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

Modul 183

Applikationssicherheit implementieren

D. A. Waldvogel
Version 2.1
8. August 2015

TECHNISCHE BERUFSSCHULE ZÜRICH

Inhaltsverzeichnis

1 Haftungsausschluss

Alle auf den folgenden Seiten gegebenen Informationen sind ausschliesslich dazu da, sich bewusst zu werden welchen Gefahren und Risiken sich heute ein normaler Internetbenutzer aussetzt, wenn er insbesondere heutige Webapplikationen benutzt und sich Ihnen sorglos anvertraut.

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 zu behandeln.

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.

2 Modulidentifikation

Modulnummer	M-183
Titel	Applikationssicherheit implementieren
Kompetenz	SW Applikationen sicher planen, entwickeln und in Betrieb nehmen
Handlungsziele	1. Aktuelle Bedrohungen erkennen und erläutern können. Aktuelle Informationen zum Thema (Erkennung und Gegenmassnahmen) beschaffen können. Mögliche Auswirkungen aufzeigen und erklären können. 2. Sicherheitslücken in einer Software Applikation erkennen und abschätzen können woher die Lücke kommt. Gegenmassnahmen vorschlagen, zuweisen, implementieren können. 3. Mechanismen kennen und für die Authentifizierung und Autorisierung umsetzen können. 4. Sichere Software Applikationen designen, implementieren, verifizieren und in die Produktion bringen. 5. Informationen für Auditing und Logging generieren. Auswertungen definieren und implementieren. Alarme definieren und in SW Applikationen einbinden können.
Kompetenzfeld	Applikations- Entwicklung
Objekt	Applikation
Niveau	4
Voraussetzungen	Modul 226 und Modul 326
Anzahl Lektionen	40
Anerkennung	Eidg. Fachausweis Informatiker/Informatikerin
Modulversion	1.0
MBK Release	R4
Harmonisiert am	Oktober 2010

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

2.1 Überblick

Es vergeht heutzutage kaum eine Woche, in welcher die Zeitungen nicht über neue Unzulänglichkeiten in der Sicherheit von Softwaresystemen im Allgemeinen und von Internet Applikationen im Speziellen informieren. Seien dies Meldungen über einen neuen Virus, 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 Mechanismen an.

2.2 Informations-Sicherheit

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 Nichtabstreitbarkeit bei Informationssystemen sichergestellt sind.

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
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
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
bedeutet, dass die Transaktionssicherheit soweit gewährleistet sein muss, dass mit Sicherheit die Identität des Sender 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.

2.3 Applikations-Sicherheit

Der Begriff „Applikationssicherheit“ umfasst alle Tätigkeiten, Prozesse und 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 basiert auf darunter liegenden Sicherheits-

Elementen, welche die beteiligten Systeme, Netzwerke, Zugangspunkte von Angriffen und Missbrauch schützen. Während die technische IT-Sicherheit (Schutz vor Hackern, Malware, Firewall, Intrusion Detection) in vielen grösseren Unternehmen bereits umgesetzt oder zumindest eingeführt und schrittweise verbessert werden, ist die Applikationssicherheit derzeit noch eine schlecht mit der IT-Sicherheit koordinierte Aufgabe gemäss Hannes P. Lubich[1]

OWASP 5-Ebenen-Modell

	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 1: 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*
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*

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

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-Security*

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*

3 Bedrohungen und Angriffe

Bedrohungen durch Pufferüberlauf-Angriffe, Viren, Würmer und Trojanische Pferde 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 Würmer ausgenutzt werden.

3.1 Bedrohungen durch Trojaner

Gemäss Gunnar Porada[2] werden heute pro Tag bis zu 55‘000 neue Viren produziert. Der Anstieg ist vorwiegend bei den Trojanern festzustellen. Es geht darum durch Ausspionieren von mobilen Geräten wie Iphone usw. an die Daten heranzukommen. Heute wird mit Datendiebstahl mehr Umsatz erzielt als mit Drogen.

Absurder Weise kann heute praktisch jeder sich einen Trojaner „zusammenstellen“ lassen. Einschlägige Webseiten bieten Service- und Dienstleistungen bereits ab CHF 1500 für einen funktionierenden Trojaner an.

Trojaner: Ein Trojanisches Pferd ist ein Programm, dessen implementierte Ist-Funktionalität nicht mit der angegebenen Sollfunktionalität übereinstimmt. Heutige Trojanische Pferde 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 selber, wenn Sie „irgendwelche“ Software installieren. Es kann aber auch beim Surfen über das Internet, oder durch E-Mails passieren.

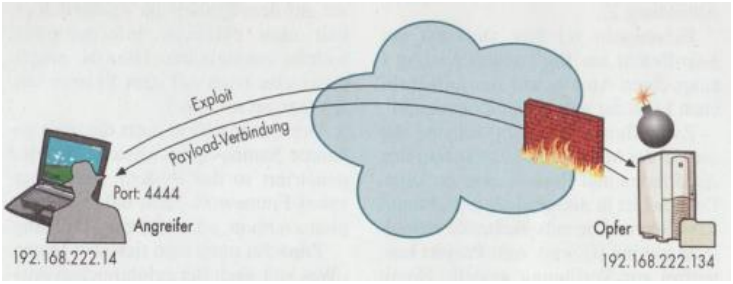


Abbildung 2: Angreifer nutzt die Schwachstelle des Opfers

Fazit: Kein Virenscanner ist in der Lage 100% aller Viren zu erkennen!

4 Web-Applikationen

In heutigen IT-Systemen sind Web-basierte Anwendungen, kurz Web-Applikationen (oder Onlineapplikationen) sehr weit verbreitet. Sie ermöglichen eine einfache, Web-basierte Zusammenarbeit zwischen einem Dienstanbieter und den Nutzern. 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

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

sind, die im Intranet des Dienstanbieters verwaltet werden. Die Nutzung eines Dienstes ist für Clients sehr einfach mittels des HTTP-Protokolls möglich.

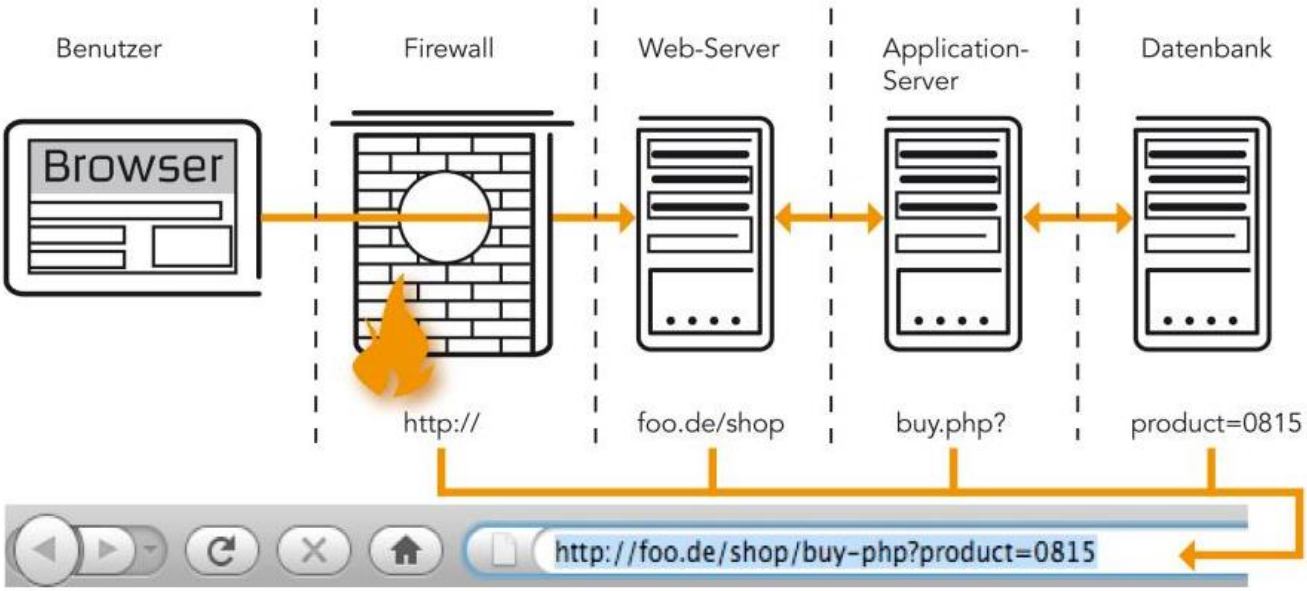


Abbildung 3: Beim Aufruf einer Web-Seite im Browser werden die einzelnen URL-Segmente an die entsprechenden Server der Web-Anwendungen weitergereicht.

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.

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. 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)

Je nach Anwendung entstehen die Daten in einer Datenbank auf verschiedene Arten. Protokolldatenbanken, die Zugriffe und Modifikationen festhalten werden automatisch durch die Applikation gefüllt. 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. Diese Form wird aber durch Fortschritte in der Scannertechnologie immer geringer. Auch die stärkere Verbreitung des e-Banking hat dazu geführt, dass die Daten beim Eingang in die verarbeitende Firma bereits elektronisch vorliegen.

Firewall

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.

Dynamische Web-Seiten

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-Script 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-Scripte 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.

4.1 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 weiter.

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 fürs Business 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. Eine der aktuell beliebtesten Attacken ist das Cross Site Scripting (XSS). Ein Beispiel: Der Angreifer versucht eine Webanwendung so zu manipulieren, dass schädlicher Scriptcode 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. Obwohl die

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

Unternehmen ihre IT-Security durchaus ernst nehmen, gehören Meldungen über erfolgreiche Angriffe zur Tagesordnung.

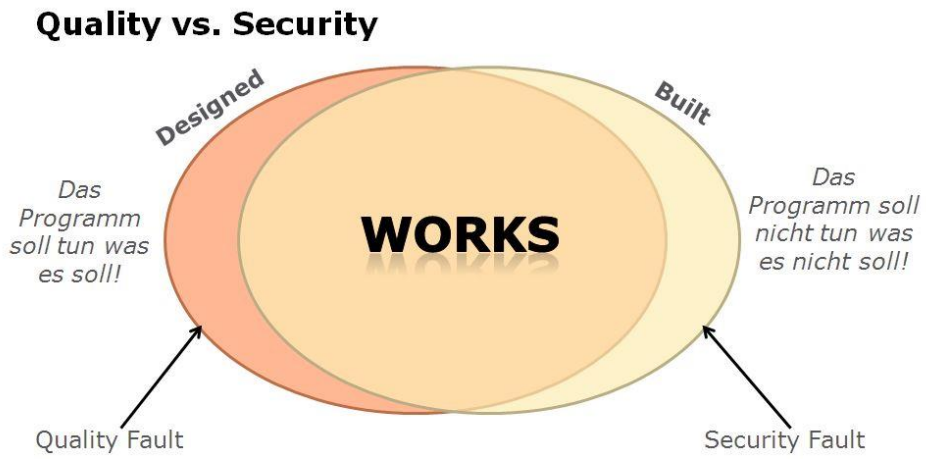


Abbildung 4:Quality vs. Security

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.

4.2 OWASP Top Ten Project

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

Um Qualität und Sicherheit von Anwendungen zu verbessern, wurde das Community-Projekt "Open Web Application Security Project" (OWASP) ins Leben gerufen. Innerhalb des OWASP gibt es verschiedene Teilprojekte, so etwa das "OWASP Top Ten Project", das regelmässig die jeweils zehn kritischsten Schwachstellen von Web-Applikationen beschreibt (z.B. Stand: 2007). Ziel ist es, Entwickler, Designer, Architekten und Unternehmen für potenzielle Schwachstellen zu sensibilisieren und aufzuzeigen, wie sich diese vermeiden lassen. Die folgenden "OWASP Top Ten" stellen unter Web-Sicherheitsexperten einen anerkannten Konsens dar, was die derzeit kritischsten Lücken in Web-Anwendungen betrifft:

- Cross Site Scripting (XSS)
- Injection-Fehler (etwa SQL-Injection)
- Die Ausführung bösartiger Dateien
- Unsichere Direct-Object-Referenz
- Cross Site Request Forgery (CSRF)
- Informationslecks und 'indiskrete' Fehlerbehandlung durch die Web-Anwendung
- Fehler bei Authentisierung und Session-Management
- Unzureichend verschlüsselte Datenablage
- Unsichere Kommunikation
- Uneingeschränkter Zugriff auf URLs

Detaillierte Beschreibung: http://www.computerwoche.de/security/582260/index.html

Weitere Quellen: [7]

4.3 19 Deadly Sins of Software Security

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

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

Die 19 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

5 Sicherheitslücken erkennen

5.1 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.

Unter einer Webapplikation versteht man im Allgemeinen Software, die auf einem Webserver ausgeführt wird und auf über *http-übertragenen* Benutzereingaben dynamisch reagiert. S.a. Kap. 4

[4] Chief Security Architekt beim amerikanischen Sicherheitsanbieter McAfee

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 „tool“ 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]

5.2 Sicherheitsanalyse / Penetrationstest

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:	Aaktive 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 Pentester bei einer raschen und möglichst korrekten Auswahl der Zielsystem 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 gefundenn Schwachstelle auch detaillierte Informationen zu den erkannten Schwachstellen und zu möglichen Risiken und Gefährdungen umfassen.

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

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

Quelle:[iX 7/2011, Backtrack 5 S. 60]

Weitere Quellen: [9]

6 Gegenmassnahmen

Als wichtige Massnahme sollen die Applikationen mit geeigneten Mitteln auf Herz und Nieren auf 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.

6.1 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

6.2 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 ApplicationFirewall 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.

Achtung: Eine Kette ist immer so stark wie ihr schwächstes Glied

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.

6.3 Massnahmen gegen die häufigsten Mängel

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 Guuidelines
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]

7 Lösungsansätze für Umsetzungsmassnahmen

7.1 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.

7.2 Umsetzungsmassnahmen

- Um zu sicheren Onlineapplikationen zu kommen, braucht es gezielte Massnahmen:
- Einführung von Sicherheitsregeln, Guidelines, Regulations
- Erstellung von Sicherheitsanforderungen und Angriffs-Cases
- Durchführung von gezielten Sicherheits-Architektur-Reviews
- Erarbeitung und Einhaltung von Secure Coding Guidelines
- Durchführung von:
  White-Box Penetrationstests
  Grey-Box Penentrationstests
  Black-Box Penetrationstests
- Kontinuierliche Überwachung der laufenden Systeme
- Assessment und Feedbackloop zum ersten Schritt

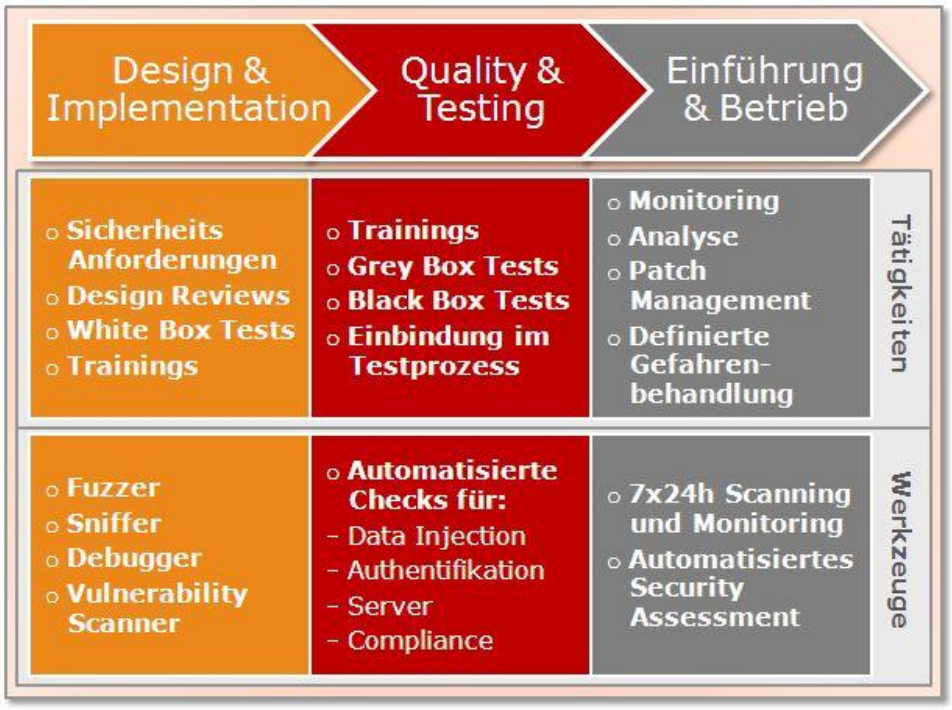


Abbildung 5:Toolbox für konkrete Umsetzungsmassnahmen

7.3 Unterstützung durch Software-tools

Sind die Tätigkeiten erst einmal definiert, können sie beschleunigt werden. Grosse Software-Hersteller wie oder IBM 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 von der Entstehung bis zur Ablösung einer Applikation den gesamten Lebenszyklus aus Sicht der Security 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.

8 Aufgabenstellungen

Aktuelle Security-Vorfälle recherchieren und bewerten

Studium der einzelnen Fachbegriffe, erstellen Sie eine Zusammenstellung mit den wichtigsten Fachbegriffen

Massnahmen und Best Practices zur Vorbeugung gegen typische Schwachstellen in Webanwendungen Sicherheits- und Risikoanalyse

Installation: Testumgebung mit Kali-Linux; Testapplikation

Evalutation: Web-Application-Scanner

Evaluation: einzelne Metaploits in virtulellen Systemen austesten

Bestehende Web-Applikationen gezielt nach Schwachstellen untersuchen

Design anpassen: Aufteilung der SW-Architektur in Presentation- Business- Data-Layer s.a. Skript Von Orelli

9 Literatur

[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

10 Anhang

10.1 Beispiele von aktuellen Bedrohungen

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

Enterprise CIO Forum: *CIOs in der Social-Media-Bredouille*

- WIK: *Verschlüsselt über die Grenze (Gemeinschaftsartikel mit RaE Dr. Hackenberg)*
- Computerwoche: *Die Gefahren des neuen Personalausweises*
- Moderner Staat: *Interview mit Gunnar Porada*
- Computerwoche: *Personalausweis - Experten warnen*
- Wik: *Wer schlau ist, leistet sich gut bezahlte Hacker...*
- Computerwoche (Interview): *Schadsoftware überall* (PDF-Version)
- Computerwoche: *WEB-Mafia Effizient, arbeitsteilig, geräuschlos*
- Computerwoche: *So wirbt die Web-Mafia Hacker an*
- CIO: *So wirbt die Web-Mafia Hacker an*
- Badische-Zeitung: *So wirbt die Web-Mafia Hacker an*
- Live-Hacking *it-sa 2009* (Video-Stream)
- MODCOMP: *Wer haftet für Schäden durch mangelnde IT-Sicherheit?*
- Computerwoche (Interview): *Schutz im Internet kann man vergessen*
- ZDF/Drehscheibe: *Kurzversion des WISO-Beitrags*
- Heise Online: *Unsichere Verarbeitung der Fingerabdrücke in Meldebehörden*
- Heise Telepolis: *Hände weg von den Fingerabdrücken!*
- TAZ: *Fingerabdrücke bei Behörden unsicher*
- eGoverment Computing: *Scanner ist Schwachstelle im Meldeamt*
- Datenschutz: *Sicherheitslücke bei Erfassung von Fingerabdrücken für Reisepässe*
- Tagespunkt: *Sicherheitslücke bei Erfassung von Fingerabdrücken für Reisepässe*
- Wikipedia: *Angriffe auf die Sicherheit von ePässen*
- ZDF/Frontal 21: *Angriff aus dem Netz*
- Spiegel Online: *Massenangriff der Virenmafia*
- Abendzeitung: *Angriff der Hacker*
- Der Westen (WAZ): *Ex-Hacker warnt Bürger und Unternehmen*
- Infosek: *Interview*
- Infosek: *Keynote Speaker: Gunnar Porada*
- Die 10 *mysteriösesten Cyber-Crimes*
- Die 10 *erfolgreichsten Technik-Hacks*
- Die 10 *der Experte*
- Khaleej Times (Dubai): *'Gunnar' Who Shoots Cyber Troubles*
- Khaleej Times (Dubai): *Surveillance Cameras Can Be Hacked: Experts*
- Zagros: شدن هک معرض در ذ ظارتـ ي و امـ ذ يـ تـي هايدوربـ ين
- WELT online: *Fachvorträge und Live-Action*

10.2 Beipiel: SQL-Injections

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

SQL-Injection-Angriffe 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, so können enthaltene SQL-Befehle die Datenbank in nicht vorhergesehener Weise manipulieren.

Dynamische PHP-Anwendungen beziehen ihre Daten meist aus zwei Quellen:

- Dateisystem und
- Datenbanken.

Die Verwendung von Daten aus Dateien ist für SQL-Angriffe nicht anfällig und wird hier nicht weiter behandelt. Setzt die Anwendung allerdings auf eine Datenbank zur Datenverwaltung, so kann je nach Anwendungsfall beispielsweise folgendes 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.

10.2.1 Angriffsvektoren

Auch bei dieser Angriffsart dringt der Schadcode durch die Nutzereingabe in die Anwendung ein. Dabei versucht der Angreifer eigenen SQL-Code in vorhandene Datenbankabfragen einzufügen und so neue SQL-Befehle zu erzeugen. Es werden dabei 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. Dabei entstehen die Schwachstellen hauptsächlich aus Fehlern in der Programmierung:

- Werteübergabe ohne umschliessende Hochkommas,
- eine fehlende Typenprüfung der Werte,
- eine fehlende Längenprüfung der Werte und
- 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 Nutzernamen erstellt, könnte wie folgt aussehen: ein Angreifer könnte den Nutzernamen manipulieren, sodass dieser zusätzlich SQL-Code führt. Verwendet die Anwendung den Nutzernamen 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. Darüber hinaus zielt der Angriff auf keinen bestimmten Nutzer sondern kann potentiell von jedem zur Ausführung gebracht werden.

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

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

10.2.2 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.

10.2.3 Logische Verknüpfungen ändern

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

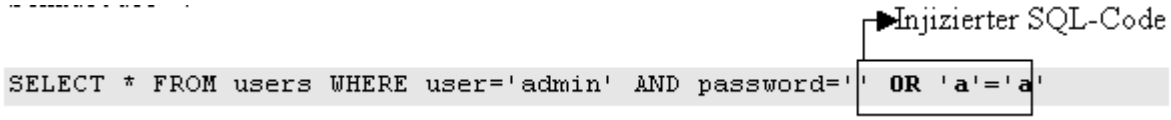
Abbildung 6: Ein für SQL-Angriffe anfälliges Skript

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.

11 Abbildungsverzeichnis

Dominik Waldvogel, 08. Aug. 2015
Skript Modul 183_V2.1.docx