigen Technologie, sowie deren Nutzung. So z.B. setzt eine Webanwendung, die sensible Daten unverschlüsselt über das Internet transferiert, nicht die richtige Technologie ein. Eine Webanwendung, die Passworte zwar verschlüsselt, dafür aber einen zu kurzen Schlüssel verwendet, setzt die richtige Technologie falsch ein.
Verantwortliche Organisationseinheit: *Fachstellen, Entwickler, Betrieb*
Fachkenntnisse: *Allg. IT-Sicherheit*

Ebene 3 – Implementierung

Die Implementierungsebene umfasst den Bereich der unbeabsichtigten Programmierfehlern („Bugs“), aber auch nicht vorhandenen oder ungenügende Prüfung von Eingabedaten („Daten Validation“). Diese Ebene beinhaltet ferner ungenügende Testverfahren, und die Vernachlässigung der Qualitätssicherung zugunsten des Inbetriebnahme Termins oder aus Kostengründen.
Verantwortliche Organisationseinheit: *Entwickler (Umsetzer)*
Fachkenntnisse: *Softwareentwicklung*

Ebene 4 – Logik

Diese Ebene betrifft sowohl die Logik der Abläufe innerhalb einer Webanwendung als auch die Interaktion mit dem Benutzer. Ist diese zweckorientiert implementiert, kann gegebenenfalls eine Missbrauchsmöglichkeit vorliegen. Wird etwa zur Verhinderung einer mehrfach wiederholten Eingabe eines Passwortes nach dem fünften fehlerhaften Loginversuch die entsprechende Benutzererkennung gesperrt, so könnte dieser Benutzer durch Dritte gezielt ausgesperrt werden, sofern keine weiteren Massnahmen getroffen werden. Diese missbräuchliche Vorgehensweise wird weiter erleichtert, wenn die Benutzererkennung einfach zu erraten ist.
Verantwortliche Organisationseinheit: *Fachstelle (Anfordere)*
Fachkenntnisse: *Kenntnisse der Geschäftsprozesse*

Ebene 5 – Semantik

Die semantische Ebene umfasst inhalts- und kommunikationsbezogene Aspekte. Sie stellt den Vertrauenskontext für die Interaktion mit einem Benutzer her. Wird in diesem Bereich nicht ein hohes Mass an Sorgfalt aufgewendet, so kann eine Webanwendung von Dritten missbraucht werden, um einen Benutzer zu täuschen. Dieser Bereich kann selten auf eine einzelne Anwendung beschränkt bleiben. Vielmehr ist eine website- oder unternehmensübergreifende Betrachtung notwendig. Missbrauchsmöglichkeiten, die sich Fehler auf der semantischen Ebene zunutze machen, sind Social Engineering, Phishing, Identitätsdiebstahl u.a.
Die Bedeutung der logischen und semantischen Ebene wird vielfach nur unzureichend wahrgenommen. Dennoch haben diese beiden Ebenen eine hohe Bedeutung, wenn man unter der Sicherheit einer Webanwendung nicht allein nur den Schutz des Servers versteht, sondern eine umfassende Perspektive zugrunde legt: Der Anbieter einer Webanwendung trägt nicht nur Verantwortung für eigene Systeme, sondern auch für alle an der Nutzung der Webanwendung Beteiligten.
Verantwortliche Organisationseinheit: *Zentrale*
Fachkenntnisse: *Corporate Identity und Unternehmenskommunikation*

2 Sicherheitslücken und Bedrohungen

Bedrohungen durch Pufferüberlauf-Angriffe, Viren, sowie mobiler Code sind heute die am häufigsten auftretenden Sicherheitsprobleme. Nicht abgefangene Pufferüberläufe (engl. Buffer Overflow) sind sehr weit verbreitete Schwachstellen, die durch nachlässige Programmierung sowohl in Betriebssystemen als auch in Anwendungsdiensten auftreten und häufig durch Computerviren ausgenutzt werden.

In diesem Abschnitt beginnen wir beispielhaft mit konkreten Angriffsszenarien, um zu zeigen, wie Schwachstellen ausgenützt werden können. Es gibt offensichtlich unzählige Angriffe. Der Schwerpunkt in der Software-Entwicklung liegt im Implementieren von Gegenmassnahmen. Somit werden wir im nächsten Schritt die Angriffe verallgemeinern und gängige Sicherheitslücken und deren Ursache in Web-Applikationen (siehe Abschnitte 2.2, 2.3) beleuchten.

Wenn man realistisch ist, muss man auch sehen, dass die Schwachstellen ebenfalls unzählig sind. Um nicht unnötigen Aufwand in das Auffinden und Beheben von gemeldeten Schwachstellen (s.a. Abschnitt 2.3) investieren zu müssen, wählen Sicherheitsfachleute heute einen anderen Ansatz: Unternehmen unterziehen ihre webbasierten Plattformen einer Risikoanalyse mit Penetrationstest (s.a. Abschnitt 2.4). Diese lässt sich in der Regel nicht verallgemeinern, sondern ist firmenspezifisch durchzuführen. Wenn man die Punkte kennt, in denen man verwundbar ist und die für das Unternehmen gravierende Folgen haben, kann man die Schwerpunkte für Verbesserungsmassnahmen geeignet setzen.

Bei konsequentem Verfolgen von Sicherheitsfragen ist es lohnenswert, eigene Sicherheitsrichtlinien zu definieren, mit denen das Unternehmen arbeiten will und – für uns wichtig – nach denen Software entwickelt und getestet wird.

2.1 Angriffsszenarien

Da die Reihe von Angriffsszenarien nicht abschliessend ist, beschränken wir uns im Nachfolgend für zwei bekannte Beispiele.

2.1.1 Beispiel: SQL-Injection

SQL-Injecton beschreiben das Injizieren von SQL-Code in eine Anwendung, sodass dieser von der Datenbank ausgeführt wird. Das geschieht typischerweise durch eine entsprechend manipulierte Benutzereingabe. Wird diese ungeprüft in eine SQL-Abfrage eingefügt, können enthaltene SQL-Befehle die Datenbank in nicht vorhergesehener Weise manipulieren.

Folgendes kann durch einen erfolgreichen SQL-Angriff erreicht werden:

- Hinzufügen, Ändern, Löschen von Datensätzen, Tabellen, Datenbanken,
- Auslesen und Stehlen von Datensätzen und
- Erzeugen von Dateien auf dem Dateisystem der Anwendung.

Durch das Hinzufügen oder Ändern von Datensätzen in einer Tabelle mit Zugangsdaten für eine Webanwendung kann sich der Angreifer selbst gehobene Rechte selbst zuweisen oder den bestehenden Nutzern Rechte entziehen. Ist es dem Angreifer möglich, Dateien zu erzeugen, so kann er gezielt Webseiten in der Anwendung ersetzen und so Nutzer oder Systemdaten ausspionieren.

Angriffsvektor

Dabei werden typische Programmcode-Schwächen ausgenutzt, die teils durch die Programmiersprache PHP gegeben sind und teils durch weit verbreitete, unvollständige Programmieranleitungen in der Literatur und im Internet vorgegeben werden.

Schwachstellen bilden folgende Programmierfehler:

- Werteübergabe ohne umschliessende Hochkommas,
- eine fehlende Typenprüfung der Werte,
- eine fehlende Längenprüfung der Werte
- die fehlende Maskierung von Sonderzeichen.

Der eingeschleuste SQL-Code kann vom Angreifer auch derart platziert werden, dass er seine Wirkung zeitverzögert entfaltet.

Ein Angriffsbeispiel auf eine Anwendung, die eine Liste der Benutzernamen erstellt, könnte wie folgt aussehen:

Ein Angreifer manipuliert den Benutzernamen so, dass diesem zusätzlich SQL-Code eingeschleust wird. Verwendet die Anwendung den Benutzernamen innerhalb einer ungesicherten SQL-Abfrage, so wird der Code eingefügt und ausgeführt. Das Besondere an einem solchen Angriff ist demnach, dass der Augenblick der Infizierung der Anwendung meist in zeitlicher Entfernung von deren Ausführung liegt.

Der kompromittierende SQL-Code kann dabei aus unterschiedlichen Quellen stammen:

- Nutzereingaben,
- Cookie-Werten,
- Session-Werten und
- anderen Datenbank-Daten.

Angriffsformen

Der Angreifer versucht bestehende SQL-Anfragen durch geeignete, eigene SQL-Abfragen zu ändern. Durch das Einfügen geeigneter SQL-Befehle kann er:

- Logische Verknüpfungen ändern,
- Teilabfragen entfernen,
- zusätzliche Abfragen erzeugen,
- den Datenbankprozess beeinflussen und
- verräterische Fehlermeldungen erzeugen.

Logische Verknüpfungen ändern

```
<?php
$query = "SELECT * FROM users WHERE user='" . $_POST['username'] . "'
AND password='" . $_POST['password'] . "'";
$response = mysql_query($query);
?>
```

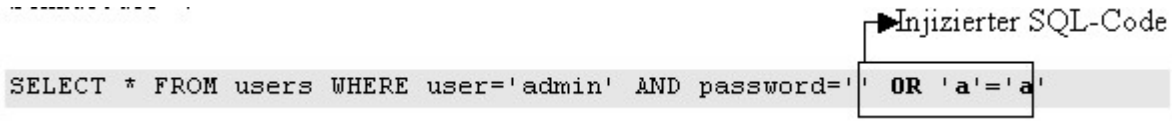
Abbildung 4: Ein für SQL-Angriffe anfälliges Skript

Dieser Code nimmt die Daten einer Eingabemaske mit "Nutzername" und "Passwort" entgegen und prüft in einer Datenbank nach dem Vorkommen dieser Kombination. Wird der Nutzer gefunden und passt das angegebene Passwort, so ist der Nutzer legitimiert.

Die Datenbankabfrage besteht aus einem festen Teil (SELECT * FROM users WHERE user=" AND password=") und aus zwei Variablen, die eingesetzt werden. Das funktioniert solange wie beabsichtigt, bis ein Angreifer folgende Eingaben macht:

```
Nutzername: admin
Passwort: ' OR 'a'='a
```

Die Abfrage, die an MySQL übermittelt wird lautet nach der Injizierung des Schadcodes[1] :



Diese neu entstandene SQL-Abfrage wird von MySQL logisch ausgewertet. Dabei wird zuerst die OR-Verknüpfung ausgeführt und anschliessend die AND-Verknüpfung. Es ergibt sich folgende SQL-Abfrage:

```
user='admin' AND password='' OR 'a'='a'
```

Das Ergebnis der OR-Verknüpfung ist WAHR, denn 'a'='a' ist WAHR. Es ergibt sich:

```
user='admin' AND TRUE
```

Die Abfrage lautet also:

```
SELECT * FROM users WHERE user='admin' AND TRUE
```

Sie ist WAHR, sofern ein Nutzer „admin“ existiert und sie liefert dessen Datensatz. Da meist im folgenden PHP-Code nur noch geprüft wird, ob die Datenbank als Antwort NULL (Nutzer oder Passwort sind falsch.) oder aber einen Datensatz (Nutzer und Passwort sind gültig.) zurückgibt, ist der Angreifer, auch ohne ein gültiges Passwort, authentifiziert.

Quelle:

http://www.erich-kachel.de/?p=223

Video:

https://www.youtube.com/watch?v=6gH4A49sPdc

2.1.2 Beispiel: Cross Site Scripting

Eine der aktuell beliebtesten Attacken ist das Cross Site Scripting (XSS). Ein Beispiel: Der Angreifer versucht eine Webanwendung so zu manipulieren, dass schädlicher Schadcode in die angezeigte Seite (z. B. in ein Gästebuch oder eine Auktionsseite) eingebettet wird. Der Browser verarbeitet diese an sich vertrauenswürdige Website inklusive des schadhaften Codes und sendet dadurch die aktuellen Login-Informationen an den Angreifer zurück.

Allgemein wird mit Cross-Site Scripting (XSS) das Einschleusen von HTML- und JavaScript-Code in eine Webseite bezeichnet. Normalerweise gilt der Grundsatz "*Von einer vertrauenswürdigen Webseite mitgelieferter JavaScript-Code ist vertrauenswürdig*": Er darf die Seite ändern und vom eigenen Server Daten nachladen. Durch Cross-Site Scripting wird dieser Grundsatz verletzt, indem bösartiger JavaScript-Code eingeschleust wird. Dafür gibt es verschiedene Möglichkeiten:

Reflektiertes XSS

Der Schadcode wird an den Webserver gesendet und von diesem an den Client zurückgegeben. Für den Angriff ist der Aufruf eines präparierten URL oder das Absenden eines präparierten Formulars durch das Opfer nötig.

Für seinen Angriff benötigt der Angreifer eine entsprechende Schwachstelle. Die besteht im Fall von Reflektierten XSS darin, dass die Webanwendung die Eingaben nicht ausreichend prüft, bevor sie sie als Bestandteil einer Webseite an den Benutzer zurückgibt.

Die einfachsten Fälle sind dabei die Fehlermeldung für eine nicht gefundene Datei mit dem HTTP-Fehlercode `404` und Suchergebnisse. Meist wird in der Fehlermeldung der Name der nicht gefundenen Datei ausgegeben, es erscheint eine Meldung nach dem Muster "`404 - Datei [Dateiname] nicht gefunden`". Wird der Dateiname nicht ausreichend geprüft, kann ein Angreifer darüber seinen Cross-Site-Scripting-Code einschleusen. Ähnlich sieht es bei der Ausgabe von Suchergebnissen aus, bei denen der gesuchte Begriff auf der Ergebnisseite ausgegeben wird. Wird der nicht ausreichend geprüft, kann darüber Cross-Site-Scripting-Code eingeschleust werden.

Als Beispiel soll im Folgenden eine Webmail-Anwendung dienen:

```
http://www.mailprovider.example/webmail/login.php?user=[User-ID]
```

Die Schwachstelle besteht darin, dass der Wert des Parameters `user` nicht ausreichend geprüft wird, bevor er auf der Login-Seite angezeigt wird. Der Angreifer stellt einen Link zusammen, der statt eines Benutzernamens XSS-Code enthält:

```
http://www.mailprovider.example/webmail/login.php?user=[XSS-Code]
```

Dann bringt er einen Benutzer dazu, diesen Link anzuklicken. Dazu kann er ihm den Link zum Beispiel per E-Mail zuschicken. Mit etwas Social Engineering ist es meist kein Problem, den Benutzer zum Anklicken des Links zu bringen.

Nach dem Klick auf den Link wird ein Request an die Webanwendung geschickt, der alle zugehörigen Parameter und ihrer Werte und damit unter anderem den XSS-Code enthält. Die Webanwendung fügt den Wert des Parameters `user` in die erzeugte Webseite ein und sendet das Ergebnis an den Benutzer.

Der eingeschleuste Code wird im Webbrowser des Benutzers im Kontext der betroffenen Webseite ausgeführt. Und genau darin besteht die Gefahr: Als Schutz vor den Auswirkungen schädlichen JavaScript-Codes verwenden die Webbrowser bekanntlich die Same-Origin Policy, die aufgrund der Herkunft des JavaScript-Codes über dessen Ausführung und Rechte entscheidet. JavaScript-Code darf nur auf die Teile der Webseite zugreifen, die von der gleichen Domain (= Origin) geladen wurden wie er selbst. Ausserdem darf er mit wenigen Ausnahmen nur mit der Domain kommunizieren, von der er geladen wurde. Diese Schutzfunktionen werden durch den XSS-Angriff umgangen. Da der eingeschleuste JavaScript-Code anscheinend von der Webseite `mailprovider.example` stammt, wird er in deren Kontext ausgeführt und darf auf die davon geladenen Seitenbestandteile zugreifen.

Den Ablauf des Angriffs zeigt folgendes Bild:

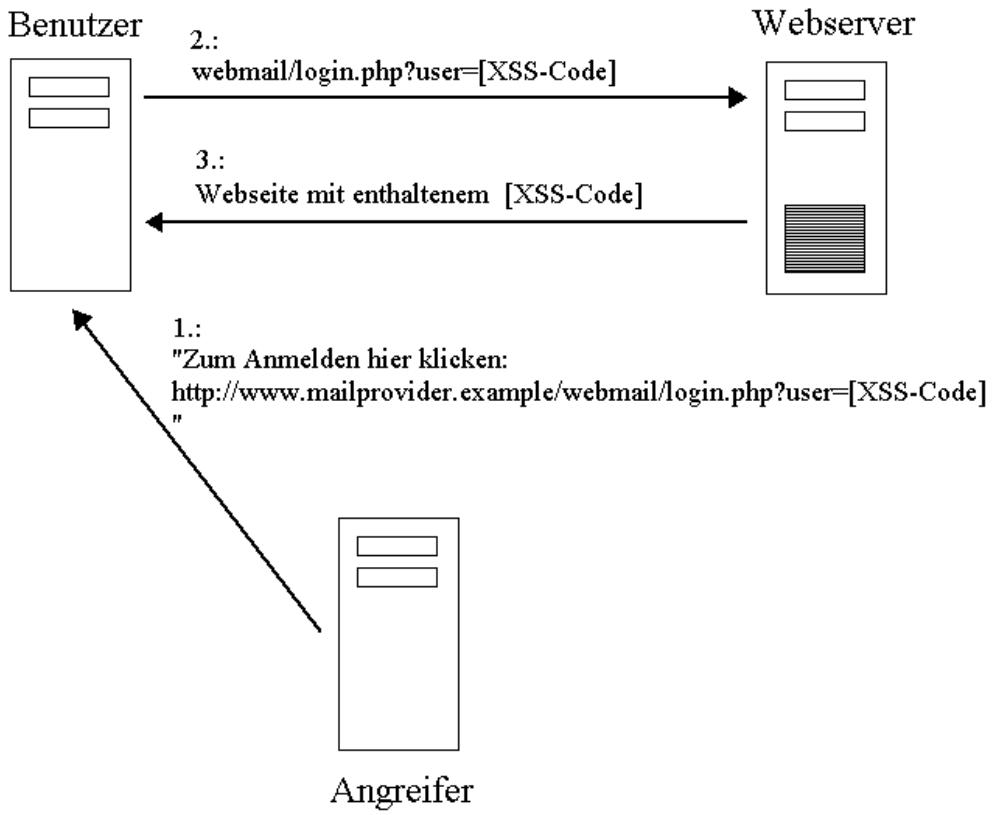


Abbildung 5: Reflektiertes XSS

Quelle:

https://www.ceilers-news.de/serendipity/622-Cross-Site-Scripting-im-UEberblick,-Teil-1-Reflektiertes-XSS.html

Persistentes XSS (auch JavaScript-Injection genannt)

Der Schadcode wird auf dem Webserver gespeichert, zum Beispiel in einem Kommentar oder einem Gästebuch-Eintrag, und danach bei jedem Aufruf der Seite mit dem betroffenen Eintrag ausgeliefert. Der Angriff erfolgt bei jedem Aufruf der Seite mit dem eingeschleusten Schadcode. Wie der Benutzer diese Seite erreicht, ist egal.

Persistentes XSS funktioniert fast genauso wie reflektiertes XSS, nur das der Angreifer nun seinen JavaScript-Schadcode zum Beispiel in einem Kommentar unter einem Blogeintrag, in einem Eintrag im Gästebuch oder einer der vielen Möglichkeiten des "User Generated Content" im Web 2.0 speichert. Immer vorausgesetzt natürlich, dort gibt es eine entsprechende Schwachstelle und der XSS-Code kann eingeschleust werden.

Immer, wenn der manipulierte Eintrag aufgerufen wird, wird der darin gespeicherte JavaScript-Schadcode des Angreifers im Webbrowser des Benutzers im Kontext der betroffenen Webanwendung ausgeführt.

Dabei ist es egal, ob ein Benutzer gezielt auf diese Seite gelockt wird, sie während des normalen Besuchs der Website erreicht oder zum Beispiel von einer Suchmaschine kommt. Auch ist es prinzipiell egal, ob die Seite von einem normalen Benutzer aus dem Internet heraus ausgerufen wird, oder ob ein Administrator sie aus dem Backend der Webanwendung heraus öffnet. Jedenfalls, so lange die Seite im Browser gerendert wird. Öffnet der Admin sie zum Beispiel in einem Editor-Fenster des Backends, wird der enthaltene Code natürlich nicht ausgeführt, sondern dargestellt.

Den prinzipiellen Ablauf des Angriffs zeigt folgendes Bild:

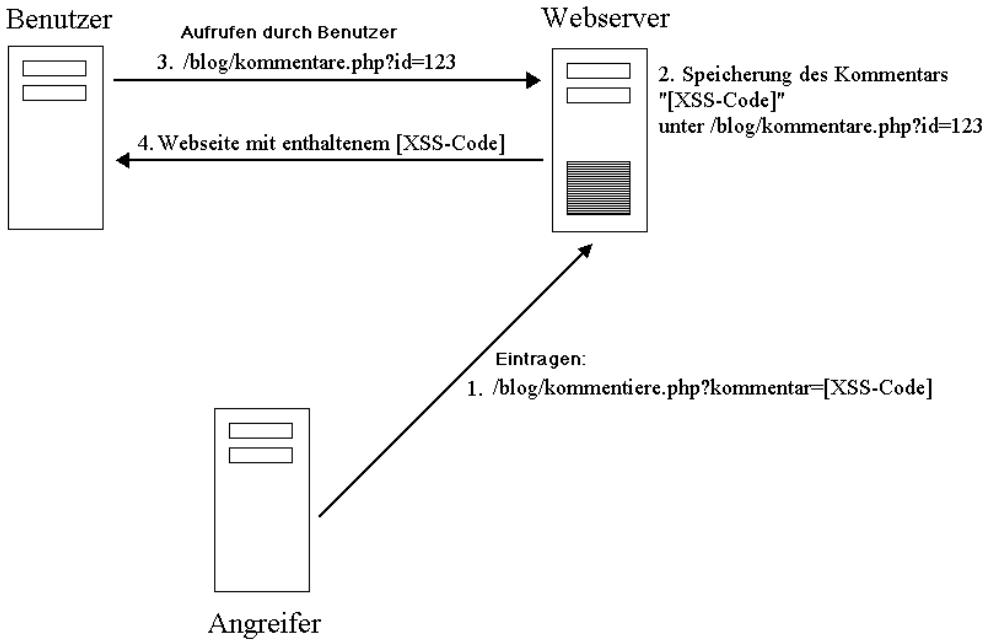


Abbildung 6: Reflektiertes XSS

Quelle:

https://www.ceilers-news.de/serendipity/625-Cross-Site-Scripting-im-UEberblick,-Teil-2-Persistentes-XSS.html

DOM-basiertes XSS

Der Angriff spielt sich ausschließlich im Webbrowser ab, im Unterschied zu Angriffen über reflektiertes oder persistentes XSS ist der Schadcode niemals Bestandteil der vom Server gelieferten HTML-Daten. Weder der Server noch ein IDS/IPS oder eine Web Application Firewall (WAF) können ihn darin also erkennen.

Der Client-seitige Code einer Webanwendung ist immer dann für DOM-basiertes XSS anfällig, wenn sie Daten aus vom Angreifer kontrollierbaren Objekten wie zum Beispiel `document.location`, `document.URL` oder `document.referrer` oder in Zeiten von HTML5 und Web 2.0 auch irgendwelche lokalen Eingaben ohne Prüfung auf eingeschleusten Code verwendet.

Eine Webseite enthält den folgenden Code:

```
Hallo
<script>
  var pos=document.URL.indexOf("name")+5;
  document.write(document.URL.substring(pos,document.URL.length));
</script>
,<br>

Herzlich Willkommen auf dieser Seite ...
```

Beim Aufruf dieser Seite wird der Benutzer mit seinem Namen begrüsst, zum Beispiel beim Aufruf von

```
http://www.beispiel.example/index.html?name=Carsten
```

mit

```
Hallo Carsten,
Herzlich Willkommen auf dieser Seite ...
```

Der übliche XSS-Test mit

```
http://www.beispiel.example/index.html?name=<script>alert('XSS')</script>
```

führt dagegen zum Öffnen der Alertbox.

Dieses Beispiel funktioniert nicht, wenn der Webbrowser den URL selbständig URL-kodiert, wie es zum Beispiel Mozilla und Firefox tun. Dadurch werden die < und > zu %3C bzw. %3E, was die spätere Codeausführung verhindert. Die Umkodierung verhindert aber keine Angriffe, die nicht auf < und > angewiesen sind.

Der Angriff ist möglich, weil der Browser beim Aufruf des präparierten URL einen HTTP-Request an `www.beispiel.example` sendet, die statische HTML-Seite mit obigen Code darin empfängt und dann beginnt, die HTML-Daten in das DOM zu parsen. Das DOM enthält das Objekt `document`, das die Eigenschaft `URL` besitzt, in der der URL der aktuellen Seite steht. Erreicht der Parser den JavaScript-Code, führt er ihn aus und ändert entsprechenden den enthaltenen Anweisungen die Seite. Der Code kopiert einen Teil des Inhalts von `document.URL` in die HTML-Seite. Im Beispiel also "`Carsten`" - oder eben

den Schadcode "`<script>alert('XSS')</script>`". Das Ergebnis sieht im zweiten Fall folgendermassen aus:

```
Hallo <script>alert('XSS')</script>,
Willkommen auf dieser Seite ...
```

Die fertiggestellte HTML-Seite wird dann geparst - und dabei der eingeschleuste Skriptcode ausgeführt.

Quelle:

https://www.ceilers-news.de/serendipity/633-Cross-Site-Scripting-im-UEberblick,-Teil-4-DOM-basiertes-XSS.html

2.2 Sicherheitslücken und ihre Ursachen

Aufbau von Web-Applikationen (three tier architectur)

In heutigen IT-Systemen sind Web-basierte Anwendungen, kurz Web-Applikationen (oder Onlineapplikationen) sehr weit verbreitet. Sie ermöglichen eine einfache Zusammenarbeit zwischen einem Dienstanbieter und den Benutzern. Typischerweise stellt der Dienstanbieter seine Web-Applikation über einen Web-Server oder ein Portal, beispielsweise einem **Apache-** oder **IIS-Server**, zur Verfügung. Web-Applikationen werden meist in Programmiersprachen wie **PHP**, **JSP** (Java Server Pages), **Perl** oder **Python** geschrieben. Sie werden auf dem Web-Server ausgeführt und verarbeiten dabei sehr häufig Daten, die in SQL-Datenbanken abgelegt sind und im Intranet des Dienstanbieters verwaltet werden. Die Nutzung eines Dienstes ist für Clients sehr einfach mittels des HTTP-Protokolls möglich.

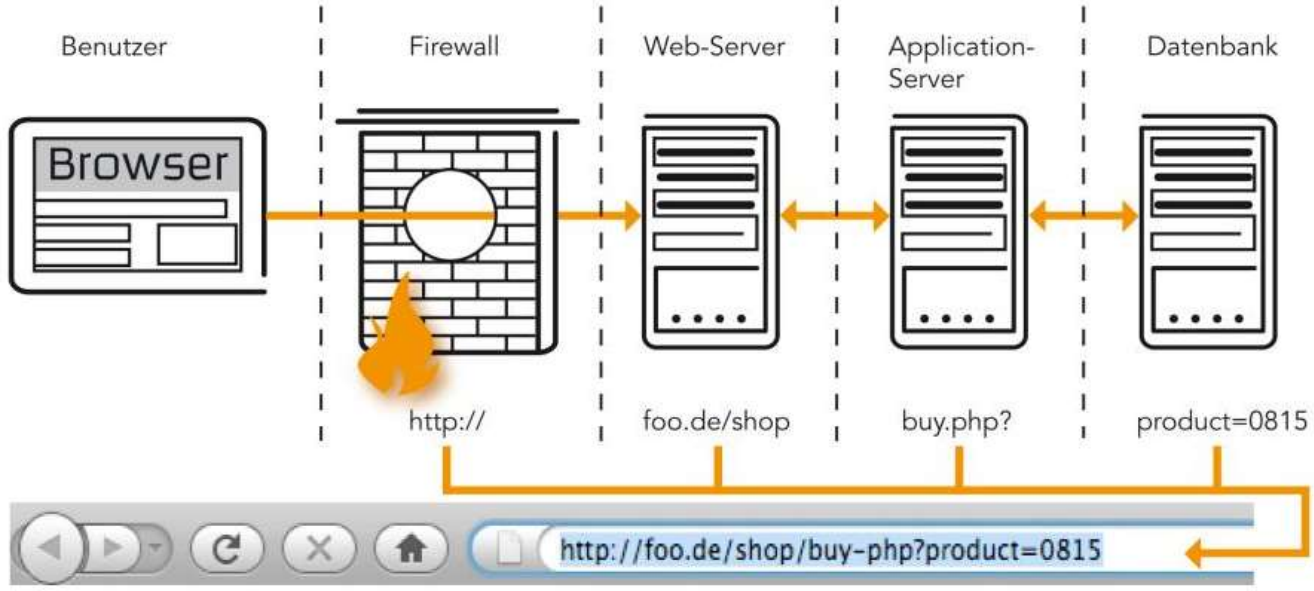


Abbildung 7: Beteiligte Komponenten beim Starten einer Webanwendung

Benutzer mit Browser

Web-Browser, wie der Microsoft Internet Explorer oder Firefox stellen Dienste zum Navigieren im WWW, zur Darstellung von Seiten sowie zur Interaktion mit Servern zur Verfügung. Mittels eines Browsers können interaktive Benutzereingaben an den Server weitergeleitet werden. Beispiele dafür sind Datenbankabfragen oder der elektronische Einkauf mit einem interaktiven Aushandeln von Vertragsbedingungen.

Um Objekte von einem Server zum Browser zu übertragen, ist es notwendig, Objekte eindeutig zu benennen. Die Benennung erfolgt über den **Uniform Resource Locater** (URL) bzw. URI (Uniform Resource Identification)

Firewall

Diese haben die Aufgabe, durch Kontrolle und Filterung von Datenpaketen die Weiterleitung solcher Pakete zu verhindern, die eine mögliche Bedrohung für die Daten und Komponenten eines Netzsegmentes bedeuten können.

Web-Server und Proxy

Aufgrund der Zustandslosigkeit des HTTP-Protokolls muss bei jedem Zugriff auf ein WWW-Objekt eine TCP-Verbindung zwischen Client und Server aufgebaut werden. Zur Effizienzsteigerung speichert man deshalb häufig das benötigte Objekt in einem Cache eines lokalen Proxy-Servers, der als Stellvertreter für den Web-Server fungiert.

Datenbank

Je nach Anwendung gelangen die Daten auf unterschiedliche Arten in eine Datenbank. Protokolldatenbanken, die Zugriffe und Modifikationen festhalten werden automatisch durch die Applikation befüllt. Webbasierte Anwendungen wie e-Shops, Gästebücher oder Foren werden zumeist durch Gelegenheitsanwender ausgefüllt. In einigen Branchen gibt es dann noch die Massendatenerfassung. Hier werden grosse Mengen an Daten durch Personen manuell erfasst.

Applikationsserver

Ursprünglich waren HTML-Seiten statische Seiten. Heute ist es jedoch üblich, dass HTML-Seiten dynamisch erzeugt werden, um beispielsweise über aktuelle Daten des Aufrufers den Seiteninhalt zu beeinflussen. Zu den ersten Technologien, gehörte die **CGI-Skriptsprache** (Common Gateway Interface). Ein CGI-Skript kann in einer beliebigen interpretierbaren Programmiersprache erstellt sein. CGI in Kombination mit Perl wird für kleinere Anwendungen nach wie vor genutzt. Für komplexere Anwendungen kommen aber heute in der Regel Technologien wie PHP, Active Server Pages (ASP.net) von Microsoft, Java Server Pages (JSP) zum Einsatz. CGI-Skripte werden auf dem Web-Server ausgeführt und nur das Ergebnis der Ausführung wird an den Client-Rechner übertragen. Hierbei ist also zunächst der Server Bedrohungen ausgesetzt.

Im Gegensatz dazu stehen die Webseiten mit sogenannten **aktiven Inhalten**, dazu zählt man Java Scripts, Applets, usw. Aktive Inhalte werden über den lokalen Browser vom Web-Server auf den Rechner des Zugreifers heruntergeladen und dort ausgeführt.

Schwachstellen und Verwundbarkeit ergeben sich sowohl für den Client-Rechner als auch für den Web-Server und die internen Datenbanken, als auch für die Daten, die über unterschiedliche Kommunikationsnetze zwischen diesen Komponenten ausgetauscht werden. Die Kommunikationsverbindung zwischen Client und Web-Server ist allen klassischen Internet-Angriffen wie beispielsweise, **Sniffing**, **Man-in-the-Middle**, **DNS-Rebinding**, **Cache-Poisoning** oder auch **Session Hijacking** ausgesetzt.

Client-Rechner sind einer Vielzahl von Angriffen wie beispielsweise durch **Phishing**, **Web-Server-Spoofing**, **Cross Site Scripting** oder auch **Session Riding** ausgesetzt.

2.3 Angriffe richten sich auf Schwachstellen der Web-Applikationen

Angriffe richten sich gezielt gegen bekannte Schwachstellen von Applikationen, wie z.B. Cross-Site Scripting, SQL Injection, Buffer Overflow und weitere.

Der Löwenanteil der IT-Security-Budgets wird in Netzwerksicherheit, z. B. für Firewalls und Intrusion Detection-Systeme investiert, obwohl nach Gartner[3] 75 Prozent der Hacker-Attacken direkt die Applikation und nicht die Netzwerke angreifen. Um Geld gezielt dort einzusetzen, wo die grösste Gefahr lauert, muss ein Umdenken stattfinden. Idealerweise wird in der Software-Entwicklung dafür gesorgt, dass erst gar keine Sicherheitslöcher entstehen oder diese unmittelbar behoben werden.

Hauptursachen für Sicherheitsprobleme bei Web-Anwendungen sind Fehler bei der Programmierung.

In Onlineapplikationen ist Sicherheit besonders wichtig, beachtet wird sie aber viel zu wenig. Onlinesicherheitsrisiken haben sich innert kürzester Zeit zum Hauptrisiko für die Unternehmungen entwickelt. Ein Grund dafür ist, dass businesskritische Applikationen heute für die Kunden, Partner und internen Benutzer online jederzeit erreichbar sein müssen. Die ständige Verfügbarkeit öffnet Tür und Tor für Freund und Feind. Interaktive Inhalte verschlimmern das Ganze. Statistiken zeigen, dass sich ein Hacker nicht zweimal bitten lässt.

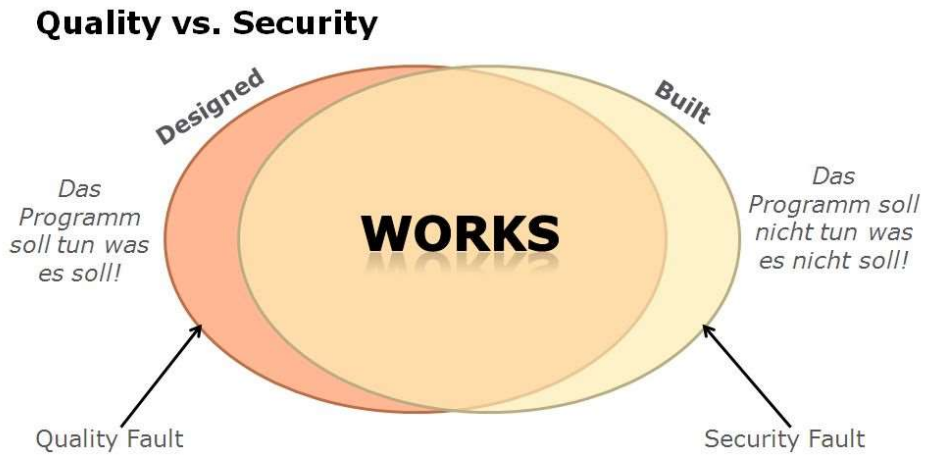


Abbildung 8:Quality vs. Security "It's not a bug, it's a feature"

Dass qualitativ schlechte Programme vielfach im Betrieb noch weitere eigentlich gar nicht geplante Eigenschaften aufweisen, oder sich unerklärlich verhalten und so zum Sicherheitsrisiko werden, sollte einem erfahrenen Software-Entwickler bekannt sein.
Gerade solche „ungeplanten Features“ werden gerne von Hackern als Angriffspunkte missbraucht.

[3] http://www.gartner.com/technology/research.jsp

2.4 Vorgehen zur Identifikation von Sicherheitslücken

Web Applikation Scanner

Ohne Unterstützung durch Web-Applikation Scanner sind komplexe Webanwendungen kaum noch umfassend, genau und wirtschaftlich zu testen. Ihre Hauptaufgabe besteht darin, Konfigurations- und Implementierungsfehler im Zusammenspiel der Applikationskomponenten aufzuspüren.

Web Scanner untersuchen eine Anwendung auf bekannte Schwachstellen und unterstützen dabei den Webentwickler beim Auffinden von Sicherheitslücken. Allerdings kann dies auch von einem Angreifer verwendet werden. Dabei wird auf einem Client-Computer der Web-Scanner installiert, um dann die auf einem anderen entfernten Server liegende Webanwendung zu analysieren. Allgemein gilt: Web-Scanner sind sinnvolle Hilfsmittel, um erfahrene Benutzer bei der Analyse der Sicherheitseigenschaften einer Webanwendung zu unterstützen. Die Tatsache, dass manche Schwachstellen von einem Werkzeug grundsätzlich nicht erkennbar sind, haben zur Folge, dass weiterhin geübte Personen für den Grossteil der Analyse benötigt werden.

Quelle: https://de.wikipedia.org/wiki/Sicherheit_von_Webanwendungen

Weitere Quellen: [5]

Sicherheitsanalyse mittels Penetrationstests

Das "Metasploit-Framework" gilt mittlerweile als Industriestandard für Penetrationstest und entwickelt sich mit über einer Million Downloads im Jahr zur populärsten öffentlich zugänglichen Exploit-Datenbank der Welt. Es geht darum Schwachstellen im System zu ermitteln und als abschliessende Massnahme Vorschläge zur Absicherung zu machen. Die Test laufen in verschiedenen Phasen: Das Bundesamt für Sicherheit in der Informationstechnik (BSI) beschreibt die folgenden fünf Phasen

Phase1:	Vorbereitung
Phase2:	Informationsbeschaffung und –Auswertung
Phase3:	Bewertung der Informationen/Risikoanalyse
Phase4:	Aktive Eindringversuche
Phase5:	Abschlussanalyse

Phase 1 – Vorbereitung: In dieser Phase werden unter anderem die Ziele festgelegt, die durch den Test erreicht werden sollen. Neben den Zielsystemen wird typischerweise die Vorgehensweise dargestellt und an die vorhandene Umgebung angepasst. An dieser Stelle wird zudem über die "Agressivität“ der Vorgehensweise diskutiert, und es wird oftmals bereits entschieden, welche Systeme unter welchen Umständen mit möglichem Exploit-Code penetriert werden dürfen bzw. wie der Ablauf und Informationsaustausch vor einem solchen Einsatz zu erfolgen hat.

Phase2 – Informationsbeschaffung und –auswertung: In der ersten technischen Phase, der Phase zur Informationsbeschaffung, wir versucht, möglichst viele Details über die prüfende Umgebung bzw. die zu prüfenden Systeme zu ermitteln. Die Herangehensweise an ein Zielsystem beginnt oftmals mit einfachen Suchabfragen über

unterschiedliche Online-Suchmaschinen. Im Anschluss an solche rein passiven Methoden kommen typischerweise auch erheblich aktivere Vorgehensweisen zum Einsatz. Zu diesen zählen typischerweise Scanningtools wie Port- und Vulnerability-Scanner

Phase3 – Bewertung der Informationen/Risikoanalyse: In dieser Phase müssen die bereits ermittelten Informationen aus Phase 2 auf mögliche Schwachstellen analysiert werden. Darauf basierend ist es möglich, weiteres Angriffspotential zu erkennen. In dieser Phase unterstützt Metasploit den Tester bei einer raschen und möglichst korrekten Auswahl der Zielsysteme und der eizusetzenden Exploits bzw. Module.

Phase4 Aktive Eindringversuche: Hier handelt es sich um die kritischste technische Phase. Es wird versucht die erkannten Schwachstellen aktiv auszunutzen, um darüber Zugriff auf die Systemumgebung zu erlangen. Es kommt dabei häufig zum Einsatz von Exploit-Code, der oftmals dazu führen kann, Dienste oder ganze Systeme und deren Verfügbarkeit negativ zu beeinflussen.

Phase5 – Abschlussanalyse: In dieser wird typischerweise eine detaillierte Auswertung und Aufbereitung aller ermittelten Ergebnisse und Informationen durchgeführt. Auf Basis dieser Informationen kommt es zur Erstellung des Abschlussreports. Dieser sollte neben einer Management-Zusammenfassung und einer Auflistung der gefundenen Schwachstelle auch detaillierte Informationen zu den erkannten Schwachstellen und zu möglichen Risiken und Gefährdungen umfassen.

Quelle:

https://www.bsi.bund.de/DE/Publikationen/Studien/Pentest/index_htm.html

[iX 8/2011, Metasploit auf Schwachstellensuche S. 115]

2.5 OWASP Top Ten

Innerhalb im bereits in Kap. 1 erwähnten "Open Web Application Security Project" (OWASP) gibt es verschiedene Teilprojekte, so etwa das "OWASP Top Ten Project", das regelmässig die jeweils zehn kritischsten Schwachstellen von Web-Applikationen beschreibt (Top Ten 2017).

Ziel soll sein: Entwickler, Designer, Architekten und Unternehmen für potenzielle Schwachstellen zu sensibilisieren und aufzuzeigen wie sich diese vermeiden lassen.

Allgemeine bedeutet Risiko immer Verlust von Geld oder sogar Menschenleben. Die nachfolgende Formel zeigt wie Sicherheitsexperten in einer Sicherheitsanalyse das Risiko bestimmen.

Risiko = Eintrittswahrscheinlichkeit x Schaden

Der erste Faktor, die Eintrittswahrscheinlichkeit setzt sich aus drei Teilen zusammen, deren Zahlenwerte gleichmässig gewichtet und als Durchschnitt zusammengefasst werden.

1. **Ausnutzbarkeit:** Wie schwierig ist es, einen Angriff durchzuführen? Die Ausnutzbarkeit ist grösser, wenn Werkzeuge vorhanden sind. Die verfügbaren Mittel und Techniken spielen für einen Angreifer eine wichtige Rolle, wenn er seinen Aufwand plant. Diese Einzelbestandteile werden auch als Angriffsvektor bezeichnet.
2. **Verbreitung von Schwachstellen:** Sind viele Anwendungen mit solchen Schwachstellen im Internet vorhanden? Software, die häufig betrieben wird, ist eher Ziel von Angriffen.

3. **Auffindbarkeit der Schwachstelle:** Wie aufwändig ist es, diese Schwachstelle zu identifizieren? Je "redseliger" eine Anwendung ist, desto einfacher ist es, genau für diese Version eine Schwachstelle zu finden.

Während ein Unternehmen gegen den ersten Punkt keine Vorkehrungen treffen kann, können bezüglich des zweiten und dritten Punkts Sicherheitsmassnahmen eingeleitet werden, indem eine Schwachstelle gezielt verbessert wird

Der zweite Faktor, der mögliche **Schaden** wird in der OWASP-Grafik mit der technischen Auswirkung beschrieben, sollte eine Ressource "gehackt" werden.

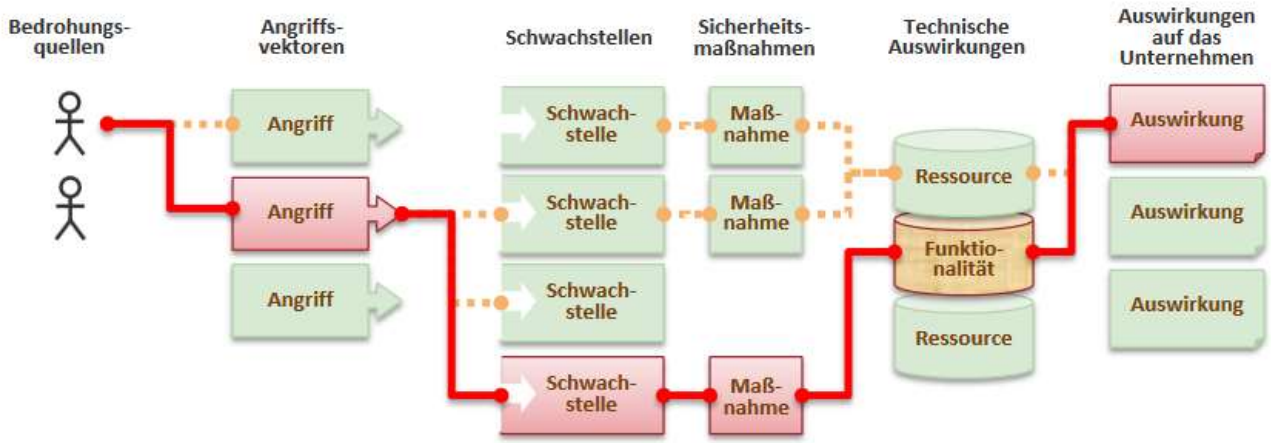


Abbildung 9:Was sind Sicherheitsrisiken für die Anwendungen

Selbst eklatante Software-Schwachstellen müssen nicht zwangsläufig ein ernsthaftes Risiko darstellen, wenn es z.B. keine Bedrohungsquellen gibt, die den notwendigen Angriff ausführen können.

Die Werte für die drei Bestandteile der **Eintrittswahrscheinlichkeit** (Ausnutzbarkeit, Verbreitung, Auffindbarkeit) und des **Schadens** (Auswirkung) ist in Form von Farben in der untenstehenden Zusammenfassung ersichtlich:

RISIKO	Bedrohungs-quellen	Angriffs-vektoren	Schwachstelle		Technische Auswirkungen	Auswirkungen auf das Unternehmen
		Ausnutzbarkeit	Verbreitung	Auffindbarkeit	Auswirkung	
A1-Injection	Anwendungs-spezifisch	EINFACH	HÄUFIG	DURCHSCHNITTLICH	SCHWERWIEGEND	Anwendungs- / Geschäfts-spezifisch
A2-Authentisierung		DURCHSCHNITTLICH	SEHR HÄUFIG	DURCHSCHNITTLICH	SCHWERWIEGEND	
A3-XSS		DURCHSCHNITTLICH	AUSSERGWÖHNLICH HÄUFIG	EINFACH	MITTEL	
A4-unsich. DOR		EINFACH	HÄUFIG	EINFACH	MITTEL	
A5-Konfiguration		EINFACH	HÄUFIG	EINFACH	MITTEL	
A6-Sens. Daten		SCHWIERIG	SELTEN	DURCHSCHNITTLICH	SCHWERWIEGEND	
A7-Funkt. Zugriff		EINFACH	HÄUFIG	DURCHSCHNITTLICH	MITTEL	
A8-CSRF		DURCHSCHNITTLICH	HÄUFIG	EINFACH	MITTEL	
A9-Komponenten		DURCHSCHNITTLICH	SEHR HÄUFIG	SCHWIERIG	MITTEL	
A10-Weiterleit.		DURCHSCHNITTLICH	SELTEN	EINFACH	MITTEL	

Abbildung 10: Top Ten Risiko-Faktoren (2013)

Beispiel: Risikobewertung A1-Injection

> Ausnutzbarkeit: **einfach**, Verbreitung: **häufig**, Auffindbarkeit: **durchschnittlich**
> Schaden: **schwerwiegend**
> **=> sehr grosses Risiko**

OWASP Top Ten Project (2017)

Die folgenden "OWASP Top Ten" aus dem Jahre 2017 stellen unter Web-Sicherheitsexperten einen anerkannten Konsens dar, was die derzeit kritischsten Lücken in Web-Anwendungen betrifft.

OWASP Top Ten - 2010	OWASP Top Ten - 2013	OWASP Top Ten – 2017
A1 – Injection Flaws	A1 – Injection Flaws	A1 – Injection Flaws
A2 - Cross Site Scripting (XSS)	A2 - Broken Authentication and Session Management	A2 - Broken Authentication and Session Management
A3 - Broken Authentication and Session Management	A3 - Cross Site Scripting (XSS)	A3 – Sensitive Data Exposure
A4 - Insecure Direct Object Reference	A4 - Insecure Direct Object Reference (fliesst in 2017 A5 ein)	A4 – XML External Entities (XXE)
A5 - Cross Site Request Forgery (CSRF)	A5 - Security Misconfiguration	A5 – Broken Access Control
A6 - Security Misconfiguration	A6 - Sensitive Data Exposure	A6 – Security Misconfiguration
A7 - Insecure Cryptographic Storage (fliesst in 2013 A6 ein)	A7 - Missing Function Level Access Control (fliesst in 2017 A5 ein)	A7 – Cross-Site Scripting (XSS)
A8 - Failure to Restrict URL Access	A8 - Cross Site Request Forgery (CSRF)	A8 – Insecure Deserialization
A9 - Insufficient Transport Layer Protection (fliesst in 2013 A6 ein)	A9 - Using Known Vulnerable Components	A9 - Using Known Vulnerable Components
A10 - Unvalidated Redirects and Forwards (Fliesst in 2013 A6 ein)	A10 - Unvalidated Redirects and Forwards	A10 – Insufficient Logging & Monitoring

Abbildung 11: Entwicklung der OWASP Top 10

A1:2017 – Injection (höchstes Risiko)

Injection-Schwachstellen, wie beispielsweise SQL-, Betriebssystem- 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.

A2:2017 – 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 Session-Tokens zu stehlen oder die Schwachstellen so auszunutzen, dass sie die Identität anderer Benutzer annehmen können.

A3:2017 – Verlust der Vertraulichkeit sensibler Daten

Viele Anwendungen schützen sensible Daten, wie Kreditkartendaten oder Zugangsinformationen nicht ausreichend. Angreifer können solche nicht angemessen geschützten Daten auslesen oder modifizieren und mit ihnen weitere Straftaten, wie beispielsweise Kreditkartenbetrug oder Identitätsdiebstahl begehen. Vertrauliche Daten

benötigen zusätzlichen Schutz, wie z.B. Verschlüsselung während der Speicherung oder Übertragung sowie besondere Vorkehrungen beim Datenaustausch mit dem Browser.

A4:2017 – XML External Entities (XXE)

Wird XML verwendet, kann es über XML External Entities manipuliert werden. Und ob die TOP 10 wirklich so eine grosse Bedrohung sind, ist auch immer relativ. Für Webanwendungen, die XML gar nicht nutzen, ist das völlig harmlos. Was nicht da ist, kann auch nicht angegriffen werden.

A5:2017 – Verlust der Zugriffskontrolle

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.

Die meisten betroffenen Anwendungen realisieren Zugriffsberechtigungen nur durch das Anzeigen oder Ausblenden von Funktionen in der Benutzeroberfläche. Allerdings muss auch beim direkten Zugriff auf eine geschützte Funktion eine Prüfung der Zugriffsberechtigung auf dem Server stattfinden, ansonsten können Angreifer durch gezieltes Manipulieren von Anfragen ohne Autorisierung trotzdem auf diese zugreifen.

A6:2017 – Sicherheits-relevante Fehlkonfiguration

Sicherheit erfordert die Festlegung und Umsetzung einer sicheren Konfiguration für Anwendungen, Frameworks, Applikations-, Web- und Datenbankserver sowie deren Plattformen. Sicherheitseinstellungen müssen definiert, umgesetzt und gewartet werden, die Voreinstellungen sind oft unsicher. Des Weiteren umfasst dies auch die regelmässige Aktualisierung der Software.

A7:2017 – Cross-Site Scripting (XSS)

XSS-Schwachstellen treten auf, wenn eine Anwendung nicht vertrauenswürdige Daten entgegennimmt und ohne entsprechende Validierung oder Umkodierung an einen Webbrowser sendet. XSS erlaubt es einem Angreifer, Skriptcode im Browser eines Opfers auszuführen und somit Benutzersitzungen zu übernehmen, Seiteninhalte zu verändern oder den Benutzer auf bösartige Seiten umzuleiten.

A8:2017 – Unsichere Deserialisierung

Werden Daten serialisiert gespeichert (vereinfacht werden dabei komplizierte Strukturen einfach hintereinander weg in eine Variable geschrieben), müssen sie später wieder deserialisiert, d. h. in die Ausgangsdaten zurückverwandelt werden. Ein Angreifer kann eine Schwachstelle dabei ausnutzen, um Daten zu manipulieren.

A9:2017– Nutzung von Komponenten mit bekannten Schwachstellen

Komponenten, wie z.B. Bibliotheken, Frameworks oder andere Softwaremodule werden meistens mit vollen Berechtigungen ausgeführt. Wenn eine verwundbare Komponente ausgenutzt wird, kann ein solcher Angriff zu schwerwiegendem Datenverlust oder bis zu einer Serverübernahme führen. Applikationen, die Komponenten mit bekannten Schwachstellen einsetzen, können Schutzmassnahmen unterlaufen und so zahlreiche Angriffe und Auswirkungen ermöglichen.

A10:2017 – unzureichende Protokollierung und Überwachung (kleineres Risiko)

Das ist wie die XXE auf Platz 4 ein Neuzugang in den Top 10 2017. Da es sich weder um eine Schwachstelle in der Webanwendung noch um einen Angriff darauf handelt, passt es eigentlich nicht richtig in die Liste. Dabei

ist es doch sehr wichtig, denn wenn Sie nicht merken, dass jemand versucht, ihre Webanwendung anzugreifen, können Sie den Angriff auch nicht abwehren.

Quelle: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Deadly Sins of Software Security

Hauptursachen für Sicherheitsprobleme bei Web-Anwendungen sind Fehler beim Design und bei der Programmierung.

Dass die Entwicklung von sicheren und zuverlässigen Software-Applikationen kein Kinderspiel ist, verdeutlichte John Viega[4] in seinem Buch "19 Deadly Sins of Software Security - Programming Flaws and How to Fix Them".

Programmier-Todsünden im Überblick

- Pufferüberläufe
- Format-String-Probleme
- Integer-Range-Probleme
- SQL-Injection
- Command-Injection
- Fehlerhaftes Error-Handling
- Cross-Site-Scripting
- Kein Schutz des Netzwerkverkehrs
- Magic-URLs und versteckte Formulare
- Fehlerhafte Nutzung von SSL
- Nutzung von schwachen, passwortbasierten Systemen
- Unsichere Speicherung von Daten
- "Hardcoden" von Geheimnissen
- Fehlerhafter Dateizugriff
- Das Trauen von Netzwerkadressen- Informationen
- Race Conditions (Wettlaufsituationen)
- Unauthentifizierter Schlüsselaustausch
- Non-Random Benutzer
- Schlechte Bedienbarkeit

Quelle:

http://www.zdnet.de/magazin/39141304/die-19-todsuenden-der-software-security.htm

[4] Chief Security Architekt beim amerikanischen Sicherheitsanbieter McAfee

3 Grundlegende Gegenmassnahmen

Als wichtige Massnahme sollen die Applikationen mit geeigneten Mitteln auf "Herz und Nieren" gegen Schwachstellen getestet werden, wobei man sich aber bewusst sein muss, dass Security beim ganzen Entwicklungsprozess ein fundamentaler Bestandteil sein sollte.

Ein neues Gebiet, das Security Engineering befasst sich mit Massnahmen und Methoden, die bei der Entwicklung qualitativ hochstehender Systeme zu beachten sind. Microsoft hat mit Software **Developement Lifecycle** (SDL) ein Ansatz vorgestellt, der auf die Praxis der Entwicklung sicherer Systeme zugeschnitten ist. Das sichere Programmieren ist dabei ein Teilaspekt. Hierzu gibt es in der Praxis eine Reihe von Best-Practice Empfehlungen und goldene Regeln, worauf ein Entwickler bei der Entwicklung sicherer Software achten sollte. Weitere Aspekte die berücksichtig werden sollten:

- Abklären des Schutzbedarfes von Systemen
- Bedrohungs- Risikoanalyse
- Sicherheitsstrategien
- Sicherheitsarchitektur

Fazit: Frühzeitig erkannte Sicherheitsmängel lassen sich deutlich schneller und billiger bereinigen.

3.1 Authentifizierung und Autorisierung

Bei der Authentifizierung handelt es sich um die Überprüfung der tatsächlichen Identität des Benutzers. Für diese Überprüfung werden die Anmeldeinformationen des Benutzers (Benutzername, Passwort) mit einer Liste von bekannten Daten verglichen. Wenn die Anmeldeinformationen korrekt sind, authentifizieren wir den Benutzer, andernfalls nicht.

Autorisierung wird als eine Sicherheitsfunktion betrachtet. Es geht darum, welcher Benutzer welchen Zugriff auf welche Funktionen hat.

Quelle:

http://msdn.microsoft.com/de-de/library/bb978972(d=printer).aspx

3.2 Einsatz von Kryptologie

Der Begriff **Kryptologie** setzt sich aus den zwei griechischen Wörtern kryptós, „verborgen“, und logos = Lehre, Kunde zusammen. Die Kryptologie ist also die Lehre des Verborgenen bzw. Verbergens. Gerade im heutigen Internetzeitalter ist das Verbergen von Daten ein nicht mehr wegzudenkender Bestandteil der gesamten Sicherheit des Datenverkehrs. Darum sollten Sie sich darüber grundlegende Kenntnisse aneignen. Beim Umgang mit-Webapplikationen sind Verschlüsselungsverfahren, Hashfunktionen, Zertifikate, usw. wichtige Elemente, die Sie genau kennen müssen.
Aufpassen muss man mit den Begrifflichkeiten in diesem Bereich, denn wenn jemand von Kryptographie z.B. redet, darf man dies nicht mit dem ähnlichen Begriff Kryptologie oder gar Kryptoanalyse verwechseln. So teilt man die **Kryptologie**, die ganz oben an der hierarchischen Struktur steht, in zwei Unterthemen ein, die wiederum viele Themen vereinen. Grundsätzlich unterscheidet man bei der Kryptologie nämlich zwischen der **Kryptographie** und der **Kryptoanalyse**. Diese Einteilung stammt vom russisch-amerikanische Kryptologe William Friedman und geht auf das Ende des Ersten Weltkrieges zurück.

Symmetrische Verschlüsselung

Abbildung 12:Symmetrische Verschlüsselung

Die Verschlüsselungsverfahren der symmetrischen Kryptografie arbeiten mit einem einzigen Schlüssel, der bei der Ver- und Entschlüsselung vorhanden sein muss. Diese Verfahren sind schnell und bei entsprechend langen Schlüsseln bieten sie auch eine hohe Sicherheit.

Der Knackpunkt liegt in der Schlüsselübergabe zwischen den Kommunikationspartnern. Vor der sicheren Datenübertragung mit Verschlüsselung müssen sich die Kommunikationspartner auf den Schlüssel einigen und austauschen

Knackpunkt ist der unsichere Schlüsselaustausch und die Authentifizierung der Kommunikationspartner. Sicher ist die Schlüsselübergabe nur dann, wenn sich zwei Personen persönlich treffen und den Schlüssel austauschen oder der Schlüssel einen anderen Weg nimmt (Seitenkanal), wie es die Daten tun. Zur Unsicherheit trägt außerdem bei, wenn einer der Kommunikationspartner den Schlüssel nur ungenügend aufbewahren.

Asymmetrische Verschlüsselung

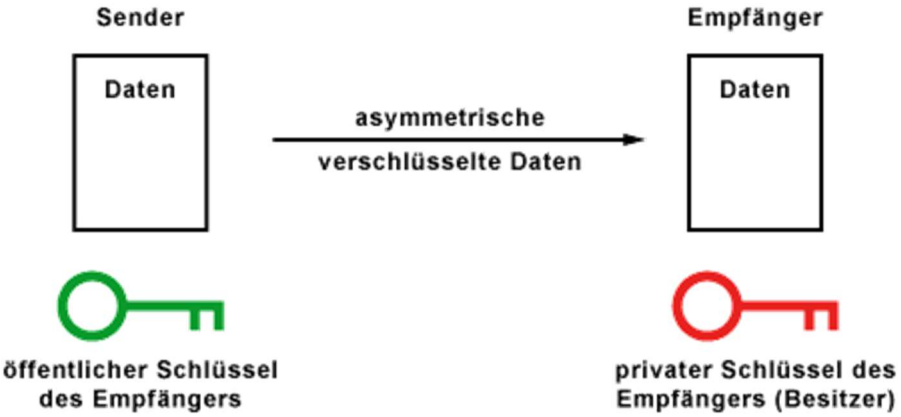


Abbildung 13:Asymmetrische Verschlüsselung

Asymmetrische Verschlüsselungsverfahren arbeiten mit Schlüsselpaaren. Ein Schlüssel ist der öffentliche Schlüssel (Public Key), der andere ist der private Schlüssel (Private Key). Dieses Schlüsselpaar hängt über einen mathematischen Algorithmus eng zusammen. Daten, die mit dem öffentlichen Schlüssel verschlüsselt werden, können nur mit dem privaten Schlüssel entschlüsselt werden. Deshalb muss der private Schlüssel vom Besitzer des Schlüsselpaares geheim gehalten werden.

Der konkrete Anwendungsfall sieht so aus: Will der Sender Daten verschlüsselt an den Empfänger senden, benötigt er den öffentlichen Schlüssel des Empfängers. Mit dem öffentlichen Schlüssel können die Daten verschlüsselt, aber nicht mehr entschlüsselt werden (Einwegfunktion). Nur noch der Besitzer des privaten Schlüssels, also der richtige Empfänger kann die Daten entschlüsseln. Wichtig bei diesem Verfahren ist, dass der private Schlüssel vom Schlüsselbesitzer absolut geheim gehalten wird. Kommt eine fremde Person an den privaten Schlüssel muss sich der Schlüsselbesitzer ein neues Schlüsselpaar besorgen.

Das Problem bei der asymmetrischen Kryptografie ist die Verteilung der öffentlichen Schlüssel. Typischerweise erfolgt die Übergabe des öffentlichen Schlüssels beim Erstkontakt. Doch hierbei stellt sich die Frage, ob dieser Schlüssel tatsächlich der echte Schlüssel des Kommunikationspartner ist.

Hinweis: Asymmetrische Verfahren benötigen viel mehr Rechenleistung als symmetrische Verfahren. Wenn man RSA und AES miteinander vergleicht, dann ist RSA ungefähr um den Faktor 1.000 langsamer als AES.

Hybride Verschlüsselungsverfahren

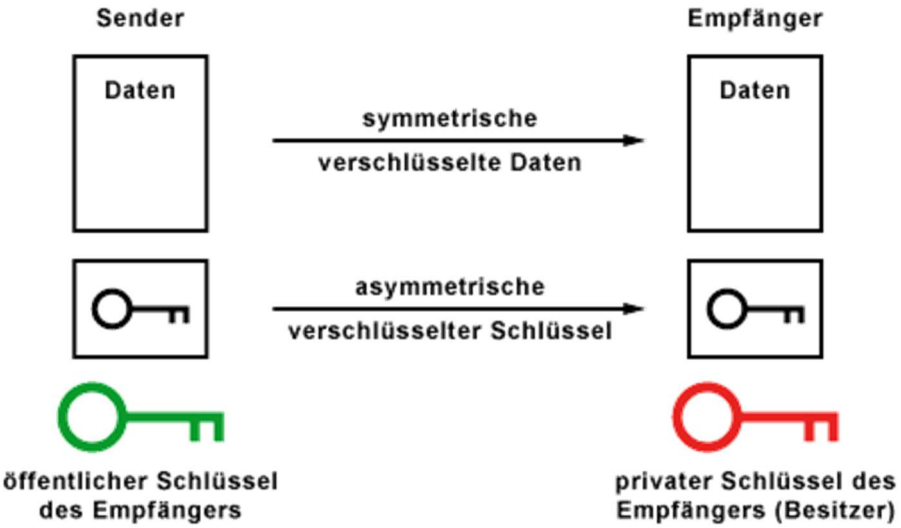


Abbildung 14:Hybride Verschlüsselung

1. Zuerst wird ein zufälliger Sitzungsschlüssel für die symmetrische Datenverschlüsselung generiert.
2. Dann verschlüsselt der Sender diesen Sitzungsschlüssel mit dem öffentlichen Schlüssel des Empfängers (asymmetrische Verschlüsselung).
3. Der Sender schickt dann den asymmetrisch verschlüsselten Sitzungsschlüssel an den Empfänger.
4. Mit seinem privaten Schlüssel kann der Empfänger den Sitzungsschlüssel entschlüsseln (asymmetrische Verschlüsselung).
5. Danach werden die Daten mit Hilfe des Sitzungsschlüssels verschlüsselt übertragen (symmetrische Verschlüsselung).

Quelle:

http://www.elektronik-kompendium.de/sites/net/1910141.htm

3.3 Web Application Firewall (WAF)

Ist eine Anwendung im Betrieb, stehen die Verantwortlichen in der Regel unter Zeitdruck und müssen schnell Fortschritt vorweisen. Dies sind die Hauptgründe, die eine nachhaltige Korrektur nicht mehr durchsetzbar machen. Umso wichtiger ist es, dass hier gewonnene Erkenntnisse kontinuierlich an die Entwicklung sowie Fachabteilung zurückfliessen. Die Application Firewall ist ein geeignetes Mittel, um eine zweite Sicherungslinie um die Anwendung herum aufzubauen. Weitere Quellen: [8]

Zusätzliche Absicherung bei älterer Webanwendungen

Das Bundesamt für Informationssicherheit (BSI) empfiehlt in seinem Leitfaden für bestehende Webanwendungen als sinnvolle Massnahme zur zusätzlichen Absicherung den Einsatz von WAF auch „Web Shields“ genannt.

Viele ältere Webanwendungen sind entweder ohne klare Sicherheitsanforderungen in den Produktivbetrieb übernommen worden oder zu einer Zeit erstellt worden, als das allgemeine Verständnis über die Anwendungssicherheit noch sehr gering gewesen ist. Zunächst ist Sorge zu tragen, bestehende Webanwendungen auf ein definiertes Sicherheitsniveau zu bringen. Dabei sollte nicht von der weit verbreiteten, aber falschen Annahme ausgegangen werden, dass eine unsichere Webanwendung dann zu keinem Schaden führt, wenn sie selbst keine sensiblen Daten bereitstellt oder keine sicherheitsrelevanten Funktionen ausführt.

3.4 Best-Practice Empfehlungen

Bewusstsein und Wissen schaffen für die erfolgreiche Umsetzung von sicheren Webanwendungen	Schulungen von Verantwortungsträgern und Umsetzern für mehr Bewusstsein und Know-how in Web Application Security
Konkrete Anlaufstellen und Zuständigkeiten definieren	Etablieren von Strukturen
Sicherheitsanforderungen für Entwicklungsphasen und –Übergänge definieren und berücksichtigen	Erarbeiten von Sicherheitsanforderungen
Abnahme sicherer Webanwendungen	Erstellen einer Roadmap für Web Application Security
Entwicklung und Implementierung sicherer Webanwendungen	Schaffung von Secure Coding Guidelines
Festlegung verbindlicher Regeln für die Entwicklung und Bereitstellung von Sicherheitskomponenten	Etablieren von Umsetzungskontrollen
Überprüfung von Sicherheitsmängeln im Code	Einsatz von Tools zur automatisierten Codeanalyse auf Sicherheitsmängel

Rechtzeitiges Erkennen von Fehlern und Finden von Lösungen	Durchführung von Sicherheitsanalysen und Penetrationstests
Prozessverbesserung	Definition von Metriken und Indikatoren, Auswertungen und Rückgabe in die Fachabteilungen

Quelle:[iX extra 7/2011, Security, sichere Anwendungen Seite11]

4 Umsetzungsmassnahmen mit Hilfe von Software Werkzeugen

4.1 Akuter Handlungsbedarf

Sicheres Programmieren und Durchführen von Penetrationstests steckt in der Schweiz noch in den Kinderschuhen. Es besteht Bedarf, die Applikationen sicherer zu machen und den ganzen Software-Entwicklungszyklus für sichere Applikationen einzubringen. Der Aufwand dafür ist jedoch kurzfristig hoch, denn es bedarf eines systematischen Trainings der Test- und Entwicklungseinheiten. Allenfalls ist ein Aufbau einer eigenen Security-Testing-Gruppe sinnvoll.

4.2 Umsetzungsmassnahmen

Um sichere Onlineapplikationen zu bekommen, braucht es gezielte Massnahmen, die den gesamten Prozess vom Design bis zur Einführung im Betrieb betreffen.

Design & Implementierung

- Einführung von Sicherheitsregeln, Guidelines, Regulation
- Erstellung von Sicherheitsanforderungen und Angriffs-Cases
- Durchführung von gezielten Sicherheits-Architektur-Reviews
- Erarbeitung und Einhaltung von Secure Coding Guidelines

Qualität & Testing

- Durchführung von:
  White-Box Penetrationstests
  Grey-Box Penetrationstests
  Black-Box Penetrationstests

Einführung & Betrieb

- Kontinuierliche Überwachung der laufenden Systeme
- Assessment und Feedbackloop zum ersten Schritt

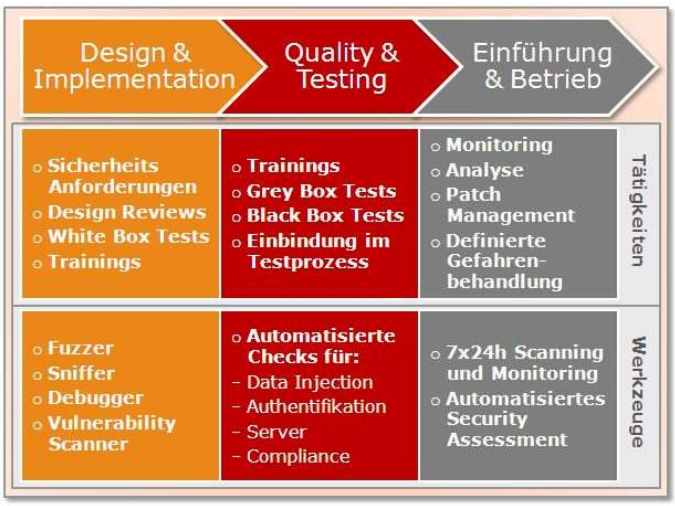


Abbildung 15:Toolbox für konkrete Umsetzungsmassnahmen

Konkrete Hilfestellung durch Software-Werkzeuge

Sind die Tätigkeiten erst einmal definiert, können sie beschleunigt werden. Grosse Software-Hersteller wie IBM oder andere haben sich jüngst im Bereich Sicherheitsprüfung von Webanwendungen durch gezielte Akquisitionen verstärkt. Für bestimmte Einsatzgebiete gibt es auch entsprechende Open-Source-Lösungen. IBM wie auch HP bieten Tools an, die den gesamten Lebenszyklus aus Sicht der Security von der Entstehung bis zur Ablösung einer Applikation unterstützen

Entwickler:

- Code wird während der Eingabe auf Sicherheit überprüft
- Auf Wunsch erscheinen Lösungsvorschläge und Links

Tester:

- Automatisierte Sicherheitstests untersuchen Webapplikationen und Services nach Fehlern
- Gefundene Fehler werden je nach Sicherheitsrisiko mit entsprechender Priorität gespeichert und dem zuständigen Entwickler zugewiesen

Security-Experten:

- Eine «7 x 24 h»-Lösung durchforstet komplette Webapplikationen nach Sicherheitslücken
- Automatische Überprüfung auf rechtliche, firmeninterne und regulatorische Bestimmungen

Die beschriebenen Tools werden laufend besser. Experten scannen täglich das Internet nach neuen Gefahren. Wie bei einer Antivirenlösung werden die Tools mit neuen Signaturen und Erkennungsmustern per Update aktuell gehalten.

Fazit

Schwachstellen in Web-Applikationen können sowohl in der Design-Phase durch das Einbinden der IT-Sicherheitsspezialisten, durch Code Reviews, Projektphasenabnahmen und Abnahmetests als auch im Betrieb durch Penetration Testing, geeignete Überwachungs- und Analysewerkzeuge behoben werden.

5 Hacking-Tools und Lernumgebung

5.1 Hacking-Tools

Kali-Linux

Kali Linux ist der offizielle Nachfolger von **Back Track**. Der Namenswechsel zu Kali Linux soll dem Hersteller zufolge anzeigen, dass es sich um eine bedeutsam fortgeschrittene Neuentwicklung handelt. Im Gegensatz zu seinem Vorgänger setzt Kali Linux nicht auf Ubuntu, sondern auf Debian. Das gesamte Betriebssystem wurde komplett neu erstellt und eine entsprechende Infrastruktur mit **Git** als Versionsverwaltung aufgebaut.

Um seinen Aufgaben gerecht zu werden, verfügt das System über mehr als 600 Sicherheits-Tools. Diese sind vor allem dafür gedacht, Sicherheitslücken und Konfigurationsfehlern aufzudecken.

Wie schon bei den bisherigen Versionen, können Sie Kali Linux als Live-System direkt von DVD oder dem USB-Stick nutzen oder, wenn Sie es bequemer mögen auch installieren. Wer das Live-System verwendet, muss als Root-Passwort "toor" eingeben.

Abbildung 16:Security-Tools Kali-Linux

Nach dem ersten Start zeigt sich Kali Linux als schicke Linux-Distribution. Schnellzugriff auf die Security-Tools gibt es über den Eintrag "Applications" in der linken oberen Bildschirmecke. Dann werden 13 Kategorien für Tools aufgelistet, ausserdem noch der Punkt "Usual Applications". Dort sind "normale" Werkzeuge untergebracht, etwa Editoren, Browser oder Taschenrechner. Klickt man auf eine Kategorie, zeigt Kali die zehn beliebtesten Tools in diesem Bereich an. Unterhalb im Menü wird der grosse Rest an Security-Werkzeugen alphabetisch sortiert aufgelistet.

OWASP Zed Attack Proxy (ZAP)

Der ZAP ist inzwischen viel mehr als ein einfacher Proxy. Er kann auch als Crawler oder Schwachstellenscanner arbeiten und enthält viele weitere Funktionen. Der Proxy muss auf dem Rechner laufen, auf dem Ihr Browser läuft.

Abbildung 17:OWASP Zed Attack Proxy

5.2 Hacking-Labor einrichten

Um die im vorherigen Kapitel vorgestellten Tools auch selbst ausprobieren zu können, brauchen wir ein geeignetes System, welches wir attackieren können. Weder eins in der Produktion noch eins, dass „einfach so im Internet ist“ oder den WLAN-Router des Nachbarn. Damit macht man sich schnell strafbar oder sorgt dafür, dass wichtige Systeme ausfallen. Um auf Nummer sicher zu gehen, sollte man sich ein eigenes Labor einrichten.

Virtuelle Systeme lassen sich auf einem PC einrichten, VMware, VirtualBox und Co können zudem sogar geschlossene Netzwerke aufbauen.

Und dann gibt es noch spezielle Betriebssysteme und Applikationen, die genau für den Pentest-Einsatz gedacht sind:

Damn Vulnerable Web App: **DVWA** ist eine Web-Applikation, in die absichtliche Fehler eingebaut wurden. Diese ist ideal, um Attacken wie SQL Injection oder Cross-Site-Scripting in Aktion zu sehen.

Metasploitable: Der Name erinnert nicht von ungefähr an Metasploit. Metasploitable3 setzt auf Windows, ist also ideal, um auf diesem System die Schwachstellen zu lernen.

Gruyere: Löchrig wie ein Käse, das ist die Pentest-Umgebung von Google Sie ist ideal für Black Box Hacking und kommt mit ein paar interessanten Aufgaben

5.3 Test- und Lernumgebung

Der Komplexitätsgrad von Webapplikationen nimmt stetig zu. Die grundlegende Vorgehensweise verändert sich jedoch auch bei sehr komplexen Anwendungen kaum. Folgende Systeme sind dazu ausgelegt, diese Vorgehensweise und die Erkennung der unterschiedlichen Schwachstellen in einer geschützten Testumgebung abzubilden und zu erlernen. Neben Laborsystemen gibt es mehrere Online-Testsysteme. Diese Testsysteme sind über das Internet erreichbar und dienen im Normalfall für erste Test- und Demo-Analysen unterschiedlicher Webapplikcation-Scanner renommierter Hersteller. Beispielsweise bieten HP, Acunetic und IBM solche Umgebungen.

Online-Testsysteme

- Acunetix-PHP-Testwebsite
- Watchfire-Testseite von IBM Appscan

Metasploitable

Metasploitable3 ist eine absichtlich unsichere Windows-Umgebung, mit der IT-Experten, Sicherheitsverantwortliche oder interessierte Hacker Angriffsszenarien durchspielen können. Die kostenlose virtuelle Maschine erlaubt das Simulieren von Angriffen, etwa zu Schulungszwecken, in technischen Interviews oder bei der Entwicklung von eigenen Exploits. Metasploitable3 besitzt einen großen Vorteil gegenüber selbstgebauten Lösungen: Damit entsteht ein Quasi-Standard, mit dem sich Trainings oder andere Vorgehensweisen und Tools in einer sicheren Umgebung nachvollziehbar ausprobieren lassen

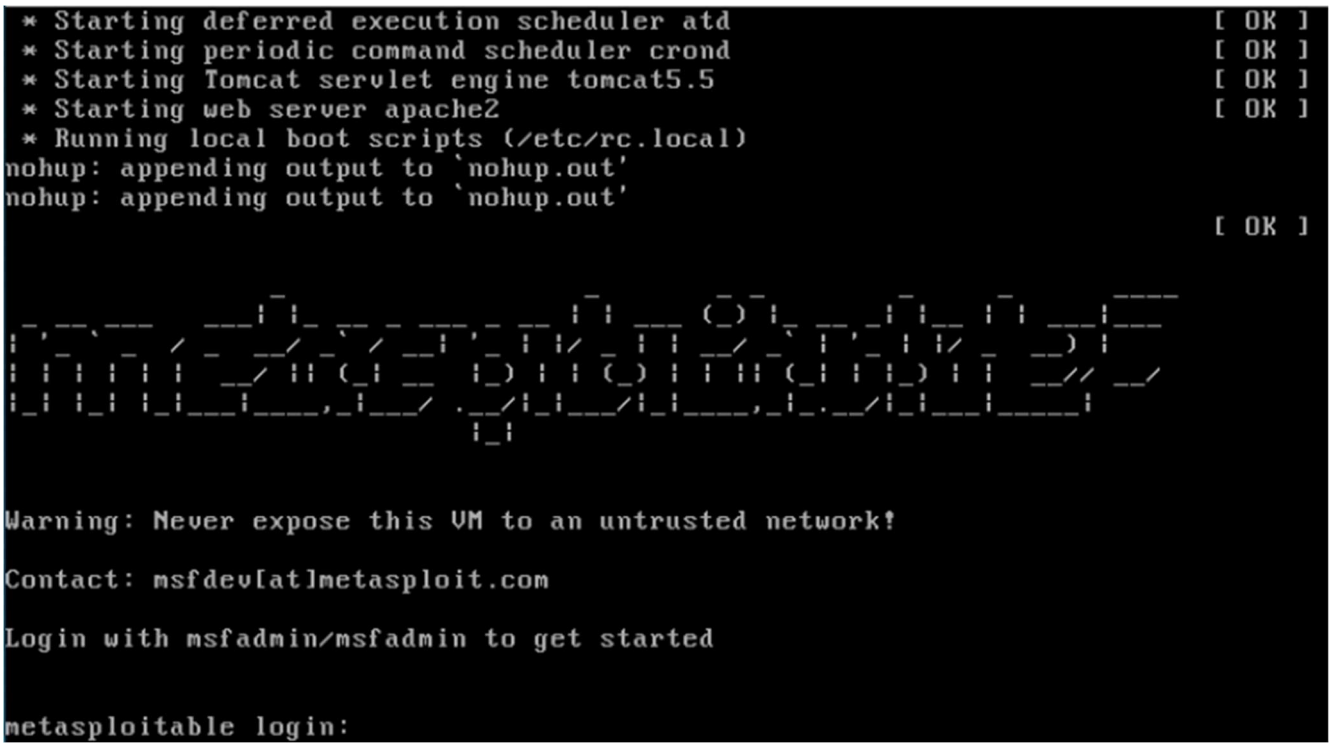


Abbildung 18:login von Metasploitable

OWASP Webgoat

eine absichtlich unsichere Webanwendung, hergestellt von OWASP als eine Anleitung für sichere Programmiermethoden.

Eine Webseite, die *step by step* an die Websecurity heranführt. Von XSS über *Session Fixation* bis zu *DOS* kann man mit WebGoat alles geführt in Form einer Anleitung nachvollziehen.

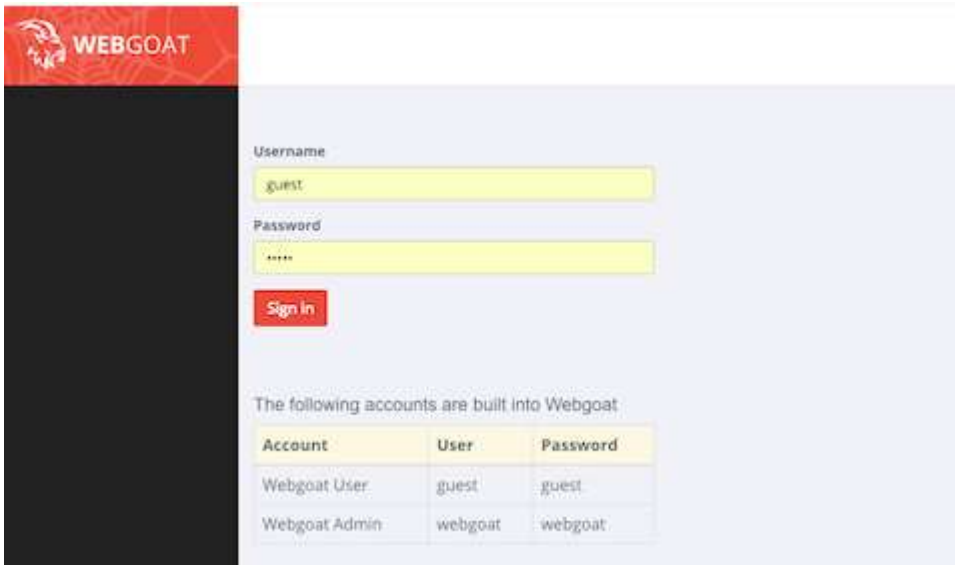


Abbildung 19:Login WebGoat

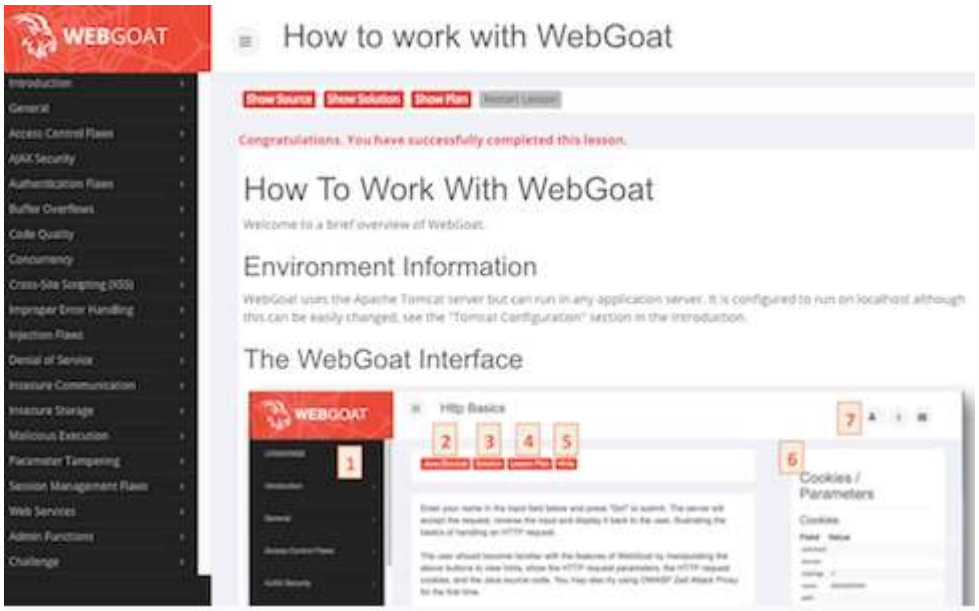


Abbildung 20:Unterrichtslektionen mit WebGoat

DVWA Damn Vulnerable Web App

Abbildung 21:DVWA Vulnerable We App

DVWA biete 3 Sicherheitsstufen

Abbildung 22:drei Sicherheitsstufen low; medium, high

Hacksplaining

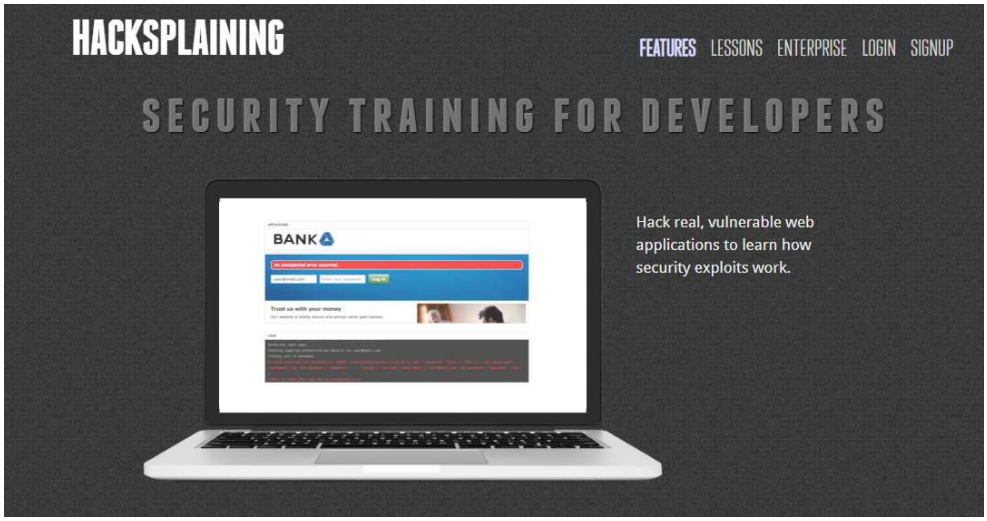


Abbildung 23:Security training for developers

Abbildung 24:Lektionen zu einzelnen Themen

https://www.hacksplaining.com/lessons

Weitere wichtige Hilfsmittel

Webseitencheck mit der **Burp-Suite**, als Java-basierter Proxy schaltet sich Burp zwischen Browser und Webserver und bietet Programmierern, Pentestern und sicherheitsaffinen Experten ein praktisches GUI mit vielen Funktionen.

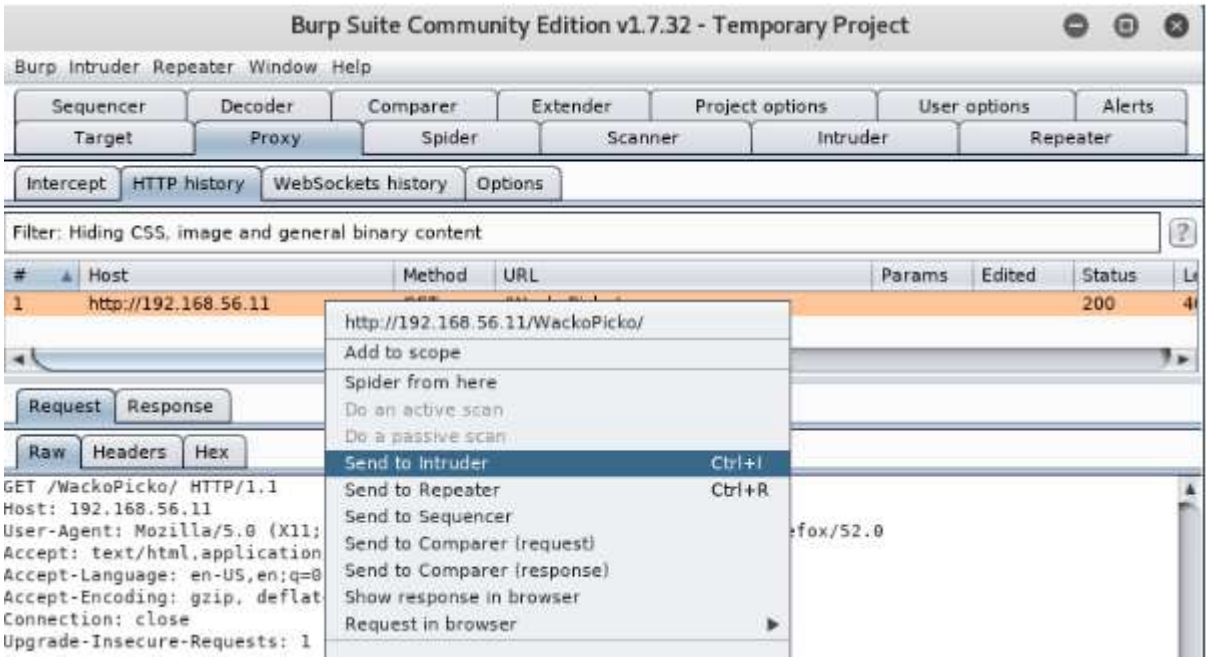


Abbildung 25:Burp Suite, Java-basierter Proxy

Add-ons zu Browser

Tamper Data

https://addons.mozilla.org/en-us/firefox/addon/tamper-data/

Firebug

https://addons.mozilla.org/en-US/firefox/addon/firebug/

Hackbar

https://addons.mozilla.org/en-US/firefox/addon/hackbar/

Cookies Manager

https://addons.mozilla.org/en-US/firefox/addon/cookies-manager-plus/

SQL Inject Me

https://addons.mozilla.org/en-us/firefox/addon/sql-inject-me/

XSS ME

https://addons.mozilla.org/en-us/firefox/addon/xss-me/

Postman for API

Postman wurde ursprünglich als eine Chrom Applikation entwickelt, steht aber inzwischen als eine eigenständige Applikation zur Verfügung. Postman wird primär zum Testen von *RESTful APIs* verwendet, kann aber allgemein für die Analyse der HTTP – Kommunikation verwendet werden. Downloads stehen für Windows und Mac zur Verfügung.

https://www.getpostman.com/apps

6 Aufgabenstellungen

6.1.1 Aktuelle Security-Vorfälle recherchieren und bewerten

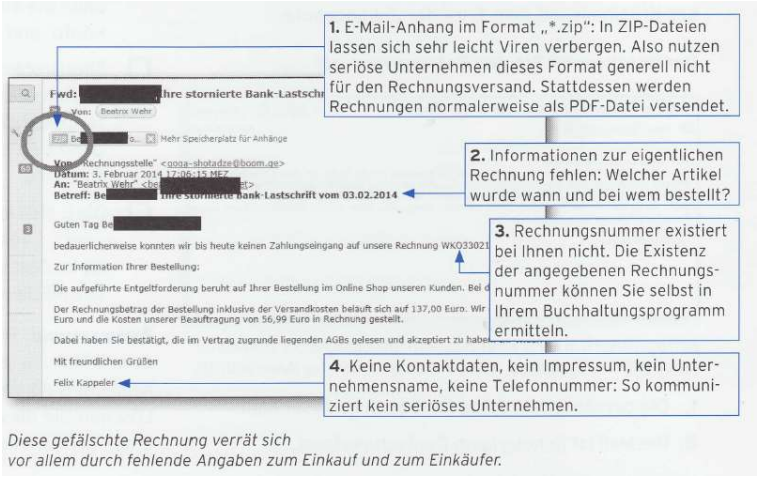
Weitere Angaben gemäss Anweisung LP.

6.1.2 Gefahrenpotential durch verdächtige E-Mails

Online-Kriminelle suchen per E-Mail das Vertrauen Ihrer potentiellen Opfer zu gewinnen und sie zu unvorsichtigem Handeln zu verführen.

Beschreiben Sie Ihren Mitarbeitern das Gefahrenpotential durch die folgenden drei Betrugsfallen und wie sie sich dagegen schützen können

Nr. 1: Die unbezahlte Rechnung

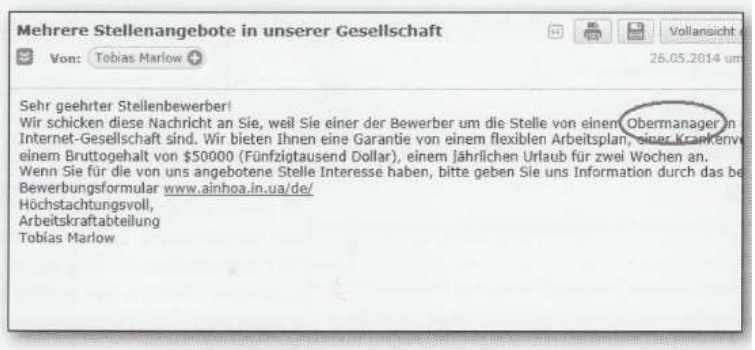


Diese gefälschte Rechnung verrät sich vor allem durch fehlende Angaben zum Einkauf und zum Einkäufer.

Nr. 2: Angebliche Probleme beim Onlinebanking

Gefälschte Postbank-Mail: Hier sollen Sie auf einen Link klicken, um sicherheitsrelevante Daten zu ändern.

Nr. 3: Das lukrative Jobangebot



Betrugsversuch in Form eines dubiosen Stellenangebots: Die Mail ist schon deshalb verdächtig, weil sie unaufgefordert bei Ihnen eintrifft.

6.1.3 Auftrag: Installation der Software Umgebung

Kali Linux; USB-Stick; oder VMware – Version

XAMPP (Apache, PHP, MySQL)

Webgoat

6.1.4 Auftrag: Datenschutz im Zeitalter der Informatik

Machen Sie konkrete Vorschläge, mit welchen technischen Mitteln Sie die vorher beschriebenen Gütekriterien des Datenschutzes umsetzen und garantieren können.

6.1.5 Auftrag: Massnahmen für Host- und Netzwerksicherheit

Erstellen Sie eine Liste mit den zehn wichtigsten Massnahmen zur Gewährung der Host- und Netzwerksicherheit.

6.1.6 Auftrag: OWASP Top Ten

Studieren Sie die Beschreibung der Top Ten von OWASP. Verschaffen Sie sich über die jeweils angegebenen Links einen Überblick. Weitere Angaben gemäss Anweisung LP.

6.1.7 Auftrag: Security Policy für KMU Webshop Velohandel erstellen

- Erstellen Sie für die Mitarbeiter einer KMU eine Security-Policy die die folgenden Themen umfasst:
- Verwendung vertraulicher Daten
- Umgang mit vertraulichen Daten
- Mobile Endgeräte
- -IT-Arbeitsplatz
- Schutzvorkehrung Router
- Goldene Regeln für Passwörter
- Virenbefall
- Private E-Mails Facebook

- Hinweis: IT-Sicherheit_für_Selbstständige_Kleinunternehmer.PD

6.1.8 Auftrag DAVWM installieren

Auftrag: SQL-Injection und XSS studieren

https://www.youtube.com/watch?v=ljws7f0Nijs

6.1.9 Auftrag: Webgoat installieren

Webgoat installieren: Weitere Angaben gemäss Anweisung LP.

6.1.10 Auftrag: Installation Testumgebung mit Kali-Linux und Testapplikation

Sie sollten in der Lage sein einzelne Tools von Kalli Linux zu benutzen. Diese Tools sollten gezielt auf eine bestehende Applikation zur Suche nach Schwachstellen angewendet werden können.

Evaluation: Web-Application-Scanner

Evaluation: einzelne Metasploits in virtuellen Systemen austesten

Bestehende Web-Applikationen gezielt nach Schwachstellen untersuchen

6.2 Projektarbeit

Angaben gemäss Anweisung LP

7 Literatur und Quellen

[1] Frank Neugebauer; Metasploit auf Schwachstellensuche; iX 8/2011; S. 115

[2] Jörg Riether; Backtrack 5; iX 7/2011; S. 60

[3] div. Autoren; Hacking Schoool, Handbuch, CSH Verlag, 2. Auflage

[4] Claudia Eckert; IT-Sicherheit; Konzepte-Verfahren-Protokolle; 6.Auflage

[5] Thomas Schreiber; iX Security extra; Application-Scanner, Web-Firewall & Co. – Sicherheit im Web 2.0

[6] John Viega; 19 Deadly Sins of Software Security - Programming Flaws and How to Fix Them"

[7] Erich Kachel; Analyse und Massnahmen gegen Sicherheitsschwachstellen; Teil 1

[8] OWASP; Best_Practices_Guide_WAF; Einsatz von Web Application Firewalls

[9] Siemens AG; Penetration Testing; GI-Proceedings.44.innen-49.pdf

[10] OWASP;Kickstart_fuer_sichere_Webanwendungen.pdf

Quellen

http://www.innosec.eu/de/unternehmen.html

8 Abbildungsverzeichnis