

Grundlagen der Sicherheit bei Webanwendungen

Christian Gekeler
Hochschule für Technik Stuttgart

November 2006

Inhaltsverzeichnis

1. Einleitung
2. Sicherheit bei Webanwendungen
 - 2.1 Über Webanwendungen
 - 2.2 Beispiel für eine Webanwendungen
 - 2.3 allgemeine Sicherheitsrisiken
3. Sicherheit auf 6 Ebenen
 - 3.1 Ebenenmodell
 - 3.2 Bereiche der Sicherheit
4. Gefahren und Gegenmaßnahmen
 - 4.1 Identitätsdiebstahl und Enumeration
 - 4.2 Maßnahmen gegen Enumeration
 - 4.3 DoS-Angriffe
 - 4.4 Phishing
 - 4.5 SQL Injektion und Data Validation
 - 4.6 Verwalten der Session
5. Fazit
6. Literatur

1 Einleitung

Jeder hat heutzutage beinahe täglich mit ihnen zu tun und genießt den Komfort den sie einem bieten. Die Rede ist von Webanwendungen. Mittlerweile sind diese Anwendungen nahezu unverzichtbar. Sie bieten mittlerweile sehr große Möglichkeiten, Prozesse zum simplifizieren. Da das Web beinahe jedem zugänglich ist, können nicht nur die potentiellen Benutzer auf den angebotenen Service zugreifen. Insbesondere ist die Anwendung auch für diejenigen frei zugänglich, deren Ziel es ist, solche Systeme auszuspionieren oder zu hintergehen.

Aus diesem Grund ist einer der wichtigsten Aspekte solcher Anwendungen die Sicherheit und diese in ausreichendem Maße zu gewährleisten. Dieser Punkt wird häufig beim Optimieren von Prozessen vernachlässigt. Webanwendungen werden oft in Betrieb genommen, ohne zu wissen ob die Sicherheit der Anwendung ausreichend Beachtung gefunden hat.

Viel mehr als in der Vergangenheit, werden Geschäftsprozesse im Internet abgebildet und durchgeführt. Dabei sind Prozesse zwischen Geschäftspartnern als auch Prozesse zwischen Unternehmen und deren Kunden (eBusiness).

Dabei reicht der Bereich dieser Anwendungen von sehr einfachen bis hin zu sehr komplexen Systemen. Unabhängig davon, sollte den Gesichtspunkten der Sicherheit immer ausreichend Beachtung geschenkt werden. Oft wird aufgrund von Zeit- oder Geldmangel diesem Aspekt nicht genügend Aufmerksamkeit geschenkt, so dass die Sicherheit nur im geringen Maße gewährleistet werden kann.

Zur besseren Klärung der Zuständigkeiten bei der Sicherheit von Webanwendungen, existiert ein 5 Ebenen Modell, welche diese klar definiert. Mittlerweile wird dieses Modell oft in sechs Ebenen aufgeteilt:

- Netzwerk und Host(*)
- System
- Technologie
- Implementierung
- Logik,
- Semantik
- Vorschriften und Bestimmungen(*)

(*) nicht direkt der Webanwendung zugeordnet

Im weiteren Verlauf werden diese Ebenen noch genauer unter die Lupe genommen.

Ein weiter Punkt sind die Gegenmaßnahmen die zu ergreifen sind und Sicherheitslücken zu vermeiden. Dazu existieren einige allgemeine Methoden um dies zu garantieren, von denen einige der wichtigsten später noch genauer besprochen werden. Im folgenden Kapitel nun aber zuerst zu den Grundlagen des Ablaufs und dem Aufbau einer Webanwendung.

2 Sicherheit bei Webanwendungen

2.1 Über Webanwendungen

Die Bezeichnung Webanwendung ist bisher nicht strikt definiert. In den meisten Fällen besteht diese aus eine Architektur in 3 Schichten:

1. Präsentation im Webbrowser des Benutzers (Darstellung)
2. dynamische Anwendungen auf dem Webserver (Applikation)
3. Datenbasis (Datalayer)

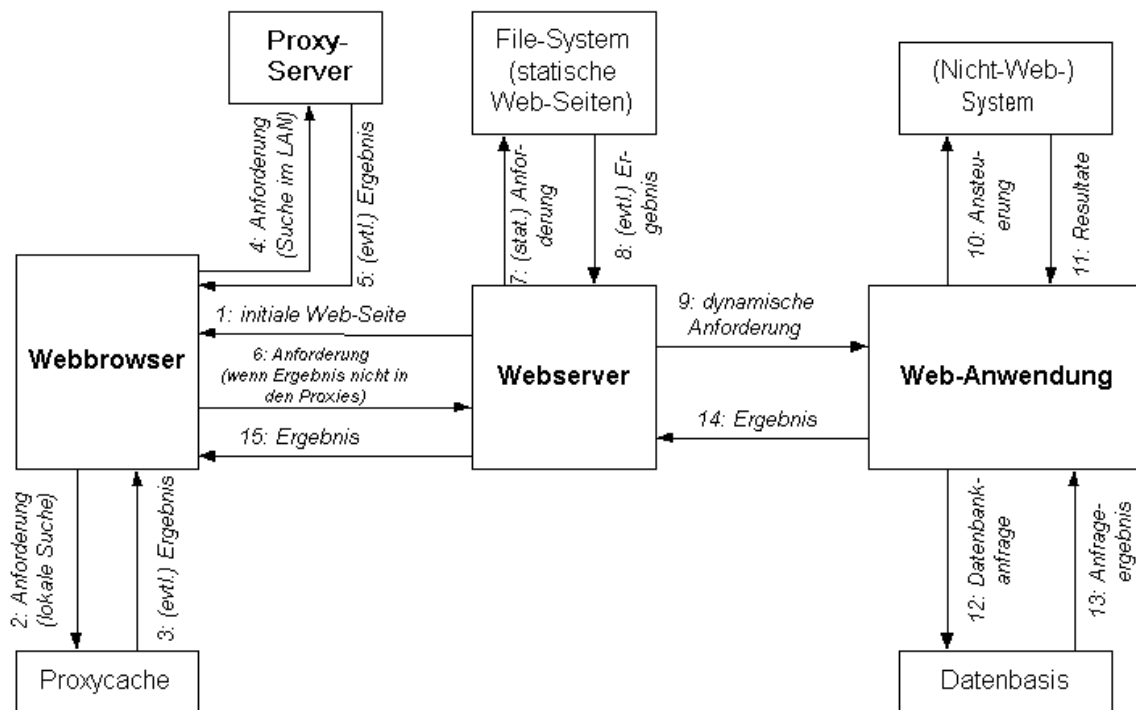


Abbildung 1: Architektur und Verlauf einer Web-Anwendung

Abbildung 1 oben zeigt die Architektur und den Ablauf einer Webanwendung. Es existieren allgemein zwei Arten von Anwendungen, Statische Webanwendungen und dynamische Webanwendungen. Dynamisch bedeutet, dass aus einer Anfrage des Benutzers auf dem Webserver eine Anforderung an die Webanwendung gestellt wird, die das Ergebnis generiert. Dies erfolgt in den meisten Fällen durch die Verarbeitung der vom Benutzer eingegebenen Daten. Diese Daten sind die Grundlage für das Ergebnis, welches dem Benutzer in seinem Browser angezeigt wird. In diesem Ablauf gibt es einige sehr wichtige Sicherheitsaspekte, die im Verlaufe der Entstehung einer Webanwendung berücksichtigt werden müssen.

2.2 Beispiel für eine Webanwendung:

Eine Werbeagentur verwaltet firmenspezifische Daten mit einer Softwareanwendung. Diese Software ist für alle Mitarbeiter zugänglich. Da die Anwendung aber auch für Kunden und externe Mitarbeiter der Agentur zugänglich sein muss, wird diese als Webanwendung zur Verfügung gestellt. Den Benutzern dieses Systems ist es nun jederzeit möglich, die aktuellen Daten über einen Webbrowser abzurufen oder zu bearbeiten. Interne sowie externe Geschäftsabläufe werden in dieser Software abgebildet. Dies ist nur ein einfaches Beispiel für eine Webanwendung. In vielen Fällen werden in einer Webanwendung ganze Unternehmensprozesse oder darüber hinaus Unternehmensübergreifende Prozesse abgebildet. Bei diesen Prozessen handelt es sich oft um vertrauliche oder sicherheitskritische Abläufe, wie z.B. bei Banken oder Versicherungen.

2.3 allgemeine Sicherheitsrisiken:

Eine solche Anwendung ist aber auch für andere Benutzer des Internets zugänglich. Diese besitzen keine Benutzerkennung und können sich nicht am System anmelden, jedoch gibt es verschiedene Gefahren, denen das System und Ihre Benutzer ausgesetzt sind. Durch nicht berechtigtes Eindringen in das System oder durch Verfälschung der Daten beim transferieren der Daten, kann den Benutzern oder dem Unternehmen selbst großer Schaden entstehen. Die Zielsetzung und Motivation von Benutzern mit böswilligen Absichten kann stark variieren. Ziel könnte z.B. sein, ein Absturz des Systems zu provozieren, damit die Anwendung für einen unbestimmten Zeitraum nicht benutzt werden kann. Ein solcher Angriff würde auf der technischen Ebene erfolgen. Eine andere Intention könnte es beispielsweise sein, über verschiedene Wege in das System einzudringen um Daten auszuspionieren oder diese im schlimmsten Fall sogar zu verfälschen. Die Angreifer können sich im Unternehmen selbst befinden oder aber auch von konkurrierenden Firmen. Aufgrund dieser Vielfältigkeit von möglichen böswilligen Angriffen auf das System, sollten Sicherheitsfragen auf verschiedene Ebenen verteilt werden. So existieren auf technischer Ebene andere Sicherheitsrisiken als auf inhaltlicher oder logischer Ebene.

3 Sicherheit auf 6 Ebenen

3.1 Ebenenmodell

Wie schon erwähnt, gibt es zur besseren Zuordnung der Zuständigkeiten im Punkt der Sicherheit einer Webanwendung ein 5(6) Ebenenmodell, welches die jeweiligen Bereiche einer Organisation den Aufgaben in diesem Sektor zuordnet.

	<i>Ebene</i>	<i>Inhalt</i>
6	Vorschriften und Bestimmungen	Einhaltung gesetzlicher Regelungen und unternehmensspezifischer Vorgaben
5	Semantik	Schutz vor Täuschung und Betrug
4	Logik	Absicherung von Prozessen und Workflows als Ganzes
3	Implementierung	Vermeiden von Programmierfehlern, die zu Schwachstellen führen
2	Technologie	Richtige Wahl und sicherer Einsatz von Technologie
1	System	Absicherung der auf der Systemplattform eingesetzten Software
0 (*)	<i>Netzwerk & Host</i>	<i>Absicherung von Host und Netzwerk</i>

Tabelle 1: Ebenenmodell

(*) nicht direkt der Webanwendung zugeordnet

Die Funktionen sind zusätzlich einer verantwortlichen Einheit(z.B. einerAbteilung) zugeordnet. Hieraus resultieren auch die erforderlichen Fachkenntnisse für die jeweiligen Ebenen.

Dazu gibt es drei Phasen der Entwicklung, denen die einzelnen Ebenen zugehörig sind:

Plan: Planung

Build: Entwicklung

Run: laufender Betrieb

Ebene 0 – Netzwerk und Host

Phase: Run

Fachkenntnisse: Netzwerk und Systemadministration

Einheit: Betrieb

Nicht direkt der Anwendung zugeordnet, eher ein allgemeiner Sicherheitsaspekt für öffentliche Server im Web. Auf dieser Ebene Sicherheit zu gewährleisten ist zwingend notwendig, da dies die Grundlage für alle weiteren Massnahmen der darüberliegenden Schichten ist.

Ebene 1 – Systemebene

Phase: Run

Fachkenntnisse: Netzwerk und Systemadministration

Einheit: Betrieb

Auf dieser Ebene werden die verwendeten Programme auf Systemebene betrachtet. Diese Programme sind diejenigen, die für die zu implementierenden Funktionalitäten notwendig sind. In diese Kategorie sind u.a. einzuordnen der Webserver oder auch Datenbank- und andere Applikationen die sich auf Systemebene befinden, auf die die Webanwendung zurückgreift.

Ebene 2 -Technologie

Phase: Build

Fachkenntnisse: IT-Sicherheit

Einheit: Betrieb, Entwickler

Hierbei handelt es sich um den Einsatz der richtigen Technologie und deren korrekten Anwendung. Dabei ist zu unterscheiden, ob die grundsätzlich falsche Technik eingesetzt wird oder die richtige Technologie verwendet wird aber diese in einer falschen Weise zum Einsatz kommt. Wird zum Beispiel eine Verschlüsselung verwendet, sollte diese auch die größtmögliche Länge für die Schlüssel verwenden, sonst werden die Möglichkeiten dieser Technologie nicht im gesamten Maße ausgenutzt.

Ebene 3 – Implementierung

Phase: Build

Fachkenntnisse: Softwareentwicklung

Einheit: Entwickler(Umsetzer)

Diese Ebene richtet sich vor allem an die Programmierer. An dieser Stelle sollen Fehler in der Programmierung verhindert werden. Ein Problem bei diesem Vorhaben stellt die Tatsache dar, dass dieser Teil der Entwicklung häufig aus Zeitmangel und Kostengründen vernachlässigt wird. Dazu zählen z.B. auch Testphasen um eventuelle Programmierfehler aufzuspüren und gegebenenfalls zu eliminieren. Viele Anwendungen werden veröffentlicht, obwohl sie nur ungenügende Tests durchlaufen haben.

Ebene 4 – Logik

Phase: Plan (Planen der Anwendung)

Fachkenntnisse: Softwareentwicklung

Einheit: der Anwendung zugehörige Fachstelle(Anforderer)

Hierbei handelt es sich um die Anwendungsbezogene Logik innerhalb des Systems, welches auch Interaktionen mit dem Benutzer beinhaltet. Beispiel dafür ist die Behandlung fehlerhafter Eingaben z.B. beim Login einer Anwendung. Ist diese nicht korrekt, wird es Dritten ermöglicht, Angriffe auf das System durchzuführen, welcher die Performance eines Systems nachhaltig beeinflussen kann. Dies könnte zum Beispiel eine Denial of Service(siehe 4.3) Attacke sein, welche zur Überlastung des Systems und dessen angebotenen Dienst führen kann. Allgemein ist dies die Ebene in der es um die Absicherung von Prozessen geht, wie das Registrieren neuer Benutzer oder dem Benutzer einen sicheren Login zu garantieren.

Ebene 5 – Semantik

Phase: Plan

Fachkenntnisse: Cooperate Identity, Unternehmensstrukturen, interne Abläufe

Einheit: Zentrale Organisation

In diesem Bereich geht es um die Integrität der Interaktionen mit dem Benutzer. So ist es hier besonders wichtig zu verhindern, dass Dritte die Anwendung missbrauchen um andere Benutzer zu täuschen. Dies sollte in den meisten Fällen nicht nur auf den Kontext der Anwendung reduziert werden. Denn aus dem Missbrauch einer Webanwendung können auch weitreichendere Schäden entstehen die die eigentliche Webanwendung nicht betreffen. Bekannte Beispiele auf die später noch genauer eingegangen wird sind : Phishing, Identitätsdiebstahl, Social Engineering.

Ebene 6 – Vorschriften und

Phase: Plan

Fachkenntnisse: Cooperate Identity, Unternehmensstrukturen, interne Abläufe

Einheit: Zentrale Organisation

In dieser Ebene kommen interne Vorschriften, Gesetzgebung und Regelungen von aussen zum tragen. Hierbei steht nicht mehr der technische Sicherheitsaspekt im Vordergrund sondern das Verhindern eventuellen Schäden durch die Nichteinhaltung dieser Regelungen und Vorschriften. Aus diesem Grund werden Teile dieser Ebene im Weiteren nicht mehr genauer beleuchtet. Abhängig vom Teilgebiet der Anwendung können die Anforderungen in diesem Bereich stark variieren.

3.2 Bereiche der Sicherheit

Sicherheit kann in folgende Elemente aufgeteilt werden:

1. Authentifizierung:

Hier geht es um das Identifizieren der Clients oder der Benutzer. So dass dokumentiert werden kann wer die Anwendung verwendet hat und zu welchem Zweck. Diese Endbenutzer werden auch *Prinzipale* genannt.

2. *Autorisierung:*
Hier geht es um den Sachverhalt, was darf der jeweilige Benutzer oder Client. Dabei kann es sich um Operationen, Dateien, Tabellen und andere Ressourcen handeln. Auf Systemebene zählen noch Konfigurationsdateien und Registrierungsschlüssel dazu. Ausser Ressourcen werden auch noch Operationen verwaltet, die der jeweiligen Anwendung zugeordnet sind, wie z.B. eine Überweisung bei eine Internet-Banking Anwendung.
3. *Überwachung:*
Hier ist das Stichwort Nachweisbarkeit, um einem Kunden einen bestimmten, von ihm abgeschlossenen Vorgang, nachweisen zu können. Es könnte passieren, dass ein Benutzer eines Webshops bestreitet, ein Produkt in einer gewissen Menge bestellt zu haben. In diesem Fall sollte der Betreiber dieses Systems dem Kunden nachweisen können, dass dieser die Bestellung in dieser Form vorgenommen hat.
4. *Vertraulichkeit:*
Anderst formuliert geht es hier um den Datenschutz. Es ist sicherzustellen, das Daten vertraulich behandelt werden und nicht in die Hände Dritter gelangen, die nicht autorisiert auf diese Daten zuzugreifen. Weitere Schlagworte sind hier Verschlüsselung und Zugriffssteuerung.
5. *Integrität:*
Daten sind vor ungewollten oder böswilligen gewollten Änderungen zu schützen. So darf bei einem Datentransfer, nach Vollendung des Transfers dürfen diese Daten keine Änderungen aufweisen. Beim Transfer über Netzwerke stellt dies eine Bedrohung dar.
6. *Verfügbarkeit:*
Das Ziel vieler Angriffe auf Anwendungen im Netz ist es diese zum Absturz zu bringen, so dass auch für die legitimen Benutzer kein zugriff mehr möglich ist. Dies sollte verhindert werden, um die Verfügbarkeit des Systems in hohem Maße zu gewährleisten.

4 Beispiele für Gefahren und Gegenmaßnahmen

Nicht nur die Webanwendung selbst ist den Gefahren böswilliger Angriffe ausgesetzt, sondern meist auch die oft ahnungslosen Benutzer. Diese Benutzer sind sich den Gefahren oft nicht bewusst, denen sie ausgesetzt sind. Oftmals denkt der Benutzer, dass alles in Ordnung ist, da er weder eine Warnung noch eine Fehlermeldung bekommt. Jedoch hat ein Dritter die eingegebenen Daten beim Transfer der Daten ausspioniert, und kann diese nun nutzen um dem Benutzer Schaden zuzufügen und zu seinem Vorteil auszunutzen. Dies kann geschehen, wenn die Daten ohne die nötigen Sicherheitsmassnahmen über das Web übertragen werden. In vielen Fällen handelt es sich um sensible Daten wie Kontonummern oder Kreditkartennummer. Jedoch kann der Benutzer dies nicht selbst beeinflussen, ausser er ist die mangelnde Sicherheit dieses Systems informiert. Durch den Sachverhalt, das solche Anwendungen heutzutage praktisch von jedem Haushalt aus zu erreichen sind, ist die Zahl

der Benutzer stetig gestiegen und steigt weiter an. Dadurch können viele Dinge bequem von zuhause abgewickelt werden. Dabei sollte auf den Bekanntheitsgrad und die Akzeptanz der Anwendung geachtet werden. Bei großen, allgemein bekannten Webanwendungen ist die Gefahr einen Schaden davonzutragen weitaus geringer als bei Anwendungen, welche nur ein kleines Publikum ansprechen. Die Mehrzahl der Benutzer hat kein Hintergrundwissen über die eingesetzte Technik des Datentransfers und Wörter wie Session oder Phishing sind für sie kein Begriff. Somit ist es die Aufgabe eines jeden Unternehmens, für die Sicherheit der Benutzer zu sorgen, so dass diese die Anwendung mit ruhigem Gewissen benutzen können. Auf der anderen Seite ist es Aufgabe des Benutzers sich über eventuelle Gefahren zu informieren.

Einige Beispiele:

4.1 Identitätsdiebstahl und Enumeration

Hierbei handelt es sich um den Diebstahl persönlicher Daten durch dritte Personen. Wenn eine Person sich Benutzerkennung und Passwort einer anderen Person zu Eigen macht, diese böswillig einzusetzt um dieser Person Schaden zuzufügen und zu seinem Vorteil zu nutzen. Es gibt verschiedene Formen dieser Handlung, die häufigsten sind:

Kreditkartenbetrug, Kontenraub und Bankbetrug. Allein dies verursachte laut einer Studie im Jahre 2002 einen Schaden von insgesamt rund 33 Milliarden Dollar für Geschäftskunden und knapp vier Milliarden Dollar bei Privathaushalten. Es gingen insgesamt 168.000 Anzeigen sowie 380.000 Beschwerden wegen Identitätsdiebstahls ein. Hieraus wird ersichtlich, wie wichtig die Verschlüsselung der Daten beim Transfer über das Web ist. Den folglich möchte kein Unternehmen verantwortlich gemacht werden, Identitätsdiebstahl zuzulassen, aufgrund von mangelnden Sicherheitsvorkehrungen. Ein Grund, weshalb ein solcher Diebstahl möglich sein könnte ist, wenn bei einer Anmeldung das Prinzip der Enumeration verwendet werden kann. Für Enumeration anfällig sind vor allem Daten wie Benutzerkennung, SessionIDs, Dokumentennummern.

ein Beispiel: Benutzerkennungen und Passwörter

Ein Benutzer darf an einem System seine Benutzerkennung selbst wählen, ist diese bereits vergeben bekommt dieser eine Fehlermeldung und wird gebeten eine andere Kennung zu wählen. Hier könnte nun durch eine Enumeration herausgefunden werden, welche Kennungen bereits existieren.

Ist die Benutzerkennung nun bekannt, kann über selbiges Verfahren das zugehörige Passwort ermittelt werden. Bei einem 5 stelligen PIN, wäre dies ohne Gegenmaßnahmen in ca. 50000 versuchten zu erreichen.

4.2 Maßnahmen gegen Enumeration

Zuerst einmal müssen diese Angriffe erkannt werden. Dabei müssen sie von normalen bzw. berechtigten Zugriffen unterschieden werden. An folgenden Kriterien kann ein solcher Angriff vermutet werden: Zugriffe in einem bestimmten Zeitraum übersteigen die normale Anzahl beträchtlich. Die verwendeten Parameter sind nach einem bestimmten Muster generiert, eventuell über einen Algorithmus, welcher diese Parameter erstellt. Viele aufeinanderfolgende Zugriffe von der selben Quelle. Jedoch können die gesendeten

Parameter und die Quelle von der sie gesendet werden auf einfache Weise verschleiert werden. Um solche Angriffe trotzdem verhindern zu können, gibt es die Möglichkeit, die Anzahl der möglichen Zugriffe in einer bestimmten Zeitspanne zu beschränken. Hierzu muss ein Durchschnitt errechnet werden, wieviele Neuanmeldungen in einem bestimmten Zeitraum stattfinden, danach kann die Menge der möglichen Zugriffe in einer solchen Zeitspanne festgelegt werden um Enumeration bei Benutzererkennung zu verhindern. Bei Passwörtern ist eine Gegenmaßnahme, dass nur Passwörter einer bestimmten Länge zugelassen werden, dabei beliebige Zeichen verwendet werden können. Gleichzeitig wird die Sicherheit dieses Passwortes geprüft und dem User mitgeteilt ob es sich um ein sicheres Passwort handelt. Jedoch machen viele Gegenmassnahmen gegen Enumeration das System anfällig für DoS-Angriffe.

4.3 DoS-Angriffe

Die Abkürzung DoS bedeutet Denial of Service. Hier zwischen primitiven DoS-Angriffen und DRDoS(Distributed Reflected Denial of Service). Bei einer primitiven Attacke wird der Dienst eines Servers so stark belastet, dass dieser mehr Anfragen bearbeiten muss als er im Normalfall im Stande ist zu verarbeiten. Dadurch werden reguläre Anfragen garnicht mehr oder nur noch sehr langsam bearbeitet. Eine andere Version dieses Angriffs ist es einen Fehler in der Programmierung auszunutzen, um einen Absturz der Serversoftware auszulösen.

Ein Szenario im Falle eines DRDoS wäre, ein Angreifer fälscht die IP Adresse seines Opfers und stellt Anfragen an einen Internetdienst, worauf die Antworten zum Opfer gelangen, was in Folge dessen den eigentlichen DoS darstellt. Solche Angriffe werden auch häufig über Würmer durchgeführt, zum Eigenschutz des Angreifers.

Varianten von DoS:

- Bandbreitensättigung
- Ressourcensättigung
- Herbeiführung von System- und Anwendungsabstürzen

Bekannte Beispiele:

21. August 2006: DDoS-Angriff auf ogame.de und alle anderen OGames

Januar 2006: DDoS gegen den Server von milliondollarhomepage.com (6 Tage Ausfall)

September 2004: DDoS u.a. gegen Symantec

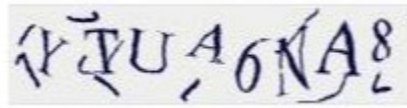
Gegenmassnahmen

Ob ein Angriff dieser Art zum Erfolg führt, hängt auch davon ab, ob ein System richtig konfiguriert ist. „Sicherheit ist kein Produkt, sondern ein kontinuierlicher Prozess.“ ist ein Zitat vieler Sicherheitsexperten. Sicherheitsvokehrungen müssen immer erweitert und weiterentwickelt werden.

Ein einfaches Verfahren um dies zu verhindern ist, den User nach einer bestimmten Anzahl von Eingabeversuchen aufzufordern, eine Eingabe zu tätigen. Diese Eingabe wird so gewählt, dass sie nicht von einer Maschine verarbeitet werden kann. Es existiert ein weiteres Verfahren, welches im Stande ist, die geistige Überlegenheit gegenüber einer Maschine auszunutzen. Bei dem Vorgang der Registrierung eines neuen Benutzers, wird dem User

eine Zeichenkette gezeigt, welches das menschliche Auge entziffern kann, jedoch ist es einer Maschine unmöglich diese zu erkennen, da sie durch Störungen beinahe unkenntlich gemacht wird. Diese Zeichen nennen sich *Captchas* (Completeloy Automated Public Turing Test to Tell Computers and Humans Apart).

Beispiel:



Hier wird der User aufgefordert die Zeichenkette „YTUA6NA8“ einzugeben. Dies ist wohl nur schwerlich von einer Maschine zu erkennen. Bei diesem Verfahren ist darauf zu achten, dass die Captchas so gewählt werden die von einer Maschine nicht lösbar sind. Davon ist die Effektivität dieses Verfahrens essentiell abhängig. Nur wenn die Zeichenfolge korrekt eingegeben wurde wird der darauf folgende Schritt durchgeführt. Eine weitere Möglichkeit ist eine Frage die eine Maschine nicht beantworten kann. Zum Beispiel die Frage: „Wie leuchtet das dritte Wort auf dieser Seite?“. Nur bei korrekter Antwort wird die Anfrage des benutzers weitergeleitet. So können maschinell durchgeführte DoS Attacken vermieden werden.

4.4 Phishing

Phishing stellt keine direkte Gefahr für Webanwendungen dar, jedoch für die Benutzer von Websystemen. Unter Phishing versteht man das Imitieren von Webanwendungen. Dies wird dazu benutzt um Zugangsdaten von Benutzern real existierender Webanwendungen herauszubekommen. Hierzu werden Mails massenweise an Emailadressen versendet, in der der User aufgefordert wird seine Zugangsdaten in einer Formular einzugeben. Zu diesem Formular gelangt er über einen Hyperlink der sich in der Mail befindet.

Hält der Benutzer diese Email für legitim, führt er dies aus und gelangt auf die imitierte Webseite, sobald er dort Seine Zugangsdaten eingegeben hat und diese abschickt, hat der Angreifer sein Ziel erreicht. Die Zugangsdaten sind nun auch im Besitz des Betrügers und können böswillig missbraucht werden, sollte der Benutzer seine Daten nicht umgehend ändern.

Schutz gegen diese Emails gibt es durch die meisten Virens Scanner oder durch die Emailprogramme. Zum Beispiel kann bei den meisten Emailprogrammen die HTML Darstellung und auch die Funktion Javascript deaktiviert werden. Programme wie Mozilla Thunderbird oder Internet Explorer 7 warnen vor Phishing Webseiten, basierend auf einer Blacklist oder typischen Merkmalen von Phishing Webseiten. In diesem Fall liegt es am Benutzer selbst sich ausreichend vor solchen Betrugsversuchen zu schützen. Durch den Einsatz der richtig konfigurierten Software und durch das Prüfen der Herkunft empfangener Emails. Ist beim Lesen eine Email als risikoreich einzustufen, so sollte diese sicherheitshalber gelöscht werden. Häufige Ziele von Phishingbetrügern sind Benutzer von Internet Banking.

4.5 SQL Injektion und Data Validation

Unter einer Sql Injektion versteht man das unerlaubte Einschleusen von Sql Befehlen in ein System um Daten auszuspionieren oder Kontrolle über den Server zu bekommen. Ein System ist besonders anfällig für solche Attacks, wenn die vom Benutzer eingegebenen Parameter nicht gefiltert und überprüft werden. Diese Prüfung der Daten sollte stattfinden bevor der Zugriff auf das Subsystem stattfindet (Datenbanken oder Systembefehle). Ein gutes Mittel um dies zu verhindern sind Stored Procedures, hierbei werden die eingegebenen Parameter an ein Programm an die Stored Procedures übergeben. Erst hier werden die SQL-Befehle erzeugt. Wird der Input gefiltert, so sind potentiell gefährliche Zeichen zu maskieren. Hierzu zählen bei SQL z.B. % (Wildcard), <> für Vergleiche, \n für einen Zeilenumbruch. Wenn eine Validierung der Eingabedaten stattfindet, ist eine Whitelist unbedingt eine Blacklist vorzuziehen. Bei einer Whitelist ist grundsätzlich alles verboten bis es explizit erlaubt wird. Bei einer Blacklist ist grundsätzlich alles erlaubt, bis es ausdrücklich verboten wird. Eine Whitelist blockiert oft etwas zuviel, jedoch ist dies keine Gefahr für das System und kann ohne Probleme geändert werden. Bei einer Blacklist besteht die Gefahr etwas vergessen zu haben, diese Lücke stellt eine erhebliche Gefahr für das System dar, bis diese behoben wurde. Filter sollten einfach gehalten werden. Kompliziertere Filter sind durch Sequenzen von einfachen Filtern zu ersetzen.

Beispiel:

Normaler Aufruf:

Aufruf: `http://webserver/cgi-bin/find.cgi?ID=42`

Erzeugtes SQL: `SELECT author, subjekt, text FROM artikel WHERE ID=42`

SQL-Injektion:

Aufruf:

`http://webserver/cgi-bin/find.cgi?ID=42;UPDATE+USER+SET+TYPE="admin"+WHERE+ID=23`

Erzeugtes SQL:

`SELECT author, subjekt, text FROM artikel WHERE ID=42; UPDATE USER SET TYPE="admin" WHERE ID=23`

Dieses Beispiel zeigt, wie man auf einfache Weise Daten manipulieren kann, wenn die Eingangsdaten nicht gefiltert werden, d.h. der Aufruf der hier vorliegenden SQL Injektion wird wie ein regulärer Aufruf behandelt, wodurch die Daten verfälscht werden.

Für Java gibt es die Klasse `PreparedStatement`, mit der sich folgende Aufrufe ergeben:

```
PreparedStatement ortabfrage = con.prepareStatement("SELECT * FROM
Benutzer WHERE Ort = ?");
ortabfrage.setString(1, ort);
ResultSet rset = ortabfrage.executeQuery();
```

Da es sich hier um einen parameterisierten Aufruf handelt, kann dieses Statement nicht für böswillige Angriffe verwendet werden. Es werden die Daten gefiltert und der Typ

entsprechend der jeweiligen Abfrage angepasst. Prepared Statements sind in viele weitere Programmiersprachen integriert, so z.B. in Microsofts .NET-Framework, PHP.

4.5 Verwalten der Session

Die Session ist das Merkmal, mit dem Anfragen an einen Webserver dem jeweiligen User zugeordnet werden. Dies geschieht mit Hilfe der SessionID, welche bei jeder Anfrage mit übertragen wird. Dies geschieht meist mit Hilfe von Cookies, die als Träger verwendet werden. Dies ist notwendig, da es sich bei http um ein verbindungsloses Protokoll handelt, die Benutzer aber trotzdem über eine Anfrage hinaus identifiziert werden müssen. Das Management der Session stellt eine hohe Problematik dar. Aus diesem Grund sollte hier stets auf wirksame Maßnahmen geachtet werden. Hierbei gibt es die Möglichkeit, durch Verschlüsselung die Entführung(Hijacking) dieser Daten zu verhindern oder zumindest erheblich zu erschweren. Dies kann über das Setzen des secure Flags geschehen. Dadurch wird garantiert, damit der Browser niemals ein Cookie über eine unverschlüsselte Verbindung sendet. Dazu ist noch das HttpOnly Flag im Cookie zu setzen, dass das Auslesen mit JavaScript verhindert. Dies stellt sich aber oft als problematisch dar. Ältere Browser unterstützen diese Funktionalität nicht, was bedeutet dass sie trotz setzen des Flags weiterhin mit JavaScript ausgelesen werden können.

Das setzen eines solchen Cookies sieht folgendermassen aus:

```
set-cookie: SESSIONID = isdnjd67.....; path=appl/portal; secure; HttpOnly;
```

Zur weiteren Sicherheit kann die SessionID an die IP-Adresse des Benutzers gebunden werden. Wenn beim Eintreffen einer Anfrage die IP-Adresse nicht mit der für die Session hinterlegten IP-Adresse übereinstimmt, wird die Session auf invalid gesetzt. Dies ist ein Problem für Benutzer, welche sich hinter einem Proxy oder Router befinden. Sollte sich die IP-Adresse einer dieser Komponenten ändern, unterscheidet sich diese von der in der Anwendung hinterlegten, somit wird die Session unterbrochen. Vor dem Einsatz dieser Technik ist zu klären ob dies ein Problem für die vorliegende Anwendung darstellt. Jedoch ist hier zu erwähnen, dass ein zu stark gesichertes System ein kleineres Problem darstellt, als ein zu wenig geschütztes.

5 Fazit

Die Sicherheit von öffentlich zugänglichen Webanwendungen gewinnt immer mehr an Bedeutung um Unternehmen oder Personen keinen Schaden zuzufügen. Infolgedessen sollten immer die notwendigen Massnahmen ergriffen werden um die Schädigung des Systems oder Personen zu verhindern. Dies kann nur durch eine gründliche Analyse der gegebenen Situation und den Prozessen der zu entwickelnden Anwendung geschehen, um die Risiken zu erkennen und diese zu minimieren. Oft ist dies nicht immer einfach zu realisieren und bei großen Anwendungen auch nur mit großem Aufwand zu erreichen. Jedoch können schon durch das Einhalten von Programmierrichtlinien die größten Schwachstellen eliminiert

werden. Die Sicherheit hat einen so großen Stellenwert, dass es sich lohnt auch einen größeren Aufwand in Kauf zu nehmen und Sicherheit bei der entwickelten Anwendung einen hohen Stellenwert beizumessen. Zusätzlichen Schutz bietet verschiedene WAF-Software(Web Application Firewall).

6 Literatur

- [1] secureNet,Sicherheit von Webanwendungen, (im Auftrag vom Bundesamt für Sicherheit in der Informationstechnik)
Maßnahmenkatalog und Best Practices
<http://www.bsi.de/literat/studien/websec/WebSec.pdf>
Version 1 vom 1. August 2006
- [2] Wikipedia(Autor unbekannt), Webanwendungen
<http://de.wikipedia.org/wiki/Webanwendungen>
- [3] Wikipedia(Autor unbekannt), Denial of Service
http://de.wikipedia.org/wiki/Denial_of_Service
- [4] Wikipedia(Autor unbekannt), Session-Hijacking
http://de.wikipedia.org/wiki/Session_Hijacking
- [5] secureNet, Die semantische Ebene der Sicherheit bei Webanwendungen.
<http://www.securenet.de>
Version vom: Whitepaper Mai 03
- [6] Dr. Micheal David, Angriffe auf Sicherheitsinfrastrukturen der IT.
Version vom: unbekannt
- [7] Thomas Schreiber , Ein Klassifizierungsschema zur Sicherheit von Webanwendungen.
http://www.securenet.de/papers/Klassifizierungsschema_Web_Application_Security.pdf
Version: It-Sicherheitskongress 2005
- [8] Detlef Weidenhammer , Sicherheit bei Webanwendungen.
http://www.gai-netconsult.de/download/security/whitepaper/WP_WAsec%20v2_0.pdf
Version: Januar 2006
- [9] Abbildung 1: Architektur und Verlauf einer Web-Anwendung
http://webeng.cs.uni-magdeburg.de/kapitel_1/Architektur_und_Verlauf_einer_Web-Anwendung.html